UNIVERSITY OF CINCINNATI

Date: 3-10-2008

I, Eric Timothy Dick,

hereby submit this work as part of the requirements for the degree of:

Master of Science

in:

Medical Physics

It is entitled:

A Survey of CT Phantom Considerations for the Study of Blooming Artifacts as Observed in CT Coronary Angiography Studies: A Preliminary Study

This work and its defense approved by:

Chair: Lisa Lemen, Ph.D.

Howard Elson, Ph.D.

Michael Lamba, Ph.D.

A Survey of CT Phantom Considerations for the Study of Blooming Artifacts as Observed in CT Coronary Angiography Studies: A Preliminary Study

A thesis submitted to the

Division of Research and Advanced Studies
of the University of Cincinnati

in partial fulfillment
of the requirements for the degree of

MASTERS OF SCIENCE (M.S.)

in the Department of Radiology of the College of Medicine

March, 2008

By

Eric Dick

B.S., University of Central Florida

Committee Chair: Lisa Lemen, Ph. D.

Abstract

The presence of image “blooming” artifacts, in particular with respect to highly calcified plaques, has been a major impediment to the implementation of Multi-detector Computed Tomography (MD-CT) as an alternative to standard angiography in routine clinical practice. A beam hardening phantom system was developed to determine the dependence of blooming with respect to measured density in Hounsfield Units (HU), of 5mm diameter plaques. Custom software was developed to provide reproducible objective measurements of plaque size. Plaque diameter was measured using a series of axially distributed profiles centered on the object. Reconstruction blur was shown to be a significant, density dependent contributor to standard clinical blooming measurements. Beam hardening was shown to not affect measurements of sample diameter in a blur corrected study. Furthermore, two additional phantom designs have been proposed and evaluated for the future study of partial volume and cone beam affects on blooming.

Table of Contents

List of Tables

List of Figures

1: Introduction

1.1: Atherosclerosis

Cardiovascular Disease (CVD) is the leading cause of death in both males and females in the United States. The seriousness of the problem cannot be overstated as CVD accounts for more than 36% of all deaths in the United States as compared to cancer which accounts for 23%. The world health organization reports that every year CVD causes 16.7 million deaths worldwide and 29% of all deaths globally. Furthermore, it is estimated that 1 in 3 American adults has some form of CVD. [15]

The American Heart Association predicts the total cost of CVD-related medical care and disability in the United States will be $431 billion at the close of 2007. Atherosclerosis, accounting for nearly three fourths of CVD mortality, is a slow and complex disease that usually starts in early childhood. This process starts when fats and other substances build up in the endothelial lining (inner layer) of the arterial wall. These fatty deposits are calcified by an excessive inflammatory-fibroproliferative response [16,17]. Atherosclerosis is the principal cause of heart attack and stroke [16].

Coronary angiography has become an established clinical procedure for some scenarios, such as coronary artery anomalies, bypass patency, and surgical planning [18]. Acute coronary events are caused by the erosion or rupture of atherosclerotic plaque into the vessel lumen (area of blood flow). Since targeted therapy can substantially reduce the risk of such events, identification of at-risk individuals for acute coronary events is of great interest. Risk prediction algorithms are commonly used in clinical practice but have limited predictive power because “frequently patients with

acute coronary events have no ‘traditional’ risk factors at all” [12]. Individualized risk stratification may be accomplished by assessing luminal integrity using various imaging modalities.

Conventional Coronary Angiography (CCA) has been considered the gold standard for the evaluation of luminal integrity; however, there is a need for less invasive coronary angiography methods [1]. CT Angiography is one such modality offering non-invasive imaging that permits detection, quantification, and possibly characterization of atherosclerotic plaque [5]. Both CCA and CTA studies are performed with the use of radio-opaque contrast agents to better visualize blood flow. The contrast material appears white where present; therefore, luminal narrowing is indicated by a dark (shadowed) region within the image of the coronary artery.

Modern CTA is typically performed using a Multi-Slice Spiral Computed Tomography (MD-CT) system which produces cross-sectional images of the patient. These images are reconstructed from the raw detector data. One of the main features limiting the clinical usefulness of modern CTA is its poor discrimination of the boundaries of coronary calcified plaques [9].

1.2: Multi-Slice CT

CT image reconstruction is a highly involved process. Most of the systems today use some variation on the back projection approach to image reconstruction. A typical reconstruction process includes preprocessing, convolution, image formation (backprojection) and post-processing. Many of the preprocessing and post-processing steps are proprietary in nature and are not available in the public domain [10]. The subtleties of the reconstruction process are beyond the scope of this document.

Raw CT data consists of the responses of several detectors to patient attenuated X-ray's over a range of angles axially around the patient. Modern systems can collect data in axial (step and shoot) or spiral (helical) modes. Each detector response is the superposition of its quantum mottle and the path integral of the X-ray attenuation from the X-ray source across the patient to the detector. In axial mode the data is collected in plane with the reconstructed image. In spiral mode the data is collected in a spiral about the length of the patient and are reconstructed using interpolation of the data. Spiral CTA is by far the most commonly used technique in coronary CTA.

Ideally while the signals through a single block of homogeneous material should be the same, in practice this is rarely the case, due to noise and artifacts. Preprocessing steps are used in an attempt to correct for non-ideal detector data. Preprocessing is not limited to but may include compensations for X-ray source, mechanical, detector, and patient induced imperfections or calibrations.

Backprojection, originally proposed by Radon in 1917, is simple in concept; it is the idea that one may reconstruct an image of the X-ray attenuation characteristics of the patient by projecting back the detector from a given angle data as a single valued streak in the image field of view. As this is done for each angle, detector streaks begin to overlap. The superposition of all the backprojected detector signals creates an image which is blurred, and has poor edge definition. Conventional CT scanners use the frequency space representation of the patient image in preprocessing. Fourier transformations are used to perform the conversion from spatial to frequency coordinates. In frequency space algorithms using convolution kernels are applied to the image to remove back projection blur and other image artifacts. The use of these techniques comes with the need to make certain assumptions about the homogeneity of the detector data. Violations in these assumptions can lead to unexpected effects when converting back from the frequency to spatial coordinates such as the appearance of streaking, shading, and banding artifacts.

1.3 MD-CT Artifacts

Perhaps the most overlooked aspect of image artifacts is how they are defined.

> *Theoretically, an image artifact can be defined as any discrepancy between the reconstructed values in the image and the true attenuation coefficients of the object. Although this definition was broad enough to cover nearly all types of non-ideal images, it has little practical value… In fact, we can further assert that the majority of the pixels in a CT image are "artifacts" in some shape or form. - Hsieh J. [10]*

In practice, we want to examine only those discrepancies that affect the accuracy of the image interpretation.

CT systems are inherently more prone to artifacts as compared to conventional radiography. CT artifacts come from several sources such as: aliasing, beam hardening, cone beam effects, interpolation, motion of the subject, partial volume averaging, scatter, noise, incomplete projections, and mechanical factors relating to X-ray tube and detector design [10]. Generally the appearance of CT artifacts is the result of the processing of specific inconsistencies in the detector data. For this reason, CT artifacts are generally categorized by the cause of the data inconsistency, rather than in their visual appearance. The most common physical causes to image artifacts are beam hardening, the presence of metal in the object being imaged, partial volume averaging, cone beam effects, and patient motion. Computer processing imposes limitations on the ideal mathematical reconstruction process, and results in a blurring of the image, characterized by the Point Spread Function (PSF) of the imaging system.

1.3.1: Beam Hardening

Beam hardening artifacts are associated with highly attenuating objects and are subdued with the use of filter algorithms, correction based algorithms and beam filters. Beam hardening artifacts can present with the appearance of pronounced edges, streaks, and environmental density suppression commonly referred to as shading artifacts [3,8,20,21].

The image reconstruction algorithm assumes that the attenuation of the incident X-ray beam is exponentially related to the thickness of the object, cf. Beer-Lambert-Bouguer law [21]. Beam hardening artifacts are the result of the energy dependence of the attenuation coefficient coupled with the polychromaticity of the source used in medical CT imaging. The high attenuation of the object results in the preferential absorption of low energy photons which causes the average energy of the X-ray beam to increase [21]. Since higher energy beams are less attenuated than the origional X-ray beam, data sets collected for regions near high-density objects exhibit attenuation inconsistencies tending to be lower on average than the actual values. Beam hardening thus results in environmental intensity suppression in the image. Extreme differences in the attenuation readings violate assumptions of the reconstruction algorithms and cause streaking.

Metal artifacts are an extreme case in which artifacts are caused by objects of such high attenuation (metal) that the image intensities are beyond the contiguous memory limitations defined in the computer algorithm. Metal artifacts can be reduced by changing the data type to one which is capable of holding higher values (switching to extended HU). Metal artifacts are highly sensitive to motion and often associated with the partial volume effect [10].

1.3.2: Partial Volume Averaging

Partial volume averaging artifacts are correlated with the size, shape, and orientation of an object relative to the imaging axis and detector size. When the slice width chosen is greater than the in plane pixel size, partial volume averaging is stronger along the axial (slice width) direction. Of note, spiral scanning uses interpolation to construct the image plane data. Thus, partial volume artifacts may vary in severity within the image plane: edge blurring, an increase in apparent size, streaking, local density averaging, and environmental density shifts may occur. Because of the divergence of the X-ray beam in the direction of the source's rotational axis, the artifact tends to appear with an angular dependence [10]. Indeed, an object in the beam path at 0^0 may not be in the beam path at 180^0, resulting in inconsistencies within the backprojection data. This phenomena leads to discrepancies in the projection data set which ultimately cause the image artifacts [10].

1.3.3: Cone Beam

Cone beam artifacts are observed uniquely with multi-slice CT systems [10]. Cone beam artifacts manifest in the image as edge distortion, streaking, patterned noise (rings and streaks), and large density shifts [10]. Standard single-slice reconstruction algorithms assume that the path of the X-ray beam is perpendicular to the detector surface, not divergent. MD-CT requires a more divergent collimator setting than does single-slice CT, resulting in an angular relationship between the detector and the beam path that violates the traditional reconstruction assumptions. "For example, on a LightSpeed ULTRA CT scanner (GE Medical Systems, Milwaukee, WI) operating in an 8x2.6mm scan mode, the corresponding cone angles for the four slices on one side are 0.13, 0.40, 0.66, and 0.93 deg. respectively… If we consider a point located at the edge of the 50cm field of view, a 0.93-deg. cone angle leads to a 4.04mm error between the backprojected position and it's actual location …" [10]. Cone beam artifacts, for large cone angles, cannot be eliminated with 3D backprojection alone; additional corrections are required [10]. Interestingly, cone beam artifacts are effectively suppressed in spiral acquisition mode for smaller scan angles due to the planar interpolation [10].

1.3.4: Motion

Motion artifacts are observed when the subject moves during the process of data acquisition: bulk patient movement, breathing, heartbeat, etc. Images with motion artifacts present with streaking near high-density structures, blurring, ill-defined boundaries, and shape distortion. Motion artifacts occur in three dimensions with the object moving within the scanning plane or into and out of it (causing inconsistencies in the projection data) [10]. Though the more severe streaking artifacts induced by patient motion are seen around high-density objects, these streaks are not caused by beam hardening. Due to the nature of the reconstruction process, data inconsistencies at transitional boundaries (from low to high attenuation) are pronounced in the image [10]. Motion gating can improve image quality, but a more subtle presentation of motion artifacts could be caused by point heterogeneities in the motion of the subject. Although motion gating may provide an opportunity for better data collection, it is possible that small irregularities in motion are not truly frozen and result in small regions of artifact distortion. Additionally, such artifacts may become pronounced near high-density objects due to the aforementioned effects of the reconstruction process.

1.3.5: Reconstruction Blur

The CT image formation involves blurring together integrated attenuation data; therefore, CT images are fundamentally blurry. Theoretically, Fourier reconstruction techniques can fully compensate for this fundamental blurriness. But due to the limitation of today's computers to finite and discrete mathematics, in practice the Fourier reconstruction technique is imperfect in correcting for natural image blur. Inherent in the reconstruction process are approximation errors that add additional noise to the image.

Filtered back projection (FBP) is the method of standard image reconstruction for conventional CT scanners. The detector data is discretely sampled by the computer and then backprojected into image data. In a computer, the Fourier transformation of the image data is in fact a truncated and discretely sampled form of the ideal continuous model. Converting back from frequency space into the discretely sampled pixels of the spatial coordinate system causes further approximations. The FBP method utilizes a combination of filters (low-pass, band-pass, and high-pass) to reduce noise and enhance edges. Use of these filters inevitably brings blurring to the resulting image.

After post processing, the image is then converted from attenuation coefficient values into unsigned integers that further truncate the image intensity data. The human eye does not have sufficient contrast discrimination to see all of the data; also computer monitors are limited by their display contrast abilities (typically 8-bytes). The result is a need to further sample the data into a specific range of intensities (the window) for optimal display.

Routine image formation practices can result in an over-representation of objects due to the inherent addition of penumbra to object edges. The penumbra (also referred to as the skirt) of an object scales with its intensity. Thus, high intensity objects are expected to have a wider skirt than those of a lower intensity. Over-representation of calcified regions, due to this inherent object blooming, is expected to be greater than resolution limitations due to detector size, which should represent the maximum theoretical resolution of the CT system.

1.3.6: Considerations for Artifact Suppression

Beam hardening, partial volume averaging, the cone beam effect, and motion artifacts are all artifacts which can be reduced if not removed entirely. Techniques to subdue image artifacts are typically specific to the physical cause of the effect. For example, beam hardening artifacts may be subdued by using a higher kV, a beam filter, or by the implementation of a variety of correction algorithms. Special techniques would be needed to combat an artifact caused by a superposition of various physical effects. However, additional artifacts may be introduced to an image (noise) while attempting to correct for another [14]. For such complex artifacts it may be most beneficial to reduce the primary contributing factor, rather than attempt a combination of simultaneous corrections. It is therefore beneficial to understand the quantitative nature of the various factors that contribute to artifacts such as blooming. Knowing the magnitude of each effect allows for a more targeted approach to its suppression.

1.4: Blooming

Blooming, as defined here, is an artifact category characterized by an increase in the apparent size of a structure often accompanied by streaking, and/or environmental density artifacts that affect the image of adjacent structures [4,13,14]. We make the distinction between blooming as pertains to the FWH-Mean of the image object (true blooming) and that of the image object at an arbitrary static level (perceptual blooming). The presence of blooming artifacts makes determination of the true size, shape and radiopacity (Hounsfield Units) of adjacent disease/normal tissue difficult. Clinically, blooming can affect the accuracy of the measurement of luminal integrity, stent restenosis, and the total burden of coronary calcified plaque [6,12,19].

Despite the general acknowledgment of its existence, blooming remains strangely ill-defined in the literature. Many authors either avoid the name altogether, refer to blooming in quotes, or use statements such as “the so called ‘blooming’ artifact.” A survey of the literature reveals disagreements between authors as to the root cause of blooming. Indeed, these authors tended to state that blooming was caused exclusively by either beam hardening or partial volume effects [2,3,6,8,9]. In most cases these authors, however, fail to provide direct experimental evidence of their claims.

Blooming tends to be correlated with the presence of high-density objects, perhaps causing some to jump to the conclusion that beam hardening is the cause. Additionally, slice widths and in plane resolution of current systems can be large in comparison to the size of thrombotic plaques and stent struts, thereby indicating partial volume effects. The nature of the reconstruction algorithm, however, is such that high-density objects tend to exacerbate many types of artifacts [10]. Therefore, the same evidence could also suggest that blooming may be a superposition of many effects.

Strong evidence exists that the blooming seen in images of stents has a significant partial volume component. Computer models have shown that slice width is a major factor in the ability to determine luminal narrowing within stents, which correlates directly with the partial volume averaging affect on blooming [8]. These computer simulations have shown that the severity of the size distortion was highest when the cross-section of the vessel was perpendicular to the slice plane, thus correlating the vessel's orientation with the amount of blooming. Due to the strong geometric dependence of the partial volume phenomena, one would not expect to see these image artifacts exclusively with high-density objects.

Improvements in the evaluation of stenosis have been observed with improvements in slice width, in particular with the move to beyond 16 slice systems [19]. That is, decreased slice width shows a strong reduction in blooming as observed with stents. However, despite the move to 64 slice scanners (offering 0.6mm slice widths or less) blooming from calcified plaques continues to remain a problem [2,9].

Aside from the partial volume averaging, contributions to the blooming artifact could come from two other sources that would not show improvement with increased spatial resolution: beam hardening or the cone beam effect. Additionally, the heterogeneous nature of point motion within the heart could cause motion artifacts that could appear as a blurring of structures (blooming).

On the other hand, beam hardening has been a particularly appealing potential cause of blooming because of its strong association with high-density objects where blooming is commonly seen (calcifications and stents). The association of the artifact with a specific material type (high-density plaques) correlates better with beam hardening than the partial volume theory. Beam hardening is the change in the multi-spectral energy distribution of the X-ray beam, which violates assumptions made by the CT reconstruction algorithms. Beam hardening would be associated with high-density objects as well as the thickness of the objects (as observed by the X-ray beam path).

An independent quantification of the various possible contributions to the blooming artifact should be accomplished to better clarify the magnitudes with which beam hardening, partial volume, and cone-beam artifacts each contribute. Stratification of the various possible contributors to blooming may help to focus efforts in correcting this problem.

2. Research Objectives

The focus of this thesis is the development of a technique by which image artifacts can be evaluated for their contributions to blooming. The phantoms developed here are for the study of the dependence of blooming on beam hardening, partial volume averaging, and cone-beam artifacts. Future projects will benefit considerably from our initial experience in fabricating these preliminary phantoms.

A study of the beam hardening artifact effects on blooming is presented. Future work will apply the analysis method developed to research to extend the decomposition of blooming into other possible artifacts.

2.1: Hypothesis

2.1.1: Beam Hardening Hypothesis

Beam hardening will not cause true geometric blooming as measured by the full width at the half mean (FWH-Mean) of its intensity distribution. Perceptual blooming will be observed in the evaluation of region size with respect to a fixed threshold, as is done in the clinic (calcium scoring).

2.1.2: Partial Volume Hypothesis

The proposed partial volume phantom can be constructed and evaluated successfully. Preliminary data will indicate the presence of blooming contributions specific to partial volume averaging.

2.1.3: Cone-beam Hypothesis

The proposed phantom for the analysis of the cone-beam effect can be constructed and evaluated for its ability to detect contributions to blooming specifically due to X-ray cone beam geometry.

3: Experimental Phantom Design

3.1: Phantom Design Considerations

This research aims, in part, to design a phantom to allow for the quantification of blooming dependent on varying amounts of beam hardening, partial volume and cone-beam effects. Such an investigation requires that plaque phantoms be engineered to sub-millimeter precision.

Anthropomorphic phantoms of coronary arteries in the heart and surrounding body are desirable to better approximate the clinical scenario. Issues involved in adopting an anthropomorphic design include, but are not limited to: precision mounting of plaques to curved tubular walls, controls to accommodate the fact that curving arteries confound beam hardening and cone-beam measurements with the partial volume effect along the axial dimension of the bore. Anthropomorphic phantoms of the heart are available; however they are expensive and adapting them to our requirements was a formidable task.

Although there is great interest in developing an anthropomorphic phantom, this preliminary study does not require one. This research focuses on developing simple plaque phantoms that will fit in commonly available quality assurance phantoms. Issues related to the design of a phantom for this study include: engineering and development of plaques of different materials (stratification of plaques by attenuation coefficient), choosing materials of sufficient geometrical stability (both thermally and structurally) to maintain dimensions to sub-millimeter accuracy [7] and the machining or molding of such materials into reproducible sizes and shapes with sub-millimeter precision.

3.2: Material Characterization

In principle, beam hardening effects are dependent on the attenuation properties of the material, so one may model the various levels of calcification of plaques by using different materials. Characterizing all materials used for supporting structures was also important. Such materials should either have tissue or dilute contrast equivalent attenuation coefficients. One should be mindful if the material holding the sample will produce beam hardening artifacts.

3.3: Beam Hardening Phantom Design

A water equivalent combined head body phantom (RMI) was augmented for the study of the affect of beam hardening on blooming. Cylindrical inserts were machined to 'slide-fit' into the rod holder. Sample holders were positioned centrally, in the longitudinal direction, to the phantom body with the use of solid inserts as fillers.

Sample cells for beam hardening experiments were constructed from Nylon® 6/6, because of it's superior machinability, dimensional stability thermally, and it's low radiopacity. Preliminary findings indicated that Nylon® 6/6 had a radiopacity of 48 +/- 18 HU (Picker with a technique of 120 kVp, Nylon rod resting on table). Each sample cell was 1.130" (2.870 cm) in diameter and 0.6 cm deep such that it 'slide-fits' into the rod holder (part of the RMI 461A kit). Two sets of beam hardening sample holders were designed: 1. Constant volume sample cells with a 5 mm diameter (11pc.) 2. Variable volume sample cell with diameters ranging from 0.5 to 5 mm at 0.5 mm increments (20pc.) (see Figure 3.1). Dilute oral contrast media, Hypaque™, was used to simulate calcified plaques. Solutions were retained in sample cells by Scotch® Clear Duct Tape #2120.

3.4: Partial Volume Phantom Design

A stepped cylindrical phantom was designed and constructed for the study of partial volume artifacts. The phantom was a polyoxymethylene (Delrin®) rod with a saw-tooth pattern machined into it (see Figure 3.2). The steps and valleys are 0.094" (2.39 mm) wide and 0.047" (1.19 mm) deep. The phantom design was intended to have step sizes equal to four times the minimum slice width of the scanner (0.6mm for smallest setting on Siemens 64-Somatom Sensation).

The phantom was designed to be placed at an arbitrary z location, such that different amounts of partial volume artifact will be generated as axial slices fall along the phantom. The stepped phantom was fixed to a water equivalent supporting block and suspended in a sealed cylindrical water phantom.

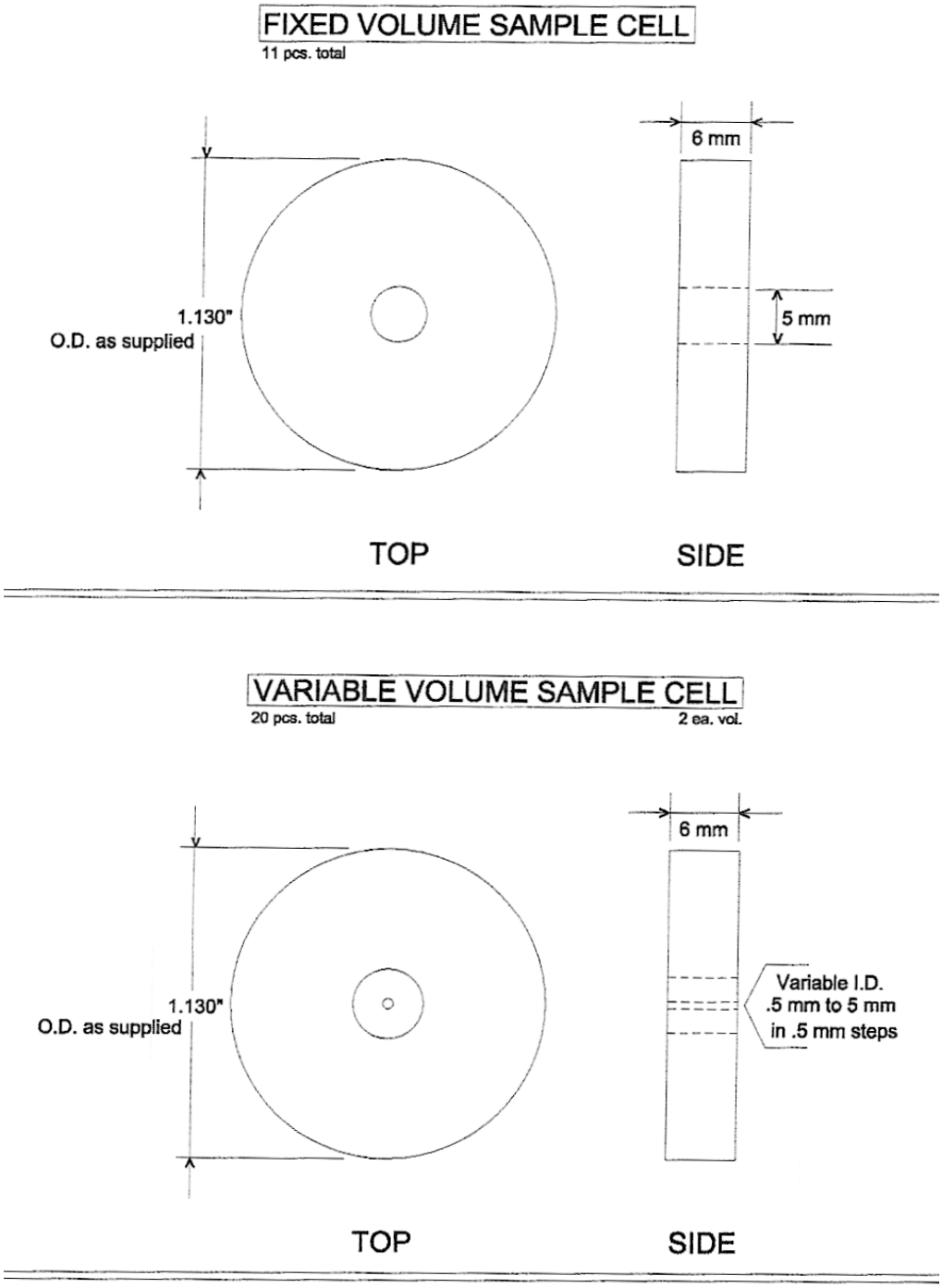


Figure 3.1: Top: Fixed diameter sample cells. Bottom: Multiple sized diameter sample cells 0.5 to 5mm diameter (0.5mm increments).

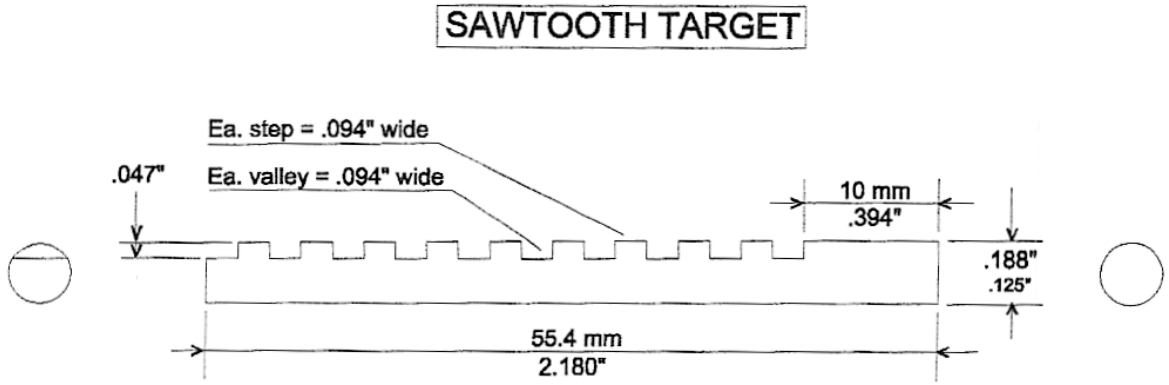


Figure 3.2: Saw-tooth phantom constructed for partial volume experiments.

3.5: Cone-beam Phantom Design

To test the cone-beam effect one needs a phantom that is at least as long as the axial X-ray beam width. A cone-beam phantom was designed and constructed to study the affect of the X-ray cone-beam geometry on blooming. 5mm test samples of Nylon® and Delrin® rods were used. The rods were fixed to a water equivalent block and suspended in a sealed cylindrical water phantom. The water phantom was fixed to a Lexan® platform that was held fixed in the scanning plane. A bridge was constructed to hold the Lexan® platform stationary despite the patient table movement beneath. A sealed cylindrical water phantom was suspended utalizing a custom suspension frame such that the sample does not move relative to the scanner despite the automatic table feed.

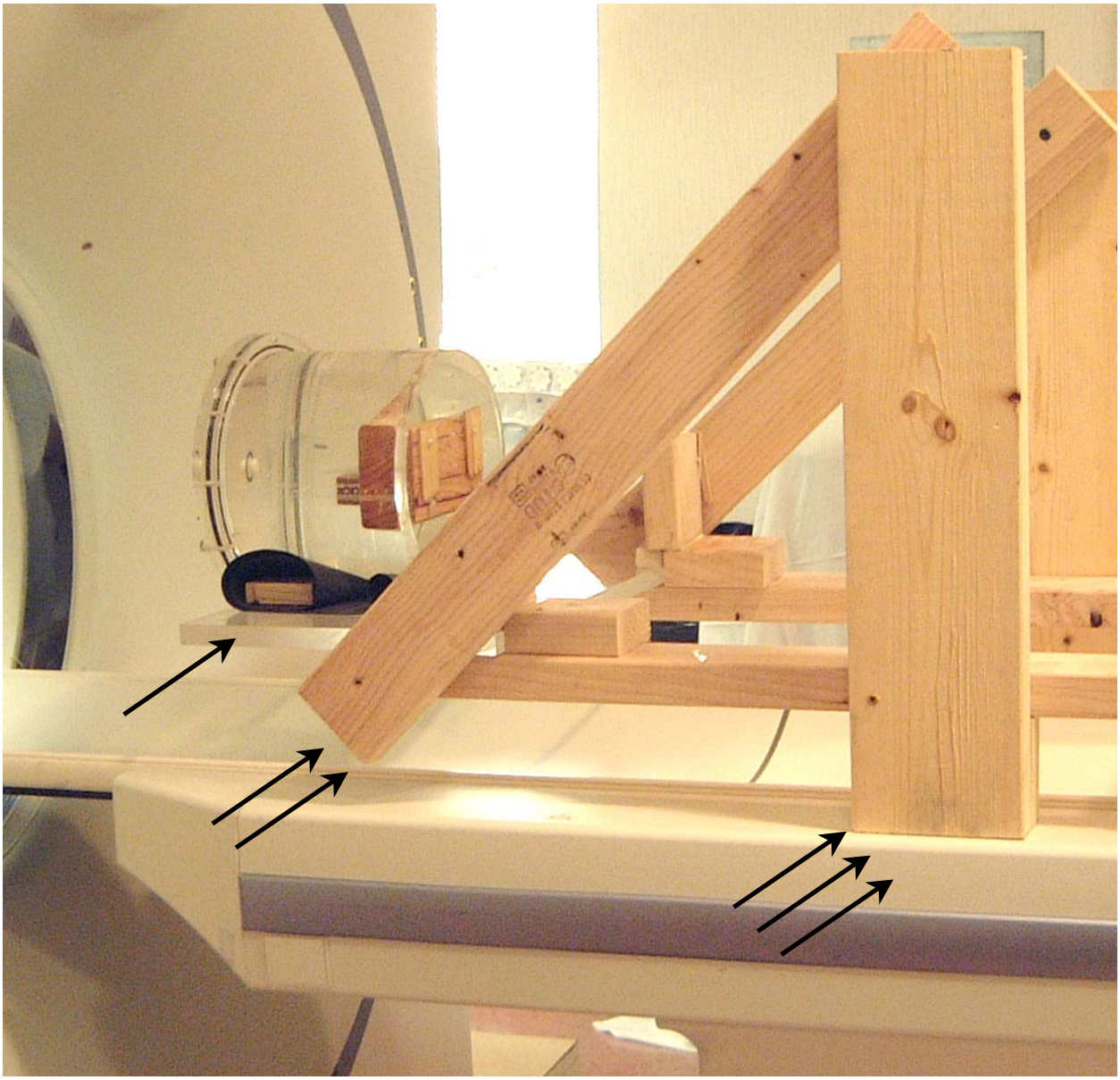


Figure 3.3: A single arrow indicates the Lexan® platform used to suspend the water phantom in to the scanning plane. The double arrow indicates the primary brace for the platform. The triple arrow indicates the placement of one of the four supporting legs. Counterweights (not shown) are at the back.

3.6: Contrast Solution for Plaque Simulation

Hypaque™ is a radio-opaque agent commonly used for contrast in CT studies. At the start of this experiment, the concentration of Hypaque™ required to simulate the attenuation of calcified plaques was unclear, so three batches were created. The first experiment was constructed to span a wide range of possible concentrations. The second and third experiments were used to collect additional data points within the clinically relevant sample intensities (Hounsfield Units [HU]). Solutions were prepared by mixing known quantities of Hypaque™ with deionized water.

Vial #	Serial #	Contrast (g)	Volume (ml)	Conc. (mg/ml)	% of Max.	ROI Mean (HU)
12	120	4.5025	15	300.2	*100.0%*	*3,029*
11	110	4.002	15	266.8	*88.9%*	*2,984*
10	100	3.502	15	233.5	*77.8%*	*2,810*
9	90	3.002	15	200.1	*66.7%*	*2,441*
8	80	2.754	15	183.6	*61.2%*	*2,284*
7	70	2.502	15	166.8	*55.6%*	*1,955*
6	60	2.253	15	150.2	*50.0%*	*1,791*
5	50	2.003	15	133.5	*44.5%*	*1,552*
4	40	1.504	15	100.3	*33.4%*	*1,232*
3	30	1.004	15	66.9	*22.3%*	*833*
2	20	0.502	15	33.5	*11.2%*	*NA*
1	10	0	15	0	*0.00%*	*23*

Table 3.1: Batch #1 was used to define the range of concentrations for the entire study; the maximum concentration was arbitrarily chosen.

Vial #	Serial #	% of Base	Conc. (mg/ml)	% of Max.	ROI Mean (HU)
10	1110	100%	33.3	11.09%	421
9	1100	90%	30	9.99%	403
8	1090	80%	26.7	8.89%	323
7	1080	70%	23.3	7.76%	307
6	1070	60%	20	6.66%	223
5	1060	50%	16.7	5.56%	190
4	1050	40%	13.3	4.43%	189
3	1040	30%	10	3.33%	106
2	1030	20%	6.7	2.23%	56
1	1020	10%	3.3	1.10%	45
11	1120	5%	1.7	0.57%	NA
0	1010	0%	0	0.00%	NA
Batch:	*3.3333 Hypaque™(g) / 100mL H2O*				

Table 3.2: Batch #2 was formulated for the purposes of generating samples in the low density range established in the first batch.

Vial #	S/N Samp.	% of Batch	mg/mL	% Max	Mean HU
9	1100	100%	160	53%	1,912
8	1090	90%	144	48%	1,718
7	1080	80%	128	43%	1,545
6	1070	70%	112	37%	1,303
5	1060	60%	96	32%	1,144
4	1050	50%	80	27%	961
3	1040	40%	64	21%	783
2	1030	30%	48	16%	557
1	1020	15%	24	8%	261
3'	**1110**	**30%**	**10**	**3%**	**119**
0	1010	0%	0	0%	3
Batch:	*16.0029 Hypaque™(g) / 100mL H2O*				

Table 3.3: Batch #3 was created for the purpose of generating samples in the intermediate range of densities established by Batch #1. The row starting with vial number 3' uses a concentration from Batch #2.

4: Development of Analysis Method

4.1: Overview

This pilot study proposes the analysis of a beam hardening phantom and the development of modular software algorithms for the future analysis of a variety of other CT phantoms. In particular an analytical technique to study blooming is desired to produce highly accurate measurements that are more reproducible and precise than classical observer-dependent measurements. Refinement and adaptation of the analytical technique to other phantoms would be possible. Therefore, the need to repetitively analyze CT images, to adapt to multiple phantom morphologies, and to be scaleable with the varying number of images acquired for each study makes an automated software analysis highly desirable.

Assessment of luminal stenosis has classically been performed with labor intensive measurements made on a case by case basis by a clinician. Many factors affect the image quality of coronary arteries, such as the image reconstruction process, the limiting resolution of the CT system, and various artifact processes which blur the luminal and plaque edges in the image. The blurred extension of an object's edge is called a penumbra. Determination of the true extent of the image object within the penumbra remains a challenge that contributes to inter-observer variability of clinical 'by-hand' measurements.

As analytical methods are refined and adapted to new phantom designs the analytical technique needs to be efficient enough to perform recalculation of all data in a timely manner. The process of debugging software often requires repeated measurement of data. A software language that provides highly refined libraries for image processing and analysis of other scientific data is an important criteria. For this reason, a scripting language designed for prototyping embedded software for mathematical sciences was utilized. In addition, MATLAB is a scripted language which removes much of the tedium required in programming manual memory management, was utilized for the development of custom scripts and functions for this project--MATLAB version 7.4.0.287 (2007a Student Version). In addition, two add-on packages were utilized (The Image Processing Toolbox and The BioInformatics Toolbox).

Since the analysis protocol proposed is to be used for multiple phantom designs, a modular software development approach was utilized (see Appendix). Within the MATLAB script editor, the code was divided into cells, each of which represented a specific task. Functions were then developed for each section of the analysis: image segmentation, FWHM calculation, and mask generation. In addition functions were developed to aid in the creation and visualization of QA data.

The automated analysis method consists of the following sections: image input and verification (preprocessing), image segmentation, region of interest statistics, background subtraction, profile collection and measurement. The automated production of montages after each processing step provides a qualitative method for continuing quality assurance of processed data.

4.2: Preprocessing

Once the CT images are collected for each experiment they must be preprocessed. In the current version, the operator must select and then sort the relevant images manually. Then the operator must generate a reference file for each experiment, hereafter called a key-file. The key-file contains information about the date of the study and the known magnitude of the independent variable under investigation. Finally, the software script uses the key-file to read in all CT images that correspond to a selected study.

The manual sorting of images is needed in this preliminary study to identify those image files which correspond to both the sample and reference images. The central three most slices for the sample and the sample holder are chosen to avoid partial volume artifacts at the ends of each insert. A sample spacer follows each sample. Each spacer is identical in construction to the sample holder, except that it has no sample well drilled through the center. This way each sample image has a corresponding reference image.

Sample and reference images are then sorted into directories by study type, date and sample serial number. The key files are then generated that indicate the file's date of the study, location within the data directory, and the known value of the independent variable for each sample. The order in which entries appear in the key-file determines the order in which input images are to be processed and displayed in intermediate montages for quality assurance.

Rather than storing all the images into memory, which would greatly limit the number of images that could be processed by the computer, a list of paths to each image is created. Thus, the first step in the automated analysis process is to create a list of the full path to all of the images. Checking that each entry corresponds with, or points to, an image file is how the software validates the key-file. Any entries within the key file that do not correspond to existing files are automatically ignored. A temporary key-file is then created which only contains information for images that actually exist in the specified subdirectory.

4.3: Segmentation

Image segmentation is the process of identifying background and foreground objects in an image. Morphological operations are commonly used to differentiate between the regions of an image. The size and shape of a segmented region can depend on the type of processing. In this study image segmentation is used to determine the geometrical center of the sample and sample holder regions (see Appendix). The measurement of sample size is performed separately using the original image data.

A common approach to image segmentation begins with the application of an intensity threshold which converts the image into black and white pixel values. The white pixels correspond to those pixels in the original image that have intensity values greater than or equal to the threshold intensity. In this study automated selection of the optimal threshold intensity proved difficult, because of the overlap in noise between the sample holder and the solid water phantom. The solid water portion of the phantom had

a mean intensity of approximately 0 +/- 60HU while the sample holder had a mean intensity of about 110 +/- 60 HU. The noise value can vary slightly with machine parameters.

The current version of the analysis software relies on the operator to manually determine the threshold. An adequate threshold may be found through an iterative process based on trial and error. The choice of the threshold value was determined such that the perturbations within the sample holder tend to combine together, forming a single large connected region.

A variety of techniques have been developed to remove noise from an image. A common noise removal technique is the application of an erosion operator. Erosion is performed through the application of a structuring element on every pixel in the image. The structuring element defines a neighborhood around each pixel that will be used during the calculation. The resultant intensity of a pixel is given by the minimum pixel value within its neighborhood as defined by the structuring element. An erosion process will tend to make objects in an image smaller. If an object, like a noise blob, is smaller than the structuring element it will be removed from the resulting image. In this study the erosion operation was applied twice using the disk shaped structuring element with a radius of two pixels (see Appendix).

A variety of morphological operations, other than erosion, may also be useful in removing noise. In particular, small regions corresponding to small sample sizes, would be entirely removed by an erosion technique rather than being segmented. In this study the samples and sample holders were much larger than the surrounding noise blobs, which permitted the repeated use of the erosion operator. Morphological openings are an example of a noise removal technique that attempts to preserve region size. A morphological opening is the application of erosion followed by the reverse process, dilation.

4.4: Region of Interest Statistics

After image segmentation the geometric center, centroid, of the sample holder and sample images was determined. The centroid is the point about which the region of interest is formed. Using the sample diameter (mm), specified in the key-file, a circular binary mask was constructed. The binary mask contains a circular region that has a diameter 80% of the known sample diameter. The binary mask is then used to read all of the pixel intensity values within the specified circular region. These pixel values are then averaged resulting in a mean sample intensity value. The mean intensity of the reference images corresponding to each sample are also evaluated using the same binary mask as the sample image.

4.5: Background Subtraction

The image background refers to all regions that are not the sample region, i.e. air, solid water, sample holder. The background ideally consists of randomly distributed fluctuations in pixel intensity values distributed evenly about a reference intensity level (bias). In this study the background noise is distributed around several levels. In particular, the sample holder provides a background bias to the pixels surrounding the sample (Figure 4.1) by increasing the region noise distribution background level with respect to zero (water value). Background subtraction methods are commonly used in many fields of science to correct for bias in the signal due to the background level. Since classic background subtraction would affect the distribution of the pixels within the sample region of the test image, the sample image is normalized to the noise level of the sample holder using the mean value obtained from the reference image.

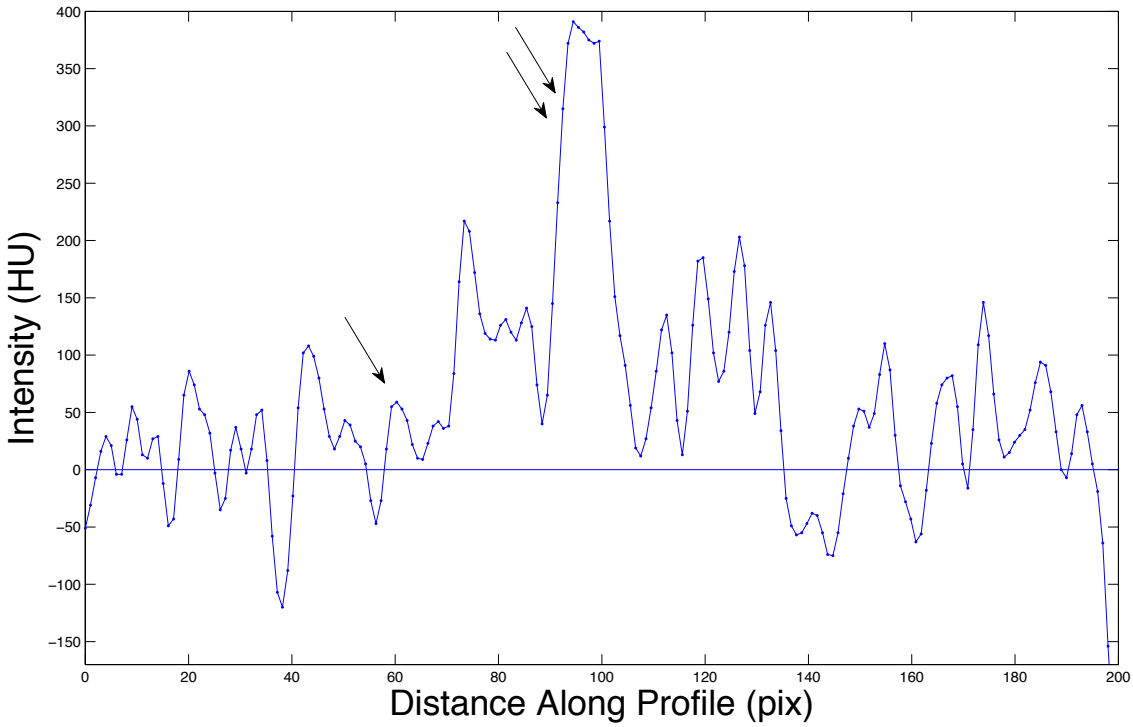


Figure 4.1: Graph of sample profile a single indicates the start of the sample holder and the double arrow indicates the sample peak.

4.6: Profile Collection and Measurement

4.6.1: Considerations for Sample Measurement

A profile of an image region can be thought of as a graph of pixel intensity values along a path, typically a straight line. In practice a profile is often not aligned perfectly with the pixels of the image, so the profile becomes a collection of interpolated values along the specified path. The shape and size of the sample image may be determined by analyzing a series of profiles distributed axially about the centroid of the sample image. The result is a series of diameter measurements taken at many different angles about the region.

Profiles were collected radially at one degree increments to meet the Nyquist sampling criteria--the minimum angle required to read every pixel at the distance of the edge of the profile (center + 20 pix). Since MATLAB treats images and matrices the same, images of the profile set for each sample can be made.

The benefit of analyzing axial profiles over bulk ROI characteristics (effective diameter, area, etc.) is that one may evaluate the angular dependence of any size distortions (image artifacts) that may appear. In this study profiles are collected axially about the sample object centroid. The number of pixels in each profile is constrained to a constant value during the interpolation process.

The edges of objects in CT images are known to be blurred as a result of the reconstruction process and various image artifacts. Pixels at the bottom of an edge are said to be at the foot of the penumbra. The penumbra is the intensity distribution extending from the edge of the sample outward, the blurring of the edge (see Appendix: Linear Blur Model). An adapted form of the full width at half maximum technique, commonly used in nuclear medicine to control for image blur, was adopted.

A contour plot can be used to illustrate the shape of a given sample (Figure 4.2). In Figure 4.2 the sample shape was contoured in 100 HU increments. The sample's FWH-Mean(θ) was analyzed, using linear interpolation, at an unrounded intensity level of 212.9243[1] HU (unrounded ROI mean = 425.8485 HU). Figure 4.2 does not have a contour that exactly matches the level used by the software to calculate the FWH-Mean. Therefore, the closest contour was utilized. All contours below 200 HU (the closest contour level to that used by the software) have been removed to better illustrate the sample shape and surrounding image noise. The sample contours were observed to deviate from that of a circle quite dramatically (Figure 4.2). Additionally, the sample region does not have a uniform intensity map. In this example the sample has a single off-center peak in the lower right quadrant of the image.

[1] The FWH-Mean threshold value is reported here without rounding to better reflect the internal operations of the code, which is designed in an effort to avoid propagation of rounding errors.

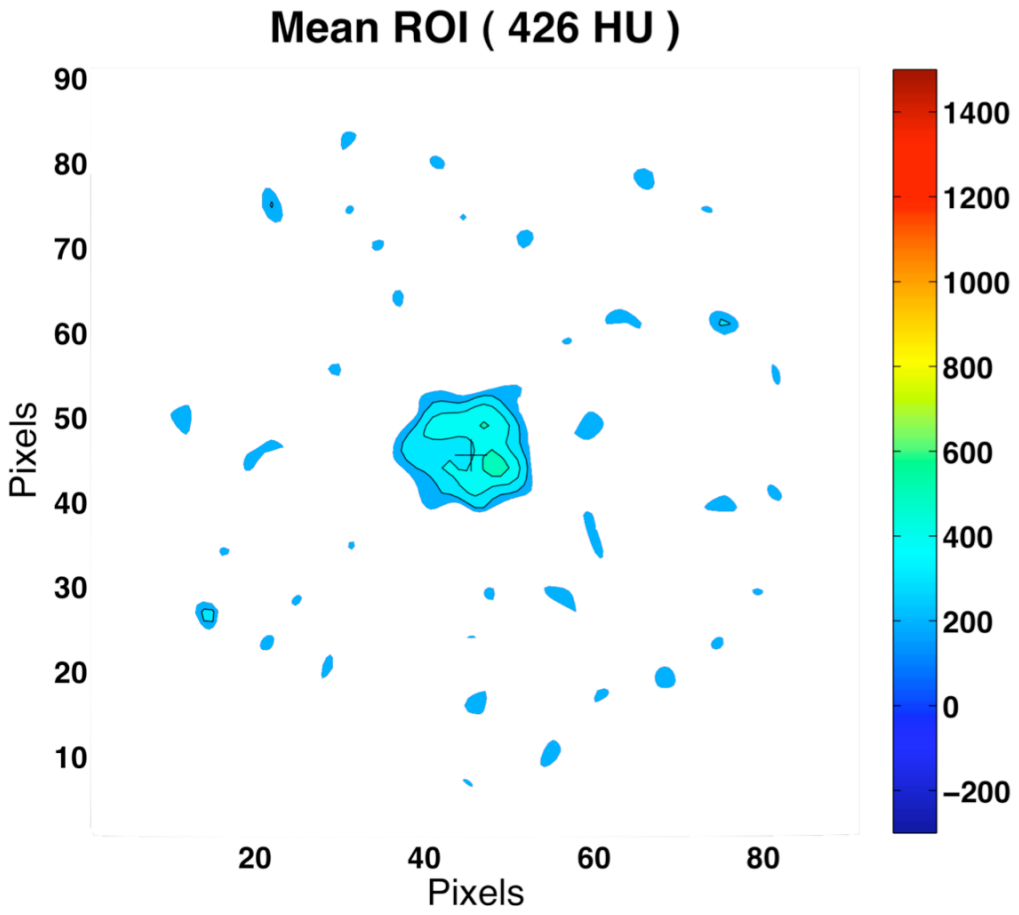


Figure 4.2: Contour plot of a 5 mm diameter sample with an ROI mean value of 426 HU. The cross denotes the detected centroid of the sample. All contours below the profiling level have been removed to better illustrate the sample shape analyzed by collecting axial profiles.

4.6.2: FWH-Mean Measurement

Typically the width of a peak or plateau is measured using an automated peak detection function that reports the FWH-Max. The non-uniform distribution of signal intensity within the sample region makes the measurement of width difficult. Additionally, the noise distribution outside of the sample region can create noise peaks which need to be filtered out. Since common peak detection functions cannot be used to measure the FWH-Mean in this study, a custom analysis routine was developed (see Appendix). Profiles were collected after background correction; therefore the intensity of the half mean threshold was also corrected using the same offset value. The rest of the analysis technique was performed sequentially on each profile from every image.

Next, the *mspeaks* function (in the MATLAB Bioinformatics Toolbox) was used to identify all of the peaks in the selected profile. The interpolated coordinates of each apex as a floating-point number were returned. The peak coordinates were then rounded to match with pixel coordinates in the profile. The peaks were then filtered by keeping only those peaks that fell within 80% of the expected sample diameter, centered along the length of the profile. The selected peaks were used as placeholders from which to begin looking for the sample edge.

The half-mean threshold was applied to the profile. All pixels with an intensity value equal to or above the threshold were set to a value of 1; all other pixels were set to 0. All connected pixels, within the profile, having a value of 1 are referred to here as being grouped. Since there may be more than one group that represents the sample region (i.e. local minimum could dip below the measurement threshold thereby

fragmenting the profile along the sample region) groups of pixels do not yet represent the FWH-Mean of the sample region, because is unlikely that these pixels fell exactly at the desired measurement intensity. The FWH-Mean is measured with the use of linear interpolation between the nearest pixel below and above the threshold, at the threshold value. Therefore pixels just above and below or exactly at the threshold needed to be identified.

Pixels on either side of the point at which the profile intensity values cross the measurement threshold were identified using a subtraction-based technique. Pixels just to the left or right of a threshold-crossing point can be identified by calculating the difference of adjacent values ($x_i' = x_{i+1} - x_i$) from the threshold profile. Once the difference of adjacent values is found, selection criteria are used to identify which side of the crossing pixels are on. Pixels at the right edge of a group of pixels are located just to the left of a crossing point; and they are indicated by a positive one value. These adjacent difference values are, however, shifted one pixel to the left. Pixels at the left edge of a group of pixels are just to the right of a threshold-crossing point; they have a value of negative one and are not shifted relative to the original profile. In this way the pixels along the edges of regions are identified. It is then necessary to identify which threshold-crossing points are to be considered part of the sample region.

The determination of the pixel values just below threshold-crossings and which threshold-crossing points correspond to the sample region was done simultaneously. The original profile intensity values were searched starting from the outer most peaks (left and right side) which survived the ROI filter. The first pixel below the threshold was identified on either side of the sample holder.

Linear interpolation was performed between the upper and lower pixel values at the desired threshold value. The equation for the fitted line was then solved for the independent variable (spatial coordinate) with the dependent variable (intensity). In this way the coordinates of the threshold-crossing points were found with interpolation. The difference in the two interpolated values is the FWH-Mean.

The measured sample diameter, FWH-Mean, corresponding to each profile angle was then averaged. This average diameter value is the measure of the bulk size of the sample region. The standard deviation of the diameter values is the measure of the consistency of the sample region in plane, that is to say the variation in its shape with angle. The data was stored in tabular form and a series of filters were generated that can be used to generate separate tables for a selected study date. The date filters can be used to graph the results from each day or each batch of samples in a plot with different colors.

5: Quality Assurance

5.1: Elliptical Test Images

Validation of the developed algorithms' ability to detect blooming and indicate angular dependence of size distortions was accomplished using a set of artificially generated images of ellipses. The analysis routine developed was demonstrated to be capable of detecting deviations in the average in plane diameter of a sample to an accuracy of 1+/- 1 pixel.

Artificial sample images were generated from an acquired reference image to provide a realistic noise contribution. A series of binary masks (black and white images) were designed to input the specified sample geometry (ellipses) into the reference image. The mask was scaled using an arbitrary value of 300HU to provide a detectable signal. Artificial sample images were then constructed by summation of the individual scaled binary images with copies of the reference background image. The result was series of images containing both the original reference image and an artificial sample with intensity at 300HU. An artificial penumbra was not imposed on the image, However, background noise pixels do result in intensity fluctuations along the sample boundary. The artificial images were then saved in standard DICOM format using the same header information as the reference image they were based from. Finally, the images were sorted into a data folder using the standard method and a key-file was manually created.

Using this method, a series of sample images were generated from a set of binary masks each of which had a slightly different ellipse. The ellipses were centered on the reference image's sample holder, using the image segmentation routine described in the analysis section. The first binary mask consisted of a 5 mm diameter circle. Subsequent binary masks were generated by incrementally expanding the horizontal axis of the ellipses. Forty images of these artificial samples were created to be used for quality assurance of the analysis protocol. Twenty images were used to determine the algorithms' ability to assess subtle changes in diameter (1%), starting from a perfect circle. Twenty images with larger increments (10%) were used to verify the ability to detect more drastic changes in diameter, in case the subtle changes were not detected.

5.2: Image QA Analysis

The developed program to detect blooming then processed both artificial data sets. To verify the performance at each step of the analysis, the results were summarized using image montages. Montages were generated of the raw DICOM images, region of interest filters and for collected profiles. Additional montages were created using the FWH-Mean threshold applied to raw images and profiles. A subtraction image was generated to indicate the difference in the blooming at a standard constant threshold relative to the FWH-Mean threshold.

5.3: QA Results

The sample holders in Figure 5.1 are well centered, indicating good precision in the detection of the sample holder centroids for each image as a result of the image segmentation process. The same can be said for the sample regions shown in Figure 5.2.

Noise fluctuations about the sample border affected the smoothness of the shape as shown in Figure 5.3. The amount of noise that was present along the edge of the sample depended on the measurement threshold (for both fixed and FWH-Mean) and the amount of noise in the reference image.

The subtraction image presented may be used as a reference for the case in which no significant signal difference between the two threshold techniques is found, irrespective of the contributions to boundary noise. This is because the artificial samples do not have a reconstruction blurring function applied to their border. Very small fluctuations in noise along the sample border with respect to the difference in level settings are present in Figure 5.3. The figure indicates in red structures present at a constant level of 200HU and not present in the half mean level image. Holes in the noise regions can be seen indicating that at higher intensity levels the noise comes to a peak.

A montage of the profile set for each artificial sample is displayed in Figure 5.5. The first column of pixels in the image is the profile collected horizontally across the sample region. The profiles continue from left to right such that the middle of the image represents the vertical profile, and the right most profile is taken at 180 degrees. A

perfectly circular sample region will appear as a light colored horizontal stripe in the profile image. The intensity of the stripe will be that of the sample region. Additionally, blobs of noise will appear curved in the profile image and pixels near the sampling center will appear to form a solid region due to oversampling.

The montage displayed in Figure 5.5 shows the power of the profile overview to show the angular dependence of the data set. As the samples increase in eccentricity from the upper left corner to the lower right corner the profiles approaching the horizontal angle show a wider streak. The curved ends of each profile set is indicative of blooming due to an elliptical size distortion in the horizontal direction with a maximum at 0 and 180 degrees.

In addition, the FWH-Mean threshold can be applied to the profile montage as a method to visually enhancing the presentation of the sample edge (Figure 5.6).

As indicated in Figure 5.7, the analysis method is capable of detecting subtle changes in diameter. The bulk size of the sample holder is represented by the average in plane diameter. The variability or consistency of the sample region shape with angle is represented by the standard deviation of the measured diameter. The standard deviation is indicated in Figure 5.7. Since the artificial samples used are elliptical the standard deviation of the diameters for each sample increases with increasing eccentricity. Displacement of the mean in plane diameter for 1% blooming indicates a high sensitivity to the detection of diameter fluctuations.

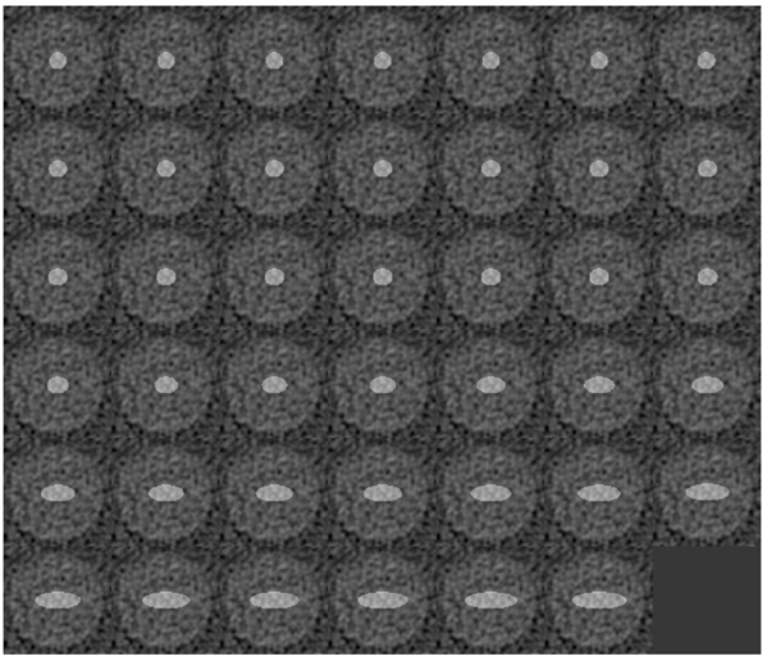


Figure 5.1: Montage of elliptical QC images cropped to sample holder. All samples have a constant semi-minor axis (5 mm) and a steadily increasing semi-major axis. The first 20 samples increase in eccentricity by a factor of 0.01 while the following samples increase by 0.1.

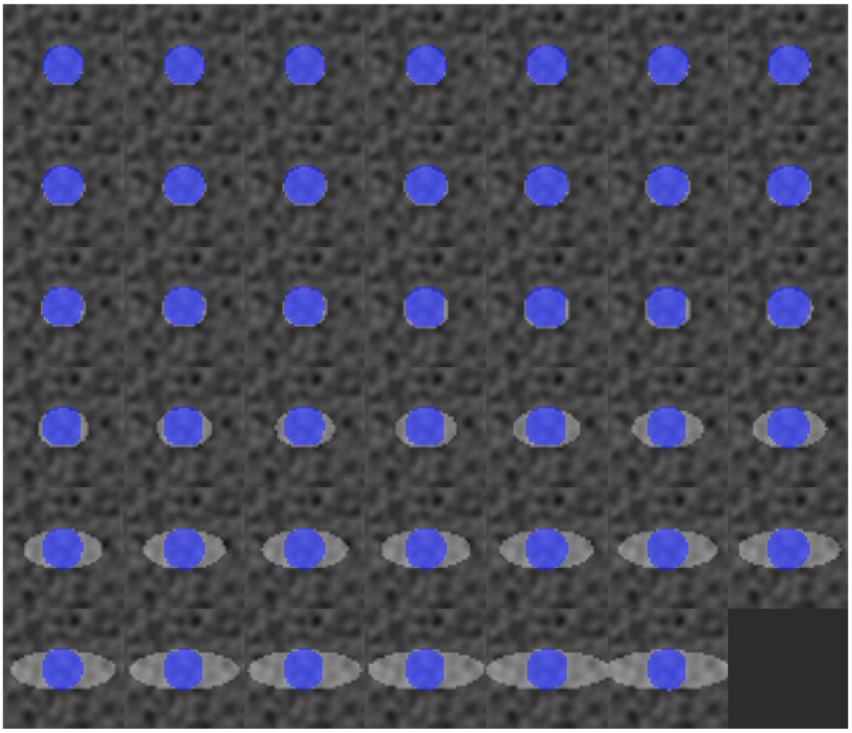


Figure 5.2: Montage of cropped images within the sample holder region. The blue filled circles indicate the ROI data collection region. The blue circles are a constant diameter of 80% of the reported physical diameter (5mm).

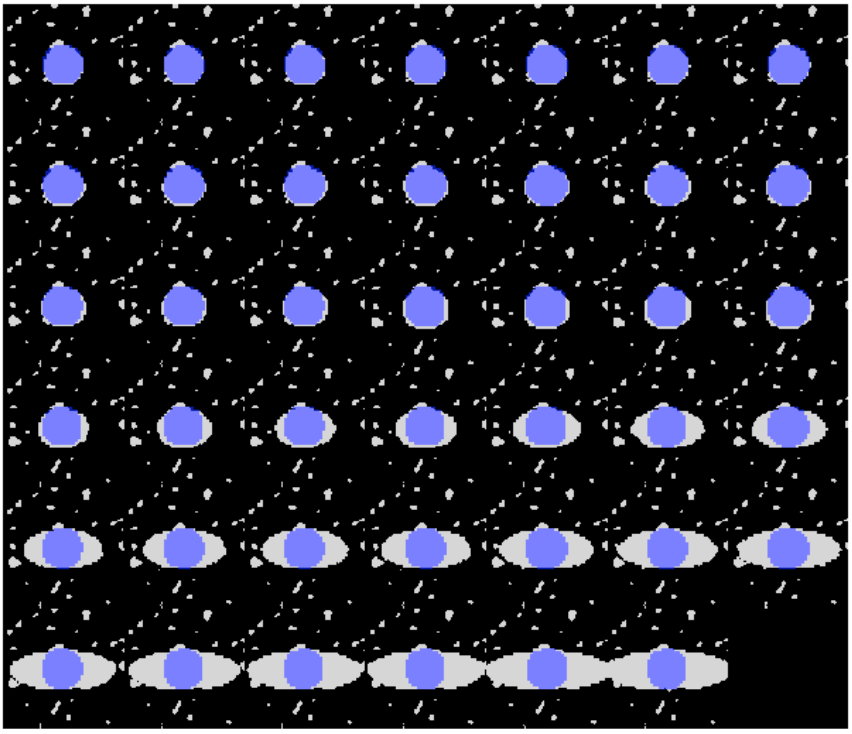


Figure 5.3: Montage of samples thresholded at the full width at half mean intensity. The blue regions indicate the sampled region used to calculate the mean sample intensity at 80% of the expected physical size (5mm).

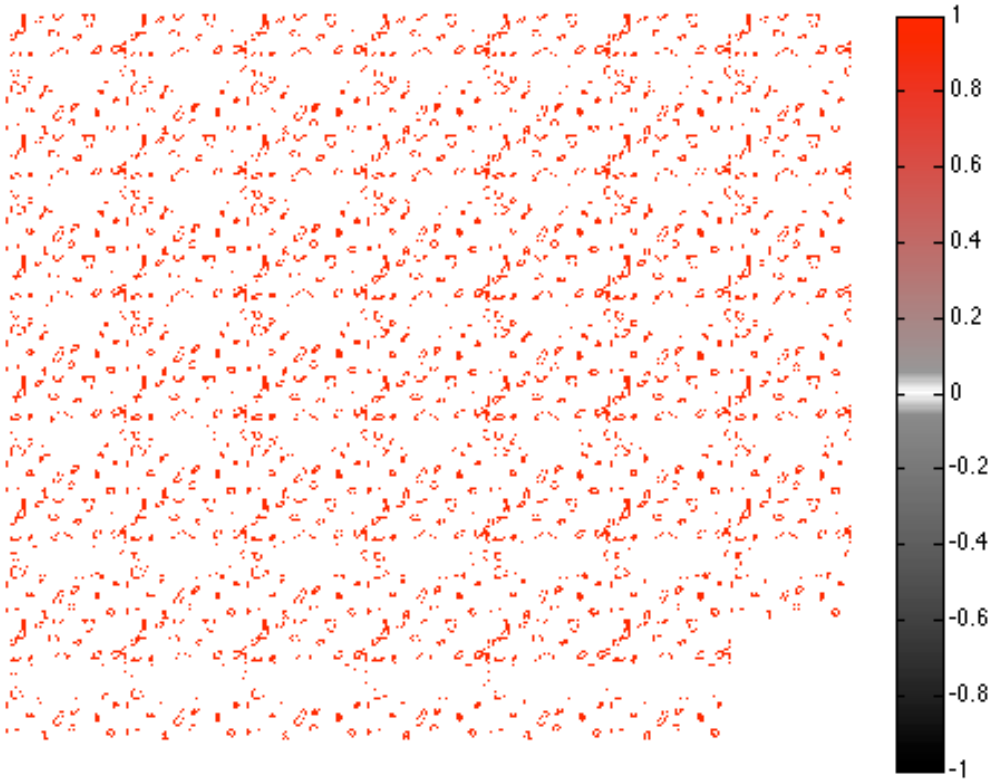


Figure 5.4: Montage of subtraction images. The subtraction images are formed by subtracting the images thresholded at half of the mean intensity value from those thresholded at a constant 200HU level. The white area indicates agreement between the two images. The red area indicates objects present in the static level image not in the half mean images.

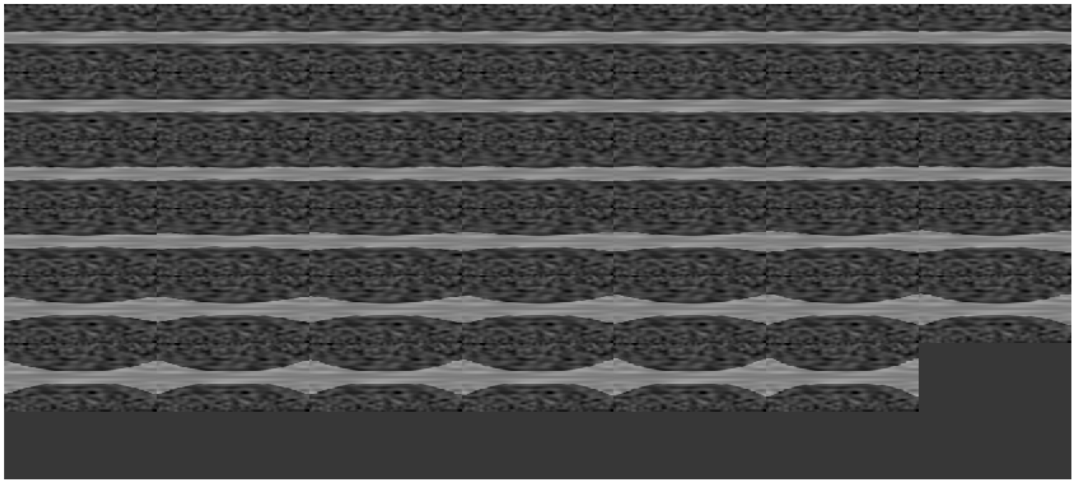


Figure 5.5: Grayscale montage of the axial profiles. The profiles are stacked horizontally such that the x-axis represents the angle at which the profile was acquired (from left to right). The y-axis is the number of pixels collected in the profile.

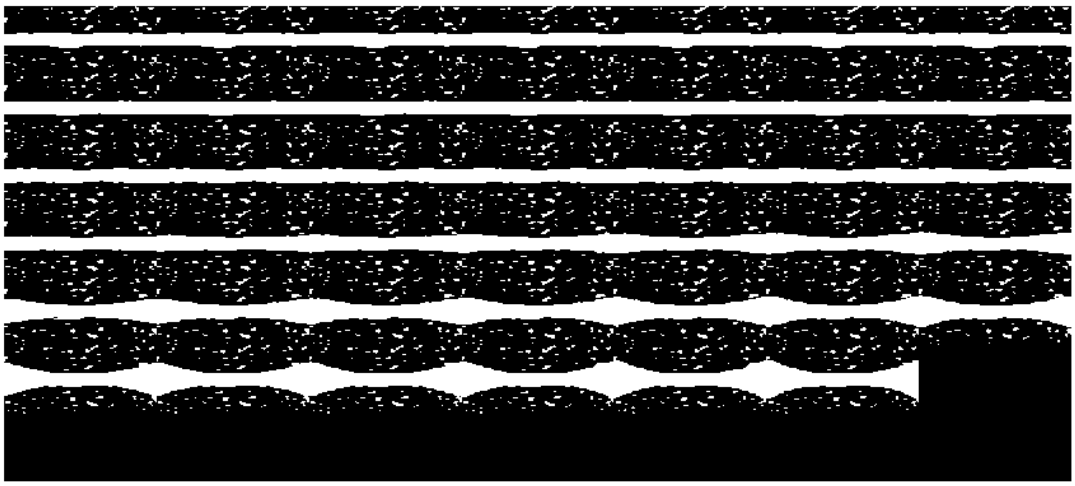


Figure 5.6: Montage of samples thresholded at the half mean value.

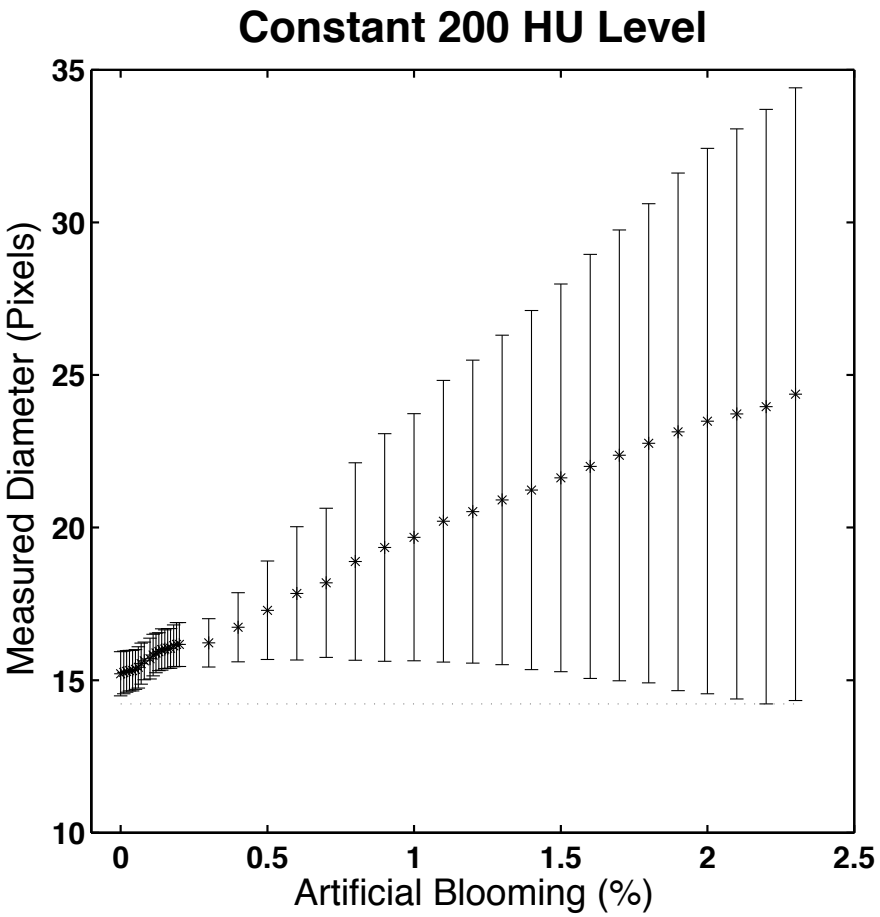


Figure 5.7: Graph of FWH-Mean vs. blooming from artificially increasing the eccentricity.

6: Methods

6.1: Data Collection

6.1.1: Beam Hardening: Varying Concentration

Three beam hardening experiments were conducted using different sets of diluted contrast material. The combined head and body water equivalent phantom was placed at the end of the patient table and positioned within the scan plane. The phantom was positioned such that it was not centered axially with the scanning geometry. The combined head and body phantom was assembled using the supplied sealing compound in an effort to remove air gaps where inserts interface with the main body. The phantom was aligned axially in the selection location with the scanner by utilizing the laser positioning system. A rubber-impregnated adhesive tape was used as a friction pad between the phantom and table surface.

The first experiment was conducted using a modified general cardiac scanning protocol. The scan was conducted axially. The scan technique used was: sequential, 120kVp, 375 effective mAs, 64x0.6 collimation, 0.60mm slice width, and a 0.7mm focal spot size. Reconstruction was performed using a 180 mm field of view (FOV) with the U30u filter.

The subsequent experiments were conducted using a different sequential protocol: 120kVp, 138 effective mAs, 2x1 mm collimation, 1.00 mm slice width, and a 1.2 mm focal spot, selected to limit the data to the central beam and detectors in order to avoid cone beam effects (clinical cardiac calcium scoring protocol, but without cardiac gating). The images were reconstructed using a 180mm FOV and the B30f convolution kernel.

6.1.2: Partial Volume Phantom

The partial volume phantom was fixed to a water equivalent scanning platform inside of a sealed cylindrical water phantom. The phantom was positioned at the end of the patient table and made level using a flexible high friction underlayment. The phantom was aligned relative to the X-ray beam geometry using the CT laser alignment system. Scans were conducted sequentially using the following parameters: 120 kVp, 128 effective mAs, 30x0.6 mm collimation, a 3mm slice width, and a 1.2 mm focal spot size. The images were reconstructed using a 180 mm FOV and the B30f convolution kernel.

6.1.3: Cone-Beam Phantom

The cone beam phantom apparatus was positioned on the patient table support, such that the cylindrical water phantom would remain stationary throughout the scan (see Figure 3.3). During the CT scan the patient table advanced unimpeded below the water phantom. The apparatus was positioned such that the cylindrical water phantom was then suspended in the scanning plane. The cylindrical water phantom was made level using a high friction underlayment between it and the plastic platform. Scanning and reconstruction was accomplished using the same protocols used for the partial volume phantom.

6.2: Data Analysis

CT images from the beam hardening experiment were analyzed by the protocol described in the Development of Analysis Method. The proper processing of the images was assured through qualitative analysis of the intermediate montages. Montages used for quality assurance of the experimental data set were generated at each step of the analysis protocol: preprocessing, image segmentation, region of interest statistics, background subtraction, and profile collection and measurement.

7: Results

7.1: Beam Hardening: Varying Concentration

7.1.1: Quality Assurance

Verification of image input and sample holder segmentation was accomplished simultaneously. Each image was cropped into a square shape centered on the sample centroid. Low variation of sample holder placement with respect to the center of the cropping window indicates that image input and sample holder segmentation was effective (Figure 7.1).

Additionally, the key-file input was verified with respect to the order in which the images are displayed; note the increasing brightness of the sample in Figure 7.1. Images were incremented from top to bottom and left to right. The first sample contains only deionized water at the top left. The last sample is that of the greatest concentration, at the bottom right. As the sample concentration increases along the first row, the sample region approaches the intensity of the surrounding sample holder. The sample area disappears from the image at this point. Eventually, sample density was increased significantly above that of the sample holder and became visible.

Furthermore, the segmentation of the sample region was verified (Figure 7.2). Images were cropped using the centroid of the detected sample region. Should a sample region fail to be segmented, the *de facto* centroid used is that of the sample holder, a log of samples that met this criterion was created.

Montages can also be presented with an applied threshold. The montage in Figure 7.3 uses a constant 200HU threshold for all images. The choice of a constant level is used clinically for the determination of bulk calcium burden. The region of interest can be displayed over the cropped image montage to verify correct alignment with the sample region (see Figure 7.2).

A montage of the profiles of images is displayed (Figure 7.4). The angular variability of measured diameter may be spot checked by visually looking for inconsistencies in the shape of the sample region streak.

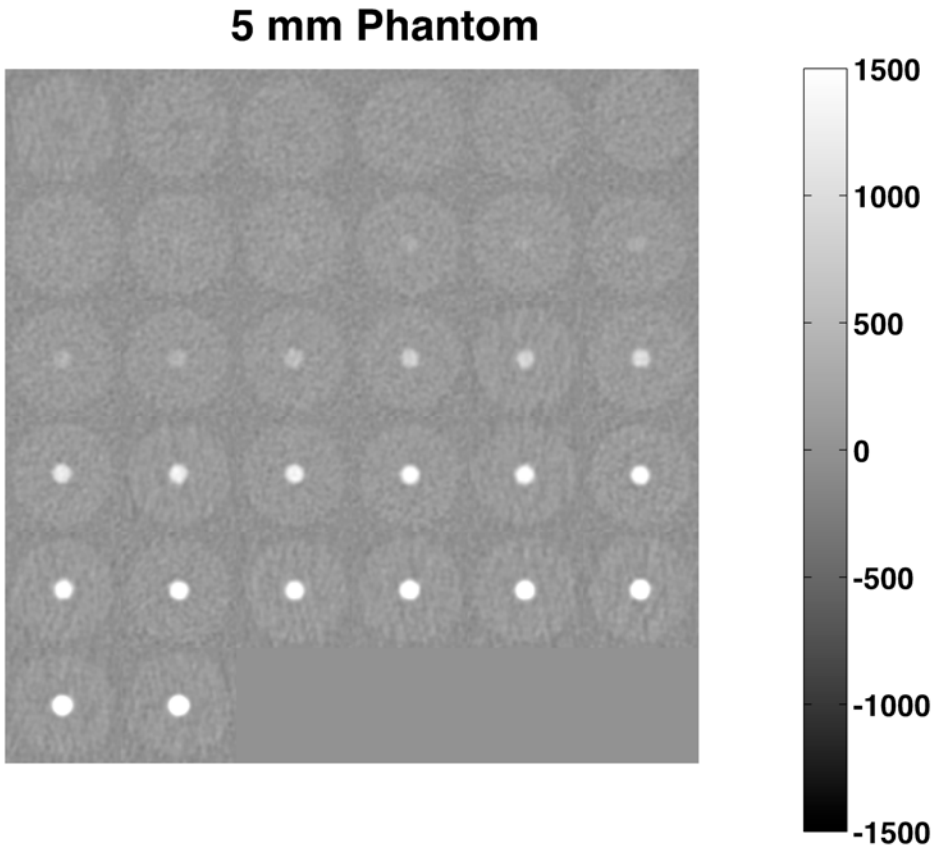


Figure 7.1: Montage of all beam hardening density phantoms cropped to sample holder. Samples are numbered from left to right and ascend with the rows. Differences in noise characteristics correlated with the date of each study, indicating some scanning variability.

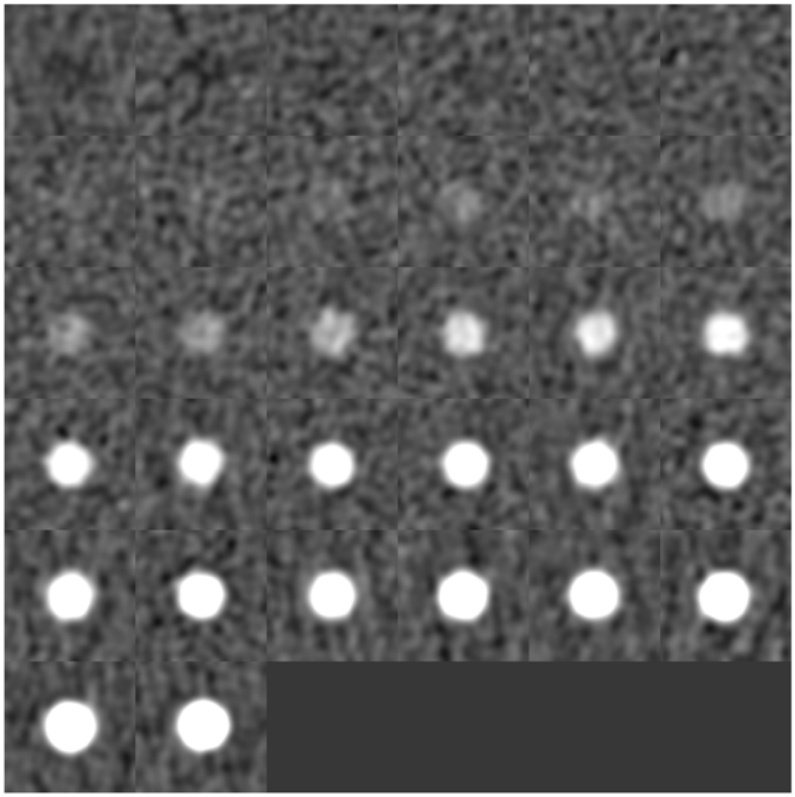


Figure 7.2: A montage of sample images cropped to the sample region. Good sample region centering indicates precise image segmentation.

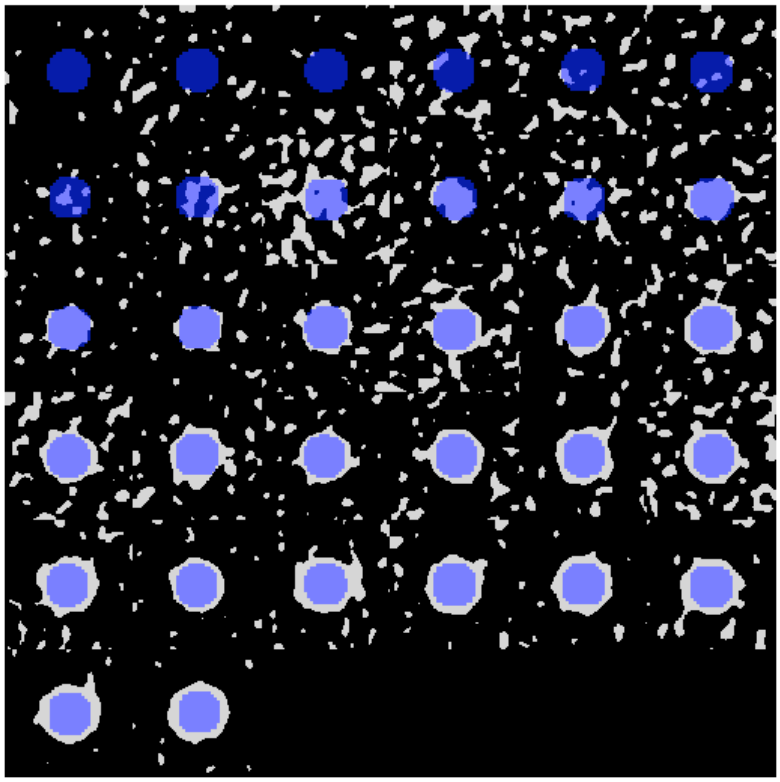


Figure 7.3: Montage of samples thresholded at constant level of 200 HU. The blue circles indicate the sampling region used to obtain the mean HU. The ROI circles 80% of the physical size of the sample phantom (5mm diameter).

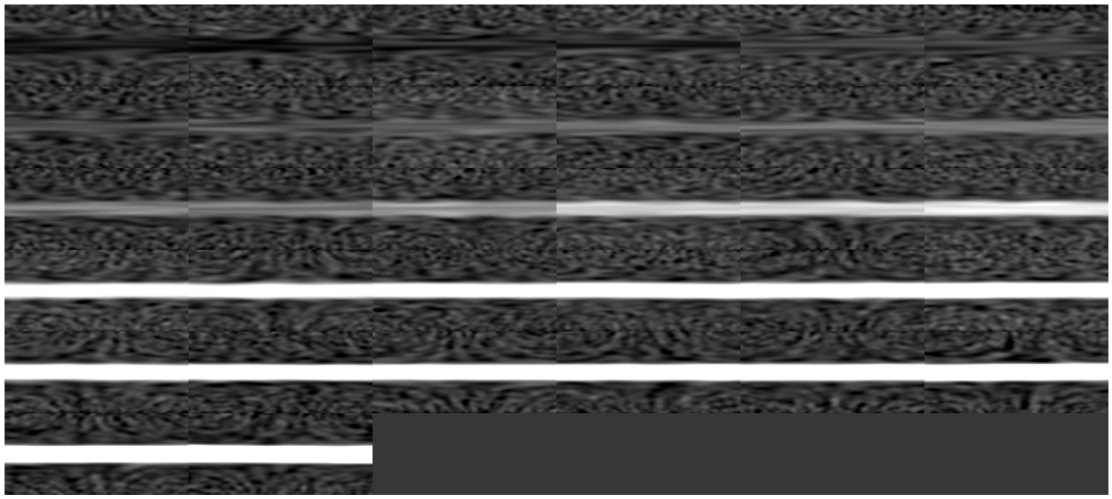


Figure 7.4: Montage of profile images.

7.1.2: Beam Hardening Results

The subtraction image revealed a difference in the blooming between the images thresholded at a constant value and those using the FWH-Mean value relative to their mean intensity. The red pixels in the subtraction image (Figure 7.6) are those which belong only to the image thresholded at a constant level. The bloomed region starts out thin and gets thicker as a function of increasing sample intensity.

The chosen static level was low enough to show frequent noise perturbations of the sample edge. Edge blooming relative to a constant threshold value depends on the relative difference in the intensity of the sample mean intensity and the chosen threshold shows little to no blooming as compared to high intensity samples.

The FWH-Mean measurements show little variability with θ (size of STD) and are consistently within the +/-1 pixel region. Measurements of the static thresholded regions increase with mean signal intensity and can exceed 50% blooming of the 5mm sample (7.5mm).

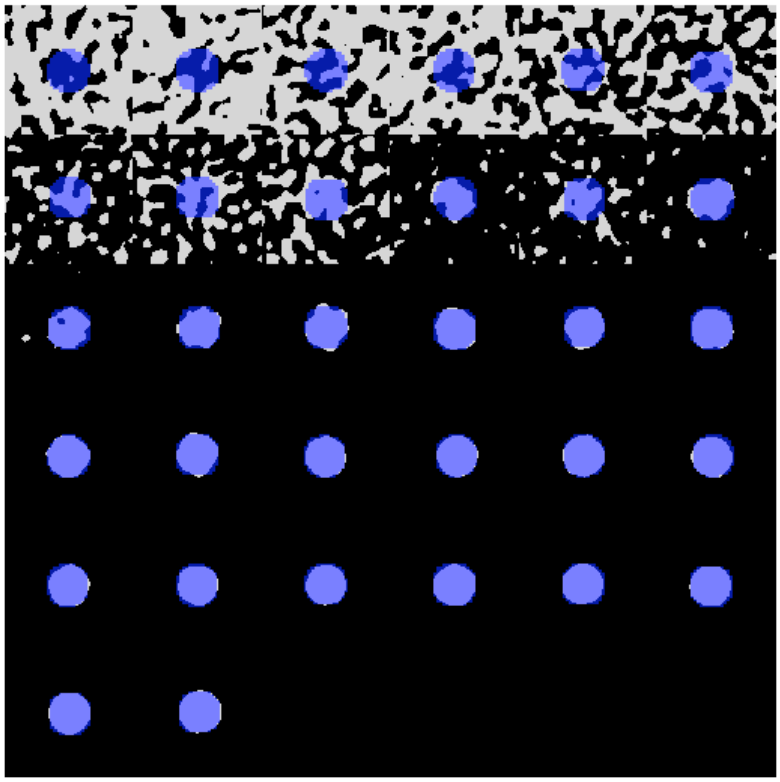


Figure 7.5: Montage of sample images thresholded at the half mean of the 80% ROI intensity. The blue circles indicate the 80% ROI sampling region.

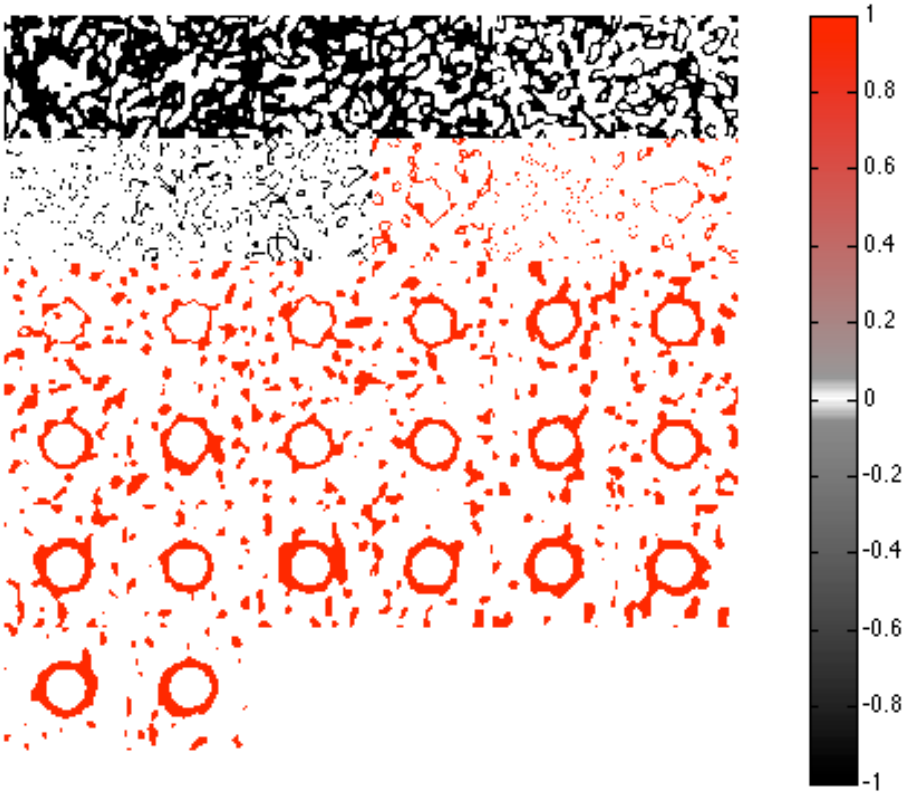


Figure 7.6: A subtraction image of images thresholded at a static level (200 HU) minus those thresholded at the FWH-Mean. The red region indicates pixels unique to the static level image.

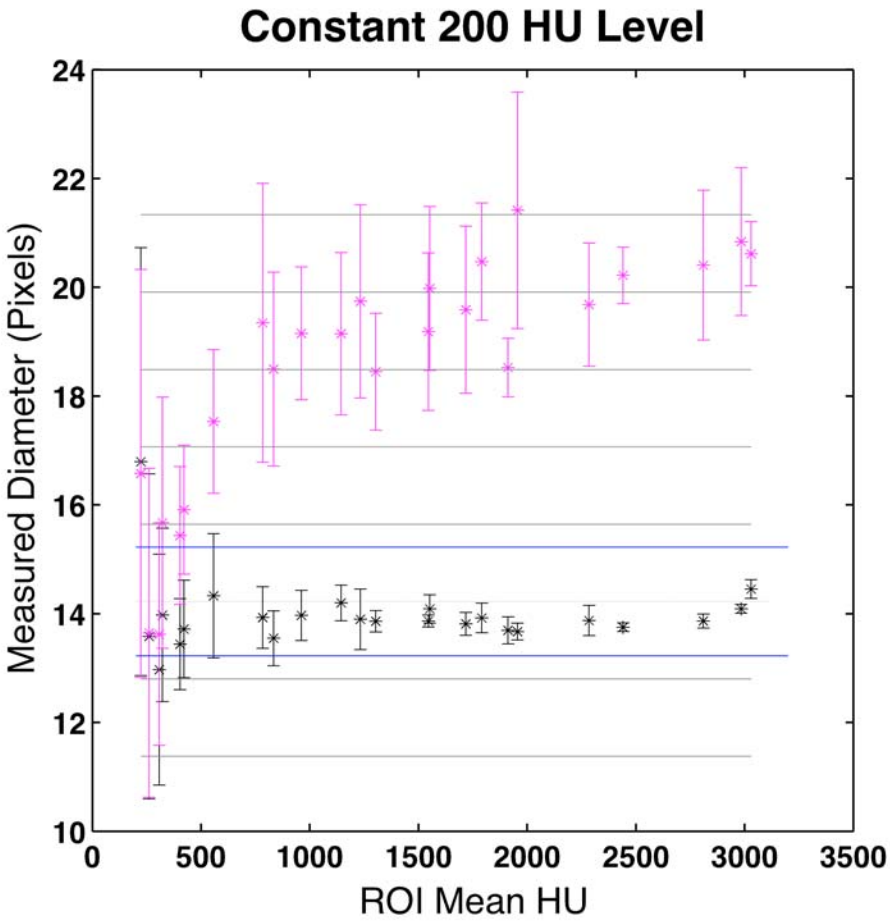


Figure 7.7: Measured sample diameters using the FWH-Mean threshold are displayed (black stars). The measured sample diameter using the constant level of 200HU is displayed (teal star). The two horizontal blue lines indicate the ideal value +/- one pixel region. Light gray horizontal lines are spaced at increments of 10% blooming.

8: Conclusions

8.1: Effect of Single Threshold Choice

Measurements of coronary plaque burden have been shown to vary with threshold choice [11]. To standardize the effect, the possible use of various threshold levels have been proposed [11]. Clinically, the choice of a low threshold of 130 HU is currently considered standard for CAC studies due to an increased sensitivity to plaque detection.

However, an evaluation of sample size with a 130 HU threshold was not possible in this study due to noise limitations resulting from the sample holder (110 +/- 60 HU). Sample blooming was evaluated at a threshold of 200HU such that all pixels greater than or equal to 200HU are white and the rest are black.

An over-representation of sample size (blooming), increasing with sample density was shown to range from 0 to 50% for samples having a 5mm diameter (approximately 14 pixels). Variability in diameter measurements of a given sample with respect to profile angle was attributable to noise and improves with higher threshold selection (see Figure 18). However, with higher threshold selection the sensitivity to low density samples was reduced.

The significant variation in blooming with increasing density stresses the importance of threshold choice. In particular, this study suggests that the choice of a single threshold when evaluating bulk calcium measurements may incur additional variability due to over-representation of plaque size relative to individual plaque density.

8.2: Threshold Choice Based on FWH-Mean

The blur inherent to CT images is well documented in the literature and is characterized by the point spread function [21]. The evaluation of blooming with respect to beam hardening requires controlling for variability in object size due to standard image blurring. Therefore, the standard clinical thresholding method cannot be used. An adapted form of the full width at half maximum technique commonly used in nuclear medicine to control for image blur was successfully adopted. The full width of the sample was determined using a threshold set at half the mean intensity of the region.

This study showed that accurate measurements of highly dense plaques are possible when using a level selected based on the FWH-Mean method to correct for image blurring. The choice of a separate threshold for each plaque was indicated for the accurate quantification of plaque size, luminal stenosis.

8.3: Beam Hardening Study

The placement of the ROI collection filter is critical to the determination of the mean sample level. Furthermore, the overlay of the region of interest montage onto the thresholded montage of the data set provided a useful static reference from which to qualitatively evaluate the presence of blooming.

No measurable effect of beam hardening on the measured in-plane diameter, blooming, was found when using the FWH-Mean technique to correct for image blur. Even at very high intensity levels corresponding to the maximum standard range of a CT scanner (3,096 HU) no measurable effect was present. The average measured in-plane diameter for samples ranging between 200 and 2,500 HU was consistent with the expected sample size of 14 pixels; measured values for this range are 14 +/- 1 pixel. The standard deviation of the measured in-plane diameter with respect to profile angle was given as a measure of the variability of sample shape; 0.5 +/- 0.5 pixels in this study. The high consistency of in-plane diameter measurements was attributable to the lack of sample holder noise at most FWH-Mean levels.

Given the lack of an effect when using the largest sample size (5mm) at very high-density values, further study of the sample size dependency of beam hardening on blooming was unnecessary. These findings suggest that the use of additional beam hardening correction algorithms is not required for discrete coronary plaques (< 5 mm).

8.3.1: Limitations

The potential impact from artifact sources other than beam hardening, as mentioned in the experimental phantom design, remain to be explained.

The phantom designed for this study was able to be used to evaluate sample densities as low as 200HU which was sufficient for examination of potential blooming resulting specifically from beam hardening. However, an evaluation of potential blooming at lower densities would require a sample holder that is more water equivalent.

The use of a single axial reconstruction technique in this study limits the ability to generalize the usefulness of the FWH-Mean technique to compensate for image blur to other clinical scenarios which use helical scan methods. Further research with respect to the use of different kernels to reduce reconstruction blur is also suggested.

Beam hardening did not cause measurable blooming of a single plaque phantom in and of itself. However the affect of beam hardening on coronary artery plaques from other larger high-density bodies within the field of view, such as the ribs and the sternum, remains to be seen.

The evaluation of the mean intensity was crucial to applying the FWH-Mean technique. The clinical utility of the FWH-Mean principle in the context standard observer-controlled measurements calculations remains to be seen. Commonly, the evaluation of vessel stenosis was subjective, with the radiologist evaluating the extent of the plaque visually. Typically an individual will use a point between the outer and inner most visible parts of the penumbra at the object edge. This method is essentially a

subjective application of some variation of the FWH-Mean principle. In dealing with manual calculations of vessel stenosis the importance of selecting a window and level that allows for the visual understanding of the plaque's entire HU range is evident, and a more objective method of evaluating the measurement intensity would be a benefit.

8.4: Partial Volume Study

The practical design and construction of the proposed partial volume phantom imposed limitations that precluded its use for study of the dependence of blooming on partial volume averaging. A result of the machining process was that the phantom (Polyoxymethylene) curved up in the direction of the notches. Since the notches of a curved phantom do not align with the image reconstruction plane, partial volume averaging will vary unpredictably along each notch. Further study is necessary to determine whether the problem of phantom warping can be resolved using different materials.

8.5: Cone Beam Study

As part of this pilot study a phantom was designed and constructed to evaluate the dependence of blooming on the cone beam affect. The phantom construction did not indicate that the phantom's use should be precluded.

9: Future Work

1. Perform analysis of data acquired for the experimental study of the contribution of the cone beam effect to the generation of the blooming artifact. This study requires that better memory management be implemented into the software protocol.
2. Investigate the design of a new partial volume phantom with special attention to the phenomena of warping of engineered plastics. The goal of this experiment would be to determine the amount of blooming generated by different known levels of partial volume averaging.
3. Study the effect of scanning and reconstruction protocols (kernel selection) on the magnitude of blooming as it relates to the dependency on increasing plaque intensity (choice of a single threshold value for all plaques).
4. Perform a study of the extent to which motion artifacts can contribute to the generation of the blooming artifact. This study would include motion in all three cardinal directions as well as rotational motion.
5. Compare the FWH-Mean technique developed here to standard hand calculations of the sample width. The purpose of this study would be to evaluate whether the application of the FWH-Mean measurement technique provides a benefit over a standard clinical technique for determining the extent of a blurred image object (by-hand).

BIBLIOGRAPHY

1. Abrams HL. Garland lecture. Coronary arteriography: pathologic and prognostic implications. Am. J. Roentgenol. 1982; 139:1-18.

2. Beohar N, Robbins JD, Cavanaugh BJ, et al. Quantitative assessment of in-stent dimensions: A comparison of 64 and 16 detector multislice computed tomography to intravascular ultrasound. Catheterization and Cardiovascular Interventions 2006; 68:8-10.

3. Choi HS, Choi BW, Choe KO, et al. Pitfalls, Artifacts, and Remedies in Multi- Detector Row CT Coronary Angiography. Radiographics 2004; 24:787-800.

4. Cody DD, PhD, Stevens DM, MS, Ginsberg LE, MD. Multi-Detector Row CT Artifacts That Mimic Disease. Radiology 2005:756 –761.

5. de Weert TT, Ouhlous M, Meijering E, et al. In Vivo Characterization and Quantification of Atherosclerotic Carotid Plaque Components With Multidetector Computed Tomography and Histopathological Correlation. Arterioscler Thromb Vasc Biol 2006; 26:2366-2372.

6. Dowe DAMD. Coronary CTA takes giant leap with 64-slice scanners -- Superior spatial and temporal resolution allowed by 4-cm detector and faster gantry speed makes coronary CTA possible in anyone. Diagnostic imaging 2005; 27:34.

7. Ferencik M, Chan RC, Achenbach S, et al. Arterial Wall Imaging: Evaluation with 16-Section Multidetector CT in Blood Vessel Phantoms and ex Vivo Coronary Arteries10.1148/radiol.2403051204. Radiology 2006; 240:708-716.

8. Flohr T, Ohnesorge BM, Bruder H, et al. Image reconstruction and performance evaluation for ECG-gated spiral scanning with a 16-slice CT system. Med. Phys. 2003; 30:2650-2662.

9. Hoffmann U, Ferencik M, Cury RC, Pena AJ. Coronary CT Angiography. J Nucl Med 2006; 47:797-806.

10. Hsieh J. Computed Tomography: Principles, Design, Artifacts, and Recent Advances. Bellingham, Washington: SPIE Press, 2003.

11. Moselewski F, Ferencik M, Achenbach S, et al. Threshold-dependent variability of coronary artery calcification measurements -- implications for contrast-enhanced multi-detector row-computed tomography. European Journal of Radiology Cardiac Imaging 2006; 57:390-395.

12. Ohnesorge BM, Flohr TG, Becker CR, Knez A, Reiser Mf. Multi-slice and Dual-source CT in Cardiac Imaging: Principles - Protocols - Indications - Outlook: Springer, 2006.

13. Oncel D, Oncel G, Karaca M. Coronary Stent Patency and In-Stent Restenosis: Determination with 64-Section Multidetector CT Coronary Angiography--Initial Experience. Radiology 2007; 242:403-409.

14. Rao PS, Alfidi RJ. The environmental density artifact: a beam-hardening effect in computed tomography. Radiology 1981; 141:223-227.

15. Rosamond W, Flegal K, Friday G, et al. Heart Disease and Stroke Statistics--2007 Update: A Report From the American Heart Association Statistics Committee and Stroke Statistics Subcommittee 10.1161/CIRCULATIONAHA.106.179918. Circulation 2007; 115:e69-171.

16. Ross R. The pathogenesis of atherosclerosis: a perspective for the 1990s. Nature 1993; 362:801-809.

17. Ross R. Atherosclerosis--an inflammatory disease. New England journal of medicine 1999; 340:115.

18. Schoepf UJ, Becker CR, Ohnesorge BM, Yucel EK. CT of Coronary Artery Disease. Radiology 2004; 232:18-37.

19. Seifarth H, Raupach R, Schaller S, et al. Assessment of coronary artery stents using 16-slice MD-CT angiography: evaluation of a dedicated reconstruction kernel and a noise reduction filter. European Radiology 2005; 15:721-726.

20. Van de Casteele E, Dyck V, Sijbers D, J. Raman E. The effect of beam hardening on resolution in x-ray microtomography [5370-239]. SPIE 2004; 5370:2089-2096.

21. Van de Casteele E. Model-based aproach for Beam Hardening Correction and Resolution Measurements in Microtomography. In:Department Natuurkunde Antwerpen: University Antwerpen, 2004.

Appendix A: Matlab Code

A Matlab script was developed to automate image processing. The functions central to the segmentation (regioncenterdetect.m) and measurement (fwpm1a.m) are included. The scripts used to coordinate image segmentation (CalcnStore.m) and to generate elliptical quality assurance images (QAImageGenerator.m) are included. In addition, a script used to demonstrate the concept of perceptual blooming is included.

Matlab's publish function was used to generate easy to read versions of the source code for each file. The reports include a table of contents followed by the code and any comments there in.

Image Processing

1. CalcnStore.m (Code)
2. regioncenterdetect.m (Code)
3. fwpm1a.m (Code)

Linear Blur Model

1. 2DConvolve_Demo_ED.m (Code)

Elliptical Q.A. Image Generation

1. QAImageGenerator.m (Code)

Contents

```
%%%%%%%%% Authorship
%  Author:  Eric Dick, M.S.
%           Department of Radiology Division of Radiation Oncology
%           University of Cincinnati, Cincinnati, OH 45243, USA
%           dicket@email.uc.edu / dickatphysics@gmail.com
%
%  Version: Beta        Revision: Aug. 3, 2007
```

0.0 Clear Work Space

```
% First Clear Workspace and Previously Used Variables
close all % closes all old figures
clc       % clears the command window of old output
clear all % clears all defined vvariablesaraibles
tic % start timer
```

NOTE: Current Design

```
display(['This code uses single stream processing independent of study'...
    'type.  This code requires that all studies listed as inputs have'...
    'directory structures specified by their keyfile.']);
display('Only one file per subdirectory is allowed as input.');
```

1.0.x Data Processing Counsel

```
%     1. Source Data Selection
%     2. Peak Detection Settings
%     3. Data Collection Settings
%     4. Image Segmentation Settings
```

Contents

```
%TODO: implement user specified constant level.

%Choose what percent of the reference intensity level to use.
PERCENTMAX     = 0.5;%Ratio of reference level
```

1.0.3 Data Collection Settings

```
percentofwindow = 1.2;%Percent of real sample size in which to accept peaks.

%Note: noisefile = 1 turns on the dicominfo and ROI detect for background
% images.  It's old purpose was also to enable the feature of collecting
% test and background profiles and then subtracting them from each other
% after the fact.
noisefile = 1;              % 1 = detect background info and ROI

classic_switch_backgroundsub = 0;%Subtract background b.f. profiling
mean_switch_backgroundsub    = 1;%Subtract mean of bkg. img. b.f. profiling


%A value > 1 indicates a wide window, these are used
%when attempting to catch peaks in a profile that isn't
%perfectly centered on the sample.

nd             = 1;  %Angle increment for profile gathering
peakdist       = 2;  %Minimum number of pixels allowed between peaks (0)

nopeakzone = 2*peakdist; %Book keeping, this is not an option to set!
```

1.0.4 Image Segmentation Settings

```
%Percent of real physical size in which to look for ROI statistics.
roipercentreal =0.80;%ROI radius % relative to expected size.

%Goal: Want background level to be high enough to clearly distinguish the
%sample holder from background noise.
%Also want a low enough sample level to detect samples near the
%sample holder intensity, but high enough to
%distinguish sample holder from sample.

bwlevela = [35, 200]; %For test image in Hounsfield unit
bwlevelb = [35, 250]; %For background image in Hounsfield units
rbig_size= 40;%(pix)radius used to collect profile over (pix)

fixedprofilelength = 1; %FIXED PROFILE LENGTH YES (1) OR NO (0)
testimgselect =1;  % GRAB FIRST FILE OF SET
noiseimgselect=1; % GRAB FIRST FILE OF SET
```

1.0.5 Display Settings

```
universalimagesize = 128;    %Will resize all cropped images (square).
display(universalimagesize);
```

1.0.6 Define General Save File Names

```
display('1.0.5 Data and Script File Paths')

% Save temporary key file to
save_tmpkey  = 'save_keyfile_tmp';
save_keeptkey= 'save_keyfile_keept';

% File name to save image variables to
save_images  = 'save_tempimages';

% File name to save misc variables to
save_misc    = 'save_mic';

% File name to save fwpm data table to as .xls format
save_table   = 'save_fwpm_datatable';
save_fwhm_mm = 'save_fwpm_mm';
```

1.0.7 Define General Path's

```
%NOTE: User defined paths should end in /

% Location of script files.
scriptdir    = '/Users/ericd/Desktop/Branch 1.0b1/MatlabWork/';

% Location of data files.
path2data    = '/Users/ericd/Desktop/Data/';

% Path to save results and temporary files to (ex.under SVN control)
save_path_parent =...
    '/Users/ericd/Desktop/Branch 1.0b1/MatlabWork/RawResults/';

% Folder to save temporary variables that are very large (ex. not in SVN)
% NOTE: Large tmp variables are stored in the DATA parent path.
save_tmplrgvar = 'TempVariableStorage/';

% Location of Test Data
Test_subPath    = 'Test/';
% Location of Noise Data
Noise_subPath   = 'Noise/';
%{
    %ALTERNATIVE: Ask user for data directory manually
    directory_name = uigetdir('/Users/ericd/Desktop/Data/')
%}
```

1.0.8 Define Study Specific Paths and File Names

```
%Location of data for each type of study
QA_gen          = 'QAGen/Data/';
QA_dir          = 'QAStaticEllipse/Data/';
QAnoise_dir     = 'QANoise/Data/';
BHSiz_dir       = 'BHDeltaSize/Data/';
CB_dir          = 'ConeBeam/Data/';
PV_dir          = 'PV/Data/';
```

```
BHRow_AxCTA_dir   = 'BHDeltaRowAxial_CTA/Data/';
BHRow_AxPCR_dir   = 'BHDeltaRowAxial_Custom/Data/';
BHRow_HeCTA_dir   = 'BHDeltaRowHelical/Data/';
BHRow_All_dir     = 'BHDeltaRow_All/Data/';

% Name of key file for each study
key_QA       = 'QA_StaticEllipse.csv';
key_QAnoise  = 'QAnoise_key.csv';
key_BHSiz    = 'bhdeltasize_key.csv';
key_CB       = 'conebeam_key.csv';
key_PV       = 'PV_key.csv';
key_BHRow    = 'bhdeltarow_key.csv';%*** Same name for each study BHRow
```

1.0.9 Choose Proper Directory for Study Type

```
%{
    NOTE: All studies share the same analysis protocol. What only need be
    done is to substitute in the correct data directory, save directory
    and keyfile name.
%}
if tempswitch == 0.1
    datadir      = [path2data        BHRow_AxCTA_dir];
    save_dir     = [save_path_parent BHRow_AxCTA_dir];
    save_dir_lrg = [path2data        BHRow_AxCTA_dir save_tmplrgvar];

    % Create save folder if needed
    mkdir(save_path_parent, BHRow_AxCTA_dir);
    mkdir([path2data BHRow_AxCTA_dir],save_tmplrgvar);
    % Specify key file name
    keyfile_name = [path2data BHRow_AxCTA_dir key_BHRow];

elseif tempswitch == 0.2
    datadir = [path2data BHRow_AxPCR_dir];
    save_dir = [save_path_parent BHRow_AxPCR_dir];
    save_dir_lrg = [path2data BHRow_AxPCR_dir save_tmplrgvar];
    % Create save folder if needed
    mkdir(save_path_parent, BHRow_AxPCR_dir);
    mkdir([path2data BHRow_AxPCR_dir],save_tmplrgvar);
    % Specify key file name
    keyfile_name = [path2data BHRow_AxPCR_dir key_BHRow];

elseif tempswitch == 0.3
    datadir = [path2data BHRow_HeCTA_dir];
    save_dir = [save_path_parent BHRow_HeCTA_dir];
    save_dir_lrg = [path2data BHRow_HeCTA_dir save_tmplrgvar];
    % Create save folder if needed
    mkdir(save_path_parent, BHRow_HeCTA_dir);
    mkdir([path2data BHRow_HeCTA_dir],save_tmplrgvar);
    % Specify key file name
    keyfile_name = [path2data BHRow_HeCTA_dir key_BHRow];

elseif tempswitch == 0.4
    datadir = [path2data BHRow_All_dir];
    save_dir = [save_path_parent BHRow_All_dir];
    save_dir_lrg = [path2data BHRow_All_dir save_tmplrgvar];
```

```
    % Create save folder if needed
    mkdir(save_path_parent, BHRow_All_dir);
    mkdir([path2data BHRow_All_dir],save_tmplrgvar);
    % Specify key file name
    keyfile_name = [path2data BHRow_All_dir key_BHRow];


elseif tempswitch == 1
    datadir = [path2data QA_dir];
    save_dir = [save_path_parent QA_dir];
    save_dir_lrg = [path2data QA_dir save_tmplrgvar];
    % Create save folder if needed
    mkdir(save_path_parent, QA_dir);
    mkdir([path2data QA_dir],save_tmplrgvar);
    % Specify key file name
    keyfile_name = [path2data QA_dir key_QA];


elseif tempswitch == 2
    datadir = [path2data BHSiz_dir];
    save_dir = [save_path_parent BHSiz_dir];
    save_dir_lrg = [path2data BHSiz_dir save_tmplrgvar];
    % Create save folder if needed
    mkdir(save_path_parent, BHSiz_dir);
    mkdir([path2data BHSiz_dir],save_tmplrgvar);
    % Specify key file name
    keyfile_name = [path2data BHSiz_dir key_BHSiz];

elseif tempswitch == 3
    datadir = [path2data CB_dir];
    save_dir = [save_path_parent CB_dir];
    save_dir_lrg = [path2data CB_dir save_tmplrgvar];
    % Create save folder if needed
    mkdir(save_path_parent, CB_dir);
    mkdir([path2data CB_dir],save_tmplrgvar);
    % Specify key file name
    keyfile_name = [path2data CB_dir key_CB];

elseif tempswitch == 4
    datadir = [path2data PV_dir];
    save_dir = [save_path_parent PV_dir];
    save_dir_lrg = [path2data PV_dir save_tmplrgvar];
    % Create save folder if needed
    mkdir(save_path_parent, PV_dir);
    mkdir([path2data PV_dir],save_tmplrgvar);
    % Specify key file name
    keyfile_name = [path2data PV_dir key_PV];

elseif tempswitch == 5
    datadir = [path2data QA_gen];
    save_dir = [save_path_parent QA_gen];
    save_dir_lrg = [path2data QA_gen save_tmplrgvar];
    % Create save folder if needed
    mkdir(save_path_parent, QA_gen);
    mkdir([path2data QA_gen],save_tmplrgvar);
    % Specify key file name
    keyfile_name = [path2data QA_gen key_QA];
```

```
elseif tempswitch == 6 % QA Noise Data

    datadir = [path2data QAnoise_dir];
    save_dir = [save_path_parent QAnoise_dir];
    save_dir_lrg = [path2data QAnoise_dir save_tmplrgvar];
    % Create save folder if needed
    mkdir(save_path_parent, QAnoise_dir);
    mkdir([path2data QAnoise_dir],save_tmplrgvar);
    % Specify key file name
    keyfile_name = [path2data QAnoise_dir key_QAnoise];

end

mkdir(save_dir_lrg);
```

2.0.x PreProcessing Summary

1. Update MatLab Path
2. File Detection and Organization
3. Resize Keyfile to Match Data Set
4. Generate DICOM Path's
5. Collect DICOM Headers
6. Store DICOMs to Varaible
7. Generate DICOM RD Variable for Montages

```
display('2.0.x Preprocessing Summary')
```

2.0.1 Update MatLab Path

```
display('2.0.1 Update MatLab Path')

% Update the Path to Include the Script Directory Tree
allsubpaths_code = genpath(scriptdir);
addpath(allsubpaths_code);
rehash;
% Update the Path to Include the Selected Data Directory Tree
allsubpaths_data = genpath(datadir);
addpath(allsubpaths_data);
rehash;
```

2.0.2 File Detection and Organization

```
display('2.0.2 File Gathering and Organization')

%Establish path variables to save data to
saveimpath = [save_dir_lrg save_images '.mat'];
savemispath = [save_dir_lrg save_misc '.mat'];

%Read in Key File
% Key file contains serial numbers and folder names for each dataset.
% Key file format: mx4 matrix (folder, SN, label, conc. rank)
%TODO: append path to keyfile's early on in code.
keyfile_tmp = csvread(keyfile_name);
```

```
%append column for binary masing inactive directories
keyfile_tmp(:,5) = zeros(size(keyfile_tmp,1),1);

% Store path and serial number (SN) from keyfile to variable
codeNames = keyfile_tmp(:,2)';
dateNames = keyfile_tmp(:,1)';

dirkey_select = arrayfun(@(parents) num2str(parents, '%08.0f'),...
    dateNames, 'UniformOutput', false);

codeName_select = arrayfun(@(subs) num2str(subs, '%04.0f'),...
    codeNames, 'UniformOutput', false);

% Preallocate some variables
counter=0;
dir_select=[];
active_dir = cell(1:length(codeNames):1);
%Preallocate Variables
codeNames_list = cell(1,length(codeNames));

% Verify expected data set is present, set binary value in keyfile_tmp
for na=1:1:length(codeNames)

    % Generate potential paths to data with path and serial number (SN)
    % Convert codeName (array) into a string
    %        codeName_select = int2str(codeNames(na));
    % Convert dateNames (array) into a string
    %        dirkey_select = int2str(dateNames(na));
    % Concatonate the sample directory to the path.
    dir_select=[datadir dirkey_select{na} '/' codeName_select{na} '/'];
    %%% final folder name has been created %%%
    %Determine if directory path combination exhists.
    tf = isdir(dir_select); %isdir returns binary yes/no existence test

    %If the directory exists add it to the list.
    if tf == 1
        counter = counter + 1;
        %Create an ordered list of paths to data subdirectories
        active_dir{1,counter}=dir_select;
        %Create an ordered list of codeNames of live directories.
        codeNames_list{1,na} = codeName_select{na};
        %Create mask column for data found in matlab path.
        keyfile_tmp(na,5) = 1; %label if data is found
        keyfile_tmp(na,6) = na; %record file "order number"
    end
end %%%Name list of data sub directories present in path complete.%%%

%Save keyfile_temp to file
savepathkey = [save_dir save_tmpkey '.csv'];
csvwrite(savepathkey, keyfile_tmp,1,0);

% **** Debug code (begin)
% Is the |codeNames_list| an Empty Set?
if isempty(codeNames_list)
    display(['Warning: Check your key file or directory structure,'...
        'no data was found.'])
end
```

```
% **** Debug code (end)
```

2.0.3 Resize Keyfile to Match Data Set

```
display('2.0.3 Resize Keyfile to Match Data Set')

%Reduce |keyfile| size to list the data subdirectories present
keyfile_mask = repmat(keyfile_tmp(:,5),[1,size(keyfile_tmp,2)]);
keyfile_keept =  keyfile_tmp(any(keyfile_mask,2),:);
```

2.0.3.1 Read In Phantom Specific Information

```
% Real diameter in mm
physsizemm = keyfile_keept(:,8);
% The known physical parameter that is varied (conc., eccentricity etc..)
indepvar = keyfile_keept(:,7);
% Independent variable integer ranking
intrank = keyfile_keept(:,4);
```

2.0.4 Generate DICOM Paths

```
display('2.0.4 Create Paths to All Files')

% Create List of Paths to Data Files
LengthActive_dir=length(active_dir); %Set Number of Maximum Iterations
samplelist = [];

%Pre Allocation of Variables
testfilepaths = cell(1,length(LengthActive_dir));
noisefilepaths = cell(1,length(LengthActive_dir));

% Create Paths to All the Files
for nb=1:LengthActive_dir
    %Create ordered list of whole paths to data folders.
    select_active_dir=(active_dir{nb});

    % Search the |/Test/| folder for DICOM files
    % Create an ordered list of the filenames by (ascending down)
    % Structure Fields: (name, date, bytes, isdir, datenum).
    testfolder=[select_active_dir Test_subPath];
    testfilestructure = dir(fullfile(testfolder,'*.dcm'));
    testfileList = sort({testfilestructure(:).name});
    %Add full path to image file to running list.
    TestSelect = fullfile(testfolder,...
        cell2mat(testfileList(testimgselect)));

    testfilepaths{1,nb}=TestSelect;

    %Retrieve the blank sample directory by appending the subdirectory
    %|/TestNoise/|.
    noisefolder = [select_active_dir Noise_subPath];
    noisefilestructure = dir(fullfile(noisefolder,'*.dcm'));

    noisefileList = sort({noisefilestructure(:).name});
```

```
    %The file name and path of the blank subject image is selected and
    %added to a running list of full path file names.
    try
        NoiseSelect = fullfile(noisefolder,...
            cell2mat(noisefileList(noiseimgselect)));
        noisefilepaths{1,nb}=NoiseSelect;
    catch
        display('You are missing the corresponding noise file')
        noisefile = 0; %disables noise file scripts any other value enables
        %FixMe: No longer fully supported many background script entries
        %are not enclosed in a if then loop.
        %ToDo: remove code duplication in noise and signal process refactor
    end
end %%% Full path list of image files created %%%
```

2.0.5 Collect DICOM Headers

```
display('2.0.5 Collect DICOM Information From Files')

%PreAllocate Variables
allprofiles_dicomslope = zeros(length(testfilepaths),1);
allprofiles_dicomintercept = zeros(length(testfilepaths),1);
dicompixsize = zeros(1,1,length(testfilepaths));

% Get DICOM information for sample files
for filenumber_idx = 1:1:length(testfilepaths)
    %Select image file to be analyzed
    %Get paths to test image file.
    file = testfilepaths{filenumber_idx};
    TestInfo = dicominfo(file);
    fileslope = (TestInfo.RescaleSlope);
    fileintercept = abs(TestInfo.RescaleIntercept);
    filepixsize = TestInfo.PixelSpacing(1);

    allprofiles_dicomslope(filenumber_idx,1) = fileslope; %Array
    allprofiles_dicomintercept(filenumber_idx,1) = fileintercept; %Array
    dicompixsize(1,1,filenumber_idx) = filepixsize; %Array
end

% Get DICOM information for background files
if noisefile ~= 0
    %Preallocate
    allprofiles_dicomintercept_bk = zeros(length(noisefilepaths),1);
    dicompixsize_bk = zeros(length(noisefilepaths),1);
    for filenumber_idx = 1:1:length(noisefilepaths)
        %Get paths to image files for sample and background
        file = noisefilepaths{filenumber_idx};
        TestInfo = dicominfo(file);
        intercept = abs(TestInfo.RescaleIntercept);
        pixsizetmp = TestInfo.PixelSpacing(1);

        allprofiles_dicomintercept_bk(filenumber_idx,1) = intercept; %Array
        dicompixsize_bk(1,1,filenumber_idx) = pixsizetmp; %Array
    end
else
```

```
    %An error has occured indicating that there are no noise files, this
    %error may also be generated if there is any number of missing noise
    %files.
    display('Insufficent number of background files, non were processed')
end
```

2.0.6 Store DICOM to Variable

```
display('2.0.6 DICOM Stored to Variable & HU Conversion')

%First save DICOM images as cell arrays (uint16 data).
dicompics_raw = permute(cellfun(@(paths) dicomread(paths), testfilepaths,...
    'UniformOutput',0),[2,1]);

%Second, convert uint16 images to the double format.
dicompicsdouble = cellfun(@(im) double(im),dicompics_raw,...
    'UniformOutput',0);

%Do same for noise files
if noisefile ~= 0

    dicompics_bk = permute(cellfun(@(paths) dicomread(paths), noisefilepaths,...
        'UniformOutput',0),[2,1]);
    dicompicsdouble_bk = cellfun(@(im) double(im),dicompics_bk,...
        'UniformOutput',0);
end

%This is useful for functions that do not accept unit16 data.
%Note MatLab offsets unit16 values by 1 before using them for color map
%When converting to double format add 1, if it's an image.
%Here we elect not to add 1 to preserve data values.  One may decide to
%add the one later for better correspondence with a color map.

%Third replicate intercept data to current file size.
dcminterceptcell = mat2cell(allprofiles_dicomintercept,ones(1,...
    length(allprofiles_dicomintercept)),1);

interceptimgcell = cellfun(@(inter, img) repmat(inter,size(img)),...
    dcminterceptcell,dicompics_raw,'UniformOutput',0);

%Then, convert the double images into Hounsfield Units.
dicomdoublehu = cellfun(@(im,intcpt) im - intcpt,dicompicsdouble,...
    interceptimgcell,'UniformOutput',0);

if noisefile ~= 0
dcminterceptcell_bk = mat2cell(allprofiles_dicomintercept_bk,ones(1,...
    length(allprofiles_dicomintercept_bk)),1);
interceptimgcell_bk = cellfun(@(inter, img) repmat(inter,size(img)),...
    dcminterceptcell_bk,dicompics_bk,'UniformOutput',0);
dicomdoublehu_bk = cellfun(@(im,intcpt) im - intcpt,dicompicsdouble_bk,...
    interceptimgcell_bk,'UniformOutput',0);
end

%Save and clear some variables to file: if first time there is no '-append'
save(saveimpath,'-regexp','^dicompics');
save(savemispath,'-regexp','^dcmint','^interc');
```

```
clear ('-regexp','^dicompics');
clear ('-regexp','^dcmint','interc');
%Variables spared in this section: dicomdoublehu*
```

2.0.7 Generate DICOM 4D Variable for Montages

```
display('2.0.7 Generate DICOM 4D Variable for Montages')

try
    %This step requires that all the images are of the same size because it
    %uses matricies to store image data. This is a requirement of the
    %montage function!  An alternative would be to create a simple test to
    %determine what image is largest and to pad all the other images so
    %that the images are all the same size.

    dicomhu_4D   = cell2mat(permute(dicomdoublehu   ,[4 2 3 1]));
    if noisefile ~= 0
    dicomhubk_4D = cell2mat(permute(dicomdoublehu_bk,[4 2 3 1]));
    end

catch
    display('A Montage of the origional data will not be available.')
    display('Images are not the same size.')
    display('Please refer to cropped image montages')
end

%Save and clear relevant variables
save(saveimpath,'-append','-regexp','dicomh');
clear ('-regexp','dicomh');

%Variables spared in this section: none

display('2.0.7 DICOM 4D Stored to Variable')
```

3.0.x Region of Interest Detection

```
1. Preallocate Variables
2. Region of Interest Detection
2.1 If the Sample Region is Not Detected Use Centroid of Sample Holder
3. Choose Size of ROI
4. Create ROI Mask
```

```
display('3.0.x Region of Interest Detection')
```

3.0.1 Preallocate Variables

```
display('3.0.1 Preallocating Variables')
testboundaries = cell(5,1,length(testfilepaths));
testlabelmatricies = cell(1,1,length(testfilepaths));

%Sample Images
xcbig               = zeros(length(testfilepaths),1);
ycbig               = zeros(length(testfilepaths),1);
xcsmall             = zeros(length(testfilepaths),1);
```

```
ycsmall             = zeros(length(testfilepaths),1);
bigDiameter         = zeros(length(testfilepaths),1);
smallDiameter       = zeros(length(testfilepaths),1);

big_eccen_ROI       = zeros(length(testfilepaths),1);
big_orientat_ROI    = zeros(length(testfilepaths),1);

small_eccen_ROI     = zeros(length(testfilepaths),1);
small_orientat_ROI  = zeros(length(testfilepaths),1);

%Background Images
xcbigbk             = zeros(length(testfilepaths),1);
ycbigbk             = zeros(length(testfilepaths),1);
bigDiameterbk       = zeros(length(testfilepaths),1);

%Debugging Arrays
debug_centerselect  = zeros(size(testfilepaths,2),1)+99;
number_o_ROI_found  = zeros(length(testfilepaths),2);

%Initialize Variables
endcellarray = 0; % to start while loop
runcount     = 0;

% Setup for Image Segmentation: ROI's

%Reorganize pixel size data for the DICOM files
dicom_mmperpix = permute(dicompixsize,[3,2,1]);

%FixMe: this diameter radius stuff seems screwed up
%Calculate desired size of ROI in pixels (uses user specified percentage)
physdiam_pix = arrayfun(@(rlsize, pxsize) (rlsize / pxsize),...
    physsizemm, dicom_mmperpix,'UniformOutput',0);
physdiamarr_pix = cell2mat(physdiam_pix); %physical diameter in pix.(array)
```

3.0.2 Region of Interest Detection

```
display('3.0.2 Region of Interest Detection')

%Loop Over Sample Images
for filenumber_idx = 1:1:length(testfilepaths)

    %Determine contours of sample holder and sample region in image.
    [centroidxtst, centroidytst, EquivalentDiametertst, boundary,labelmatrix,numberofbo
undaries,eccen_ROI,orien_ROI] =...
        regioncenterdetect(testfilepaths, filenumber_idx, bwlevela,...
        physdiamarr_pix);

    %Store boundary pixels in cell array.  One cell per boundry detected
    for k = 1:size(boundary,1)
        testboundaries(k,1,filenumber_idx)={boundary{k}};
    end

    %Next is a variable for debugging, it tells you how well ROI detection
    %is working on each of it's 2 passes for every file.

    %Note that it is possible for a large ROI to dissapear in the second
```

```
    %round if it has a hole in it.

    %Also it is possible for a large ROI to be detected twice if the
    %centroid of it's second discover is not exactly the same as the first.
    %This happens because the second pass uses different black and white
    %conversion level settings.

    number_o_ROI_found(filenumber_idx,[1 2]) = numberofboundaries;

    %Prepare loop to run next iteration
    endcellarray = 0;

    %Store label matrix for each file
    testlabelmatricies(1,1,filenumber_idx)= {labelmatrix};

    %Get centroid coordinates baised on the number of detected regions
    if size(centroidxtst,2) < 2
        %Less than two regions were detected, use the first one.
        xcbig(filenumber_idx,1) = centroidxtst(1); % centroid of sample holder
        ycbig(filenumber_idx,1) = centroidytst(1);% centroid of sample holder
        bigDiameter(filenumber_idx,1) = EquivalentDiametertst(1);
        big_eccen_ROI(filenumber_idx,1) = eccen_ROI(1);
        big_orientat_ROI(filenumber_idx,1) = orien_ROI(1);

        %Error detection: 1 = Only one region was detected.
        %display('<2')
        debug_centerselect(filenumber_idx,1) = 1;

    elseif size(centroidxtst,2) == 2
        %Two regions were detected.  Use the fist two.
        xcbig(filenumber_idx,1) = centroidxtst(1); % centroid of sample holder
        ycbig(filenumber_idx,1) = centroidytst(1); % centroid of sample holder
        xcsmall(filenumber_idx,1) = centroidxtst(2); % centroid of sample region
        ycsmall(filenumber_idx,1) = centroidytst(2); % centroid of sample region

        bigDiameter(filenumber_idx,1) = EquivalentDiametertst(1);
        smallDiameter(filenumber_idx,1) = EquivalentDiametertst(2);

        big_eccen_ROI(filenumber_idx,1) = eccen_ROI(1);
        big_orientat_ROI(filenumber_idx,1) = orien_ROI(1);
        small_eccen_ROI(filenumber_idx,1) = eccen_ROI(2);
        small_orientat_ROI(filenumber_idx,1) = orien_ROI(2);

        %Error detection: 0 = Things went as planned (2 regions detected)
        %display('==2')
        debug_centerselect(filenumber_idx,1) = 0;

    elseif size(centroidxtst,2) > 2
        %Temporary: Need some identification
        %More than two regions were detected.  Use the fist two.
        xcbig(filenumber_idx,1) = centroidxtst(1); % centroid of sample holder
        ycbig(filenumber_idx,1) = centroidytst(1); % centroid of sample holder
        xcsmall(filenumber_idx,1) = centroidxtst(2); % centroid of sample region
        ycsmall(filenumber_idx,1) = centroidytst(2); % centroid of sample region

        bigDiameter(filenumber_idx,1) = EquivalentDiametertst(1);
        smallDiameter(filenumber_idx,1) = EquivalentDiametertst(2);
```

```
        big_eccen_ROI(filenumber_idx,1) = eccen_ROI(1);
        big_orientat_ROI(filenumber_idx,1) = orien_ROI(1);
        small_eccen_ROI(filenumber_idx,1) = eccen_ROI(2);
        small_orientat_ROI(filenumber_idx,1) = orien_ROI(2);

        %Error detection: 3 = More than 2 Regions Were Found.
        display('Warning: More than 2 ROIs were found!')
        debug_centerselect(filenumber_idx,1) = 3;

    else
        display('NO REGIONS DETECTED!')
        centroidxtst;
        debug_centerselect(filenumber_idx,1) = 9;
    end %Variable format: file number in row order

end % Centroid data collected for test sample
%Generated: testlabelmatricies, xcbig, ycbig,bigDiameter (also small)

%Bug Notification
if ~isempty(find(debug_centerselect,1))
    display(['Warning: Unexpected events occured in detecting ROIs.'...
        'Please check the BW debug_centerselect output and BW level.'])
    display(['debug_centerselect = 1 if only one ROI found,'...
        '3 if more than one found and 9 if no ROI found'])
end

% Background Images
%TODO: Add changes to this section ( like above )
if noisefile ~= 0
    for loopindex = 1:1:length(noisefilepaths)
        %Select image file to be analyzed
        filenumber_idx = loopindex;
        %Determine contours of sample holder and sample region in image.
        [centroidxtst, centroidytst, EquivalentDiametertst,eccen_ROI,orien_ROI] = ...
            regioncenterdetect(noisefilepaths,filenumber_idx,bwlevelb,...
            physdiamarr_pix);
        %Get centroid coordinates
        xcbigbk(filenumber_idx,:) = centroidxtst(1); %centroid of holder
        ycbigbk(filenumber_idx,:) = centroidytst(1);%centroid of holder
        bigDiameterbk(filenumber_idx,:) = EquivalentDiametertst(1);
        %Variable format: file number in row order
    end % Centroid data collected for blank sample
elseif noisefile == 0
    %An error has occured indicating that there are no noise files, this
    %error may also be generated if there is any number of missing noise
    %files.
    display('Since there is an insufficent number of background files')
    display('non were processed')
end

%Generated: xcbigbk, ycbigbk, bigDiamterbk

%Save relevent variables by appending them to the file already created.
save  (savemispath,'-append','-regexp','^labelm','^testlabelma');
clear ('-regexp','^labelm','testlabelma');
```

```
%Variables spared in this section: dicomdoublehu*
```

3.0.2.1 If The Sample Region is Not Detected Use Centroid of Holder

```
display(['3.0.2.1 If The Sample Region is Not Detected'...
    'Use Centroid of Holder'])
zerosfoundidx = find(~xcsmall);
if size(zerosfoundidx,1) ~= 0
    xcsmall(zerosfoundidx) = xcbig(zerosfoundidx);
    ycsmall(zerosfoundidx) = ycbig(zerosfoundidx);
    display(['xc and yc were not detectable for some images'...
        'the centroid of the sample holder was used (xc_big)'...
        'Please check cropped images for accuracy.']);
end

save(savemispath,'-append','-regexp','zerosfoundidx');
```

3.0.3 Choose Size of ROI, using Known Size (input from keyfile)

```
display('3.0.3 Calculate ROI Size')

%Reorganize pixel size data for the DICOM files
dicom_mmperpix = permute(dicompixsize,[3,2,1]);

%Calculate desired size of ROI in pixels (uses user specified %)
roirad_pix = arrayfun(@(rlsize, pxsize) (roipercentreal*rlsize)/(2*pxsize),...
    physsizemm,dicom_mmperpix,'UniformOutput',0);

%Convert from cell to array
roirad_pix = cell2mat(roirad_pix);
```

3.0.4 Create ROI Mask

```
display('3.0.4 Create ROI Mask')

%Create circular ROI boundary for standard image files
thetaidx = 0:pi/360:2*pi;
xcircwall = arrayfun(@(xb,rad) (arrayfun(@(angle)(rad*cos(angle)+xb),...
    thetaidx')),xcsmall,roirad_pix,'UniformOutput',0);
ycircwall = arrayfun(@(yb,rad) (arrayfun(@(angle)(rad*sin(angle)+yb),...
    thetaidx')),ycsmall,roirad_pix,'UniformOutput',0);

%Create circular ROI boundary for standard image files
thetaidx = 0:pi/360:2*pi;
xcircwall_bk = arrayfun(@(xb,rad) (arrayfun(@(angle)(rad*cos(angle)+xb),...
    thetaidx')),xcbigbk,roirad_pix,'UniformOutput',0);
ycircwall_bk = arrayfun(@(yb,rad) (arrayfun(@(angle)(rad*sin(angle)+yb),...
    thetaidx')),ycbigbk,roirad_pix,'UniformOutput',0);



%Create Mask in Circular Region: 0's are Outside ROI
dicommask = cellfun(@(im,xs,ys) roipoly(im,xs,ys),dicomdoublehu,...
    xcircwall,ycircwall,'UniformOutput',0);
```

```
dicommask_bk = cellfun(@(im,xs,ys) roipoly(im,xs,ys),...
    dicomdoublehu_bk,xcircwall_bk,ycircwall_bk,'UniformOutput',0);

%Convert mask into double type data, the easiest way is to 0+ == +.
dicommaskdouble = cellfun(@(mask) double(+mask),dicommask,...
    'UniformOutput',0);
dicommaskdouble_bk = cellfun(@(mask) double(+mask),dicommask_bk,...
    'UniformOutput',0);

%Save relevent variables to file: if first time there is no '-append'
save(saveimpath,'-append','-regexp','dicommask$','dicommask_bk$');
clear ('-regexp','dicommask$','dicommask_bk$');

%Variables spared clearing cumulaitaive:
%dicomdoublehu* dicommaskdouble dicommaskdouble_bk
```

4.0.x Region of Interest Statistics

1. Apply Masks to Images in Hounsfield Units
2. Calculate ROI Statistics from Raw Images
3. Generate Background Subtraction Images and Apply ROI
4. Calculate ROI Statistics from Bkg. Subtraction Images
5. Reconfigure Background Subtraction Variables into 4D for Montages
6. Save and Clear Varaibles

4.0.1 Apply Masks to Images in Hounsfield Units

```
display('4.0.1 Apply Masks to Images in Hounsfield Units')

%Set all area outside of ROI to 0.  Note these are doubles in HU already!
roiimg = cellfun(@(im,mask)im.*mask,dicomdoublehu,dicommaskdouble,...
    'UniformOutput',0);
roiimg_bk = cellfun(@(im,mask)im.*mask,dicomdoublehu_bk,...
    dicommaskdouble_bk,'UniformOutput',0);

%Save relevent variables to file: if first time there is no '-append'
save(saveimpath,'-append','-regexp','roiimg');
clear ('-regexp','roiimg');
```

4.0.2 Calculate ROI Statistics from RAW Images in HU

```
%This section calculates and stores ROI data about images before background
%subtraction.  Please see next section for ROI statistics from images that
%have gone though background subtraction.

display('4.0.2 Calculate ROI Statistics from RAW Images in HU')

%Convert mask to logical values
dicommaskdouble_logical = cellfun(@(mask) logical(mask),dicommaskdouble,...
    'UniformOutput',0);
dicommaskdouble_bk_logical = cellfun(@(mask) logical(mask),...
    dicommaskdouble_bk,'UniformOutput',0);

%Collect data from within the ROI for analysis
%Use same sized ROI for background and sample images.
```

```
roidat = cellfun(@(im,immask) im(immask) ,dicomdoublehu,...
    dicommaskdouble_logical,'UniformOutput',0);

roidat_bk = cellfun(@(im,immask) im(immask) ,dicomdoublehu_bk,...
    dicommaskdouble_bk_logical,'UniformOutput',0);

%Calculate sample ROI statistics
meanROI = cellfun(@(imagef) mean(imagef),roidat,'UniformOutput',0);
stdROI = cellfun(@(imagef) std(imagef),roidat,'UniformOutput',0);
stderrorROI = cellfun(@(imagef) std(imagef)/sqrt(length(imagef)),...
    roidat,'UniformOutput',0);

%Calculate background ROI statistics
meanROI_bk = cellfun(@(imagef) mean(imagef),roidat_bk,'UniformOutput',...
    false);
stdROI_bk = cellfun(@(imagef) std(imagef),roidat_bk,'UniformOutput',0);
stderrorROI_bk = cellfun(@(imagef) std(imagef)/sqrt(length(imagef)),...
    roidat_bk,'UniformOutput',0);
```

4.0.3 Generate Background Subtraction Images

```
display('4.0.3 Generate Background Subtraction Images and Apply ROI')

%Create classic background subtraction images
dicomclassicsub = cellfun(@(im,bkg) im - bkg,dicomdoublehu,...
    dicomdoublehu_bk,'UniformOutput',0);

%Apply ROI mask to dicomclassicsub image
roiclassicsubimg = cellfun(@(img,immask) img .* immask,dicomclassicsub,...
    dicommaskdouble_logical,'UniformOutput',0);

%Create background subtraction images from mean of background image
replicationsize = size(dicomdoublehu{1,1});
meanbackgroundim = cellfun(@(meanval)...
    repmat(meanval,replicationsize),meanROI_bk,'UniformOutput',0);
dicommeansub = cellfun(@(im,bkg_mean) im - bkg_mean, dicomdoublehu,...
    meanbackgroundim,'UniformOutput',0);

%Apply ROI mask to dicommeansub
roimeansubimg = cellfun(@(im,mask)im.*mask,dicommeansub,...
    dicommaskdouble,'UniformOutput',0);
```

4.0.4 Calculate ROI Statistics from Bkg. Subtraction Images

```
display('4.0.4 Background Subtraction Statistics')

%Collect data from within the ROI of the classicsub img. for analysis
roidat_classicsub = cellfun(@(im,immask) im(immask),dicomclassicsub,...
    dicommaskdouble_logical,'UniformOutput',0);

roidat_meansub = cellfun(@(im,immask) im(immask) ,dicommeansub,...
    dicommaskdouble_logical,'UniformOutput',0);

%Calculate classic background subtraction image statistics
meanROI_classicsub = cellfun(@(imagef) mean(imagef),roidat_classicsub,...
```

```
    'UniformOutput',0);
stdROI_classicsub = cellfun(@(imagef) std(imagef),roidat_classicsub,...
    'UniformOutput',0);
stderrorROI_classicsub = cellfun(@(imagef) std(imagef)/...
    sqrt(length(imagef)),roidat_classicsub,'UniformOutput',0);

%Calculate mean background subtraction image statistics
meanROI_meansub = cellfun(@(imagef) mean(imagef),roidat_meansub,...
    'UniformOutput',0);
stdROI_meansub = cellfun(@(imagef) std(imagef),roidat_meansub,...
    'UniformOutput',0);
stderrorROI_meansub = cellfun(@(imagef) std(imagef)/sqrt(length(imagef)),...
    roidat_meansub,'UniformOutput',0);



%|usebiggestROI| = 1 indicates that a data series contains the same
%intensity and should be treated as such.  Since sampeling statistics favor
%larger samples we will select the largest sample of each series and apply
%it's mean to all other members of that series.

%NOTE: this section of code assumes that the input data set contains at
%least one series of data in which there is a maximum physical size
%(indicated in the keyfile).  Multiple sets of images are allowed, but they
%are all assumed to be of identical dimensions (there should be as many
%repeated maximum values as there are series in the input).  Sets may
%differ in density form eachother, but are assumed to have constistant
%density within any set.

%This section determines the level from ROI data that has undergone
%background subtraction, because the profile's they are applied to are from
%background subtracted images.

if usebiggestROI == 1
    if classic_switch_backgroundsub == 1
        mask_maxsamp = zeros([size(keyfile_keept,1),1]);
        mask_date = zeros([size(keyfile_keept,1),1]);

        %Get the mean ROI statistic for the largest sample size's.
        maxsize = max(keyfile_keept(:,4),[],1);
        indiciesfilt = find(keyfile_keept(:,4)==maxsize);
        meanROI_tmp = cell2mat(meanROI_classicsub(indiciesfilt));

        %Replicate the statistic for all indicies with same measurement
        %index.  If multiple sample sets are never on the same day then
        %this index is also essentially a date index.
        for i = 1:1:size(meanROI_tmp,1)
            %Look up date index for first value
            dateidx = keyfile_keept(indiciesfilt(i),3);
            keyfile_ind_select = find(keyfile_keept(:,3)==dateidx);
            select_statistic = meanROI_tmp(i);
            mask_date(keyfile_ind_select) = select_statistic;
        end

        %Redefine
        meanROI_classicsub = mat2cell(mask_date,ones([size(mask_date),1]),1);
```

```
    elseif mean_switch_backgroundsub == 1
        mask_maxsamp = zeros([size(keyfile_keept,1),1]);
        mask_date = zeros([size(keyfile_keept,1),1]);

        %Get the mean ROI statistic for the largest sample size's.
        maxsize = max(keyfile_keept(:,4),[],1);
        indiciesfilt = find(keyfile_keept(:,4)==maxsize);
        meanROI_tmp = cell2mat(meanROI_meansub(indiciesfilt));

        %Replicate the statistic for all indicies with same measurement
        %index.  If multiple sample sets are never on the same day then
        %this index is also essentially a date index.
        for i = 1:1:size(meanROI_tmp,1)
            %Look up date index for first value
            dateidx = keyfile_keept(indiciesfilt(i),3);
            keyfile_ind_select = find(keyfile_keept(:,3)==dateidx);
            select_statistic = meanROI_tmp(i);
            mask_date(keyfile_ind_select) = select_statistic;
        end

        %Redefine
        meanROI_meansub = mat2cell(mask_date,ones([size(mask_date),1]),1);

    end
end
```

4.0.5 Reconfigure Background Subtraction Variables into 4D for Montages

```
display('4.0.4 Generate Background Subtraction 4D Variable for Montages')

try
    %Background subtraction images
    dicomclassicsub_4D = cell2mat(permute(dicomclassicsub,[4 2 3 1]));
    dicommeansub_4D = cell2mat(permute(dicommeansub,[4 2 3 1]));
    %Background subtraction images with ROI mask
    roiclassicsubimg_4D = cell2mat(permute(roiclassicsubimg,[4 2 3 1]));
    roimeansubimg_4D = cell2mat(permute(roimeansubimg,[4 2 3 1]));
catch
    %Try fails if images are not the same size.
    display('A Montage of the origional data will not be available.')
    display('Images are not the same size.')
    display('Please refer to cropped image montages')
end

%Save relevent variables to file: if first time there is no '-append'
save(saveimpath,'-append','-regexp','_4D$');
clear ('-regexp','_4D$');
%Variables spared from clearing (cumulaitive):
%dicomdoublehu dicomdoublehu_bk roidat roidat_bk meanROI dicomclassicsub
%roiclassicsubimg roidat_classicsub roidat_meansub
```

4.0.6 Save and Clear Variables

```
%Save relevent variables to file: if first time there is no '-append'
%save(saveimpath,'-append','-regexp',);
```

```
save(savemispath,'-append','-regexp','roidat$','roidat_bk$',...
    '^meanROI_');
save(saveimpath,'-append','-regexp','^dicommaskd','dicomm',...
    'meanback');
save(savemispath,'-append','-regexp','^std','meanROI_bk$',...
    'replication');

clear ('-regexp','^dicommaskd','[^dicommeansub]dicomm','meanback','^std',...
    'replication');
clear ('-regexp','roidat$','roidat_bk$');
%Variables spared from clearing (cumulaitive):
%dicomdoublehu dicomdoublehu_bk meanROI dicomclassicsub
%roiclassicsubimg roidat_classicsub roidat_meansub
```

5.0.x Crop Images for Montage Display

1. Calculations for Cropping to Sample Holder
2. Calculations for Cropping Images
3. Crop Images to Sample Holder
4. Crop Images to Sample (Constant Sized Cropping Window)
5. Crop Images to Sample (Dynamic Size Cropping Window)
6. Generate 4D Array of Images Cropped to Holder for Montage Function
7. Generate 4D Array of Images Cropped to Sample (Static)
8. Generate 4D Array of Images Cropped to Sample (Dynamic)

```
display('5.0.x  Crop Images for Montage Display')
```

5.0.1 Calculations for Cropping Images

```
display('5.0.1 Calculations for Cropping Images')

%Cropping dimensions for sample holder with padding
diam_holder   = rbig_size*2 + 10;
widthhold     = diam_holder;
heighthold_mm = diam_holder;

%Create cropping window for sample holder with padding
constsizediam = repmat(max(physsizemm),size(physsizemm));
diam_sample   = constsizediam + 10;%Constant valued reguardless of sample size
widthsamp     = mat2cell(diam_sample,ones(size(diam_sample)),1);
heightsamp_mm = widthsamp;
calctmp       = cell2mat(heightsamp_mm)/pixsizetmp;
boxsize_pix   = num2cell(calctmp);
widthsamp     = boxsize_pix;

%Create cropping window for sample with padding (varying window size)
diam_sample_realwpad_mm = physsizemm + 10;
widthsamp_dynamic       = mat2cell(diam_sample_realwpad_mm,ones(size(...
    diam_sample_realwpad_mm)),1);
heightsamp_dynamicmm    = widthsamp_dynamic;
calctmp                 = cell2mat(heightsamp_dynamicmm)/pixsizetmp ;
boxsize_pixd            =  num2cell(calctmp);
widthsamp_dynamic       = boxsize_pixd;
```

5.0.2 Calculations for Cropping Images

```
display('5.0.2 Calculations for Cropping Images')

%Coordinates to center cropping on: use small or big ROI's centers
xcsmallcell  = mat2cell(xcsmall,ones(1,length(xcsmall)),1);%sample
ycsmallcell  = mat2cell(ycsmall,ones(1,length(ycsmall)),1);%sample
xcbigcell    = mat2cell(xcbig,ones(1,length(xcbig)),1);%sample
ycbigcell    = mat2cell(ycbig,ones(1,length(ycbig)),1);%sample
xcbigcellbkg = mat2cell(xcbigbk,ones(1,length(xcsmall)),1);%Bkg holder
ycbigcellbkg = mat2cell(ycbigbk,ones(1,length(ycsmall)),1);%Bkg holder

%Upper left coordinates for croppng to sample holder
xminsm_hold     = cellfun(@(xc) xc-(diam_holder/2), xcsmallcell,...
    'UniformOutput',0);
yminsm_hold     = cellfun(@(yc) yc-(diam_holder/2), ycsmallcell,...
    'UniformOutput',0);
xminbig_hold    = cellfun(@(xc) xc-(diam_holder/2), xcbigcell,...
    'UniformOutput',0);
yminbig_hold    = cellfun(@(yc) yc-(diam_holder/2), ycbigcell,...
    'UniformOutput',0);
xminbig_bk_hold = cellfun(@(xc) xc-(diam_holder/2), xcbigcellbkg,...
    'UniformOutput',0);
yminbig_bk_hold = cellfun(@(yc) yc-(diam_holder/2), ycbigcellbkg,...
    'UniformOutput',0);

%Upper left coordinates for croppng to sample (STATIC WINDOW)
xminsm_samp     = cellfun(@(xc,diam) xc-(diam/2), xcsmallcell,boxsize_pix,...
    'UniformOutput',0);
yminsm_samp     = cellfun(@(yc,diam) yc-(diam/2), ycsmallcell,boxsize_pix,...
    'UniformOutput',0);
xminbig_samp    = cellfun(@(xc,diam) xc-(diam/2), xcbigcell,boxsize_pix,...
    'UniformOutput',0);
yminbig_samp    = cellfun(@(yc,diam) yc-(diam/2), ycbigcell,boxsize_pix,...
    'UniformOutput',0);
xminbig_bk_samp = cellfun(@(xc,diam) xc-(diam/2), xcbigcellbkg,boxsize_pix,...
    'UniformOutput',0);
yminbig_bk_samp = cellfun(@(yc,diam) yc-(diam/2), ycbigcellbkg,boxsize_pix,...
    'UniformOutput',0);

%Upper left coordinates for croppng to sample (DYNAMIC WINDOW)
xminsm_samp_dynamic     = cellfun(@(xc,diam) xc-(diam/2), xcsmallcell,boxsize_pixd,...
    'UniformOutput',0);
yminsm_samp_dynamic     = cellfun(@(yc,diam) yc-(diam/2), ycsmallcell,boxsize_pixd,...
    'UniformOutput',0);
xminbig_samp_dynamic    = cellfun(@(xc,diam) xc-(diam/2), xcbigcell,boxsize_pixd,...
    'UniformOutput',0);
yminbig_samp_dynamic    = cellfun(@(yc,diam) yc-(diam/2), ycbigcell,boxsize_pixd,...
    'UniformOutput',0);
xminbig_bk_samp_dynamic = cellfun(@(xc,diam) xc-(diam/2), xcbigcellbkg,boxsize_pixd,...
    'UniformOutput',0);
yminbig_bk_samp_dynamic = cellfun(@(yc,diam) yc-(diam/2), ycbigcellbkg,boxsize_pixd,...
    'UniformOutput',0);
```

5.0.3 Crop Images to Sample Holder

```
totalmem_ed (whos);

display('5.0.3 Crop Images to Sample Holder')

%Compatable with different sized samples.
%Primary images cropped using the sample center only (even bkg).
%Use rectdb as an output parameter for information about each cropped
%images minimum xy position and size.

[holdcrophu_DICOM] = cellfun(@(im,xm,ym)...
    imcrop(im,[xm ym widthhold heighthold_mm]),dicomdoublehu,xminsm_hold,yminsm_hold,..
.
    'UniformOutput',0);
[holdcrophu_bk_DICOM] = cellfun(@(im,xm,ym)...
    imcrop(im,[xm ym widthhold heighthold_mm]),dicomdoublehu_bk,xminbig_bk_hold,yminbig
_bk_hold,...
    'UniformOutput',0);

%Classic background subtraction images cropped using the sample center
[holdcrophu_calssicsub] = cellfun(@(im,xm,ym)...
    imcrop(im,[xm ym widthhold heighthold_mm]),dicomclassicsub,xminsm_hold,yminsm_hold,
...
    'UniformOutput',0);

%Mean background subtraction images cropped using the sample center
[holdcrophu_meansub] = cellfun(@(im,xm,ym)...
    imcrop(im,[xm ym widthhold heighthold_mm]),dicommeansub,xminsm_hold,yminsm_hold,...
    'UniformOutput',0);

%ROI masked images cropped using sample center
[holdcrophu_roi_classicsub] = cellfun(@(im,xm,ym)...
    imcrop(im,[xm ym widthhold heighthold_mm]),roiclassicsubimg,xminsm_hold,yminsm_hold
,...
    'UniformOutput',0);
[holdcrophu_roi_meansub] = cellfun(@(im,xm,ym)...
    imcrop(im,[xm ym widthhold heighthold_mm]),roimeansubimg,xminsm_hold,yminsm_hold,..
.
    'UniformOutput',0);

%Intermediate images cropped using sample center
totalmem_ed (whos);
```

5.0.4 Crop Images to Sample (Constant Sized Cropping Window)

```
display('5.0.4 Crop Images to Sample (Constant Sized Cropping Window)')

%Primary images cropped using the sample center only (even bkg).
[sampcrophu_DICOM] = cellfun(@(im,xm,ym,wid,heig)...
    imcrop(im,[xm ym wid heig]),dicomdoublehu,xminsm_samp,yminsm_samp,widthsamp,...
    boxsize_pix,'UniformOutput',0);
[sampcrophu_bk_DICOM] = cellfun(@(im,xm,ym,wid,heig)...
    imcrop(im,[xm ym wid heig]),dicomdoublehu_bk,xminsm_samp,yminsm_samp,...
    widthsamp,boxsize_pix,'UniformOutput',0);

%Classic background subtraction images cropped using the sample center
[sampcrophu_calssicsub] = cellfun(@(im,xm,ym,wid,heig)...
```

```
    imcrop(im,[xm ym wid heig]),dicomclassicsub,xminsm_samp,yminsm_samp,...
    widthsamp,boxsize_pix,'UniformOutput',0);

%Mean background subtraction images cropped using the sample center
[sampcrophu_meansub] = cellfun(@(im,xm,ym,wid,heig)...
    imcrop(im,[xm ym wid heig]),dicommeansub,xminsm_samp,yminsm_samp,...
    widthsamp,boxsize_pix,'UniformOutput',0);

%ROI masked images cropped using sample center
[sampcrophu_roi_classicsub] = cellfun(@(im,xm,ym,wid,heig)...
    imcrop(im,[xm ym wid heig]),roiclassicsubimg,xminsm_samp,yminsm_samp,...
    widthsamp,boxsize_pix,'UniformOutput',0);
[sampcrophu_roi_meansub] = cellfun(@(im,xm,ym,wid,heig)...
    imcrop(im,[xm ym wid heig]),roimeansubimg,xminsm_samp,yminsm_samp,...
    widthsamp,boxsize_pix,'UniformOutput',0);

totalmem_ed (whos);
```

5.0.5 Crop Images to Sample (Dynamic Size Cropping Window)

```
display('Crop Images to Sample (Dynamic Size Cropping Window)')

%Primary images cropped using the sample center only (even bkg).
[sampcrophu_DICOM_dyn] = cellfun(@(im,xm,ym,wid)...
    imcrop(im,[xm ym wid wid]),dicomdoublehu,xminsm_samp_dynamic,yminsm_samp_dynamic,wi
dthsamp,...
    'UniformOutput',0);
[sampcrophu_bk_DICOM_dyn] = cellfun(@(im,xm,ym,wid)...
    imcrop(im,[xm ym wid wid]),dicomdoublehu_bk,xminsm_samp_dynamic,yminsm_samp_dynamic
,...
    widthsamp,'UniformOutput',0);

%Classic background subtraction images cropped using the sample center
[sampcrophu_calssicsub_dyn] = cellfun(@(im,xm,ym,wid)...
    imcrop(im,[xm ym wid wid]),dicomclassicsub,xminsm_samp_dynamic,yminsm_samp_dynamic,
...
    widthsamp,'UniformOutput',0);

%Mean background subtraction images cropped using the sample center
[sampcrophu_meansub_dyn] = cellfun(@(im,xm,ym,wid)...
    imcrop(im,[xm ym wid wid]),dicommeansub,xminsm_samp_dynamic,yminsm_samp_dynamic,...
    widthsamp,'UniformOutput',0);
clear rectdb;
%ROI masked images cropped using sample center
[sampcrophu_roi_classicsub_dyn] = cellfun(@(im,xm,ym,wid)...
    imcrop(im,[xm ym wid wid]),roiclassicsubimg,xminsm_samp_dynamic,yminsm_samp_dynamic
,...
    widthsamp,'UniformOutput',0);

[sampcrophu_roi_meansub_dyn] = cellfun(@(im,xm,ym,wid)...
    imcrop(im,[xm ym wid wid]),roimeansubimg,xminsm_samp_dynamic,yminsm_samp_dynamic,..
.
    widthsamp,'UniformOutput',0);

totalmem_ed (whos);
```

5.0.6 Generate 4D Array of Images Cropped to Holder for Montage Function

```
display('5.0.6 Generate 4D Array of Images Cropped to Holder for Montage Function')

%Primary images cropped using the sample center only (even bkg).
holdcrophu_DICOM_4D = cell2mat(permute(holdcrophu_DICOM,[4 2 3 1]));
holdcrophu_bk_DICOM_4D = cell2mat(permute(...
    holdcrophu_bk_DICOM,[4 2 3 1]));

%Classic background subtraction images cropped using the sample center
holdcrophu_calssicsub_4D = cell2mat(permute(...
    holdcrophu_calssicsub,[4 2 3 1]));

%Mean background subtraction images cropped using the sample center
holdcrophu_meansub_4D = cell2mat(permute(holdcrophu_meansub,[4 2 3 1]));

%ROI masked images cropped using sample center
holdcrophu_roi_classicsub_4D = cell2mat(permute(...
    holdcrophu_roi_classicsub,[4 2 3 1]));
holdcrophu_roi_meansub_4D = cell2mat(permute(...
    holdcrophu_roi_meansub,[4 2 3 1]));

%Intermediate images cropped using sample center
totalmem_ed (whos);
```

5.0.7 Generate 4D Array of Images Cropped to Sample (Static)

```
display('Generate 4D Array of Images Cropped to Sample (Static)')

%Resize all images to the standard montage size established above
scaled_sampcrophu_DICOM = cellfun(@(im) imresize(im,'OutputSize',...
    [universalimagesize universalimagesize]),sampcrophu_DICOM,'UniformOutput',0);
scaled_sampcrophu_bk_DICOM = cellfun(@(im)imresize(im,'OutputSize',...
    [universalimagesize universalimagesize]),sampcrophu_bk_DICOM,'UniformOutput',0);
scaled_sampcrophu_calssicsub = cellfun(@(im) imresize(im,'OutputSize',...
    [universalimagesize universalimagesize]),sampcrophu_calssicsub,'UniformOutput',0);
scaled_sampcrophu_meansub = cellfun(@(im) imresize(im,'OutputSize',...
    [universalimagesize universalimagesize]),sampcrophu_meansub,'UniformOutput',0);
scaled_sampcrophu_roi_classicsub = cellfun(@(im) imresize(...
    im,'OutputSize',[universalimagesize universalimagesize]),sampcrophu_roi_classicsub,
...
    'UniformOutput',0);
scaled_sampcrophu_roi_meansub = cellfun(@(im) imresize(im,...
    'OutputSize',[universalimagesize universalimagesize]),sampcrophu_roi_meansub,...
    'UniformOutput',0);

%Primary images cropped using the sample center only (even bkg).
sampcrophu_DICOM_4D = cell2mat(permute(scaled_sampcrophu_DICOM,[4 2 3 1]));
sampcrophu_bk_DICOM_4D = cell2mat(permute(...
    scaled_sampcrophu_bk_DICOM,[4 2 3 1]));

%Classic background subtraction images cropped using the sample center
sampcrophu_calssicsub_4D = cell2mat(permute(...
    scaled_sampcrophu_calssicsub,[4 2 3 1]));
```

```
%Mean background subtraction images cropped using the sample center
sampcrophu_meansub_4D = cell2mat(permute(scaled_sampcrophu_meansub,...
    [4 2 3 1]));

%ROI masked images cropped using sample center
sampcrophu_roi_classicsub_4D = cell2mat(permute(...
    scaled_sampcrophu_roi_classicsub,[4 2 3 1]));
sampcrophu_roi_meansub_4D = cell2mat(permute(...
    scaled_sampcrophu_roi_meansub,[4 2 3 1]));

%Intermediate images cropped using sample center

totalmem_ed (whos);
```

5.0.8 Generate 4D Array of Images Cropped to Sample (Dynamic)

```
display('5.0.8 Generate 4D Array of Images Cropped to Sample (Dynamic)')

%Resize all images to the standard montage size established above
scaled_sampcrophu_DICOM_dyn = cellfun(@(im) imresize(im,'OutputSize',...
    [universalimagesize universalimagesize]),sampcrophu_DICOM_dyn,'UniformOutput',0);
scaled_sampcrophu_bk_DICOM_dyn = cellfun(@(im)imresize(im,'OutputSize',...
    [universalimagesize universalimagesize]),sampcrophu_bk_DICOM_dyn,'UniformOutput',0)
;
scaled_sampcrophu_calssicsub_dyn = cellfun(@(im) imresize(im,'OutputSize',...
    [universalimagesize universalimagesize]),sampcrophu_calssicsub_dyn,'UniformOutput',
0);
scaled_sampcrophu_meansub_dyn = cellfun(@(im) imresize(im,'OutputSize',...
    [universalimagesize universalimagesize]),sampcrophu_meansub_dyn,'UniformOutput',0);
scaled_sampcrophu_roi_classicsub_dyn = cellfun(@(im) imresize(...
    im,'OutputSize',[universalimagesize universalimagesize]),sampcrophu_roi_classicsub_
dyn,...
    'UniformOutput',0);
scaled_sampcrophu_roi_meansub_dyn = cellfun(@(im) imresize(im,...
    'OutputSize',[universalimagesize universalimagesize]),sampcrophu_roi_meansub_dyn,..
.
    'UniformOutput',0);

%Primary images cropped using the sample center only (even bkg).
sampcrophu_DICOM_4D_dyn = cell2mat(permute(scaled_sampcrophu_DICOM_dyn,[4 2 3 1]));
sampcrophu_bk_DICOM_4D_dyn = cell2mat(permute(...
    scaled_sampcrophu_bk_DICOM_dyn,[4 2 3 1]));

%Classic background subtraction images cropped using the sample center
sampcrophu_calssicsub_4D_dyn = cell2mat(permute(...
    scaled_sampcrophu_calssicsub_dyn,[4 2 3 1]));

%Mean background subtraction images cropped using the sample center
sampcrophu_meansub_4D_dyn = cell2mat(permute(scaled_sampcrophu_meansub_dyn,...
    [4 2 3 1]));

%ROI masked images cropped using sample center
sampcrophu_roi_classicsub_4D_dyn = cell2mat(permute(...
    scaled_sampcrophu_roi_classicsub_dyn,[4 2 3 1]));
sampcrophu_roi_meansub_4D_dyn = cell2mat(permute(...
    scaled_sampcrophu_roi_meansub_dyn,[4 2 3 1]));
```

```
%Intermediate images cropped using sample center

%Save relevent variables to file: if first time there is no '-append'
save(saveimpath,'-append','-regexp','^dicom','dyn$');
%Save variables ending in roi to misc. file
save(savemispath,'-append','-regexp','^roi');
%clear ('-regexp',);
clear ('-regexp','^roi','^dicom','dyn$');

totalmem_ed (whos);
```

6.0.x Data Aquisition

1. Preallocate of Variables
2. Set Profile Collection Length
3. Collect Sample Profiles
4. Generate 4D Montage Matricies

```
display('6.0.x Data Aquisition')
```

6.0.1 Preallocate of Variables

```
display('6.0.1 Preallocate of Variables')

%allprofiles_rbig_a = cell(1,1,length(testfilepaths));
%allprofiles_rbig_b = cell(1,1,length(testfilepaths));
%allprofiles_backgnd = cell(1,1,length(testfilepaths));
%allprofiles_dicomintercept = zeros(length(testfilepaths),1);

%allprofiles_dicomintercept_bk = zeros(length(testfilepaths),1);
```

6.0.2 Set Profile Collection Length

```
display('6.0.2 Set Profile Collection Length')

if fixedprofilelength == 1
    %Use manually entered value (see 1.4) for profile length.
    %"Manual Setting"
    rbig = rbig_size;
elseif fixedprofilelength == 0
    %Uses the estimated diameter from ROI, use 1 file info for everything!
    %"Semi-Automatic Setting"
    rbig = bigDiameter(10,1)/2;
    %rsmall = smallDiameter(10,1)/2;
else
    %Error checking
    display('THE PROFILE LENGTH TECHNIQUE HAS NOT BEEN SET!')
    display('THE DEFAULT MODE CHOSEN (Manual): rbig=rbig_size')
    rbig = rbig_size;
end
```

6.0.3 Collect Sample Profiles

```
totalmem_ed (whos);
display('6.0.3 Collect Sample Profiles')

if classic_switch_backgroundsub == 0 && mean_switch_backgroundsub == 0
    display(['classic_switch_backgroundsub and '...
        'mean_switch_backgroundsub are 0.'])
    display('This does not make any sense, so I will just use the')
    display('closest approximation to raw image data without background')
    display('subtraction.  mean_switch_backgroundsub == 1')
    mean_switch_backgroundsub = 1;
    num_o_input = 1;

elseif classic_switch_backgroundsub == 1 && mean_switch_backgroundsub == 0
    load(saveimpath,'dicomclassicsub_4D');
    num_o_input = 1;

elseif classic_switch_backgroundsub == 0 && mean_switch_backgroundsub == 1
    load(saveimpath,'dicommeansub_4D');
    num_o_input = 1;

    % elseif classic_switch_backgroundsub == 1 && mean_switch_backgroundsub == 1
    %     num_o_input = 2;
    %     load(saveimpath,'dicomclassicsub_4D');
    %     load(saveimpath,'dicommeansub_4D');
    %     %display('Collection of profiles for two data sets simultainously')
    %     %display('is not currently supported.  See source section 6.0.3')
end

% Preallocate
profilestorage = zeros(floor(2*rbig),180/nd);
linex = zeros(2,180/nd);
liney = zeros(2,180/nd);

allprofiles_rbig_classicsub = cell(1,1,1,length(testfilepaths));
allprofiles_rbig_meansub = cell(1,1,1,length(testfilepaths));

%NOTE: good attempt at accepting multiple styled processing simulatniously

%FIXME: Don't do this since going to functionalization.

%Loop over all input variables.
for inputnumber_idx = 1:1:num_o_input
    if inputnumber_idx == 1 && classic_switch_backgroundsub == 1
        testimagea = dicomclassicsub_4D;
        clear dicomclassicsub_4D;
    elseif inputnumber_idx == 1 && mean_switch_backgroundsub == 1
        testimagea = dicommeansub_4D;
        clear dicommeansub_4D;
    elseif inputnumber_idx == 2 && classic_switch_backgroundsub == 1
        clear testimagea
        testimagea = dicomclassicsub_4D;
        clear dicomclassicsub_4D;
    elseif inputnumber_idx == 2 && mean_switch_backgroundsub == 1
        clear testimagea
        testimagea = dicommeansub_4D;
        clear dicommeansub_4D;
```

```
    end

    for filenumber_idx = 1:1:length(testfilepaths)
        testimageselect = testimagea(:,:,1,filenumber_idx);

        %Calculate Points of a Circle
        %format compact;
        %format long e;
        thetan = 0:nd:360;
        xi = rbig * cosd(thetan) + xcsmall(filenumber_idx,:);
        yi = rbig * sind(thetan) + ycsmall(filenumber_idx,:);

        % Collect Profile Data
        for ni = 1:1:(180/nd);
            %Create Array of Profile Start and End Points.
            distx = [xi(ni) xi(ni+(180/nd))];
            disty = [yi(ni) yi(ni+(180/nd))];

            %Bicubic provides the beat approximation of local pixels
            %Alternatives are available: nearest, bilinear.
            profiledata = improfile(testimageselect,distx,disty,2*rbig,'bicubic');

            profilestorage(:,ni) = profiledata;
            linex(:,ni) = distx';
            liney(:,ni) = disty';
        end
        %FIXME: I changed (:,:,bob) to (1,1,1,bob) fix cell ref to var
        %below.
        %Store raw profiles
        if inputnumber_idx == 1 && classic_switch_backgroundsub == 1
            allprofiles_rbig_classicsub(1,1,1,filenumber_idx) = {profilestorage};

        elseif inputnumber_idx == 1 && mean_switch_backgroundsub == 1
            allprofiles_rbig_meansub(1,1,1,filenumber_idx) = {profilestorage};

        elseif inputnumber_idx == 2 && classic_switch_backgroundsub == 1
            allprofiles_rbig_classicsub(1,1,1,filenumber_idx) = {profilestorage};

        elseif inputnumber_idx == 2 && mean_switch_backgroundsub == 1
            allprofiles_rbig_meansub(1,1,1,filenumber_idx) = {profilestorage};
        end

        %linexstorage_samp(:,filenumber_idx) = {linex};
        %lineystorage_samp(:,filenumber_idx) = {liney};
    end % Sample profile data collected
end

clear testimagea;
totalmem_ed (whos);
```

6.0.4 Generate 4D Montage Style Matrices

```
display('6.0.4 Generate 4D Montage Style Matrices')

%FIXME: doesn't have a 4D array type structure
if classic_switch_backgroundsub == 1
```

```
    allprofiles_rbig_classicsub_4D = cell2mat(permute(...
        allprofiles_rbig_classicsub,[4 2 3 1]));

elseif mean_switch_backgroundsub == 1
    allprofiles_rbig_meansub_4D =cell2mat(permute(...
        allprofiles_rbig_meansub,[4 2 3 1]));
end
```

```
%Save and clear some variables to file: if first time there is no '-append'
save(saveimpath,'-append','-regexp','_4D$');
clear ('-regexp','_4D$');

totalmem_ed (whos);
```

7.0.x Post-processing

1. Select Profile Image Set To Analyze and Define Average Signal
2. Set Level for FWPM Detection
3. Preallocate Variables
4. Detect Peaks in Profile Data
5. FWPM Acquisition

```
display('7.0.x Post-processing')
```

7.0.1 Select Profile Image Set To Analyze

```
display('7.0.1 Select Profile Image Set To Analyze')

%The remaining code below does not currently support analysis of multiple
%background subtraction methods simultaneously.
%TODO: Add support for performing multiple background subtraction at same
%time.

%If multiple background subtraction methods are selected the following
%analysis will be performed on the default setting (mean background
%subtraction data).

if classic_switch_backgroundsub == 1
    allprofiles_sample_hu_sub = allprofiles_rbig_classicsub;
elseif mean_switch_backgroundsub == 1
    allprofiles_sample_hu_sub = allprofiles_rbig_meansub;
end
```

7.0.2 Set Level for FWPM Detection

```
display('7.0.2 Set Percent Level for FWPM Detection')

load(savemispath,'meanROI_meansub','meanROI_classicsub');

%Note: Data has already been through background subtraction, so the values
%here are not pure Hounsfield units (they are lower).

if doublydynamiclevel == 1
```

```
    %The level at which FWPM is calculated changes with every theta value
    %with this setting.  Due to the dependence on theta the code for this
    %option lies in the for loop rather than here.
    %Note: This method is a remnant from an early alpha and is only
    %kept as an option for reference and curiosity.

elseif dynamiclevel_meanROI == 1
    %Use the ROI mean as the reference from which to calculate the % level.

    if classic_switch_backgroundsub == 1
        ythreshcell = cellfun(@(estmean) PERCENTMAX*estmean,...
            meanROI_classicsub,'UniformOutput',0);

    elseif mean_switch_backgroundsub == 1
        ythreshcell = cellfun(@(estmean) PERCENTMAX*estmean,...
            meanROI_meansub,'UniformOutput',0);
    end

elseif staticlevel == 1
    %This switch was broken in previous versions (it stated that it would
    %use the max value in a ROI (err. profile set) as the file number's max
    %and ythresh reference.  In fact later it is used as if it has a static
    %value over all file.  It was in fact a crude attempt at doing a fully
    %static level.
    %ythresh =
    %PERCENTMAX*max(max(allprofiles_sample_hu_sub{1,1,filenumber}))

elseif userspecifiedconstlevel == 1;

    %This value is hard coded in the loop.
end
```

7.0.3 Preallocate Variables

```
display('7.0.3 Preallocate Variables')
```

7.0.4 Detect Peaks in Profile Data

```
display('7.0.4 Detect Peaks in Profile Data')

% Peak's are detected using mspeaks (bioinformatics toolbox) and is
% vectorized rather than having 2 loops.  If all profiles were converted to
% a mxnxk matrix you might be able to vectorize both loops.

%Preallocate r and dummy variables to the size of the first profile set.
[r,dummy] = size(allprofiles_sample_hu_sub{1}(:,:));
x=(1:1:r)'; % Row number list

%Preallocate memory for variable
Peaks = cell(size(allprofiles_sample_hu_sub{1,1,1,1},2),1,...
    size(testfilepaths,2));

% Find peaks that are a minimum of peakdist apart.
for loopindex = 1:1:size(testfilepaths,2);
    %loopindex %for debugging
```

```
    y = allprofiles_sample_hu_sub{1,1,loopindex};
    peakdist = 0; %temp fix don't combine close peaks
    Peaks(:,:,loopindex) = mspeaks(x,y,'DENOISING',false,...
        'OverSegmentationFilter', peakdist, 'HeightFilter',0);
end %Cell's of Arrays of Peaks for All Files and Angles
```

7.0.5 FWPM Acquisition

```
display('7.0.5 FWHM Acquisition')

load(saveimpath,'dicom_mmperpix');

%Get size of a sample image (image 1)
[r,c] = size(allprofiles_sample_hu_sub{1}(:,:));

%Preallocate memory for variables
fwatpercentmaxlist    = zeros(c,length(testfilepaths));
fwthreshold           = zeros(c,length(testfilepaths));
fwwindowsize          = zeros(c,length(testfilepaths));
ditchstatus           = zeros(c,length(testfilepaths));

fwwindowpts           = zeros(c,2,length(testfilepaths));

fwcoordinates         = cell(c,length(testfilepaths));
fwpeaks               = cell(c,length(testfilepaths));
fwfirstpts            = cell(c,length(testfilepaths));
fwpm_errmsg           = cell(c,length(testfilepaths));
fwpm_errid            = cell(c,length(testfilepaths));
fwpm_errst            = cell(c,length(testfilepaths));


fwatpercentmaxlist_av = zeros(length(testfilepaths));
fwthreshold_av        = zeros(length(testfilepaths));
fwwindowsize_av       = zeros(length(testfilepaths));
ditchstatus_av        = zeros(length(testfilepaths));

fwwindowpts_av        = zeros(length(testfilepaths),2);

fwcoordinates_av      = cell(length(testfilepaths));
fwpeaks_av            = cell(length(testfilepaths));
fwpm_errmsg_av        = cell(length(testfilepaths));
fwpm_errid_av         = cell(length(testfilepaths));
fwpm_errst_av         = cell(length(testfilepaths));
fwfirstpts_av         = cell(length(testfilepaths));

%Collect data for each file in a loop

for filenumber_idx = 1:1:length(testfilepaths)
    %Loop over angles and get FWHM for each profile

    for angle = 1:1:c

        %Look at Y peak values for selected subtraction image
        peakspro = Peaks{angle,:,filenumber_idx}(:,2);

        %Find the row-index of the peaks
```

```
        %Index location in peakspro ONLY
        [rowpk] = find(peakspro);
        %Index location in profile
        %Note: values are be rounded because the output of mspeaks is
        %formatted as x.0000 and can contain extra digits that can't be
        %displayed.  Despite having integer input for the x parameter
        %mspeaks can output slightly larger values which are visually
        %reported in a deceiving way 15.0000 > 15 returns true.
        x_index = round(Peaks{angle,:,filenumber_idx}(rowpk,1));

        %Get the values of those peaks
        selectedpeak_val = peakspro;

        %Convert peak data to format that the fwhm.m file expects Column
        %headings: Peak#, index (x), Y(value)
        Pav = zeros(length(selectedpeak_val), 4); %Setup |Pav| structure
        if isempty(selectedpeak_val)
            %Don't try and list the count of peaks
        else
            Pav(:,1) = 1:1:length(selectedpeak_val); %Peak number
        end
        Pav(:,2) = x_index; %Index in whole profile
        Pav(:,3) = selectedpeak_val; %Peak height

        %Prepare profile for fwhm.m function
        Ihu = allprofiles_sample_hu_sub{1,1,1,filenumber_idx}(:,angle);

        %Acquire the FWHM using homemade function fwhm.m
        %[FWHM_SingleFileList,keepcrossing] = fwhm(Ihu, Pav, PERCENTMAX);

        %% Request ythreh number from cell array.
        if doublydynamiclevel == 1
            %This method is dependent on theta and file number.
            ythresh = PERCENTMAX*max(Ihu); %Level(theta,file)

        elseif dynamiclevel_meanROI == 1
            %Since threshold doesn't depend on theta it should be constant
            %So, select the first value for each file.
            ythresh = ythreshcell{filenumber_idx}(1,1);

        elseif staticlevel == 1
            %This value is set in section 7.0.2 and is independent of theta
            %or filenumber_idx.

        elseif userspecifiedconstlevel == 1;
            %Mean background value (HU) is subtracted from threshold (HU)
            ythresh = THRESH_HU - meanROI_bk{1}(:);
            %filenumber_idx
        end


        %Narrow window to ROI

        %Specify the region, in the center of the profile, in which to keep
        %peaks that have been detected.  This feature helps to reduce the
        %probability that noise will be detected and thought to be inside
        %the sample.
```

```
        %Convert physical size in mm to pixels and expand region by the
        %window scaling factor |percentofwindow|.
        windowsizepx = (physsizemm(filenumber_idx)/...
            dicom_mmperpix(filenumber_idx))*percentofwindow;
        centerx = [];

        [FWHM_SingleFileList, keepcrossing, peakskeept, windowpts,...
            firstptsx, threshold, ditch, windowsizepx,...
            errmsg, errid, errst] = ...
            fwpm1a(Ihu, Pav, ythresh, windowsizepx, centerx, 'linear');

        %Main Output variable
        fwatpercentmaxlist(angle,filenumber_idx) = FWHM_SingleFileList;

        %Archive Variables for Graphing


        fwthreshold(angle,filenumber_idx)        = threshold;
        fwwindowsize(angle,filenumber_idx)       = windowsizepx;
        fwwindowpts(angle,[1,2],filenumber_idx)  = windowpts;

        fwcoordinates(angle,filenumber_idx)      = {keepcrossing};
        fwpeaks(angle,filenumber_idx)            = {peakskeept};
        fwfirstpts(angle,filenumber_idx)         = {firstptsx};


        %Archive Errors
        ditchstatus(angle,filenumber_idx)        = ditch;
        fwpm_errmsg(angle,filenumber_idx)        = {errmsg};
        fwpm_errid(angle,filenumber_idx)         = {errid};
        fwpm_errst(angle,filenumber_idx)         = {errst};

    end %FWHM collected for all angles for a file

    %***--- FWHM (half max) of Average Profile ---***%

    %Get Average signal for each sample over all profile angles.
    allprofiles_sample_hu_sub_av = permute(cell2mat(cellfun(@(images) ...
        mean(images,2),allprofiles_sample_hu_sub,'UniformOutput',0)),...
        [1 4 2 3]);

    %Get peaks from averaged profile set
    yav = allprofiles_sample_hu_sub_av(:,:,1,1); %for mspeaks
    peaks_av(:,:) = mspeaks(x,yav,'DENOISING',false,...
        'OverSegmentationFilter',peakdist,'HeightFilter',0);

    %Select peak set for file number
    peaksprofile_av = peaks_av{filenumber_idx,1}(:,2);

    %Find maximum peak value and pixel location
    [maxpeak_val_av,maxpeak_index_av] = max(peaksprofile_av);

    %Set minimum value of peak height for peaks to be kept
    threshold_av = maxpeak_val_av*PERCENTMAX;

    %Find the row-index of thoes peaks that meat the minimum height
```

```
    %requirements and round floating point values of peak pixel locations
    %to ensure mspeaks correlates with integer peak location values.
    [rowpk_av] = find(peaksprofile_av>threshold);
    x_index_av = round(peaks_av{filenumber_idx}(rowpk_av,1)); %pixel loc.

    %Get the intensity values of those peaks
    selectedpeak_val_av = peaksprofile_av(peaksprofile_av>threshold);

    %Convert peak data to format that the fwhm.m file expects
    %col. headings: Peak#, index (x), Y(value), fwhm detected
    numberopeaks_av = length(selectedpeak_val_av);
    Pav_av = zeros(numberopeaks_av, 4); %Setup |Pav_av| structure

    try
        Pav_av(:,1) = 1:1:numberopeaks_av; %Peak number
    catch
        Pav_av(:,1) = 1:1:1;
    end

    try
        Pav_av(:,2) = x_index_av; %Index in whole profile
        Pav_av(:,3) = selectedpeak_val_av; %Peak heights (intensity)
    catch
        if isempty(x_index_av) || isempty(selectedpeak_val_av)
            Pav_av(:,:) = [];
        end
    end

    %Prepare profile for fwpm function
    ihu_av = allprofiles_sample_hu_sub_av(:,filenumber_idx);

    %Prepare physical dimensions for fwpm function
    windowsizepx_av = (physsizemm(filenumber_idx)/...
        dicom_mmperpix(filenumber_idx))*percentofwindow;

    %%% ** Obselete ** %%%
    %    ideal = expectedmm/dicom_mmperpix(filenumber_idx); obselete
    %    widthonasidepx = (windowsizepx/2);
    %    widthonasidemm = (windowsizepx/2)*dicom_mmperpix(filenumber_idx);
    %%% ** Obselete ** %%%

    %ABOVE: boundary of measurement in mm
    centerx_av = [];

    [FWHM_SingleFileList_av, keepcrossing_av, peakskeept_av,...
        windowpts_av, firstptsx_av, threshold_av, ditch_av,...
        windowsizepx_av, errmsg_av, errid_av, errst_av] = ...
        fwpm1a(ihu_av, Pav_av, threshold_av, windowsizepx_av, centerx_av,...
        'linear');

    %Main Output Variable
    fwatpercentmaxlist_av(filenumber_idx) = FWHM_SingleFileList;

    %Archive Variables for Graphing
    %Fix Me: if threshold_av or peakskeept_av is [] error.
    fwthreshold_av(filenumber_idx)          = threshold_av;
    fwwindowsize_av(filenumber_idx)         = windowsizepx_av;
```

```
    fwwindowpts_av(filenumber_idx,[1,2])    = windowpts_av;

    fwcoordinates_av(filenumber_idx)        = {keepcrossing_av};
    fwpeaks_av(filenumber_idx)              = {peakskeept_av};

    %Archive Errors
    ditchstatus_av(filenumber_idx)          = ditch_av;

    fwpm_errmsg_av(filenumber_idx)          = {errmsg_av};
    fwpm_errid_av(filenumber_idx)           = {errid_av};
    fwpm_errst_av(filenumber_idx)           = {errst_av};

end %FWPM values for signal and average signal have been collected

sizeofFWHMList = size(fwatpercentmaxlist);
sizeofAVfwhmlist = size(fwatpercentmaxlist_av);

save(savemispath,'-append','errmsg_av','errid_av','errst_av');
save(savemispath,'-append','errmsg','errid','errst');
```

8.0.x Get FWPM Statistics

```
1. FWH-Mean Statistics (FWH-Mean averages etc...)
```

```
display('8.0.x Get FWPM Statistics')
```

8.0.1 FWHM Statistics (FWPM_AV etc..)

```
display('8.0.1 FWHM Statistics (FWPM_AV etc..)')

load(save_misc,'stdROI')
%Look at rows of |fwatpercentmaxlist| and gather statistics. Format:
%filenumber_idx, filenumber_orig, lable matrix (date indicator),
%number of fwpm valid calc,mean, std., std. error, max, min,

%Preprossing step to help filter out bad fwpm calculations whicha are
%automatically set to 0.
numberlist = 1:1:length(testfilepaths);
[ra,ca,va] = find(fwatpercentmaxlist);
colsizenonzero = arrayfun(@(count) length(find(ca == count)),numberlist)';

%Create a matrix storing statistics data from fwhm.m in dimensions of pixel
fwstat_pix(:,1)  = 1:1:length(testfilepaths);%file number_index first col
fwstat_pix(:,2)  = keyfile_keept(:,6); %file number in original keyfile
fwstat_pix(:,3)  = keyfile_keept(:,3);% label matrix (denotes date of study)
fwstat_pix(:,4)  = colsizenonzero; %number of valid fwpm calculations
fwstat_pix(:,5)  = arrayfun(@(count) mean(fwatpercentmaxlist(...
    fwatpercentmaxlist(:,count)~=0,count)), numberlist); %mean for non 0 values
%Standard deviation is computed using default method. (matrix,2,1) is the
%code to use the second definition (normalizing to n).
fwstat_pix(:,6)  = arrayfun(@(count) std(fwatpercentmaxlist(...
    fwatpercentmaxlist(:,count)~=0,count),1,1), numberlist); %std. (n-1) version
fwstat_pix(:,7)  = fwstat_pix(:,6)./sqrt(colsizenonzero); %standard error
fwstat_pix(:,8)  = max(fwatpercentmaxlist);% max of columns (default)
fwstat_pix(:,9)  = min(fwatpercentmaxlist);% min of columns (default)
```

```
fwstat_pix(:,10) = cell2mat(meanROI);
load(savemispath,'stderrorROI','stderrorROI_bk','stdROI_bk','meanROI_bk');
fwstat_pix(:,11) = cell2mat(stdROI);
fwstat_pix(:,12) = cell2mat(stderrorROI);
fwstat_pix(:,13) = cell2mat(meanROI_bk);
fwstat_pix(:,14) = cell2mat(stdROI_bk);
fwstat_pix(:,15) = cell2mat(stderrorROI_bk);


%fwstat_pix(:,7) = ;% %>3std. %are these data points greater than 3 std.
%outlier detection

%Save fwpm in pixels to file
savepathtable = [save_dir save_table '.csv'];
%headera = 'File Number','Label','Samp. Size','Mean','std','std error',...
%'max','min';
%flmwrite(savepathtable, headera,'-ascii');
csvwrite(savepathtable, fwstat_pix,1,0);

%Create version of |fwstat_pix| that is in mm rather than pixels.
%TODO: check for need to do error propagation on multip.
fwstat_pix_real=fwstat_pix*dicom_mmperpix(filenumber_idx); %converts pixels to mm
%Fix data that wasn't in units of pixels to begin with
fwstat_pix_real(:,1)  = 1:1:length(testfilepaths);%fixes file number's in col1
fwstat_pix_real(:,2)  = keyfile_keept(:,6);
fwstat_pix_real(:,3)  = keyfile_keept(:,3);
fwstat_pix_real(:,4)  = colsizenonzero; %number of valid fwpm calculations
fwstat_pix_real(:,10)   = fwstat_pix(:,10);
fwstat_pix_real(:,11)   = fwstat_pix(:,11);
fwstat_pix_real(:,12)   = fwstat_pix(:,12);
fwstat_pix_real(:,13)   = fwstat_pix(:,13);
fwstat_pix_real(:,14)   = fwstat_pix(:,14);
fwstat_pix_real(:,15)   = fwstat_pix(:,15);


%Save fwpm in mm to file
savepathfwhmmm = [save_dir save_fwhm_mm '.csv'];
%headera = 'File Number','Label','Samp. Size','Mean','std','std error',...
%'max','min';
%flmwrite(savepath, headera,'-ascii');
csvwrite(savepathfwhmmm, fwstat_pix_real,1,0);

%Create matrix storing averaged profiles' FWPM data in Pixels
fwstat_pix_av(:,4) = (fwatpercentmaxlist_av(1,:))';
fwstat_pix_av(:,1) = 1:1:length(testfilepaths);
fwstat_pix_av(:,2) = keyfile_keept(:,6); %file number in original keyfile
fwstat_pix_av(:,3) = keyfile_keept(:,3);% label matrix added (denotes date of study)
%Create matrix storing averaged profiles' FWPM data in mm
fwstat_pix_av_real(:,4) = (fwatpercentmaxlist_av(1,:)'.*dicom_mmperpix(filenumber_idx))
;
fwstat_pix_av_real(:,1) = 1:1:length(testfilepaths);
fwstat_pix_av_real(:,2) = keyfile_keept(:,6); %file number in original keyfile
fwstat_pix_av_real(:,3) = keyfile_keept(:,3);% label matrix added (denotes date of stud
y)

%Save keyfile_kept to disk
savepathkey_keept = [save_dir save_keeptkey '.csv'];
```

```
csvwrite(savepathkey_keept, keyfile_keept,1,0);
```

9.0.x Prepare Data for Display

1. Preallocate Variables
2. Reorganize Statistics Data by Date of Acquisition for Graphing

```
display('9.0.x Prepare Data for Display')
```

9.0.1 Preallocate of Variables

```
display('9.0.1 Preallocate of Variables')

fwstat_pix_mask = cell(1,1,max(fwstat_pix(:,3)));
fwstat_pix_masked = cell(1,1,max(fwstat_pix(:,3)));
fwstat_pix_real_mask = cell(1,1,max(fwstat_pix(:,3)));
fwstat_pix_real_masked = cell(1,1,max(fwstat_pix(:,3)));
fwstat_pix_av_mask = cell(1,1,max(fwstat_pix(:,3)));
fwstat_pix_av_masked = cell(1,1,max(fwstat_pix(:,3)));
fwstat_pix_av_real_mask = cell(1,1,max(fwstat_pix(:,3)));
fwstat_pix_av_real_masked = cell(1,1,max(fwstat_pix(:,3)));
```

9.0.2 Reorganize Statistics Data by Date of Acquisition for Graphing

Store statistics values for each experimental date ordered in increasing fashion. This cell array contains the same number of pages as the maximum value in the label matrix.

```
display('9.0.2 Reorganize Statistics Data by Date of Acquisition for Graphing')
%Loop through all label numbers: 1,2,3,4..., max; This requires labels to
%be some ordered set of multiples.

%Save FWPM Data in Pixel Form
for naa = 1:1:max(fwstat_pix(:,3))
    %Use cell's to store data because there may be a different number of
    %samples collected for each date.
    fwstat_pix_mask(1,1,naa) = {repmat((fwstat_pix(:,3)==naa),[1,size(fwstat_pix,3)])};
 %Construct mask
    fwstat_pix_masked(1,1,naa) = {fwstat_pix(any(fwstat_pix_mask{1,1,naa},3),:)}; %Save
 ordered data
end

%Save FWPM Data in mm Form
for nbb = 1:1:max(fwstat_pix(:,3))
    fwstat_pix_real_mask(1,1,nbb) = {repmat((fwstat_pix_real(:,3)==nbb),[1,size(fwstat_
pix_real,3)])}; %Construct mask
    fwstat_pix_real_masked(1,1,nbb) = {fwstat_pix_real(any(fwstat_pix_mask{1,1,nbb},3),
:)}; %Save ordered data
end
%Save FWPM of Averaged Profiles in Pixel Form
for ncc = 1:1:max(fwstat_pix(:,3))
    fwstat_pix_av_mask(1,1,ncc) = {repmat((fwstat_pix_av_real(:,3)==ncc),[1,size(fwstat
_pix_av_real,3)])}; %Construct mask
    fwstat_pix_av_masked(1,1,ncc) = {fwstat_pix_av_real(any(fwstat_pix_av_real_mask{1,1
```

```
,ncc},3),:)}; %Save ordered data
end
%Save Version of Averaged Profiles in mm Form
for ndd = 1:1:max(fwstat_pix(:,3))
    fwstat_pix_av_real_mask(1,1,ndd) = {repmat((fwstat_pix_av_real(:,3)==ndd),[1,size(f
wstat_pix_av_real,3)])}; %Construct mask
    fwstat_pix_av_real_masked(1,1,ndd) = {fwstat_pix_av_real(any(fwstat_pix_av_real_mas
k{1,1,ndd},3),:)}; %Save ordered data
end
```

10.0.x Clean Up Leftover Variables

1. Save All Leftover Variables
2. End of Data Acquisition Section (Stop Timers)

```
display('10.0.x Clean Up Leftover Variables')
```

10.0.1 Save All Leftover Variables

```
%Save all leftover variables to the miscellaneous file
save(savemispath,'-append');
```

10.0.2 End of Data Acquisition Section

```
toc %stop timer
display('end');
%Note
%meanROI = cell2mat(meanROI);
%stdROI = cell2mat(stdROI);
```

11.0.x Display Quick Data Summary

```
%Display FWPM Measurements as Surface Plot
figure,surf(fwatpercentmaxlist)

%%%%%%%% The End: See Presentation Scripts
```

Variables Left In Memory Summary

```
totalmem_ed (whos);
```

Published with MATLAB® 7.4

Contents


```
function [centroidxb, centroidyb, Eqdiama, Bound, L, numb, E, O] = regioncenterdetect(t
estfilelist,filenumber,bwlevela, physdiam_pix)
%regioncenterdetect attempts two image segmentations using repeated erosion
%   regioncenterdetect is a function that accepts a list of image file
%   paths, a file number, set of threshold values, and the physical
%   diameter of the sample to be segmented.
%
%   OUTPUT: centroid x&y, eq. diameter(pix), # ROI found, eccentricity and
%   orientation.  Also labelmatricies and boundaries can be
%   exported, but have not been thoroughly tested.
%
%   INPUT:          path list, file # selection, set of BW conversion
%   levels, diameter of the physical object.
%
%  Method:
%           image conversion to BW (user specified threshold)
%           two erosion steps
%
%           *Adjust BWlevel setting or pix size assumptions until function
%           gives desired results.
%
%   Future Development Considerations:
%           morphological opening when used with an appropriate sized
%           structuring element may be desired in attempt to preserve
%           region size
%
%   See also erosion, dilation, imopen, imclose
%
%
%   Author: Eric T. Dick
%   Created:5/24/07
```

Bug Detection

```
%{
close all
testfilelist = noisefilepaths
filenumber = 1
bwlevela =[35,200]
%}
```

Initialize variables

```
Eqdiama = [];
E =[];
O = [];
centroidxb = [];
centroidyb = [];
L = [];
numb = [];
Bound = {0};
norun = 0;
```

Start Loop (detect big obj, detect center)

```
%Calculate area from expected diameter of the object (in pixels)
physarea_pix = ceil(pi*(physdiam_pix(filenumber)/2)^2);

%Set first pass area minimum to be 100 pixels^2.
bigsize    = 100;

%Scale second pass area minimum with expected sample area.
littlesize = ceil(physarea_pix * 0.1);
```

Import Dicom Files and Information

```
file             = testfilelist{filenumber};
ImageThrown      = dicomread(file);
ImageThrown_Info = dicominfo(file);
offset           = abs(ImageThrown_Info.RescaleIntercept);

%Setup segemetnation tool.
se = strel('disk',2);

%figure, imshow(ImageThrown,[0,2200])
```

Loop Over the Number of Regions Desired

```
%ab = 1 looks for sample holder
%ab = 2 looks for sample

for ab = 1:1:2

    bwlevel = bwlevela(ab);

    %Calculate the level for grayscale to BW conversion.
    %Use the DICOM metadata to determine the integer to HU units
    %conversion.
    bwlevel = offset + bwlevel;

    %Convert image type to double from unit16.
    %Note: Matlab uses a +1 shift in color-mapping, so to preserve true
    %pixel color data one needs to adjust by +1.
    level = (bwlevel + 1) / 2^16;

    %% Convert Image to BW
    BW = im2bw(ImageThrown, level);
```

```
    %figure, imshow(BW) %Show BW Image
    %figure, imshow(cropI,[0,2000]) %Optionally Crop Image

    %%Start Image Segmentation Routine

    %Erode the Image
    I2 = imerode(BW,se); %opposit of dialate
    %figure, imshow(I2);

    %Fill the Image Holes
    BWn = imfill(I2,'holes');
    %figure,imshow(BWn)

    %Dilate the Image
    I3 = imdilate(~BWn,se); %Same as eroding

    %figure,imshow(I3)

    %Remove Objects < User Specified Area in Pixels
    if ab==1
        %<100 pixels in size
        %bigsize = 100;
        close1 = bwareaopen(~I3, bigsize);
        %figure,imshow(close1);
    else
        %<50 pixels in size
        %littlesize = 50;
        close1 = bwareaopen(~I3, littlesize);
        %figure,imshow(close1);
    end

    %% Discover Object Boundaries

    % Create a label list of objects in image.
    [B,L,N] = bwboundaries(close1); %,'noholes'

    %B is an array of cells which have a 2 column matrix specifying
    %coordinates of the boundary pixels.
    %The first N cells of B are object boundaries, the rest are holes.
    %L is a label matrix the same size as the image that indicates all the
    %regions and numbers them.

    if iscell(B) ~= 1 && ab == 2
        %It seems possible for B to be exported as a non cell object if
        %only 1 boundary is found.  Since the code depending on this
        %function assumes it is getting a cell function we need to make
        %one.
        B(1,1) = {B};
    end

    %% Debugging (Display Boundaries and Label Matrix)
    %{
        figure, imshow(ImageThrown,[1000 2000])
        %figure, imshow(dicomdoublehubk,[0 1000])
        figure,FillSpace = imshow(label2rgb(L, @jet, [.5 .5 .5]));

        hold on
```

```
        % Determine boundaries of objects.
        for k = 1:length(B)
            boundary = B{k};
            %plot(boundary(:,2),boundary(:,1),'w','LineWidth',2)
        end

    %}

    %% Region Statistics / Calculate Centroid Etc..
    stats = regionprops(L,'Area','Centroid','Eccentricity',...
        'Orientation','EquivDiameter','Image');

    %Concatenate multiple entries as [x1,y1; x2, y2]. |var(row, column)|
    %This makes temporary vectors of info. for all labeled items
    %Each row is a different labeled item (ROI).
    Centroid            = cat(1, stats.Centroid);
    EquivalentDiameter  = cat(1, stats.EquivDiameter);
    Eccentricity        = cat(1, stats.Eccentricity);
    Orientation         = cat(1, stats.Orientation);

    %Second loop condition...
    if size(Centroid,1) == 1 && ab == 2 && Centroid(1,1) == centroidxb && Centroid(1,2)
== centroidyb
        %Don't record data set in second loop if only one region is found
        %and it has the same centroid as the one found in the preceding
        %loop, because it is very likely that no unique second region was
        %detected.
        %This if statement allows for 1 ROI detect on first run and 0 on
        %second.
        Centroid = [];

        %Record number of regions found in second run.
        numb     = [numb, N];
        norun    = 1;
    end
```

If a unique centroid was detected then store output

```
    if ~isempty(Centroid) && ~logical(norun)

        %Second Loop Condition
        if size(Centroid,1) == 1 && ab == 2
            %Note: we choose to keep the second region even if it closely
            %resembles the first as long as it is not the first.

            %Set region detection to 1, start loop over again.
            ab = 1;
        end

        %Store centroid data for reporting out of the loop.
        centroidxb  = [centroidxb Centroid(ab,1)];
        centroidyb  = [centroidyb Centroid(ab,2)];

        %Store shape data for reporting out of the loop.
        Eqdiama     = [Eqdiama, EquivalentDiameter(ab)];
```

```
        E           = [E, Eccentricity(ab)];
        O           = [O, Orientation(ab)];

        %Stores number of detected regions (runs are columns)
        numb = [numb, N];

        %Store all B's but not redundantly!

        %There could be a dimension mismatch if we tried to use |horzcat|
        %in Bound and B. So, we want to add B manually to the record
        %Boundary horizontally.
        lastline = size(Bound,1);
        numberofnewlines = size(B,1);

        for ind = 1:numberofnewlines
            nextline = lastline + ind;
            Bound(nextline,:) = B(ind,1);
        end

    elseif isempty(Centroid) == 1
        numb = [numb, 0];
    end

end

end
```

Published with MATLAB® 7.4

Contents


```
function [varargout] = fwpm1a(varargin)
% fwpm1a performs an interpolated threshold measurement on a sample object
%   fwpm1a relies on input from a peak detection function such as |mspeaks|
%   in the bioinformatics toolbox.  The peaks serve as a reference from
%   which to find the sample edge.  Samples are assumed to be centered in
%   the profile.
%
%  This function is designed to determine the widths of a data set above a
%  certain threshold value, determined by percent-of-mean * meanval.
%  This function accepts a peak combine feature that allows the
%  user to specify a window in which any peaks that are detected are
%  considered one unit, all other peaks are ignored.
%
% Input:
%   ythresh         threshold level for y (used for constant threshold over
%                   all theta, otherwise need maxpeak*percent-peak change
%                       to function.
%   peakdist        specifies how many pixels mspeaks uses on either side
%                   to determine if 2 peaks should be counted as the same.
%   combinewidth    specify combinewidth in pixels.
%   peaks           Peak#, Index(x), Y(intensity value): as columns.
%   trackback       # of Pixels before and after crossing values to use in
%                       interpolation.
%
% Output:
%   fwpm            the calculated width in pixels
%   fwpmx / fwpmy   the coordinates of the interpolated points col(lft,rit)
%   leftptabovex    first pt. above level (left crossing) x value
%   leftptabovey    first pt. above level (left crossing) y value
%   leftptabovewinx first point in window above level (l.c) x value
%   leftptabovewiny (above)
%
% Requirements:
%   The data must have peaks of interest with points going above and below
%   the threshold truncation level (Y) or they will be ignored.
%
%   The profiles should be centered over the feature of interest to
%   use the combine window feature properly.
%
%   At present the function does not tolerate missing parameters, make sure
%   you at lease specify the parameter even if it's [].
%
%   Peaks input must have the following format (matrix with columns):
%   Peak#, index (x), Y(value).  Peaks need to be sorted in x value from
%   minimum value down to maximum value.
```

```
%
% Special Note:
%   If the function returns a value of 0 then the level was either too high
%   or the entire data set's width was calculated (typically when signal is
%   so low that the level is well in the noise).
%
% Note: All fwpm values are calculated in pixels
%
%  Authorship:
%  Created by: Eric Dick                    Start Date: July 30, 2007
%  Status: Beta                             Revision  :
%%%%%%%%%%%%%%%%%%%%%%%%%%% End of Introduction %%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Prepare to Begin Function

```
lasterror('reset')
nPrimaryOutPuts = nargout;
nPrimaryInPuts  = nargin;
```

Check Input

```
parseinput;
defaultscheck;

try
    flagbaddata;

    %% Get First Points Above Level for Each Peak
    %Take the difference of adjacent elements in the profile data.
    %Get X indices for first point on either side of a peak; peak must be
    %identified as dipping below the intensity threshold on both sides.

    %Proceed only if there are no errors.

    leftptabovex  = find(diff(profiledata >= ythresh) > 0)+1;
    rightptabovex = find(diff(profiledata >= ythresh) < 0);
    %TODO: if only one value is found, jump straight to the interpolation
    %method, as a speed enhancement.
    if isempty(leftptabovex) || isempty(rightptabovex)
        error('leftptabovex or rightptabovex is an emptyset');
    else
        try
            %Trim off partial peaks at both ends and collect intensity
            %values for selected points.
            if rightptabovex(1) < leftptabovex(1);
                rightptabovex(1) = [];
                rightptabovey = profiledata(rightptabovex);
            end

            if leftptabovex(end) > rightptabovex(end);
                leftptabovex(end) = [];
                leftptabovey  = profiledata(leftptabovex);
            end

        catch
```

```
            error(['Tried to trim rightptabovex or leftptabovex'...
                'and failed']);
        end
    end

    %% Combine Peaks Over Peak Collection Window

    %If peaks kept window is unspecified skip this section,(i.e. don't use
    %this feature).
    if windowsizepx ~= 0 && ~isempty(windowsizepx)
        %Determine which peaks and crossing points to keep for the calc..
        %Define peak combine window x boundaries
        if windowsizepx >= length(profiledata)
            %If window setting is wider than data set, select whole data
            %set.
            windowleft  = 1;
            windowright = length(profiledata);
        elseif windowsizepx < length(profiledata)
            %If window setting is less than the whole data set, use it.
            windowleft  = centerx - windowsizepx/2;
            windowright = centerx + windowsizepx/2;
        end
        %Get indexes of only thoes peaks within the window boundaries and
        %above the threshold yvalue.
        peakkeepindx = peaksed(...
            (find(peaksed(:,2)>= windowleft & peaksed(:,2)...
            <= windowright & peaksed(:,3) >= ythresh)),2);
        if isempty(peakkeepindx)
            %No peaks were kept, abort calculation.
            error('No peaks were kept in |peakkeptindx|');
        else
            %Get index of left and right "above crossing points" that are
            %to the outside of the peaks even if they are beyond the window
            %boundaries.
            if peakkeepindx(end)>rightptabovex(end)
                %Check if there is a valid right sided crossing point
                %after the last kept peak.  If not then there is no way to
                %determine the right edge.  It should either be a failed
                %attempt or the outer edge of the profile.  It can be shown
                %that in pure noise signals it is possible to have several
                %peaks, but no valley low enough to dip past the threshold
                %of measurement.  Thus, we choose to ditch the calculation
                %attempt.
                error (['Last kept peak has no threshold'...
                    'crossing on right side.']);
            end
            %get x location of farthest left and right above crossing points
            leftmin= leftptabovex(find(leftptabovex <= peakkeepindx(1),...
                1,'last'));
            rightmax = rightptabovex(find(rightptabovex >= ...
                peakkeepindx(end),1,'first'));

            %If there is no |leftmin| found then the data set does not dip
            %below the threshold value before the first kept peak.
            if isempty(leftmin)
                error (['The left most crossing point is to'...
                        'the right of the first peak inside the peak'...
```

```
                    'collection window.']);
                %TODO: modify rightmax and leftmin to have the logical
                %test separate and then modify these if statements to
                %depend directly on that logical test.  This would
                %support the use here of the error message pretending
                %to know what the problem is.  Because there it would.
            end

            if isempty(rightmax) % ---> Bc. mspeaks rounding err?
                error (['The right most crossing point is to'...
                    'the left of the last peak inside the peak'...
                    'collection window.']);
                %TODO: modify rightmax and leftmin to have the logical
                %test separate and then modify these if statements to
                %depend directly on that logical test.  This would
                %support the use here of the error message pretending
                %to know what the problem is.  Because there it would.
            end
            %Toss out left crossing points that go beyond the right window,
            %i.e. that are beyond the right most kept peak.
            %I don't think this can ever happen anyway, by definition.
            %TODO: remove this section of code and change variable names.
            leftptabovewinx = leftptabovex(find(leftptabovex >=...
                leftmin & leftptabovex <= windowright));
            %Keep the right crossing points between the first kept peak
            %(on left) and the right most crossing point beyond the last
            %peak).  Include rightcrossing points that are end points.
            rightptabovewinx = rightptabovex(find(rightptabovex >= ...
                peakkeepindx(1,1) & rightptabovex <= rightmax));

            leftptabovewiny = profiledata(leftptabovewinx);
            rightptabovewiny = profiledata(rightptabovewinx);
            %Note: so these are trimmed versions of the left and right
            %crossing points.  The strange thing is that at this point I've
            %already identified the desired crossing points, there doesn't
            %seem to be a condition that would allow one of those points to
            %be more than one number. rightmax and leftmin.
        end
    end



    %% Interpolate to Get Crossing Coordinates
    if windowsizepx == 0 || isempty(windowsizepx)
        display('No peaks window specified, feature disabled')
        %TODO: change the values below to correspond with not using the
        %window.
        windowsizepx = 0;
        leftptabovewinx = leftptabovex;
        rightptabovewinx = rightptabovex;
        leftptabovewiny = leftptabovey;
        rightptabovewiny = rightptabovey;
        peakkeepindx = peaksed(:,2);
    end

    if isempty(rightptabovewinx) || isempty(rightptabovex)
        error ('Rightptabovewinx or rightptabovex were empty sets');
```

```
    end
    %Indices of the crossing points (x values)
    lxcrss = leftptabovewinx;
    rxcrss = rightptabovewinx;
    %Crossing points (y values)
    %lycrss = leftptabovewiny;
    %rycrss = rightptabovewiny;
    peaksx = peakkeepindx; %The peaks kept after windowing and thresholding

    %Use farthest left and right threshold crossings.
    %If you don't specify a window you are essentially hoping that your
    %threshold is above all noise.

    %Debugging: if the crossing point is too close to the end of
    %the data trackback less.
    if isempty(lxcrss) || isempty(rxcrss)
        error('lxcrss or rxcrss were empty sets');
    end

    if lxcrss(1) <= trackback
        %If there is a risk of tracking back beyond the profile data set
        %change trackback to minimum (2-1 = 1)
        trackbackl = trackback-1;%Magic number bad! let = absmin == 1
        %TODO: or itterate (-1) until not true or 0
    elseif lxcrss(1) == 1
        %If the left most crossing point is the first in the data set then
        %tracking back is not possible, set to 0.
        trackbackl = 0;
    else
        %Set trackback on left equal to intended parameter
        trackbackl = trackback;
    end

    if length(profiledata) - rxcrss(end) <= trackback
        %If there is a chance that the trackback will go beyond the data
        %set remove one. TODO: itterate to support trackback>1, no magic #
        trackbackr = trackback - 1;
    elseif rxcrss(end) == length(profiledata)

        trackbackr = length(profiledata);
    else
        %Set trackback to the desired value
        trackbackr = trackback;
    end

    %Debugging: if the selected point by trackback is too high
    if profiledata(lxcrss(1)) < profiledata(lxcrss(1)-trackbackl)...
            || profiledata(lxcrss(1)-trackbackl) > ...
            profiledata(lxcrss(1)-trackbackl+1)
        %The way we are using interp1 we need to have y continually
        %increasing!
        trackbackl = 1;

        if profiledata(lxcrss(1)) < profiledata(lxcrss(1)-trackbackl)
            trackbackl = 0;
        end
    end
```

```
    if profiledata(rxcrss(end)) < profiledata(rxcrss(end)+trackbackr) ||...
            profiledata(rxcrss(end)+trackbackr) > ...
            profiledata(rxcrss(end)+ trackbackr-1)
        trackbackr = 1;
        if profiledata(rxcrss(end)) < profiledata(rxcrss(end)+trackbackr)
            trackbackr = 0;
        end
    end

    if strcmp(intrpmethod, 'cubic')
        %The input data must represent a function thus the profile
        %vlaues must be steadily increasing!  Too many trackback data
        %points involved will guarantee violation of this requirement.

        %However, it can be shown that with cubic interpolation at
        %least one point of padding on each side is required to get a
        %good fit relative to cubic interpolation of the whole data
        %set.
        leftminx    = lxcrss(1) - trackbackl;
        %index of interpolation starting point 2pixels over
        rightmaxx   = rxcrss(end) + trackbackr;
        %interpolation ending point to right
        leftpeakx   = peaksx(1); %+ trackbackl;
        rightpeakx  = peaksx(end); %- trackbackr;

    elseif strcmp(intrpmethod, 'linear')
        %Linear interpolation requires only two distinct points on
        %either side of the desired interpolation point.
        leftminx   = lxcrss(1)-1; %index of interpolation starting point 2 pixels over
        rightmaxx  = rxcrss(end)+1 ;%interpolation ending point to right
        leftpeakx  = peaksx(1) ;
        rightpeakx = peaksx(end) ;
    end

%     if  rightmax ~= rightmaxx
%         %Interpolation starting point and ending point are not identical on
%         %the right side, which frankly is ok.
%         %TODO: remove this section of code.
%         error('rightmax and rightmaxx are not equivalent');
%     end


    if trackbackr~=0 && trackbackl~=0
        if leftminx > leftpeakx || rightmax<rightpeakx
            error(['Threshold crossing points are not on the'...
                'correct side of the kept peaks']);
            %
            % * Condition 1: the left crossing point should be on the left
            % * Condition 2: right crossing point should be on the right
            % * Condition 3: both methods for calculation the last right
            % crossing point should be equivalent.  If they aren't then
            % perhaps something like the crossing point not being on the
            % right could be happening.
        end

        %If you have valid data thus far continue processing.
```

```
        if ythresh == profiledata(lxcrss(1))
            lfwpm = lxcrss(1);
        else
            %Works if requested point is in range (not inclusive) or is
            %peak)
            if strcmp(intrpmethod, 'linear')
%                 lfwpmc = arrayfun(@(lm,ylvl,lpkx) interp1(...
%                     profiledata(lm:lpkx),(lm:1:lpkx)',...
%                     ylvl,'linear'),round(leftminx),ythresh,...
%                     round(leftpeakx));
                %by hand
                %The diff function: x(n) - (xn-1)
                if round(leftmin-leftminx)>1
                   error(['LinearInterpolation: more than one point was'...
                       ' selected for interpolation.']);
                end
                slope = diff(profiledata(leftminx:leftmin))/(leftmin-leftminx);
                intercept = profiledata(leftmin)-(slope*leftmin);
                lfwpm = (ythresh-intercept)/slope;
                %difflx = lfwpmc-lfwpm;

            elseif strcmp(intrpmethod, 'cubic')
                lfwpm = arrayfun(@(lm,ylvl,lpkx) interp1(...
                    profiledata(lm:lpkx),(lm:1:lpkx)',...
                    ylvl,'cubic'),round(leftminx),ythresh,...
                    round(leftpeakx));
            end
        end
        if ythresh == profiledata(rxcrss(end))
            rfwpm = rxcrss(end);
        else
            if strcmp(intrpmethod, 'linear')
                %{
               Round command added to deal with floating-point numbers
               from peaks variable.
                 rfwpmc = arrayfun(@(rm,ylvl,rpkx)...
                     interp1(profiledata(rm:-1:rpkx),(rm:-1:rpkx)',...
                     ylvl,'linear'),round(rightmaxx),ythresh,...
                     round(rightpeakx));
                 %}
                if round(rightmaxx-rightmax)> 1
                   error('more than two points were chosen for linear interpolation');
                end
                rslope = diff(profiledata(rightmax:rightmaxx))/(rightmaxx-rightmax);
                rintercept = profiledata(rightmax)-(rslope*rightmax);
                rfwpm = (ythresh-rintercept)/rslope;
                %diffrx = rfwpm-rfwpmh

            elseif strcmp(intrpmethod, 'cubic')
                % Round command added to deal with non hanky Xloc data
                % from peaks variable.
                error(['Cubic interpolation is not yet properly' ...
                'supported the current version interpolates a mirror'...
                'image graph on the right hand side.'])
                rfwpm = arrayfun(@(rm,ylvl,rpkx) interp1(...
                    profiledata(rm:-1:rpkx),(rm:-1:rpkx)',...
                    ylvl,'cubic'),round(rightmaxx),ythresh,...
```

```
                    round(rightpeakx));
            end
        end
        fwpm  = rfwpm - lfwpm;
        fwpmx = [lfwpm, rfwpm]';
        fwpmy = [ythresh,ythresh]';

    end
    if trackbackr == 0 && trackbackl ==0
        error('trackbackr and trackbackl are 0');
    end


    if fwpm == length(profiledata)
        error('Length of detected region is the entire profile length');
    end

    %% If FWP-Mean calculation fails, set outputs to 0 and store error message
catch

    %|fwpm = 0| is used to indicate a loss of data. One could use NaN's
    %instead, but ultimately post processing is likely to require the
    %replacement of NaN's with an actual number.

    ditch = 1; %Binary indicator of error status
    fwpm  = 0;
    fwpmx = 0;
    fwpmy = 0;

    leftptabovewinx  = 0;
    rightptabovewinx = 0;
    leftptabovewiny  = 0;
    rightptabovewiny = 0;

    %Window Coordinates
    windowleft = 0;
    windowright = 0;
end
```

Format Output

```
peakmask  = (find(peaksed(:,2)>= windowleft & peaksed(:,2) <=...
    windowright & peaksed(:,3) >= ythresh));
fwpmpts   = [fwpmx fwpmy];
peakskept = [peaksed(peakmask,1) peaksed(peakmask,2) peaksed(peakmask,3)];
windowpts = [windowleft windowright];
firstptsx = [leftptabovewinx' leftptabovewiny' rightptabovewinx' ...
    rightptabovewiny']; %points of A above threshold & in window are kept
threshold = ythresh;

%Get error log and structure
[errmsg, errid, errst] = errorcode(lasterror);
```

Generate Output Variables for |varargout|

```
format_output
%(fwpm,fwpmpts,peakskept,windowpts,firstptsx,...
%   threshold,ditch,windowsizepx,errorlog,errst,nargout);

%The following are input variables for the nested function |format_output|,
%however they are not needed because they should all be used in the primary
%function and thus will be accessible by all nested functions.
%If you experience an error here you can either specify all inputs or you
%can simply call the variable in the primary function.
%
%(fwpm,fwpmpts,peakskept,windowpts,firstptsx,...
%    threshold,ditch,windowsizepx,errorlog,errst,nargout);
```

Nested Functions

```
    function parseinput

        if nPrimaryInPuts == 6
            profiledata     = varargin{1};
            peaksed         = varargin{2};
            ythresh         = varargin{3};
            windowsizepx    = varargin{4};
            centerx         = varargin{5};
            intrpmethod     = varargin{6};
        else
            error('Invalid number of inputs')
        end

        %Varible initialization for boolean tests
        fwpm = 0;
        leftptabovewinx = 0;

    end

%%%%%%
    function format_output

        tmpout{1}  = fwpm;          %FWHM_SingleFileList
        tmpout{2}  = fwpmpts;       %keepcrossing
        tmpout{3}  = peakskept;    %peakskept
        tmpout{4}  = windowpts;     %windowpts
        tmpout{5}  = firstptsx;     %firstptsx
        tmpout{6}  = threshold;     %threshold
        tmpout{7}  = ditch;         %ditch
        tmpout{8}  = windowsizepx;  %windowsizepx
        tmpout{9}  = errmsg;        %errmsg
        tmpout{10} = errid;         %errid
        tmpout{11} = errst;         %errst

        %|varargout| is within the scope of the outer most function as an
        %output parameter and is thus modifiable via this nested function.
        varargout = cell(nPrimaryOutPuts,1);

        for k = 1:1:nPrimaryOutPuts
            varargout(k) = {tmpout{k}};
        end
```

```
        %%[FWHM_SingleFileList, keepcrossing, peakskept, windowpts,...
        %%  firstptsx, threshold, ditch, windowsizepx,fwpm_errorlog,errst]


    end

%%%%%%
    function defaultscheck()

        %If sample center coordinates are 0 or empty use center of profile.
        if isempty(centerx) || centerx == 0
            centerx = round(length(profiledata)/2);
        end

        %TODO: Remove trackback setting except for nonlinear interpolation
        %Set number of points to involve in interpolation if method is
        %nonlinear.
        trackback = 2;
        ditch = 0; %Code did not error out

    end %defaultscheck

%%%%%%
    function flagbaddata
        %If the profile contains invalid data (NaN's) return error.
        if max(isnan(profiledata)) == 1 || isnan(windowsizepx) == 1
            error('Data contains NaN values, fwpm calculation aborted');

        elseif ythresh > max(profiledata)
            %If level is above all profile data then return error.
            error('Threshold level is larger than all data');

        end

    end % flagnans


%%%%%%%
    function [errmsg, errid, errst] = errorcode(lasterror)
        errorlogtmp = lasterror; %set errorlog to value of last error
        errid       = errorlogtmp.identifier;
        errmsg      = errorlogtmp.message;

        %Return the structure of the error (line etc..)
        errst       = errorlogtmp.stack;
    end

lasterror('reset')
end %fwpm1a
```

Code Archive: Flip Profile about 0 if Sample Region is an Intensity Hole

```
% %If the profile is inverted flip it and retake peak data
% if abs(min(profiledata)) >= max(profiledata)
```

```
%     %Profile is down dominant indicating a water filled sample or contrast
%     %below that of the sample holder.
%     xcoord = (1:1:length(profiledata))';
%     peakdist = 2; default peak seperation minimum (in pixels)
%     profiledata = -profiledata;
%     clear ('peaksed')
%     peaksed = mspeaks(xcoord,profiledata,'DENOISING',false,...
%     'OverSegmentationFilter', peakdist, 'HeightFilter',0);
%     peaksed = [(1:length(peaksed))' peaksed];
% end
```

Published with MATLAB® 7.4

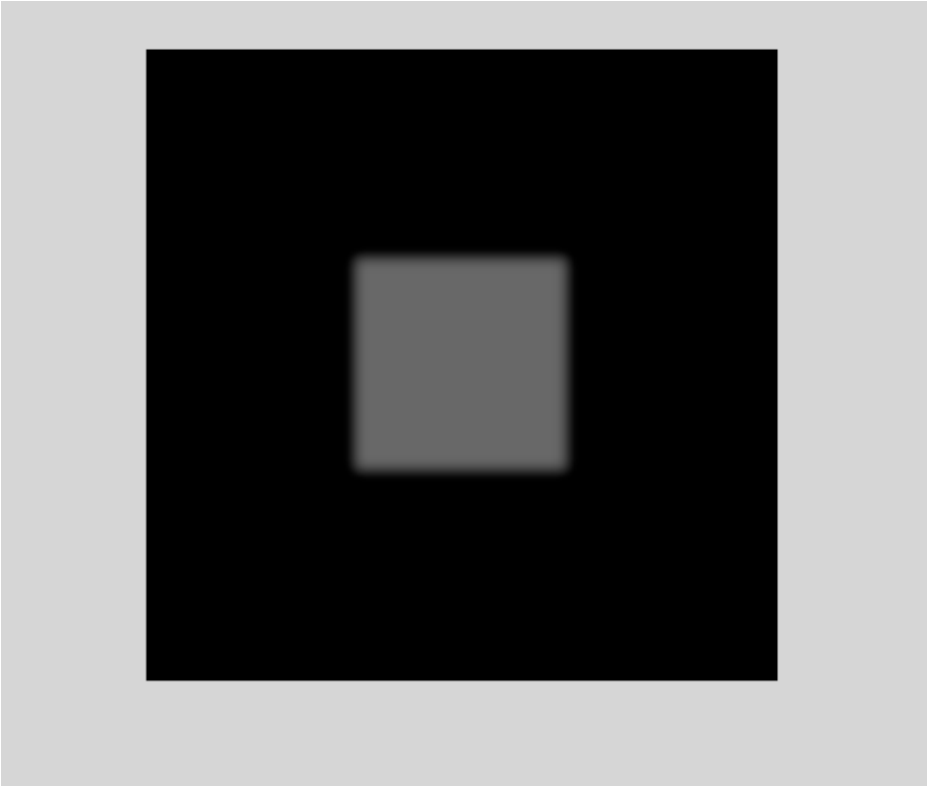
Contents


```
% 2DConvolve_Demo_ED shows demo of the impact of sample intensity on blur
%   Here we give an example of profiles from the binary image of a white box
%   (1) which have been blurred.  Blurring is accomplished using a symmetric
%   2D gaussian kernel. The convolution (conv2) of the blurring kernel and
%   the image is performed, resulting in a slightly larger image (padding).
%   Image registration, via correlation analysis, is performed and only the
%   central portion of the result is keept.  This returns the convolution
%   image essentially cropped back to the size of the image that formed it.
%   The default mode here displays the intensity and threshold dependence of
%   images blured by a model 2D gaussian kernel.
%   One may look at the effects of an additional blur created by a second
%   gaussian function.  This would occur for example with a reconstructed
%   image (modeled by 2D blur) and a physical artifact that scale's with
%   intensity).  In our example the artifact is a blur modeled by a 2D
%   gaussian function.
%
%   Author: Eric Dick 2008/02/10


close all
```

Create our input and smoothed objects

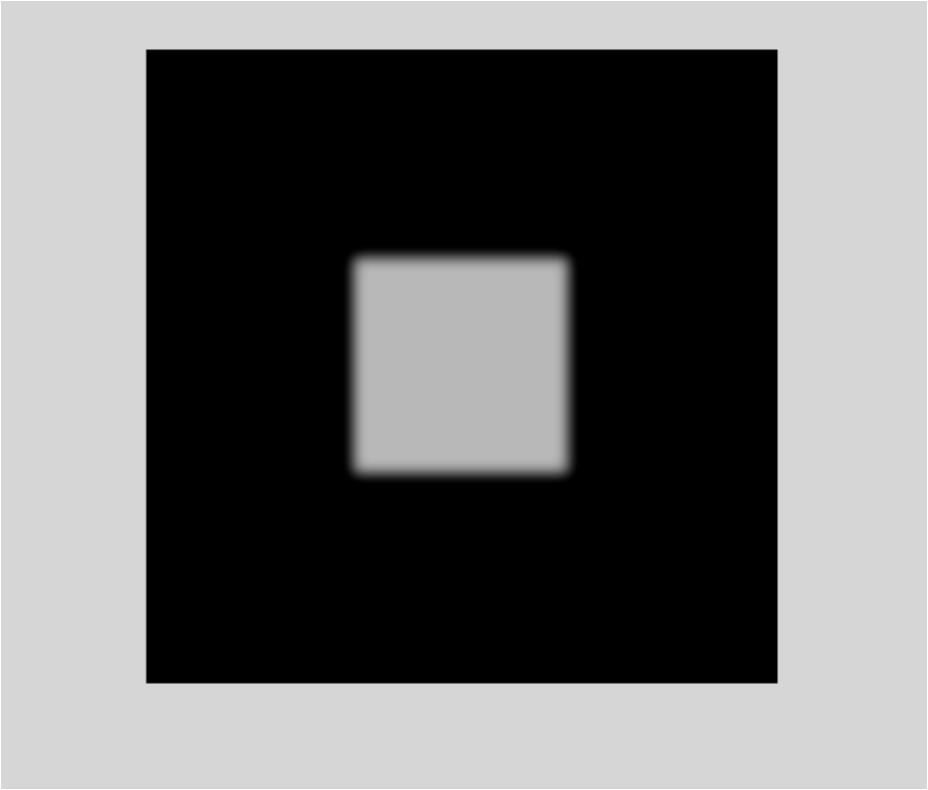
First we establish a 300 by 300 black canvas.

```
bobzeros = zeros(300,300);

% A 200 by 200 white box is centered in the image (white > 0)
yc = 100;
xc = yc;
bobzeros(xc:200,yc:200) = 50;

% A 2D symmetric gaussian kernel image is created.  It is 15 by 15 with a
% SD of 3 pixels.
kernel=fspecial('gaussian', 15, 3);

% We apply the convolution and image registration in one step using the
% filter2 command, which intern uses the 2D convolution command conv2.
smoothedimage1 = filter2(kernel,bobzeros);
imshow(smoothedimage1,[0 150]);
```



We generate a second figure using a brighter square

```
figure,

bobzeros = zeros(300,300);
yc = 100;
xc = yc;
bobzeros(xc:xc+100,yc:yc+100) = 100;
kernel=fspecial('gaussian', 15, 3);
smoothedimage2 = filter2(kernel,bobzeros);
imshow(smoothedimage2,[0 150]);
```

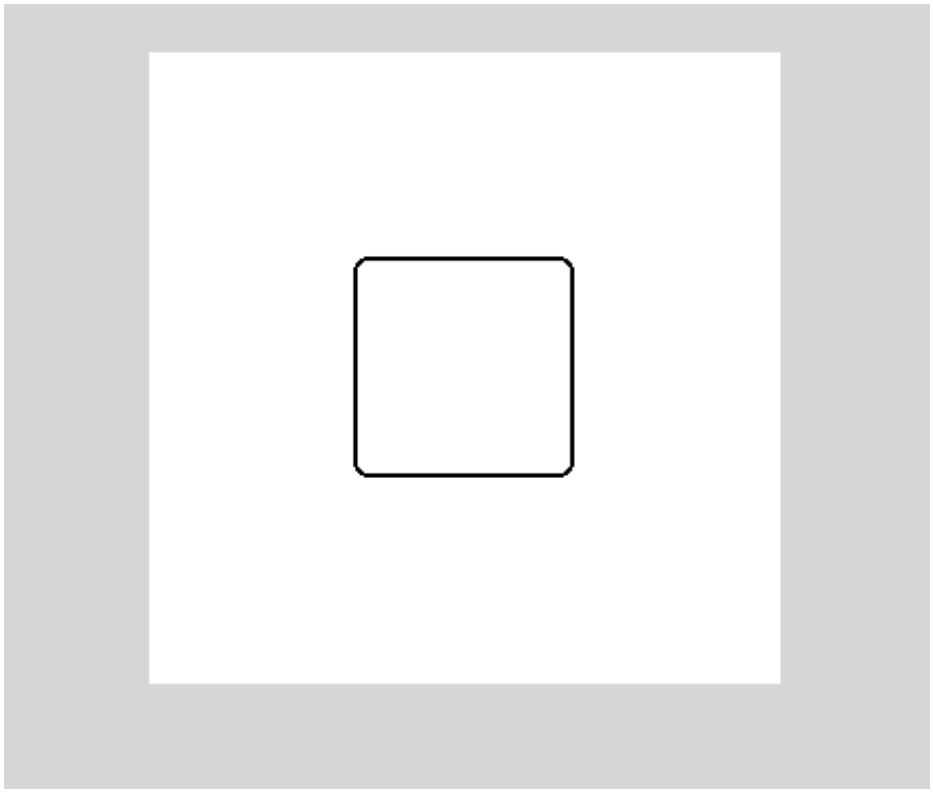


Apply additional artifact option

```
%kernel2=fspecial('gaussian', 15, 3);        %second artifact
%smoothedimage3 = filter2(kernel2,smoothedimage2); %apply second artifact
```

Subtraction images at static and FWH-Mean thresholds are created

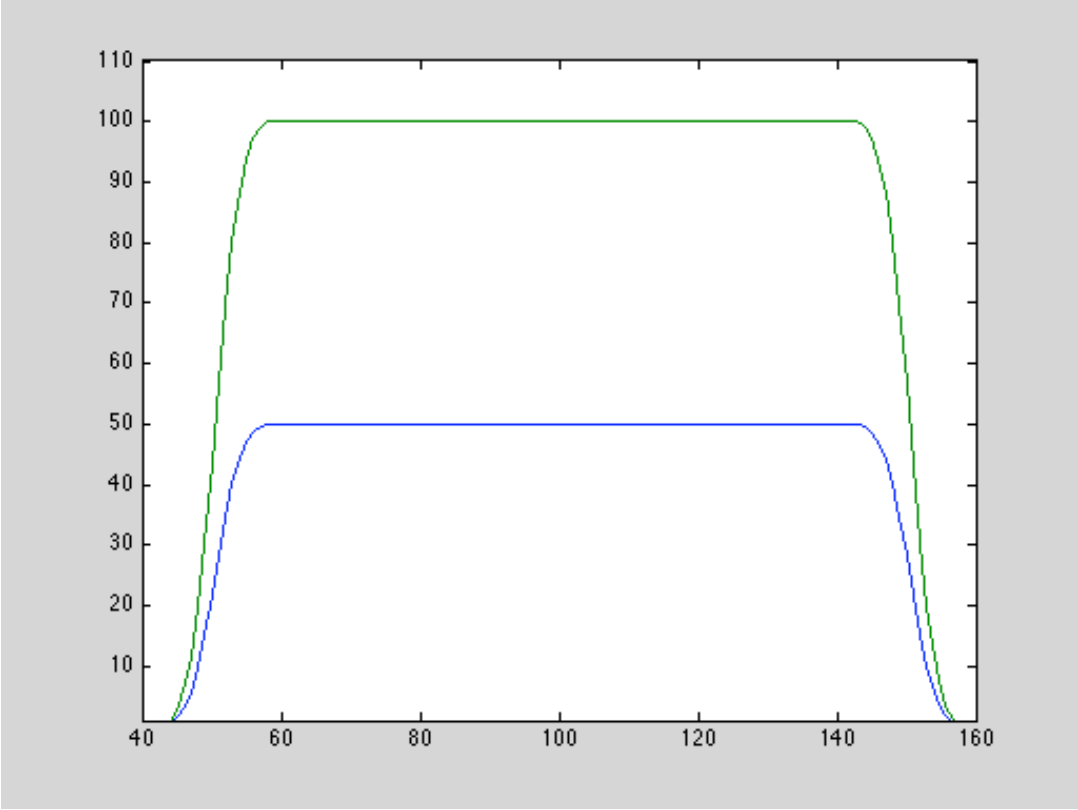
```
%Subtraction images are inverted to use a white background.  The black
%pixels represent pixels in the second image (larger intensity) that are
%not in the first image (smaller intensity), blooming.

figure,
% Using a static threshold
staticThresh = 25;
imshow( ~((smoothedimage2 > staticThresh)-(smoothedimage1 > staticThresh)),[0 1]);

figure,
% Using a FWHM or other percentatge
percnt = 0.5 ;%Percentage modifier

halfMax1 = max(max(smoothedimage1))* percnt;
halfMax2 = max(max(smoothedimage2))* percnt;
imshow( ~((smoothedimage2 > halfMax2)-(smoothedimage1 > halfMax1)),[0 1]);
```



A profile is collected across each image at the same location

```
figure,
c1 = improfile(smoothedimage1, [150 150],[50 250]);
```

```
c2 = improfile(smoothedimage2, [150 150],[50 250]);

c = [c1 c2];

plot(1:length(c), c);

ylim([1 100+10]);
```



Published with MATLAB® 7.4

Contents


```
%%%%%
%
%   Description
%       This script's purpose is to increase the DC level of an
%       elliptical region within a noise file (centered on the sample
%       holder).  The elliptical region is created at the center of the
%       detected sample holder (xcbig and ycbig).  The resultant images
%       are those of an artificial sample of known size and geometry.
%
%   USE:
%       1. Arrange Input Data into Input Directory
%       2. Add KeyFile to Input Directory (can be too big)
%       3. Run CalcNStore.m with QASwitch = 1  QAGenerate = 1
%       4. Run this Script to Create QA Image Files
%       5. Run CalcNStore.m with QASwitch = 1 and QAGenerate = 0
%          to process QA data set.
%
%   TODO:
%       0. Add ability to rotate ellipse
%       1. Add option to allow for the addition of a 'linear' edge blur.
%
%   Example:
%   --------
%     bob = dicomread('/Users/ericd/Desktop/Data/QA_Tmp/Data/StaticEllipse/QA0001_21-Fe
b-2008.dcm');
%     theta1 = 0:pi/180:2*pi;
%     r = pixRadius_diam/2;
%     x =@(theta) r * cos(theta) + xcenter(1);
%     y =@(theta) r * sin(theta) + ycenter(1);
%     imshow(bob,[]);
%     hold on
%     plot(x(theta1),y(theta1));
%%%%%
```

1.0.x Settings

```
1. Settings for Processing Modes and Output Files
2. Settings for Input Path Specification
```

1.0.1 Choose the settings for output image files.

```
%numberoimag = 10 %number of images wanted
mmStepBI_Fine       = 0.01 %B.I. Setp. Incriment A
mmStepBI_Coarse     = 0.10 %B.I. Setp. Incriment B

NumberFine          = 20  %Number of Images to Collect with Fine Step
NumberCoarse        = 20  %Number of Images to Collect with Coarse Step

HUweight            = 350 %Value to scale elliptical region

mmPhysicalSize_diam = 5 %Starting Circle Diameter (mm)

subpathSaveQA       = 'QA_Tmp/Data/StaticEllipse/'

%%%%%%%%
IdxTotalImgs = NumberFine+NumberCoarse;
```

1.0.2 Set Path's to Input Data

```
% Specify Parent Path's
Save_images    = 'Save_tempimages';
Save_misc      = 'Save_mic';

path2data      = '/Users/ericd/Desktop/Data/';
BHRow_dir      = 'BHDeltaSize/Data/';

Save_tmplrgvar = 'TempVariableStorage/';
Save_tmpkey    = 'Save_keyfile_tmp';
Save_keeptkey  = 'Save_keyfile_keept';

Save_path_parent  =...
    '/Users/ericd/Desktop/trunk/MatlabWork/Reports/ThisVersionOutput/';
Save_dir          = [Save_path_parent    BHRow_dir];
Save_dir_lrg      = [path2data BHRow_dir Save_tmplrgvar];

% Specify Saved File Locations with a Variable Name
Savepathkey       = [Save_dir Save_tmpkey   '.csv'];
Savepathkey_keept = [Save_dir Save_keeptkey '.csv'];

Saveimpath        = [Save_dir_lrg Save_images '.mat']; %Image Files
Savemispath       = [Save_dir_lrg Save_misc   '.mat']; %Miscellaneous Files
```

2.0.x Create Static Elliptical Plots Varying Semi-Major Axis

1. Acquire Input Image
2. Create Ellipse Coordinates
3. Draw Binary Ellipses & Fill

```
load(Saveimpath ,'dicomdoublehu_bk');
load(Savemispath,'dicom_mmperpix','testfilepaths','noisefilepaths');
load(Savemispath,'xcsmall','xcbig','ycsmall','ycbig');
```

2.0.1 Acquire Input Image

```
metadataDICOM   = dicominfo(testfilepaths{1});
mmPixSize       = metadataDICOM.PixelSpacing(1);
InterceptDICOM  = metadataDICOM.RescaleIntercept;

ImgBackground_HU   = uint16(dicomdoublehu_bk{1}+abs(InterceptDICOM));
clear dicomdoublehu_bk
```

2.0.2 Calculate Stepping Sizes and End Points (Pixels)

```
%Convert step size from percent blooming to pix. size
% Really we are just converting the step size and not yet adding on the
% original size.
pixStepFine   = mmStepBI_Fine *   mmPhysicalSize_diam / mmPixSize;
pixStepCoarse = mmStepBI_Coarse * mmPhysicalSize_diam / mmPixSize;

%Calculate radius of the perfect circle (in pix. coordinates)
pixRadius_diam = mmPhysicalSize_diam / mmPixSize;

%Calculate starting and ending sizes
pixStartFine = pixRadius_diam;
pixEndFine   = NumberFine*pixStepFine + pixStartFine;

pixStartCoarse = pixEndFine + pixStepCoarse; % First Step Here
pixEndCoarse = (NumberCoarse-1)*pixStepCoarse + pixStartCoarse; % n - 1 steps here
```

2.0.3 Calculate Elliptical Axes Sizes (Pix)

```
%Calculate all sizes of semi-major axis (major axix/2)
semimajorFine   = [pixStartFine:pixStepFine:pixEndFine]/2;
semimajorCoarse = [pixStartCoarse:pixStepCoarse:pixEndCoarse]/2;

%Calculate lists of semi-major and semi-minor values
pixSemimajor = [semimajorFine semimajorCoarse]';
pixSemiminor = repmat(pixRadius_diam/2,1,length(pixSemimajor))';

%Calculate Expected Eccentricity and BI for Each Image
eccen = arrayfun(@(minor,major) sqrt(1- ((minor^2)/(major^2))),...
    pixSemiminor,pixSemimajor)';
BI = arrayfun(@(minor,major) ((major)/(minor))-1,...
    pixSemiminor,pixSemimajor)';

xcenter = repmat(xcbig(1),size(pixSemimajor));
ycenter = repmat(ycbig(1),size(pixSemiminor));
```

2.0.5 Generate Elliptical Images

```
%Convert everything into cell arrays
cellImgBackground = repmat({ImgBackground_HU},[size(pixSemimajor),1]);

cellxcenter       = num2cell(xcenter);
cellycenter       = num2cell(ycenter);
cellpixSemimajor  = num2cell(pixSemimajor);
```

```
cellpixSemiminor  = num2cell(pixSemiminor);

%Generate circular mask function handle
circle_maskImg  = @(xc,yc,smj,smin) full( ellipticalmask([xc, yc, smj, smin],...
    size(cellImgBackground{1}), smj, smin) );
```

```
%Get Binary Ellipses
BinaryEllipse = cellfun(@(xc,yc,smj,smin) circle_maskImg(xc,yc,smj,smin),...
    cellxcenter, cellycenter,...
    cellpixSemimajor, cellpixSemiminor,...
    'UniformOutput',0);

uint16Ellipse = cellfun(@(im) uint16(im), BinaryEllipse,'UniformOutput',0);

%Add binary ellipse image to input image with weighting factor in intg.
%compatible format
 ImgEllipse = cellfun(@(im,elips) (im + (elips*HUweight)),...
     cellImgBackground, uint16Ellipse,'UniformOutput',0);

%BW only
% ImgEllipse = cellfun(@(im,elips) ( (elips*HUweight+1024)),...
%     cellImgBackground, uint16Ellipse,'UniformOutput',0);
```

2.0.6 Convert New Image to DICOM Standard Format

```
savepathQA = [path2data subpathSaveQA];

if isdir(savepathQA) == 0
    mkdir(savepathQA)
end

maxfielid = length(ImgEllipse);

%Create filenames and files
for idxFileNum = 1:1:maxfielid
    if idxFileNum > 0 && idxFileNum < 10
        nameid = ['000' num2str(idxFileNum)];
    elseif idxFileNum >= 10 && idxFileNum < 100
        nameid = ['00' num2str(idxFileNum)];
    elseif idxFileNum > 100 && idxFileNum < 1000
        nameid = ['0' num2str(idxFileNum)];
    elseif idxFileNum > 1000 && idxFileNum < 10000
        nameid = num2str(idxFileNum);
    end

    genfilename = ['QA' nameid '_' date '.dcm']
    path2saveimg{idxFileNum,1} = [savepathQA genfilename];

    %Encode a slice order
    metadataDICOM.InstanceNumber    = idxFileNum;
    CellmetadataDICOM{idxFileNum,1} = metadataDICOM;
end
```

Write Files to Disk

```
    cellfun(@(imgs,filepaths,metadata) dicomwrite(imgs,...
        filepaths, metadata, 'CreateMode', 'copy'),...
        ImgEllipse,path2saveimg,CellmetadataDICOM,'UniformOutput',0);
```

Example QA for Function

bob = bwlabel(uint16Ellipse{1}); bobr = regionprops(bob,'EquivDiam'); bobr
bobr.EquivDiameter*0.3516 %mm Graphed

(bobr.EquivDiameter*0.3516)/5 %equivalent diameter of BW image

fwstat_pix(1,5)/(5/0.3516) %profiling method results

Published with MATLAB® 7.4