THE PRODUCTION OF HYDROXYAPATITE STANDARDS, USED FOR POST CBCT SCAN HOUNSFIELD UNIT CALIBRATION

by

KAVEH VARGHAI

Submitted in partial fulfillment of the requirements for the degree of

Master of Science

Biomedical Engineering

CASE WESTERN RESERVE UNIVERSITY

January 2018

CASE WESTERN RESERVE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

We hereby approve the thesis/dissertation of

Kaveh Varghai

candidate for the degree of **Master of Science**.

Committee Chair

Steven Eppell, Ph.D.

Committee Member

Russell Wang, DDS, MSD

Committee Member

David Wilson, Ph.D.

Date of Defense

August 16, 2017

*We also certify that written approval has been obtained

for any proprietary material contained therein.

Table of Contents

The Production of Hydroxyapatite Standards, Used for Post CBCT Scan Hounsfield Unit Calibration

Abstract

by

KAVEH VARGHAI

We think that we can help ease patients' concerns by altering diagnostic x-rays. Our goal is to use CT information to pre-surgically determine the likelihood of success of an operation. This technique could help expectation setting or, moreover, could help oral physicians make procedural decisions which result in higher rates of implant stability and functionality.

Currently, cone beam computerized tomographies (CBCT) are commonly used in the dental community for morphological assessment of the mandible and maxilla. Previous studies have shown the potential of using a single set of phantoms in the oral cavity, to derive Hounsfield Units (HU). However, due to the polychromatic x-ray source in CBCT scanners, variation in average grey level for a single phantom exists, as a function of location.

This thesis will focus on using common biomaterials, used in biomedical implants, to design and fabricate a set of standards, small enough so that a single set is fixed adjacent to the region of interest (ROI). This set of phantoms would allow dental clinicians to quantitatively assess bone quality by ultimately comparing derived HU to mass density without the need for expensive medical software.

Chapter 1: Introduction

Biomedical Significance

Dental visits are often associated with anxiety and discomfort. Dental procedures can introduce uncertainties in pain level, recovery time and functionality. Our group believes that we can help with patients' expectation by utilizing X-ray imaging that pre-surgically determines the likelihood of a successful operation. Moreover, we can assist oral physicians in making procedural decisions, which result in a higher rate of functionality in patients who suffer from lower bone mineral density (BMD).

Animal studies indicate that subjects with lower BMD levels tend to show worse initial bone/implant interface in the site of a dental implant, and ergo, lower implant stability. [1] Primary dental implant stability, a prerequisite for implant survival, prevents micro-movement and subsequent formation of connective tissue layers between the bone and the implant, thus ensuring bone healing and osseointegration. Biomechanically, three factors determine primary dental implant stability: implant screw type, drilling condition, and quality of bone. [2]–[4] While the drilling conditions depend on the bit diameter and rotational velocity of the drilling instrument, bone quality varies from healthy patients to aforementioned patients suffering from a deficiency in dense cortical bone. [5]–[8]

A previous publication has shown that relative contributions of implants' final insertion torque range from 62% -74% from cortical bone. [9] Therefore, if we operate under the assumption that final insertion torque positively correlates to primary implant stability, we find it crucial for clinicians to have the ability to quantitatively distinguish between areas with high portions of dense cortical bone and region of interest (ROI) with small portions of dense cortical bone.

Use of Cone Beam Computerized Tomography (CBCT)

Due to the high resolution of the CBCT scanners, dentists and oral surgeons heavily rely on these instruments to qualitatively assess bone quality in the mandible and maxilla. However, grey levels from the CBCT scanner are arbitrary and do not allow for accurate bone quality assessment. [10], [11] Without a standardized scale, it is difficult to interpret and compare the grey levels acquired from different machines. On the other hand, Hounsfield units (HU) use a standardized scale for material attenuation comparison across any scan or machine. Therefore, clinicians commonly use HU acquired from a clinical fan beam CAT scanner to quantitatively distinguish density/quality of bone in a ROI. Our group's big picture goal is to use a clinical Cone Beam CT (CBCT) scan to pre-surgically determine the likelihood of a successful dental implantation. This is done by utilizing the HU of the ROI to predict the maximum insertion torque required to insert a specific implant. Moreover, this task requires a method to relate grey levels from a CBCT to HU.

Grey level and HU are commonly related by a water air phantoms calibration. This calibration relates grey levels of water and air to known HU values of water and air. HU values for objects with grey levels in between water and air can be interpolated using a linear function. However, HU values for dense objects (like bone) exceed HU of water by ~2000-3000 units. Moreover, an extrapolation of the water air linear function is not appropriate to relate grey scale level of bone to the HU for bone. Previous groups have successfully related grey level for a bony ROI to calibrated HU by fixing a set of standards with known attenuation coefficients to the roof of the oral cavity. [10], [11] Their calibration requires making a single calibration plot relating grey levels and HU for the entire field of view (FOV). However, our preliminary testing using an Inveon Micro PET/CT scanner revealed a maximum of 19.9% difference in average grey level

for a Hydroxyapatite disc with known uniform mass densities as a function of its location on top of a human hemi-mandible. Moreover, characterization tests done for the Inveon system revealed a 15.7% variation in a single phantom grey level when it was moved ~4.5 cm inside the FOV. While 8.5% of this variation is credited to machine start up inconsistency, the remaining 7.2% is thought to depend on the phantom's xyz location inside the scanner (the report detailing these tests and results is included in Appendix A). The grey level variation observed created a concern that the location of the standards in respect to the ROI affected the calibration curve parameters relating grey levels to HU values. This, in turn, was the motivation behind placing calibration standards adjacent to the ROI throughout our study.

Moreover, due to the polychromatic nature of the X-ray source in the CBCT scanners, beam hardening or exomass problems are common occurrences in CBCT scans. Beam hardening occurs when low-energy X-ray photons are attenuated more easily than high-energy X-ray photons. This results in artifacts known as streaking, cupping, ring, etc. [12] Likewise, exomass results from the X-ray beam hardening by objects outside of the FOV. Our solution to exomass problem is to line the ROI with a small set of standard with known HU values. We hypothesize that doing so will ensure that standards are effected by exomass in the same manner as the ROI and will allow us to generate calibration plots for each ROI to correct any offsets due to exomass.

Commercially manufactured standards with HU values in the desired range are too large to fit along a ROI comparable to a dental implant. Therefore, this thesis focuses on designing and fabricating a set of standards with HU numbers spanning trabecular and cortical bone values and small enough for multiple standards to fit along the ROI.

Preliminary testing of our hypothesis was done by utilizing our standards in two separate scans. The first scan was completed using the Inveon Micro PET/CT scanner (Case Center for Imaging Research) while the second study was performed with a clinical Carestream 9300 CBCT scanner (Case Western Reserve University School of Dental Medicine). In each study, calibrated HU values for a chosen volume of interest (VOI) were compared as a function of standard location in respect to the VOI.

This thesis is divided into three parts. The first part focuses on the calibration standards' specifications and fabrication protocol, while part 2 details the computer programing used to analyze the DICOM data, imaging methods as well as the data analysis methods. Lastly, Part three includes the results and conclusions of the studies mentioned above.

This Thesis also includes a reporting on the discoveries made when characterizing the Inveon Micro PET/CT scanner (Appendix A). Lastly, Appendix B and C includes the Matlab code and standard fabrication protocol.

Reference

[1] G. Giro *et al.*, “Impact of osteoporosis in dental implants: A systematic review,” *World J. Orthop.*, vol. 6, no. 2, pp. 311–315, Mar. 2015.

[2] R. Adell, U. Lekholm, B. Rockler, and P.-I. Brånemark, “A 15-year study of osseointegrated implants in the treatment of the edentulous jaw,” *Int. J. Oral Surg.*, vol. 10, no. 6, pp. 387–416, Jan. 1981.

[3] T. Albrektsson, P. I. Brånemark, H. A. Hansson, and J. Lindström, “Osseointegrated titanium implants. Requirements for ensuring a long-lasting, direct bone-to-implant anchorage in man,” *Acta Orthop. Scand.*, vol. 52, no. 2, pp. 155–170, 1981.

[4] G. A. Zarb and A. Schmitt, “The longitudinal clinical effectiveness of osseointegrated dental implants: the Toronto study. Part III: Problems and complications encountered,” *J. Prosthet. Dent.*, vol. 64, no. 2, pp. 185–194, Aug. 1990.

[5] E. Bałczewska, L. Klimek, and A. Palatyńska-Ulatowska, “SEM analysis of dental enamel morphological structures on the basis of their replicas in patients with systemic calcium deficiency,” *Cent. Eur. J. Biol.*, vol. 4, no. 2, pp. 190–195, Jun. 2009.

[6] A. H. FRIEDLANDER, “The physiology, medical management and oral implications of menopause,” *J. Am. Dent. Assoc.*, vol. 133, no. 1, pp. 73–81, Jan. 2002.

[7] M. R. Norton and C. Gamble, “Bone classification: an objective scale of bone density using the computerized tomography scan,” *Clin. Oral Implants Res.*, vol. 12, no. 1, pp. 79–84, Feb. 2001.

[8] M. Yamazaki *et al.*, “Bone reactions to titanium screw implants in ovariectomized animals,” *Oral Surg. Oral Med. Oral Pathol. Oral Radiol. Endodontology*, vol. 87, no. 4, pp. 411–418, Apr. 1999.

[9] R. Wang, S. J. Eppell, C. Nguyen, and N. Morris, “Relative Contribution of Trabecular and Cortical Bone to Primary Implant Stability: An In Vitro Model Study,” *J. Oral Implantol.*, vol. 42, no. 2, pp. 145–152, Jun. 2015.

[10] P. Mah, T. E. Reeves, and W. D. McDavid, “Deriving Hounsfield units using grey levels in cone beam computed tomography,” *Dentomaxillofacial Radiol.*, vol. 39, no. 6, pp. 323–335, Sep. 2010.

[11] T. Reeves, P. Mah, and W. McDavid, “Deriving Hounsfield units using grey levels in cone beam CT: a clinical application,” *Dentomaxillofacial Radiol.*, vol. 41, no. 6, pp. 500–508, Sep. 2012.

[12] “CT artifacts: Causes and reduction techniques - Semantic Scholar.” [Online]. Available: /paper/CT-artifacts-Causes-and-reduction-techniques-Boas-Fleischmann/916896f8af9acae46c5a9a4c28e51b31d061c0c5. [Accessed: 10-Aug-2017].

Chapter 2: Phantom Specifications and Fabrication

Introduction:

The focus of this chapter is the calibration standards. Here we discuss the materials used, technical specifications of the standards as well the method utilized to determine target mass ratio of material. Moreover, the fabrication protocol is discussed as well as the measurements applied to verify the technical specifications.

Material:

Synthetic hydroxyapatite (HA) as well as high-density polyethylene (HDPE) are commonly utilized in bone substitute application. Patients with orthopedic injures or defects (e.g., osteoporosis) are treated with HA reinforced implants due to their mechanical properties and bioactivity. [1] Moreover, HA particles also contain similar elements to bone apatite. This is a desirable feature because the change in attenuation as a function of X-ray energy is not the same for every material. Therefore, standards made up of similar components as bone ensure that the attenuation of the standards change similarly to the attenuation of bone as a function of X-ray energy. For these reasons, standards in this study were mainly produced using HA and HDPE.

Furthermore, to avoid concerns regarding standards' biocompatibility (cytotoxicity, sensitization, irritation, carcinogenicity, etc.) standards were embedded into a dental X-ray film pouch used in clinical scans (Figure 2.1).

Figure 2.1: Image of dental X-ray film pouch.

Standard Specifications:

As mentioned in the previous chapter, our group's big picture goal is to use calibration standards to relate grey levels from a clinical CBCT to HU values for a ROI in the oral cavity. This task requires the use of standards with HU values ranging from HU of trabecular bone to cortical bone (700-3000). [2], [3] Commercially manufactured phantoms satisfying these needs exist; however, none are small enough to fit along a ROI with the size of a dental implant. Therefore, the goal of this thesis is to design and fabricate three standards with HU values ranging from 700-3000 HU all while having a length of 4 mm ± 1 mm and a thickness of 1.5 mm ± 0.5 mm.

To determine standards' HA mass percentage, one must be familiar with theoretical HU calculations. HU calculations require the understanding of effective energy and the techniques to attain it. Below, we detail these concepts.

Determining Effective Energy:

To compute the theoretical HU for a standard, we used equation 2.1 provided below:

$$HHHH = \frac{uu_{ii} - uu_{wwwwwwwwww}}{uu_{wwwwwwwwww} - uu_{wwiiww}} * 1000 \qquad EEEEuuEEEEEEEEEE \quad 2.1$$

Where u_i is the linear attenuation coefficient of the standard at a given X-ray photon energy. We obtained u_i from the NIST website. [4] This site lists the mass attenuation coefficient (u_i/ρ) as a function of the X-ray energy incident on the sample (Figure 2.2). The mass attenuation coefficient attained from the NIST website can easily be converted to linear attenuation coefficient by multiplying it by the mass density of the material.

Clinical X-ray Sources have a broad energy distribution (Figure 2.3). Therefore, to locate the appropriate theoretical attenuation coefficient of the standards from the NIST tables, the

effective energy of the scanner has to be determined. The effective energy is "the energy of a mono-energetic beam of photons that has the same penetrating ability as the spectrum of photons" emitted from the scanner's X-ray source. [4] To determine the effective energy the half value length (HVL) needs to be measured first.

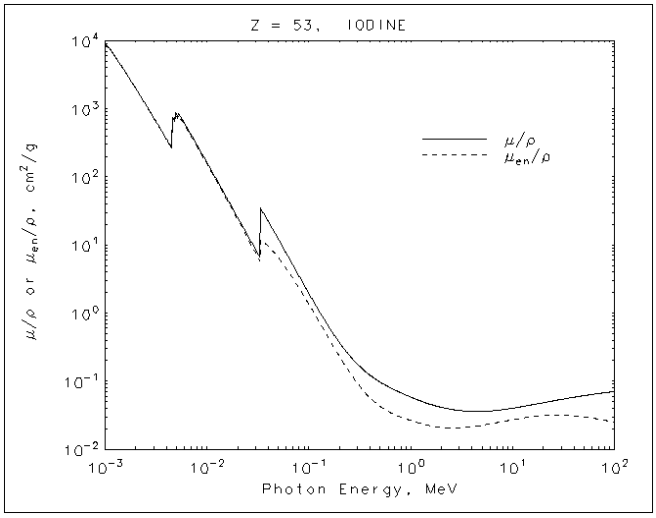


Figure 2.2: Plot illustrating the mass attenuation coefficient as a function of x-ray energy incident for Iodine. This plot was generated by NIST Website.

Finding Half Value Length:

The Half Value Length (HVL) is the thickness of the aluminum sheet that absorbs half of the X-ray photons emitted from the X-ray source. Aluminum standards made for such testing (RDP Inc. cat. #115-500) along with an ion-detecting chamber and an electrometer (660, Victoreen, Cleveland, OH) were used to record

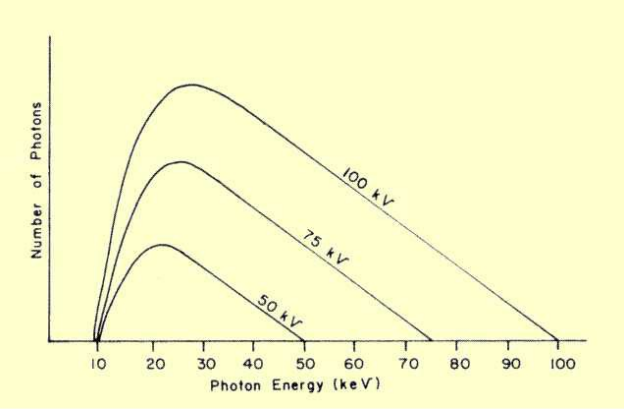


Figure 2.3: X-ray photon energy profile in CT scanners.

the number of photons that penetrated through different thicknesses of aluminum. To determine HVL, data points for photon intensity (*I*) as a function of aluminum thickness (*t*) were plotted and fitted to equation 2.2 (Figure 2.4).

$$II = AA + II_0 ee^{-uuuw} \qquad EEEEuuEEEEEEEEE\ 2.2$$

The constant *A* is included in this fit to account for sources of noise like scattered photons and dark currents. Dark currents are generated by the applied field in the machine and are not due to any incident X-ray ions. [5] Equation 2.2 was used to calculate a range of intensities (*I)* for a

range of thickness (t = 0 mm - 3 mm). The constant *A* was then subtracted from the intensity values and HVL was calculated by solving for *t* when $I/I_0 = ½$.

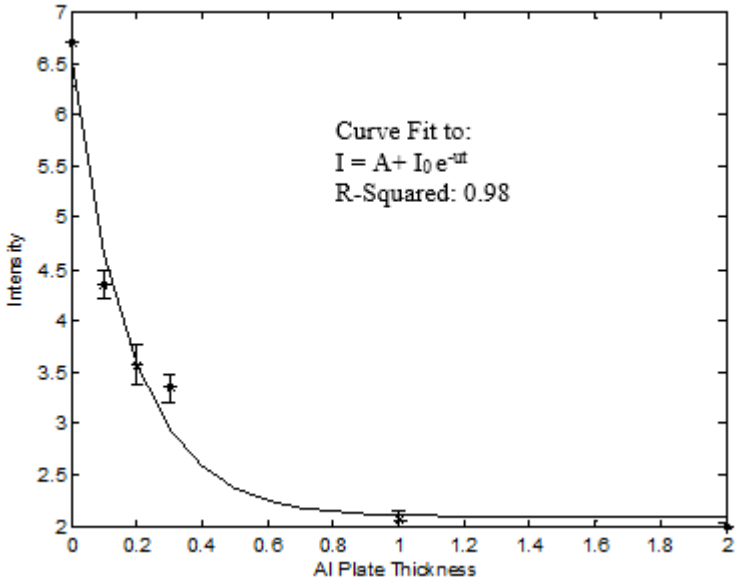


Figure 2.4: This plot generated using Matlab, show a curve fit to data collected for photon intensity as a function of aluminum thickness.

To find the effective energy, the standard X-ray absorption equation 2.3 was used to solve for μ when $EE = HHHHHH$ and $I/I_0 = ½$.

$$II = II_0 ee^{-uuww} \quad EEEEuuEEEEEEEEE \quad 2.3$$

Using the NIST website, we looked up the attenuation values for aluminum as a function of energy. The energy that corresponded to the attenuation value computed for μ was defined as the effective energy of the scanner with the indicated accelerating voltage and beam current. [1]

[1] If other users wish to use different accelerating voltages and/or different beam currents, they will need to reproduce this measurement at their desired X-ray source settings. It is important to note that the effective energy is subject to change as a function of time. In a 14 month period, we observed that the effective energy of our machine had dropped from 25.4 keV to 21.8 keV.

Determining weight percentage based on theoretical HUs:

Utilizing the tables on the NIST website, the mass attenuation coefficients as a function of photon energy for HA, HDPE and air were used to calculate the linear attenuation coefficient of samples containing 0-100% by weight of HA. Linear attenuation coefficients, μ_L, of each sample was calculated using the density of HA, HDPE and air in solution and their respective mass attenuation coefficient, μ_m, as a function of photon energy (Equation 2.4).

$$uu_{LL} = uu_{mmmmmm}PP_{mmmm,ssssss} + uu_{mmmmmmmmmm} * PP_{mmmmmmmm,ssssss} + uu_{mmwwiiww} * PP_{wwiiww,ssssss} \quad EEEEuuEEEEEEEEEE \quad 2.4$$

$$PP_{mmmm,ssssss} = \frac{MM_{mmmm}}{HH_{mmmm} + HH_{wwiiww} + HH_{mmmmmmmmm}} \quad EEEEuuEEEEEEEEEE \quad 2.5$$

$$PP_{wwiiww,ssssss} = \frac{MM_{wwiiww}}{HH_{mmmm} + HH_{wwiiww} + HH_{mmmmmmmmm}} \quad EEEEuuEEEEEEEEEE \quad 2.6$$

$$PP_{mmmmmmmm,ssssss} = \frac{MM_{mmmmmmmm}}{HH_{mmmm} + HH_{wwiiww} + HH_{mmmmmmmmm}} \quad EEEEuuEEEEEEEEEE \quad 2.7$$

It is important to include air in these calculations because the HA particles were not fully dense HA. Sintered HA is known to reach a mass density as high as 3.05 g/cc. [6] However, the HA powder purchased by our group (Bonding Chemical, Katy, TX) contained a mass density of only 1.67 g/cc. This is likely due to the precipitation process of the powder.[2] Figure 2.5 provides SEM images of the HA powder.[3] Images on the top left and top right illustrate the HA powder shape as well as the polydispersity of the particle sizes. Moreover, the bottom image in Figure 2.5 illustrates a magnified view of a single HA particle. It is in this image where the void spaces (the darker regions) filled with air are visible.

[2] Through private communication with the tech support of the company who provided the HA powder (Bonding Chemical) we were able to determine that the powder had a certain amount of porosity which dilated the volume of the powder while keeping the mass constant.

[3] SEM images were provided by the Bonding Chemical's Sales Team

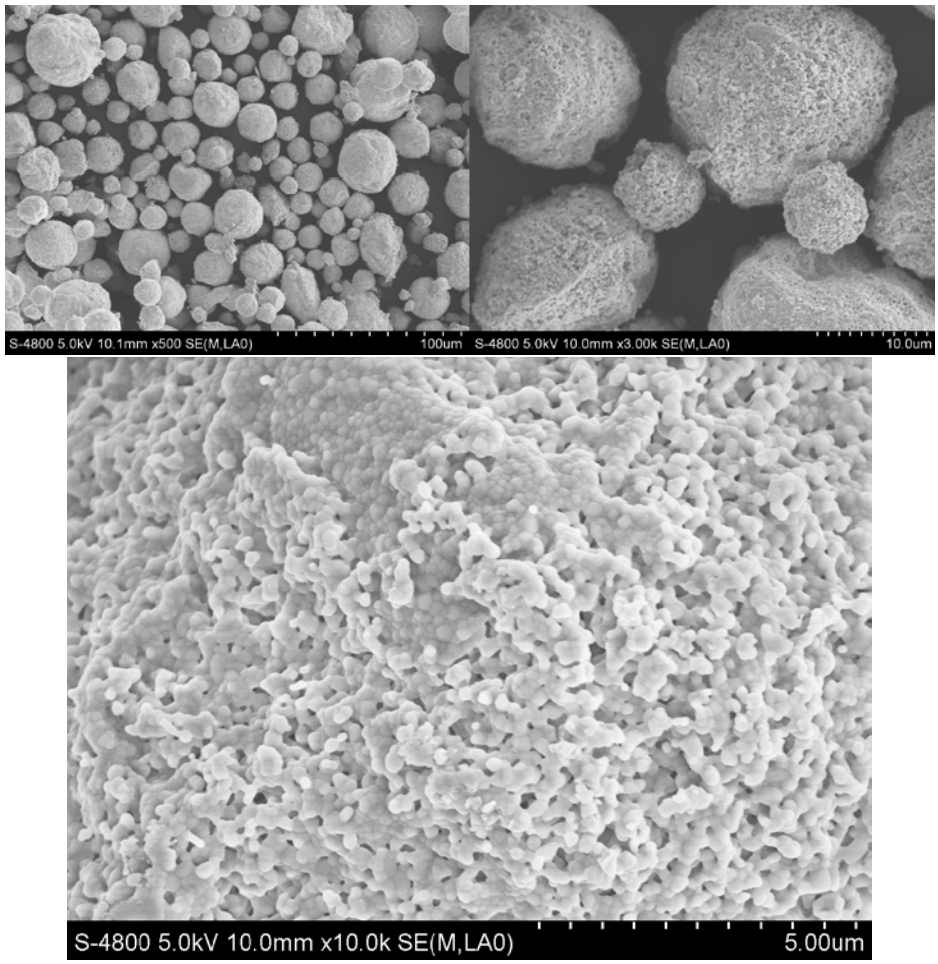


Figure 2.5: SEM images of HA particles provided by the manufacturing company, Bonding Chemical (Katy, TX). Magnified images reveal spaces between the HA molecules occupied by air.

Using the calculated linear attenuation coefficient, HU was calculated using equation 2.1. HUs as a function of HA mass percentage were then plotted for mass percent HA particle = 0 – 100% in 10% increments (Figure 2.6). A quadratic function was utilized to relate HU to HA particle weight percentages. This plot allowed us to choose a specific HU for our standards and determine the amount of HA powder needed to obtain such X-ray attenuation.

Fabrication Protocol:

A substantial amount of research went into creating the protocol for production of the ceramic/polymer standards containing a combination of HA and HDPE. Techniques including ultra-sonication as well as heat pressing were mainly used to fabricate the standards. Moreover, surfactants like Polysorbate 20 was utilized in attempts to decorate HDPE molecules with HA particles in solution. These methods included many drawbacks which will be touched upon in Appendix C; however, the main drawback with these techniques included the high viscosity and the low flow rate of the polymer melt which resulted in the drastic inhomogeneity of the standard. Eventually, methods involving utilizing a twin screw extruder were used. This was done under the assumption that the shear force generated by the twin screws could lower the viscosity of the polymer melt and lead to better distribution of the HA powder inside the melt. The finalized protocol is introduced in the remaining section of the chapter; however, the various steps taken towards writing this protocol are listed in Appendix C.

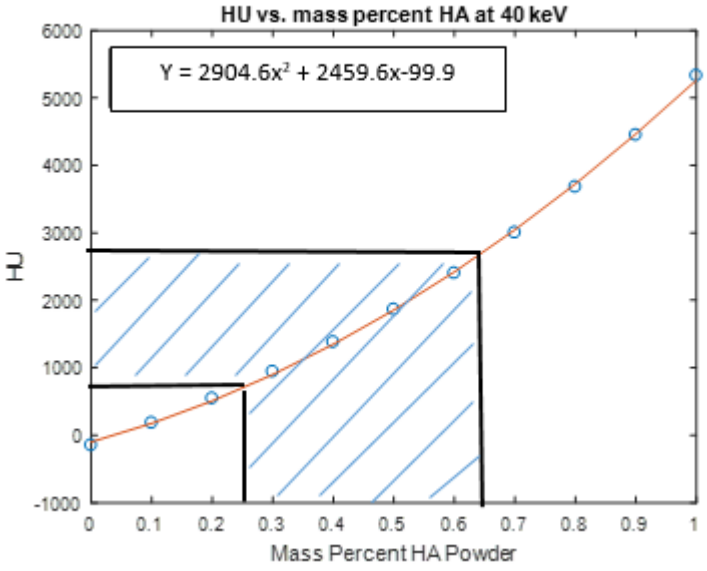


Figure 2.6: Square fit relating Theoretical HU of standards to mass percentage of HA powder, ranging from 0% to 100% by weight HA. Shaded region represent the range in which HA-HDPE standards were made.

Fabrication using twin-screw extruder:

Utilizing a twin-screw extruder (HAAKE MiniLab II Micro Compounder, Thermo Fisher Scientific, Waltham, MA), standards with known mass density were fabricated using a combination of HA and HDPE. Before use, the twin-screw extruder was heated to 180°C and

pressure and torque sensors were tared. The first sample, containing 27% by weight of HA, was produced by adding HA (Bonding Chemical, Katy, TX) powder to a combination of HDPE (Alfa Aesar, Tewksbury, MA) beads and mineral oil at a 27:68:5 ratio. Mineral oil was applied to wet HA powder and help the powder coat onto the HDPE beads. Moreover, oil and polyethylene are made from similar elements. Therefore, the addition of mineral oil was not expected to effect the standards' properties.

The ingredients were mixed for one minute before slowly being funneled into the opening of the MiniLab. A total of 5 grams of material was added to occupy the machine crevice (shown by blue arrows in Figure 2.7) in its entirety. After processing inside the machine for 15 minutes, the HA-HDPE mixture was extruded onto an aluminum sheet. The samples were then sliced into

Figure 2.7: Image of MiniLab twin-screw extruder (HAAKE MiniLab II Micro Compounder, Thermo Fisher Scientific, Waltham, MA). Image (Top Left) shows the opening in which the mixture is inserted into the machine. Image (Top right) shows the inside of the machine with the twin screws in place. Image (Bottom) illustrate the crevices which have to be occupied by mixture so material can be cycled through.

smaller samples using a single edge razor blade. Initial cutting was done before the sample cooled to room temperature. The sample lengths were measured in millimeters and a razor blade was used to further adjust the sample length to satisfy the dimensional specification mentioned above.

Samples containing 45% (1,595 HU) and 68% (2,922 HU) by weight of HA were made by adding HA powder to a combination of HDPE beads and mineral oil at a 45:50:5 and 68:27:5 weight ratio respectively. Production methods detailed above were replicated.

As evident from Figure 2.6, due to the low density of the HA powder, at least 70% by mass powder was needed to achieve HUs above 3000. This created concern because prior experience with the MiniLab machine had revealed its incapability to process mixtures of powder and polymer melt at ratios higher than 68:32. Low amounts of polymer melt caused powder to clump together and back up the machine. For this reason, a fourth standard made of aluminum was added to the set of standards. Aluminum was chosen because of its use in prior studies.[7], [8] Moreover, aluminum has a mass density of 2.7 g/cc and an atomic number of 13, which is similar to the average atomic number of all the elements (more commonly known as the *effective atomic number*) in bone, 13.8.

Measuring Techniques:

Standards' dimensions were measure using a caliper (Mitutoyo, Aurora, Illinois). The length and thickness of the standards were measure in three trials. Average length and thickness measurement were recorded in Table 4.1.

X-ray density was determined by imaging the standards using a clinical fan beam CT scanner (Philips IQon) operating at 120 kvp with an effective energy of 40 keV. After imaging, the

average volumetric HU for each standard was measured using Matlab functions further described in Chapter 3. HU measurements were recorded in Table 4.1.

Reference

[1] R. K. Roeder, M. M. Sproul, and C. H. Turner, "Hydroxyapatite whiskers provide improved mechanical properties in reinforced polymer composites," *J. Biomed. Mater. Res. A*, vol. 67A, no. 3, pp. 801–812, Dec. 2003.

[2] I. M. de C. C. Silva, D. Q. de Freitas, G. M. B. Ambrosano, F. N. Bóscolo, and S. M. Almeida, "Bone density: comparative evaluation of Hounsfield units in multislice and cone-beam computed tomography," *Braz. Oral Res.*, vol. 26, no. 6, pp. 550–556, Dec. 2012.

[3] "Hounsfield units - scale of HU, CT numbers | Classifications, online calculators, and tables in radiology." [Online]. Available: http://radclass.mudr.org/content/hounsfield-units-scale-hu-ct-numbers. [Accessed: 11-Aug-2017].

[4] C. Suplee, "X-Ray Mass Attenuation Coefficients," *NIST*, 17-Sep-2009. [Online]. Available: https://www.nist.gov/pml/x-ray-mass-attenuation-coefficients. [Accessed: 10-Aug-2017].

[5] J. B. Frey, "An experimental and theoretical study of the dark current and x-ray sensitivity of amorphous selenium x-ray photoconductors," Jan. 2013.

[6] S. Schweizer *et al.*, "Preparation and characterization of calibration standards for bone density determination by micro-computed tomography," *Analyst*, vol. 132, no. 10, pp. 1040–1045, Sep. 2007.

[7] P. Mah, T. E. Reeves, and W. D. McDavid, "Deriving Hounsfield units using grey levels in cone beam computed tomography," *Dentomaxillofacial Radiol.*, vol. 39, no. 6, pp. 323–335, Sep. 2010.

[8] T. Reeves, P. Mah, and W. McDavid, "Deriving Hounsfield units using grey levels in cone beam CT: a clinical application," *Dentomaxillofacial Radiol.*, vol. 41, no. 6, pp. 500–508, Sep. 2012.

Chapter 3: Imaging Method/ Image Analysis

Introduction:

Standards made from polymer/ceramic mixtures tend to lack total homogeneity. Moreover, as the mass percentage of the polymer melt decreases the more inhomogeneous the standards appeared in an X-ray scan. Therefore, it was important to develop a method to calculate an average grey level for a given VOI. Doing so allows for a single quantitative measure which can characterize the standards. In this chapter, Matlab functions written to analyze CBCT scans of the standards are detailed. Below the three main Matlab functions are described, and methods of imaging, pixel value calculation as well as data processing are detailed.

Background:

Inherently, Inveon PET/CT scanners output DICOM files in a staggered and randomly organized manner. Moreover, filenames do not contain the slice order. This randomization inhibits the use of DICOM slices to render a coherent three-dimensional image of the scan. Therefore, several Matlab functions were written to open and reorder DICOM files outputted by the Inveon Micro PET/CT system.[4] This was accomplished by collecting acquisition information found in the header of each file. In the following section, we detail the functionality of Matlab functions used for analysis.

[4] A NOBL group member, Elyh Lapetina, initially wrote Matlab functions. The functions were modified by myself once Elyh left the group.

DICOM2Volume and Genrate3DMatrix:

Reordering and displaying the DICOM slices in order could be accomplished by utilizing two functions: *DICOM2Volume* and *Generate3DMatrix* (Matlab Code included in Appendix B). *DICOM2Volume* determines the number of slices in an image by determining the number of DICOM files in a given directory. This function loops through each DICOM slice and obtains the slice information from the file headers. *Acquisition Number* information from each slice header is used to reorder the files into a matrix called *Volume*. The *Generate3DMatrix* function creates a three-dimensional matrix, *matrix,* containing the same number of rows and columns as each DICOM slice as well as a third dimension mirroring the number of DICOM slices stored in the *Volume* matrix. These two functions are illustrated in pseudo code format below.

Pseudo Code Parameters: *DICOM2Volume*	Explanation
Inputs and Parameters	(i) Directory
Output	*Volume* matrix containing name of each DICOM file in order of acquisition

Algorithm	(1) Determine number of files in directory *(2)* Loop into each file, retrieve *Acquisition Number* from file header and store in a matrix named *slices* (3) Order numbers in *slices* matrix from smallest to largest (4) Create new matrix (*Volume)* to store file names *(5)* Loop through each file in directory, retrieve *Acquisition Number*, find number in *slices* matrix that matches *Acquisition Number* and record its index location in variable (*index)* (6) Record name of given file in matrix, *Volume*(*index*)

Pseudo Code Parameters *Generate3DMatrix*	Explanation
Inputs and Parameters	(i) *Volume*: Matrix containing name of each slice in order
Output	Three-dimensional matrix (*matrix)* containing pixel values
Algorithm	(1) Determine length and width of each slice (in number of pixels) (2) Determine number of file names stored in *Volume* matrix *(3)* Generate three-dimensional matrix of zeros called *matrix* a. *matrix* contains same number of rows and columns as number of pixels in each slice b. *matrix* also contains third dimension of pixels mirroring number of files in the *Volume* matrix (4) Loop through each file in directory and copy pixel value into corresponding index in *matrix*

DICOMViewer:

The third and main function, *DICOMViewer,* operates the user interface illustrated in Figure 3.1. Once the DICOM slices are reordered and displayed, the user interface utilizes the *DICOMViewer* function to scroll through the slices simulating a three-dimensional rendering of the scan. *DICOMViewer* also uses inputs to distinguish the center of a ROI and determine the number of slices where the VOI is visible. After a user identifies the center and number of slices containing the VOI, the function can also calculate an average volumetric pixel value in grey scale or HU. If the VOI is not perpendicular to the axial plane of the machine, *DICOMViewer* can account for the rotational offset of the standards and adjust the center of the ROI in each slice using two linear functions. The first linear function adjust the x location of the center point as a function of slice number, while the second function adjust the y location of the center point as a function of slice number. The *DICOMViewer* function allows for two calibration types including water-air calibration and calibration using an aluminum and three HA-HDPE standards with various mass ratios of HA. These calibrations require a method to measure the average pixel values in a given volume. The remaining portion of this chapter illustrates how the Matlab functions were used to obtain volumetric average pixel values. The final portion of this chapter is dedicated to describing how calibration is performed using these functions.

Figure 3.1: Image of the GUI used to analyze a cylindrical volumetric HU of our samples, using DicomViewer function. Mark 1 and Mark 2 buttons are used to indicate slice boundaries, while the radius button allows the user to indicate the radius of the cylinder. Pixel Button allows the user to input a second ROI.

Volumetric Average Pixel Value:

To reach our ultimate goal in this project, we had to develop a method that allowed us to measure grey level for a three-dimensional region representing the volume of a dental implant. Clinical software interface available to our group did not allow for such measurements. Below we describe how the developed user interface allows an individual to make average volumetric pixel intensity measurements.

Using the *DICOMViewer* function, the user can scroll to the first slice where the VOI[5] is visible. This slice must be marked by pressing the *Mark 1* pushbutton (shown in Figure 3.1). In this slice,

[5] While ROI refers to a two dimensional region in the scan, a volume of interest refers to a volumetric region in the scan. Average grey levels for a volume of interest is calculated by averaging the grey values in ROIs.

the user must choose a point that represents the center of the ROI. From the center, the user can mark the outer limits of the ROI by indicating a radius. The user may scroll until the last slice in which the VOI is visible. This final slice will be marked by the *Mark 2* pushbutton (Figure 3.1). If there is a discrepancy between the center of the ROI in the slice indicated by *Mark1* and *Mark2*, the user can press the *Pixel* button (Figure 3.1) and mark the center of the ROI once more in the *Mark 2* slice. Mechanism used to correct for offsets is described above in section DICOMViewer.

After the volume of interest is outlined, the function first calculates an average pixel value by looping through each slice, finding the pixel marked as the center and calculating the mean of all pixel values in a circular area with a specified radius. The volumetric average is then determined by calculating the mean of all circular average pixel values for slices between *Mark 1* and *Mark 2*. Before determining the VOI's HU, it is important to calibrate the ROI using standards with known X-ray attenuations. In the following section, we discuss a working method to generate calibrated HU for a VOI.

Calibration Using HA-HDPE Phantoms:

Calibration requires making a plot of un-calibrated pixel values (in grey scale) outputted by the CBCT scanner against known HU from a set of calibration samples. The relationship between HU and grey scale value is assumed to be linear as represented in the equation 3.1. [1]

$$HHHH = (RReeRRRREERee\ SSRREESSee * GGSSHH) + RReeRRRREERee\ IIEEEEeeIIRReeSSEE \quad \text{Equation 3.1}$$

The rescale slope and rescale intercept are parameters that are included in each DICOM file header. While differences in rescale slope and rescale intercept between machines is not surprising, we have discovered that rescale slopes and intercept in the Inveon system are not consistent between scans. Furthermore, the mechanism used to generate rescale slope and rescale intercept is unknown to us. Therefore, it was important to develop a method that allowed us to generate our own rescale slope and rescale intercept. Below, we detail how the *DICOMViewer* function was used to generate calibrated rescale slope and intercept.

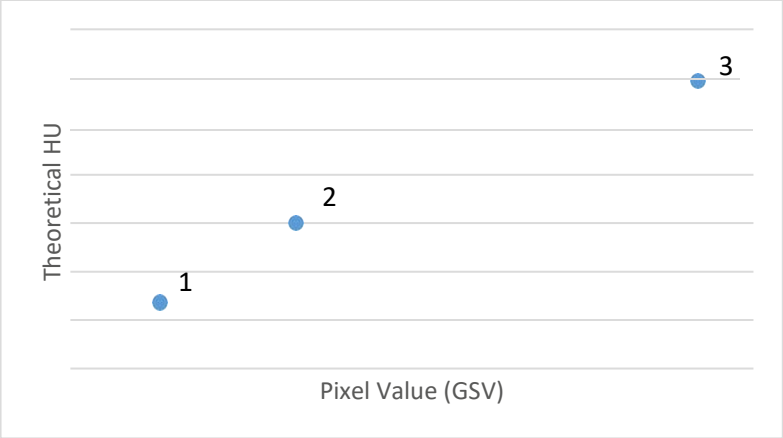


Figure 3.2: Example of theoretical HU values plotted vs the volumetric average pixel values for each standard. The number 1, 2 and 3 represent the data point for standards 1, standard2 and standard 3 respectively.

Before volumetric average pixel values are measured for the ROI, the *DICOMViewer* function is used to measure the volumetric average pixel values (in grey scale) for each standard. When

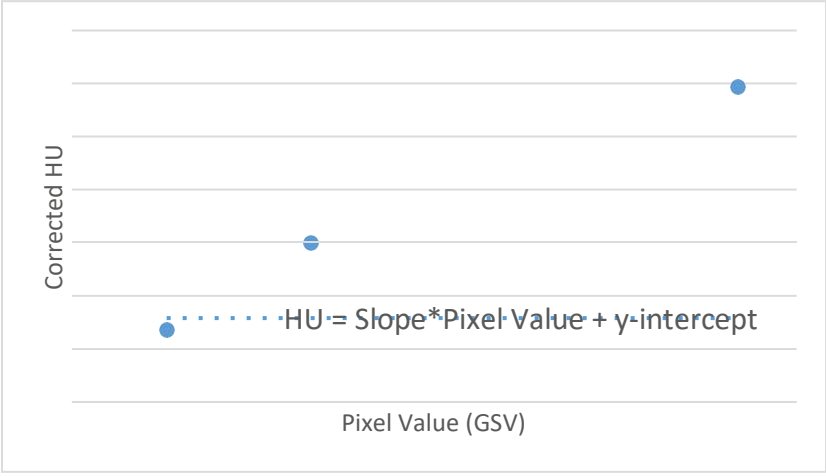


Figure 3.3 Example plot for theoretical HU values plotted versus the volumetric average pixel values for three standards.

outlining the standard, it is crucial to choose a *Mark 1*, *Mark 2* and radius that evade the edges of the standards. Doing so helps avoid partial volume effects, underestimation of standard pixel value and a large standard deviation of pixel values. The theoretical HU value for each standard is plotted against the volumetric average pixel value for each standard (Example shown in Figure 3.2). The average volumetric pixel values outputted by the machine are then related to corrected HU's by fitting a linear function to the data (Figure 3.3). The *DICOMViewer* function then replaces the rescale slope and intercept in the header of each file with the slope and the y-intercept of the linear fit. The newly determined slope and intercept are then used to convert the average volumetric pixel value (in grey scale value) outputted by the machine to corrected HU.

Imaging Method:

As mentioned in the first chapter of this thesis, we hypothesize that placing the standards near the ROI will ensure that standards are effected by exomass in the same manner as the ROI. Therefore, we predict that grey values for the same standards will differ when placed in two different locations in the FOV. This in return will result in two different calibration curve for the same set of standards and a variation in calculated HU values for a chosen ROI.

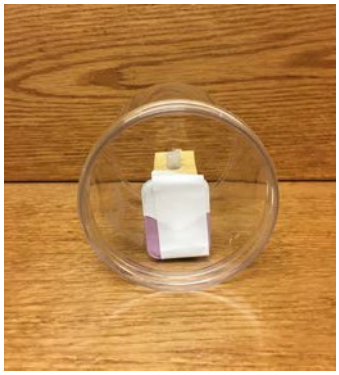
Preliminarily testing of this hypothesis was done by scans using an Inveon Micro PET/CT scanner (at an accelerating voltage of 80 kvp and a beam current of 0.5 mA) and a Carestream 9300 CBCT scanner (at an accelerating voltage of 90 kvp and a beam current of 5.0 mA). The remaining of this chapter will discuss the imaging technique utilized in both machines. Moreover, Image analysis will also be discusses as well as the statistical analysis.

Imaging With Inveon Micro PET/CT Scanner:

First, the effective energy (at 80 kvp and 0.5 mA) of the Inveon scanner was determined using the method detailed in chapter 2. Once the effective energy was determined, A 40 mm x 60 mm

polyurethane saw bone block (Density = 0.80 g/cc) was imaged inside an acrylic container (89 mm and height of 102 mm) filled with water. A small cylindrical vial (8.0 mm diameter, 32.0 mm tall) filled with a lead nitrate solution in distilled water was imbedded in the polyurethane block for each scan. This was done to outline a clear VOI in the block. The water and polyurethane block were meant to act as a surrogate for soft tissue and trabecular bone, respectively. Two scans were performed with the calibration standards fixed onto the polyurethane block. Calibration standards were taped onto a thin plastic board ~0.5 cm apart from each other. The samples were imbedded into a dental X-ray film pouch and taped onto the polyurethane block with waterproof tape. In the first scan, the pouch containing the standards was fixed onto the block 1.25 cm away from the ROI (marked by the imbedded Lead (II) Nitrate vial). The second scan involved taping the standards onto the block 3.25 cm away from the ROI. Figure 3.4 illustrates the orientation of the polyurethane block inside the container in each scan.

A *B* *C*



D E

Figure 3.4: This image illustrate the assembly of the surrogate used in the Inveon Micro PET/CT scans. Image A illustrates all of the individual components including the polyurethane bloc, the vial containing lead (II) Nitrate, X-ray film pouch containing the calibration standards, the strip of waterproof tape used to stick the pouch to the block and the acrylic container. Image B illustrates when the lead (II) Nitrate vial was imbedded into the block and the standards were fixed on the block, 1.25 cm away from the ROI. C illustrates the block orientation when placed in the scanner. Image D illustrates when the lead (II) Nitrate vial was imbedded into the block and the standards were fixed on the block 3.25 cm away from the ROI. Finally, image E illustrates the block orientation when placed into the scanner.

Figure 3.4 attempts to diagrams the surrogate model inside the Micro CBCT scanner. Image A in Figure 3.4 lays out the different parts of the model individually. This image includes the polyurethane saw bone block with a 9 mm diameter hole drilled into it, the acrylic container with a white lid, a plastic vial containing Lead (II) Nitrate and the purple X-ray pouch laying above the strip of white waterproof tape. Images B illustrates the imbedded Lead (II) Nitrate vial and the placement of the X-ray pouch when the standards were placed 1.25 cm away from the ROI. The block was then placed into the container similar to image C and the it was over filled with water. Once the container was filled over the top with water, the lid was used to contain the water inside the container and allow no air bubbles to form inside the container. The Container was then placed into the Inveon Micro CBCT X-ray tube in the same orientation as shown in Image C. Similar procedures were completed in the second scan where the standards were placed 3.25 cm away from the ROI. However, the standards were taped on the long end of the polyurethane block as shown in Image D in Figure 3.4. Moreover, the block was oriented as

shown in Image E. A side view of the scan is shown in Figure 3.5 when the standards were placed both near the ROI (left) and farther away from the ROI (Right).

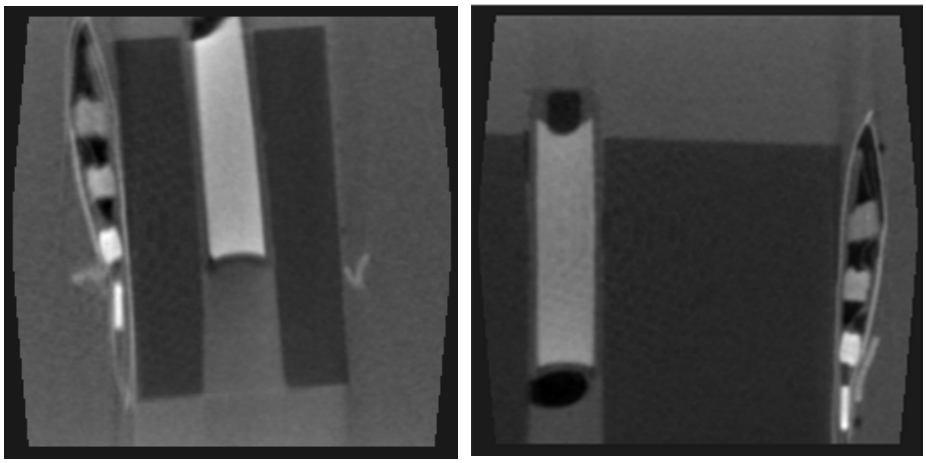


Figure 3.5 :X-ray images of polyurethane density block with an imbedded Lead (II) Nitrate sample and Calibration standards attached 1.25 cm (Left) and 3.25 cm (Right) away from the ROI.

Once we were done testing our standards with the Inveon Micro PET/CT scanner, a pilot test was also done by imaging our standards inside a human skull using a clinical CareStream 9300 CBCT scanner. This test studied the difference in standards' grey values when moving the standards from the top of the oral cavity to a location nearest to the ROI. These two locations were chosen to discover if our phantom location/orientation introduced any advantage over fixing standards to the top of the oral cavity as done by a previous group. [2]

Imaging With Clinical CBCT Scanner:

Similar to the Inveon scanner, the effective energy (at 90 kvp and 5.0 mA) of the CBCT scanner was determined using the method detailed in chapter 2. Once the effective energy was determined, a human skull was scanned two times. In the first scan, an X-ray pouch containing the standards was taped onto the roof of the oral cavity. In the second scan, the pouch was attached to a bitewing and held in place by placing the bitewing in between the top and bottom teeth of the skull. It was important that in the second scan, the standards were placed near the

chosen VOI. The VOI was determined to be a cylindrical volume between the first molar on the left to the left Mental Foramen (Figure 3.6).

Image Analysis:

Average volumetric grey scale values were measured using the method described above. First, an average grey scale value was measured for each standard three separate times and recorded in Tables 4.2, 4.3, 4.7 and 4.8. A mean average grey scale value for each standard was plotted versus the known HU value for the standards. The data points were fitted to a linear function and the slope and y-intercept of the function were stored for later use (Figures 4.3 and 4.4). Due to the lack of automation in outlining the VOI, twelve iterations of average volumetric grey scale values measurements were made for the same VOI in each scan. Between each iteration, the interface was shut down and restarted completely to avoid any bias from the previous measurements. Average grey scale values were recorded in Tables 4.5, 4.6, 4.10 and 4.11. HU calculated using the initial rescale slope and intercept and HU calculated using the slope and intercept of the linear fit were also recorded.

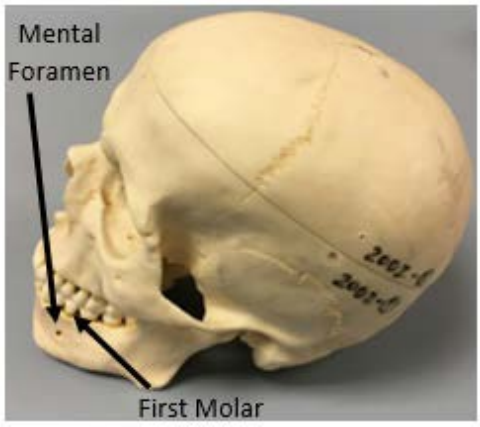


Figure 3.6: Image of Skull used in Clinical CBCT scan. This image points out the First Molar to the Mental Foramen to illustrate the top and bottom borders of the ROI.

HU mean calculated using the slope and intercept from the linear fit were then compared using an independent sample Bootstrap Analysis with a 90% confidence interval.

Reference:

[1] T. Razi, M. Niknami, and F. Alavi Ghazani, “Relationship between Hounsfield Unit in CT Scan and Gray Scale in CBCT,” *J. Dent. Res. Dent. Clin. Dent. Prospects*, vol. 8, no. 2, pp. 107–110, 2014.

[2] T. Reeves, P. Mah, and W. McDavid, “Deriving Hounsfield units using grey levels in cone beam CT: a clinical application,” *Dentomaxillofacial Radiol.*, vol. 41, no. 6, pp. 500–508, Sep. 2012.

Chapter 4: Results and Discussion

Introduction:

This chapter is dedicated to reporting the results for the dimensional measurements of the three HA-HDPE calibration standards shown in Figure 4.1. Moreover, HU for the standards in a clinical fan beam CAT scanner is reported as well as the average grey level values for the standard in two CBCT scanners: Inveon Micro PET/CT and Carestream 9300 Clinical CBCT scanner. Once calibration is performed using the grey values of the standards, corrected HU values for a VOI in each scan are reported. Lastly, the remaining portion of the chapter discusses the results.

Results:

Dimensional Specifications of HA-HDPE Standards

The three rectangular HA-HDPE standards had lengths ranging from 3.9 mm to 4.9 mm. The thickness of the standards ranged from 1.2 mm to 1.8 mm. Extruded strips of the ceramic/polymer mixture ranged from 10 cm - 20 cm in length. In practice, only the initial 10 – 12 cm of the strip was extracted from the extruder and sliced into 25-30 standards with the given dimensional criteria, mentioned in chapter 3. Furthermore, it was noted that thickness of the standards decreased as the HA mass percentage increased. As mentioned in chapter 2, it was observed that the MiniLab machine could not process mixtures with a powder mass percentage higher that 68%. Exceeding such mass ratios resulted in powder remains to build up and back up the machine.

Figure 4.2: Image of three HA-HDPE standards produced using (from top to bottom) 27% by mass HA, 45% by mass HA and 68% by mass HA. Standards ranged from 4 mm - 5 mm in length and width, with a thickness ranging from 1.5 mm - 2.5 mm.

Measuring the average volumetric HU measurements from a clinical fan beam CAT scan revealed HU numbers of 746 ± 6 HU, 1,502 ± 40 HU, and 2,778 ± 47 HU at an effective energy of 40 keV. Detailed verification measurements are illustrated in the Table 4.1. The ± in Table 4.1 rows one and two represent the standard deviation for an average of three length and three thickness measurements. The ± in the third row represent the standard deviation of average grey scale for the VOI.

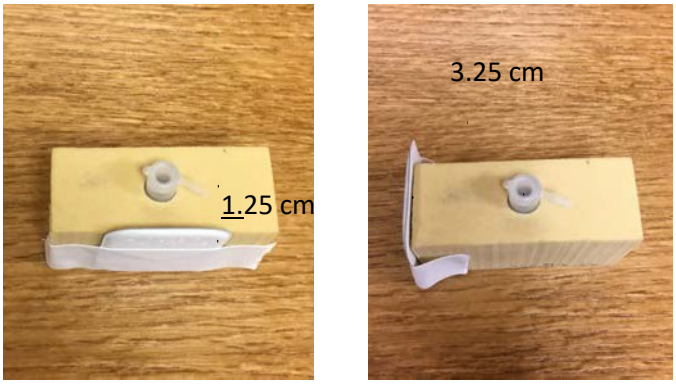
Table 4.1: Length, width, and thickness of each standard. Mass density measurements as well as HU measurements at 40 keV are recorded. HU were measured from scans taken in a clinical fan beam CT scanner.

Measurement	27% HA	45% HA	68% HA
Length	3.9 ± .02 mm	4.8 ± .11 mm	4.1 ± .02 mm
Thickness	1.8 ± .01 mm	1.6 ± .03 mm	1.3 ± .02 mm
Hounsfield Units (40 keV)	746 ± 6 HU	1,502 ± 40 HU	2,778 ± 47 HU

Once the standards were produced with features that satisfied our technical specifications, they were further tested in two CBCT scanners: Inveon Micro PET/CT scanner (at an accelerating voltage of 80 kvp and a beam current of 0.5 mA), Carestream 9300 clinical CBCT scanner (at an accelerating voltage of 90 kvp and a beam current of 5.0 mA). Imaging methods are detailed in chapter 3. The remaining of this chapter will report and discuss calibrating the scans using the standards in the two scanners and report on the calculated average HU for a VOI based on new calibration parameters generated from the standards.

Grey Level Measurements using the Inveon PET/CT

As mentioned in chapter 3, two scans were taken using the Inveon PET/CT system. The first scan contained a sample with the standards placed 1.25 cm away from the ROI (Figure 4.2A), and the second scan contained a sample with the standards placed 3.25 cm away from the ROI (Figure 4.2B). Tables 4.2 and 4.3 include average cylindrical grey levels for the HA-HDPE standards containing 27%, 45%, 68% by mass HA and an aluminum standard. The values in Table 4.2 represent the grey levels for the four standards in the Inveon scanner when placed 1.25 cm away from the ROI (Figure 4.2A). The values in Table 4.3 include grey levels for the four standards in the Inveon scanner when placed 3.25 cm from the ROI (Figure 4.2B). As mentioned in chapter 3, due to the lack of automation in outlining the standard location the grey level measurements were made three times for each standard. The mean grey level for each standard when placed close to the volume of interest was 2,963 ± 150 for the 27% by mass HA, 8,936 ± 301 for the 45% by mass HA, 22,885 ± 540 for the 68% by mass HA, and 25,349 ± 470 for the aluminum standard (Table 4.4). These numbers shifted to -4,042 ± 227, 2,088 ± 167, 18,421 ± 292 and 23,967 ± 352 respectively, when the standards were moved 2 cm farther from the original standard location (Table 4.4).



A B

Figure 4.2: Illustrations when the lead (II) Nitrate vial was imbedded into the block and the standards were fixed on the block, A) 1.25 cm away from the ROI and B) 3.25 cm away from the ROI.

Table 4.2: Grey level measurements for four standards when placed as close as possible to the ROI (1.25 cm away). GSV for each standard was measured in three iterations. Standards containing 27% by mass HA, 45% by mass HA and 68% by mass HA are represented by the 27%, 45% and 68%, respectively. Mark 1 and Mark 2 indicate slice boundaries. Mean grey level is calculated for all pixels inside a circular area with radius of 5 (in pixels). STD represents the standard deviation of average grey level for slices between Mark 1 and Mark 2. HUs are calculated using the initial rescale slope and rescale intercept provided by the scanner.

	GSV	STD	HU
27% Trial 1	3,049	987	-52
27% Trial 2	3,051	968	-52
27% Trial 3	2,790	1,313	-62
45% Trial 1	8,851	1,391	175
45% Trial 2	8,687	1,454	168
45% Trial 3	9,271	1,260	191
68% Trial 1	23,412	1,243	744
68% Trial 2	22,912	1,148	725
68% Trial 3	22,332	1,887	701
Al Trial 1	25,954	2,251	844
Al Trial 2	26,768	2,373	877
Al Trial 3	23,325	5,682	741

Table 4.3: Grey level measurements for four standards when placed 3.25 cm away from the ROI. GSV for each standard was measured in three iterations. Standards containing 27% by mass HA, 45% by mass HA and 68% by mass HA are represented by the 27%, 45% and 68%, respectively. Mark 1 and Mark 2 indicate slice boundaries. Mean grey level is calculated for all pixels inside a circular area with radius (in pixels) Radius. STD represents the standard deviation of average grey level for slices between Mark 1 and Mark 2. HUs are calculated using the initial rescale slope and rescale intercept provided by the scanner.

	GSV	STD	HU
27% Trial 1	-4,077	1,050	-85
27% Trial 2	-4,250	1,073	-93
27% Trial 3	-3,800	603	-73
45% Trial 1	2,212	985	195
45% Trial 2	2,154	812	192
45% Trial 3	1,898	1,124	181
68% Trial 1	18,709	1,142	930
68% Trial 2	18,125	993	904
68% Trial 3	18,429	1,205	917
Al Trial 1	19,656	4592	972
Al Trial 2	23,684	434	1,151
Al Trial 3	20,974	5,080	1,031

Table 4.4: Mean grey level measurements for four standards. Average GSV for each standard is represented when placed both 1.25 cm and 3.25 cm away from the ROI. Standards containing 27% by mass HA, 45% by mass HA and 68% by mass HA are represented by the 27%, 45% and 68%, respectively. Known HU represents the calculated HU for the standards using the lookup tables from the NIST website.

Standard	1.25 cm Avg.	STD	3.25 cm Avg.	STD	Known HU
27% (GSV)	2,963	150	-4,042	227	723
45% (GSV)	8,936	301	2,088	167	1,447
68% (GSV)	22,885	540	18,421	292	2,728
Aluminum (GSV)	25,349	470	21,438	352	4,640

Grey level data from Table 4.4 was then plotted versus the known HU values for the standards. Once plotted, the data points were fitted by the linear equation 3.1 (Figure 4.3). The slope and y-intercept relating grey levels to HU when the standards were near the ROI were 0.15 and 168, respectively. The slope and y-intercept relating grey levels to HU when the standards were 2 cm farther from the ROI were 0.12 and 1132, respectively.

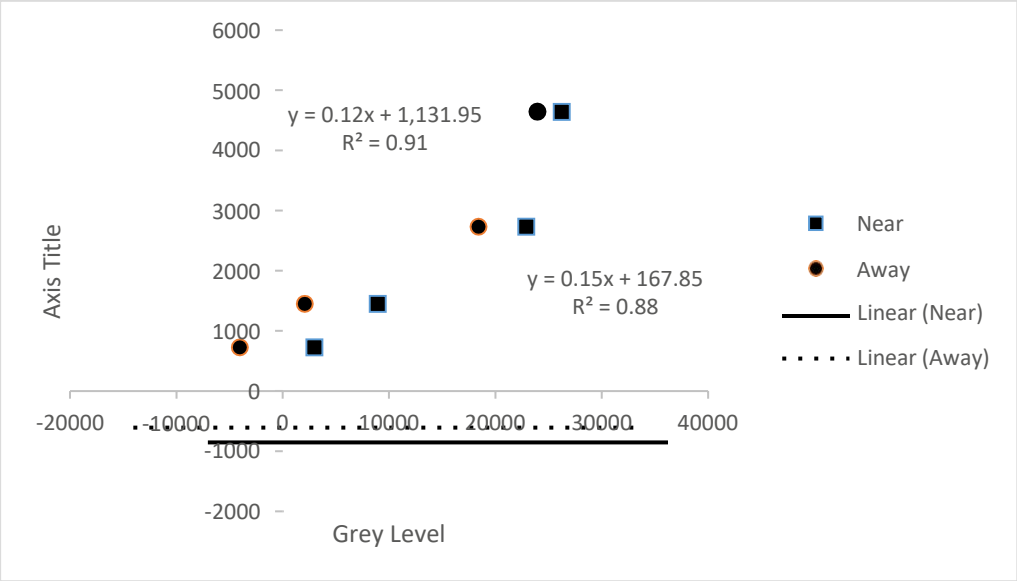


Figure 4.3: Data points representing the known HU of standards at an effective energy of 40 keV versus grey levels outputted by the Inveon micro PET/CT scanner. Solid and dotted lines represent the linear fit demonstrating the relationship between grey level measurements for a chosen ROI and calibrated HU values.

Once standards were used to generate a new rescale slope and rescale intercept, the average grey level for the chosen VOI was recorded in Table 4.5 and 4.6. These tables contain 12 trials of grey

level measurements for a given volume of interest when the standards were placed near (Table 4.5) and away from (Table 4.6) the ROI. When measuring the volume of interest's grey level in the two scans, we discovered an average grey level of 8,302 ± 135 for when the standards near the ROI and -447 ± 116 for when the standards farther away from the ROI. Using the new rescale slope and rescale intercept generated by calibration standards, we found that the HU for the VOI are 1,374 ± 20 and 1,077 ±14, respectively, resulting in a 24.3% difference. Here the ± represent the standard deviation of average grey level measurement and the calculated HU values from the 12 iterations, while the standard deviation reported in Tables 4.5 and 4.6 are standard deviation of average grey level of slices within the volume of interest. While the ± values illustrate the variation between twelve average grey scale measurements for a single volume of interest, the STD values reported in GSV is the standard deviation of grey level within the volume of interest. The mean STD within the volume of interested is 1,230 and 972 for the first and second scan respectively. These values are recorded in the last row of the STD column in Tables 4.5 and 4.6.

Table 4.5: 12 iterations of grey level measurements for a chosen ROI when standards were placed as close as possible to the ROI (1.25 cm away). *Mark 1* and *Mark 2* indicate slice boundaries. Mean grey level is calculated for all pixels inside a circular area with radius of 10 (in pixels). STD represents the standard deviation of average grey level for slices between *Mark 1* and *Mark 2*. *HUs* are calculated using the initial *rescale slope* and *rescale intercept* provided by the scanner. *Calculated HUs* are calculated using slope and intercept from the linear fit generated using the calibration standards.

	GSV	STD (GSV)	HU	Calculated HU
Trial 1	8,458	1,535	159	1,415
Trial 2	8,413	1,273	158	1,408
Trial 3	8,308	1,207	153	1,393
Trial 4	8,375	1,455	156	1,403
Trial 5	8,370	1,248	156	1,402
Trial 6	8,228	1,127	150	1,381
Trial 7	8,156	978	147	1,370
Trial 8	8,157	1,039	148	1,371
Trial 9	8,187	1,032	149	1,375
Trial 10	8,360	1,338	155	1,400
Trial 11	8,094	956	145	1,361
Trial 12	8,515	1,575	162	1,423
Mean	8,302	1,230	153	1374

Table 4.6: 12 iterations of grey level measurements for a chosen ROI when standards were placed 3.25 cm away from the ROI. Mark 1 and Mark 2 indicate slice boundaries. Mean grey level is calculated for all pixels inside a circular area with radius of 10 (in pixels). STD represents the standard deviation of average grey level for slices between Mark 1 and Mark 2. HUs are calculated using the initial rescale slope and rescale intercept provided by the scanner. Calculated HUs are calculated using slope and intercept from the linear fit generated using the calibration standards.

	GSV	STD (GSV)	HU	Calculated HU
Trial 1	-418	1,010	78	1,107
Trial 2	-256	1,300	85	1,128
Trial 3	-201	1,426	88	1,135
Trial 4	-515	957	74	1,095
Trial 5	-423	756	78	1,107
Trial 6	-435	955	77	1,105
Trial 7	-519	867	73	1,094
Trial 8	-605	805	70	1,083
Trial 9	-515	970	74	1,095
Trial 10	-444	880	77	1,104
Trial 11	-500	891	74	1,097
Trial 12	-530	843	73	1,093
Mean	-447	972	77	1077

Comparing the two independent data sets using a Bootstrap analysis reveals a mean difference of -286 with a 90% confidence interval (-309,-286). These results illustrate that with 90% certainty the mean difference between HU values in a volume of interest will be between -309 and -286 when standards are placed near (1.25 cm away) the ROI and when standards are placed father away (3.25 cm away) from the ROI.

Standards produced for this thesis project were further tested by imaging them inside a human skull using a Carestream clinical CBCT scanner. Below we report on the grey level difference in standards when moving the standards from the top of the oral cavity to a location nearest to the ROI (between the first molar on the left to the Mental Foramen).

Grey Level Measurements using the Clinical CBCT scanner:

Tables 4.7 and 4.8 include average cylindrical grey levels for the HA-HDPE standards containing 27%, 45%, 68% by mass HA and a single aluminum standard. The values in Table 4.7 represent the grey levels for the four standards in a CBCT scan when the standards are placed

Table 4.3: Grey level measurements for four standards when fixed on the inside of the mandible as close as possible to the ROI. GSV for each standard was measured in three iterations. Standards containing 27% by mass HA, 45% by mass HA and 68% by mass HA are represented by the 27%, 45% and 68%, respectively. *Mark 1* and *Mark 2* indicate slice boundaries. Mean grey level is calculated for all pixels inside a circular area with radius of 3 (in pixels). STD represents the standard deviation of average grey level for slices between *Mark 1* and *Mark 2*. *HUs* are calculated using the initial *rescale slope* and *rescale intercept* provided by the scanner.

	GSV	STD	HU
27% Trial 1	1453	33	453
27% Trial 2	1461	25	461
27% Trial 3	1424	23	424
45% Trial 1	1887	37	887
45% Trial 2	1869	23	869
45% Trial 3	1902	41	902
68% Trial 1	2630	48	1630
68% Trial 2	2646	21	1646
68% Trial 3	2643	40	1643
Al Trial 1	3486	36	2486
Al Trial 2	3461	70	2461
Al Trial 3	3476	73	2476

as close as possible to the first molar on the left of the skull. The values in Table 4.8 include grey levels for the four standards when placed at the top of the oral cavity. Once again, due to the lack of automation in outlining the standard location the grey level measurements were made three times for each standard.

Table 4.8: Grey level measurements for four standards when fixed on the roof of the oral cavity. GSV for each standard was measured in three iterations. Standards containing 27% by mass HA, 45% by mass HA and 68% by mass HA are represented by the 27%, 45% and 68%, respectively. Mark 1 and Mark 2 indicate slice boundaries. Mean grey level is calculated for all pixels inside a circular area with radius of 3 (in pixels). STD represents the standard deviation of average grey level for slices between Mark 1 and Mark 2. HUs are calculated using the initial rescale slope and rescale intercept provided by the scanner.

	GSV	STD	HU
27% Trial 1	1495	67	495
27% Trial 2	1479	62	479
27% Trial 3	1402	55	502
45% Trial 1	1596	56	596
45% Trial 2	1589	48	589
45% Trial 3	1594	65	594
68% Trial 1	2679	106	1679
68% Trial 2	2635	93	1635
68% Trial 3	2690	126	1690
Al Trial 1	3548	101	2548
Al Trial 2	3462	195	2462
Al Trial 3	3461	204	2461

Table 4.9: Mean grey level measurements for four standards. Average GSV for each standard is represented when placed both on the mandible **near the** ROI and on the roof of the oral cavity. Standards containing 27% by mass HA, 45% by mass HA and 68% by mass HA are represented by the 27%, 45% and 68%, respectively. Known HU represents the calculated HU for the standards using the lookup tables from the NIST website.

Standard	Near Avg. GSV	STD	Roof Avg. GSV	STD	Calculated HU
27%	1446	19	1459	50	723
45%	1886	17	1593	4	1447
68%	2640	9	2668	29	2728
Aluminum	3474	13	3490	50	4640
ROI (GSV)	2126	6	2111	4	-
ROI Calculated HU	1932	12	2047	7	-

The mean grey level for each standard when placed close to the ROI was 1,446 ± 19 for the 27% by mass HA, 1,886 ± 17 for the 45% by mass HA, 2,640 ± 9 for the 68% by mass HA and 3,474 ± 13 for the aluminum standard. These numbers shifted to 1,459 ± 50, 1,593 ± 4, 2,668 ± 29 and 3,490 ± 50, respectively, when the standards were moved from the ROI to the roof of the mouth (Table 4.9).

Grey level data from Table 4.9 was then plotted versus the known HU values for the standards. Once plotted, the data points were fitted by the linear equation 3.1 (Figure 4.4). The slope and y-intercept relating grey levels to HU when the standards were near the ROI were 1.92 and -2,152, respectively. The slopes and y-intercepts relating grey levels to HU when the standards were placed at the top of the oral cavity were 1.77 and -1,680, respectively.

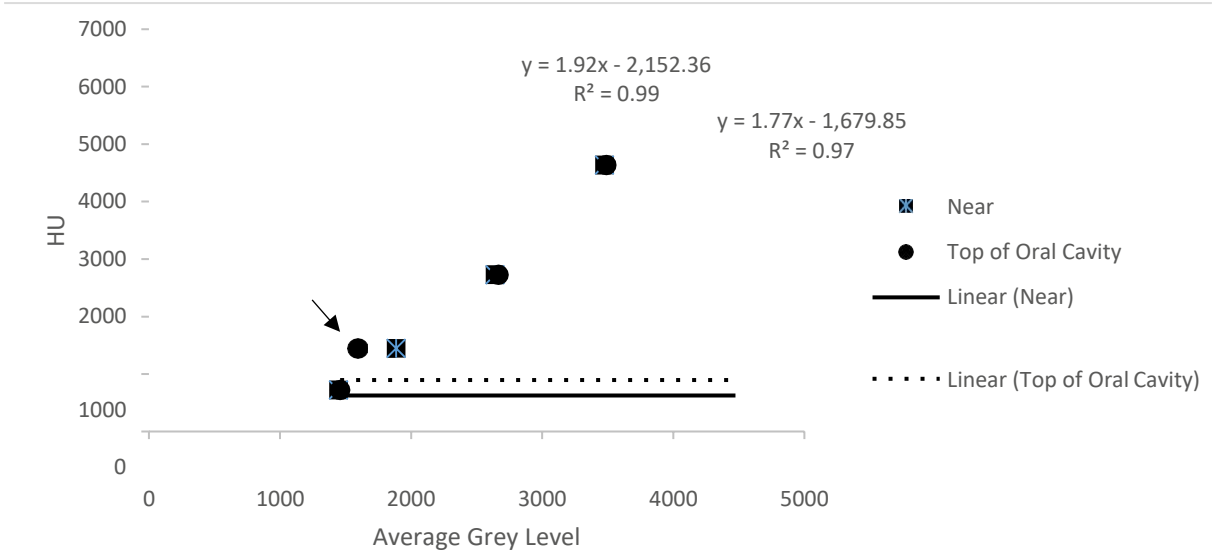


Figure 4.4: Data points representing the known HU of standards at an effective energy of 40 KeV versus grey levels outputted by the Clinical CBCT scanner. Solid and dotted lines represent the linear fit demonstrating the relationship between grey level measurements for a chosen ROI and calibrated HU values.

Once standards were used to generate a new rescale slope and rescale intercept, the average grey level for the chosen VOI was recorded in Tables 4.10 and 4.11. These tables contain twelve trials

of grey level measurements for a given VOI when the standards were placed near the left first molar (Table 4.10) and on the roof of the oral cavity (Table 4.11).

Table 4.10: 12 iterations of grey level measurements for a chosen ROI when standards were fixed on the inside of the mandible as close as possible to the ROI. Mark 1 and Mark 2 indicate slice boundaries. Mean grey level is calculated for all pixels inside a circular area with radius or 10 (in pixels). STD represents the standard deviation of average grey level for slices between Mark 1 and Mark 2. HUs are calculated using the initial rescale slope and rescale intercept provided by the scanner. Calculated HUs are calculated using slope and intercept from the linear fit generated using the calibration standards.

	GSV	STD	HU	Calculated HU
Trial 1	2123	27	1123	1926
Trial 2	2127	26	1127	1934
Trial 3	2125	27	1125	1930
Trial 4	2124	28	1124	1928
Trial 5	2123	23	1123	1926
Trial 6	2113	28	1113	1907
Trial 7	2138	32	1138	1955
Trial 8	2124	27	1124	1928
Trial 9	2133	30	1133	1946
Trial 10	2127	28	1127	1934
Trial 11	2134	26	1134	1947
Trial 12	2123	25	1123	1926
Mean	2126	27	1126	1932

Table 4.11: 12 iterations of grey level measurements for a chosen ROI when standards were fixed on the roof of the oral cavity. Mark 1 and Mark 2 indicate slice boundaries. Mean grey level is calculated for all pixels inside a circular area with radius of 10 (in pixels). STD represents the standard deviation of average grey level for slices between Mark 1 and Mark 2. HUs are calculated using the initial rescale slope and rescale intercept provided by the scanner. Calculated HUs are calculated using slope and intercept from the linear fit generated using the calibration standards.

	GSV	STD	HU	Calculated HU
Trial 1	2103	25	1103	2032
Trial 2	2110	21	1110	2045
Trial 3	2114	21	1114	2052
Trial 4	2113	24	1113	2050
Trial 5	2113	24	1113	2050
Trial 6	2105	24	1105	2036
Trial 7	2111	25	1111	2047
Trial 8	2112	22	1112	2048
Trial 9	2114	26	1114	2052
Trial 10	2116	27	1116	2055
Trial 11	2112	18	1112	2048
Trial 12	2111	23	1111	2047
Mean	2111	23	1111	2047

When measuring the volume of interest's grey level in the two scans, we discovered an average grey level of 1,126 ± 6 for the standards near the ROI and 1,111 ± 4 for the standards farther away from the ROI. Using the new rescale slope and rescale intercept generated by calibration standards, we found that the HU for the volume of interest are 1,932 ± 12 and 2,047 ±7, respectively, resulting in a 5.8% difference. Here the ± represent the standard deviation of twelve average grey level measurement and calculated HU values.

Comparing the two independent data sets using a Bootstrap analysis reveals a mean difference of 0 with a 90% confidence interval (-8.75, 7.84). These results illustrate that with 90% certainty the mean difference between HU values in a volume of interest will be between -8.75 and 7.84 when standards are placed near the first molar and for when standards are placed on the roof of the oral cavity.

Discussion:

In this thesis project a preliminary test quantitatively contrasted HU of a chosen VOI when standards are both near (1.25 cm) and farther away (3.25 cm) from the ROI. Previous results from characterization tests done for the Inveon scanner supported two conclusions. First, grey scale values vary up to 8.5% based on three scans of the same standard and parameters. Second, grey scale values vary up to 15.7% when the standard was moved ~4.4 cm. These results can be further studied in Appendix A. These conclusions indicate that machine start up issues as well as the location inside the machine affect grey scale values significantly. While previous groups have had success relating grey levels and HU by using one calibration curve for the entire FOV, [1] we fabricated standards small enough so that multiple standards with varying HUs fit along a chosen ROI. This allows for generation of a calibration curve for each ROI.

After looking at the intensity measurements collected for the phantom in the micro CBCT scanner, a positive correlation between the standards' grey levels and HA mass percentage was observed. Additionally, data collected from the two micro CBCT scans revealed that the grey level for each standard changes significantly between the two scans. However, the difference in grey level for each standard between the two scans decreases as the HA mass percentage of the standard increases. For example, the 68% by mass HA standard has the smallest difference between grey levels, while the 27% by mass HA standard has the greatest difference between grey levels. Based on the reporting from the characterization testing shown in Appendix A, we concluded that the observed difference in grey levels might also be a result of the standards' changes in location in the FOV. When characterizing the Inveon Micro PET/CT scanner, it was reported that about 7.2% variation in grey levels is due to the standard location inside the FOV. A 7.2% variation in grey level of the VOI resulted up to a 6% variation in HU for the VOI. Therefore, it can be concluded that about ~18% of the HU variation is likely due to the exomass

and beam hardening phenomena described in chapter 1. Therefore, placing the standards away from the ROI may result in HU measurement for a volume of interest that do not properly represent the attenuation of the region.

When studying the results obtained from the Carestream clinical CBCT scanner, it is evident that the difference in grey levels for each standard between scans is lower than observed in the Inveon scanner. The only exception to this statement is the difference between grey levels of the 45% by mass HA standard in the two scans. This difference is due to an underestimation of grey level when the standards were placed on the roof of the oral cavity. The underestimation most likely occurred because of the standards' orientation when fixed onto the roof of the oral cavity. The X-ray beams had to travel through both the 68% by mass HA standard and the aluminum standard. Thus, when the X-ray photons reached the 45% by mass HA standard, they had "hardened" (high energy). The data point labeled with an arrow in Figure 4.3 represents the underestimated sample. This data point is responsible for the visible difference in the two linear fits, and is therefore responsible for the 5.8% difference in calculated HU for the ROI between scans.

The Bootstrap analysis performed on the micro CBCT scanner data produced a 90% confidence interval (-300, -277) and a mean difference of -288. This means that when comparing the calculated HU values for the VOI, on average, the HU is 288 values lower when the standards are placed 2 cm farther away from the VOI. From this analysis, we concluded that the standards' grey level measurements changed significantly between scans. On the other hand, even though we observe a 5.8% difference in the ROI HU between the two Carestream scans, the difference does not prove to be significant (according to the Bootstrap analysis).

Conclusion:

As mentioned in chapter 1, the focus of this thesis was to design and fabricate a set of standards with HU numbers spanning 700-3000 HU while small enough so multiple standards with varying X-ray densities could fit along a ROI. Doing so, we hypothesized, would allow users to generate a calibration curve for a chosen VOI and correct any offsets due to exomass. While, commercially manufactured standards with X-ray densities in the desired range exist, there was a need for smaller sized standards. Through this thesis, we have displayed the ability to generate three standards made of hydroxyapatite and high-density polyethylene with HU ranging from ~700 to ~2800 HU. While production of a standard with 3000 HU was not successful, we were able to use an aluminum standard as a substitute. This thesis also demonstrated developed computer programs that were used to calibrate and analyze three-dimensional X-ray data.

While preliminary testing in the Clinical Carestream 9300 CBCT scanner did not support our hypothesis, promising discoveries were made using the research grade Inveon Micro PET/CT scanner. In conclusion, our calibration standards as well as our Matlab functions have proven to be useful in the research imaging. Currently, our standards are being tested on animals to determine if accurate quantitative measures could be obtained for mice spines. Eventually, we think we can develop our methods so that any researcher can use our standards and Matlab programs to quantitatively analyze bone using the Inveon system.

In the future, our goal is to use the calibration standards to relate grey levels from a CBCT scan to bone density. Doing so will allow us to relate quantitative measures outputted by a CBCT machine with stability of a dental implant. Future research is required to fabricate standards with a density between 1.3 g/cc - 1.8 g/cc. This can be achieved by producing HA particles with higher mass densities.

Reference:

[1] T. Reeves, P. Mah, and W. McDavid, “Deriving Hounsfield units using grey levels in cone beam CT: a clinical application,” *Dentomaxillofacial Radiol.*, vol. 41, no. 6, pp. 500–508, Sep. 2012.

Appendix A:

Appendix A is dedicated to a report written and presented to the personnel in the Case Center for Imaging Research, characterizing the Inveon Micro PET/CT scanner.

Background:

Cone beam CT (CBCT) scans, captured and processed using the Inveon PET-CT scanner, do not produce accurate Hounsfield Units (HU). Our goal is to determine what aspects of the data collection and analysis contribute to this inaccuracy and to develop augmented experimental and image processing techniques so that quantitatively reliable HU may be obtained.

Characterizing the machine:

Issue 1: How reproducible is the data using the same phantom in the same location?

Our first test involved imaging a single phantom consisting of a small cylindrical vial (8.0 mm diameter, 32.0 mm tall) filled with a lead nitrate solution in distilled water at a concentration of (0.15g/ml). We placed the phantom in the middle of the apparatus and collected three images, one right after the other, while keeping all of the scan parameters unchanged. The purpose of this test is to observe any difference in pixel value due to some inconsistency with the machine start up. Due to beam hardening, the gray scale values of all of the voxels in the phantom were not the same. Thus, we measured the average voxel intensity in a cylindrical volume. Given the average pixel value, we converted to HU by using an equation of a straight line with a slope equal to something called the “rescale slope” and the y-intercept equal to the “rescale intercept.” The rescale values are stored in the DICOMM file header and they are different for each image collected. We are not sure at this time how the instrument arrives at the values that end up being stored in the headers.

$$HHHH = RRSS * PPHH + RRII \quad EEEEuuEEEEEEEEEE\ AA. 1$$

In this equation, PV equals the average voxel value in the volume. While the RS and RI are the rescale slope and rescale intercept.

Table A.1: Gray Scale values and HU values outputted by the machine for a phantom. Each scan was done with the sample placed in the same location in the scanner.

Location of Lead Nitrate	Gray Value	HU
Mid-Mid A	21,662± 115	10,048
Mid-Mid B	20,906 ± 161	9,953
Mid-Mid C	20,449 ± 174	10,041

It is evident from table 1, that the gray scale values can vary by up to 8.47% while the HU can vary by up to 2.26% when we take the same scan three times. This shows that machine start-up issues affect gray scale values significantly. Much, but not all of this variation, is accounted for when converting the gray values into HU using the scanner’s provided parameters.

Issue 2: How reproducible is the data using the same phantom but moved to different locations within the scanner?

Our second test involved measuring the HU for a phantom as a function of location inside the scanner. To perform this test, we used the same phantom as in Issue 1. We moved our phantom around a cylindrical acrylic container with a diameter of 3.5 inches and height of 4 inches. We placed the phantom in four different locations labeled Top-Mid, Top-Back, Right-Mid and Right-Back. We recorded the Gray-values and the generated HU in table 2

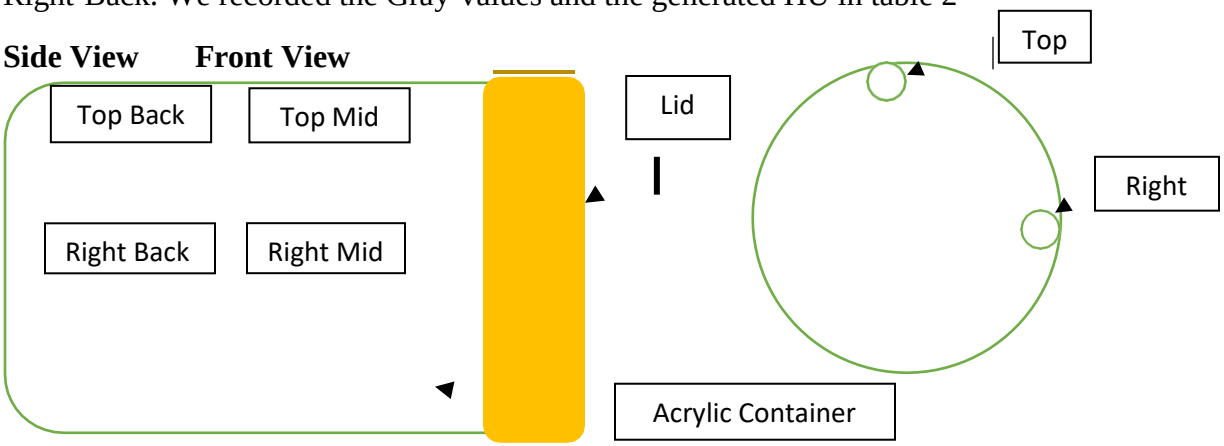


Figure A.1: Sketch of the side and front view of the cylindrical, acrylic container used as well as the four locations the standard was placed in each scan.

Table A.2: Gray Scale values and HU values outputted by the machine for a phantom of Lead Nitrate. Each scan was done with the sample placed in a different location in the scanner.

Location of Lead Nitrate	Gray Value	HU
Top-Mid	19,775 ± 74	10,637
Top-Back	18,926 ± 374	10,700
Right-Mid	22,038 ± 93	10,650
Right-Back	21,785 ± 357	10,800

It is evident from table 2, that the gray scale values can vary by up to 15.7% while the HU can vary by up to 5.57% when we move the phantom around by 1.75 inches. Focusing on the HU values, we can assume that ~2.3% of the variation seen here is due to start-up issues and the remaining 3.3% variation is due to changing locations.

Based on the fact that HU values were much more tolerant to moving the same phantom around in the scanner, we decided to look more closely at the specific quantitative values the instrument provided for the conversion of gray values to HU, the rescale slope and the rescale intercept. The conversion is accomplished using the by plugging in the gray scale value for PV in equation A.1.

Issue 3: Are the HU values for air and water equal to the expected values of -1000 and 0 respectively?

When analyzing a scan containing water and air, we found that the calculated HU for air was

-1029, which is off by 3% from the expected value of -1000 HU. The calculated HU for water was recorded at ~965, which is considerably higher than the expected value of 0. Based on this, we decided to create our own process for determining the rescale slope and rescale intercept parameters and to replace these values in the DICOMM header with our own Matlab function.

Determining rescale slope and rescale intercept for Water/Air scan:

Utilizing several Matlab functions written by us, we are capable of opening and reordering DICOMM files outputted by the Inveon system. This is necessary because the order of the slices is not contained in the filenames. The information is in the header of each file. Once again, we measured the average voxel intensity in a cylindrical volume.

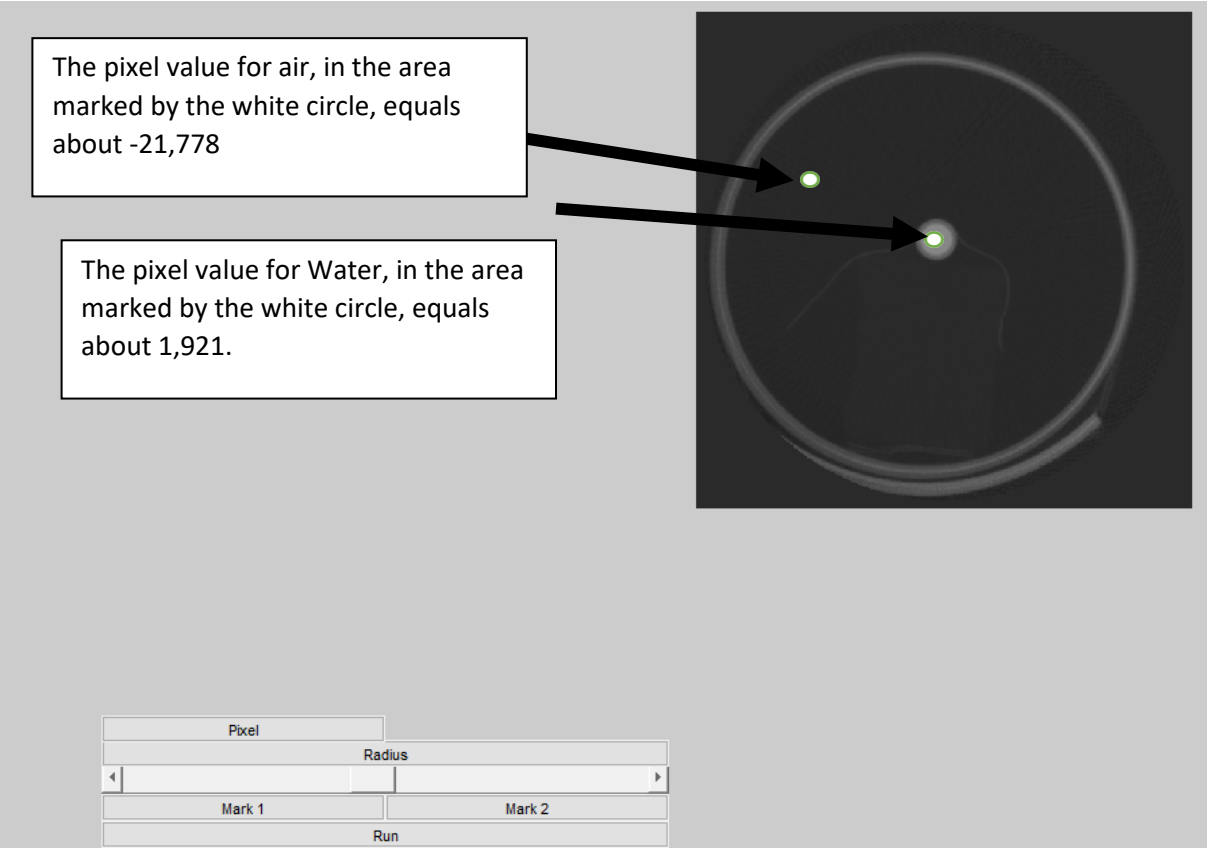


Figure A.2: Image of the GUI used to analyze a cylindrical volumetric HU of our samples, using a Matlab function written by our lab. Mark 1 and Mark 2 buttons are used to indicate slice boundaries, while the radius button allows the user to indicate the radius of the cylinder.

We then imaged a water phantom. Table 3 shows the measured gray scale value, the HU output using the Inveon rescale slope and intercept, the expected HU value obtained from the NIST website, and finally, the HU value obtained when we generated our own rescale slope and intercept using the measured Gray values and the NIST HU values to plot a best-fit line.

Table A.3: Gray Scale values and HU values outputted by the machine for water and air.

Material	Gray Value	Expected HU	Inveon HU	Percent Error %	Our HU
Air	-21,778 ± 47	-1,000	-1029	2.9	-1,000
Water	1,921 ± 61	0	965	~965%	0

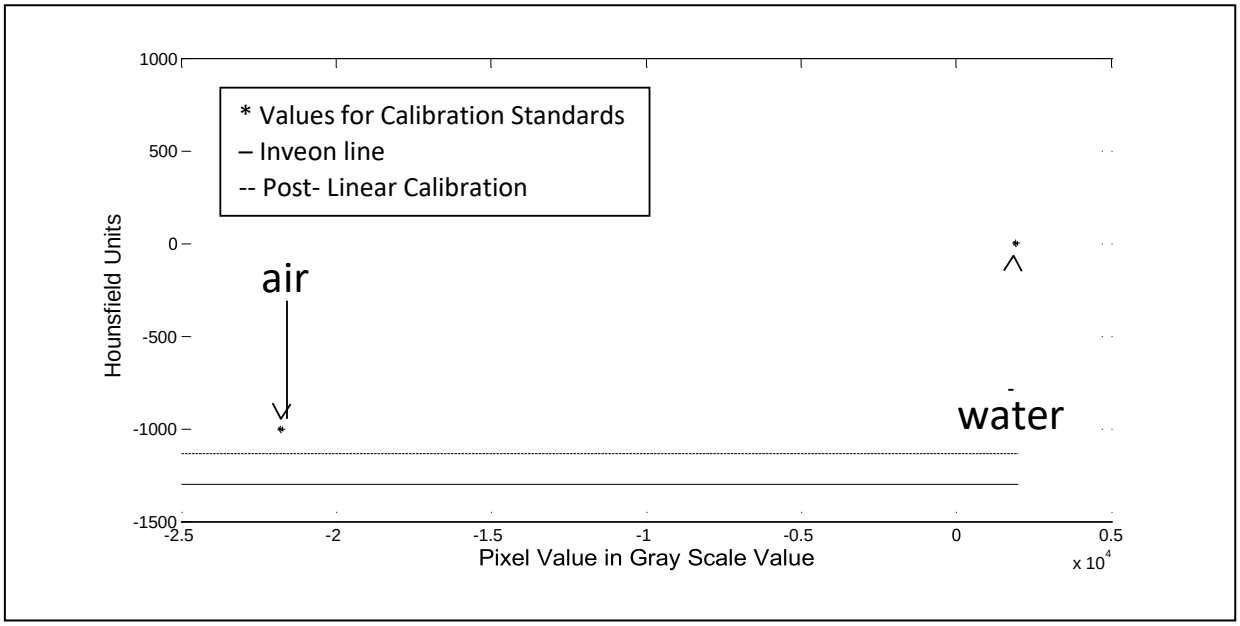


Figure A.3: This plot shows the Gray Scale Values for water and air plotted at their corresponding expected HU of 0 and -1000 respectively. The two lines represent the conversion between gray scale value and HU using the rescale slope and intercept (0.0841 and 803 respectively), provided by the machine (Solid) and (0.0422and -81 respectively), defined by the line of best fit to our phantom measurements (Dashed).

Based on the data above, if a user wants their HU to be accurate in the soft tissue region, they need to ensure that there is a region of their scanned volume that contains pure water and another that contains pure air. If their sample does not contain such regions, then they need to include water and/or air phantoms with each scan. They must then find their own rescale slope and intercepts by finding the best fit line to air and water. Finally, they must insert these values into their headers of their DICOMM files.

Issue 4: If scans of objects substantially more x-ray dense than water (like bone or mineral deposits in vasculature) are scanned, is it necessary to use phantoms other than air and water to determine the rescale slope and intercept.

To address this problem, we made four water phantoms containing lead nitrate at concentrations that produce x-ray attenuation coefficients ranging from trabecular to cortical bone. To compute the theoretical HU for these phantoms, we can use equation A.2 provided below:

$$HHHH = \frac{uu_{ii} - uu_{wwwwwwwwww}}{uu_{wwwwwwwwww} - uu_{wwiiww}} * 1000 \qquad EEEEuuEEEEEEEEE\ AA.2$$

Where u_i is the attenuation coefficient of our phantom at a given x-ray photon energy. We obtained u_i from the NIST website
http://physics.nist.gov/PhysRefData/Xcom/html/xcom1.html.
This site lists the u_i as a function of the x-ray energy incident on the sample.

Cone beam scanners have x-ray sources with a broad energy distribution. Therefore, to locate the appropriate theoretical attenuation coefficient of our phantoms from the NIST tables, we first had to figure out the effective energy of the Inveon PET-CT. The effective energy is "the energy of a mono-energetic beam of photons that has the same penetrating ability as the spectrum of photons" emitted from the scanner's x-ray source (http://www.sprawls.org/ppmi2/RADPEN/)

Finding the Effective Energy:

We first determined the half value layer (HVL) of the Inveon PET-CT x-ray source operated at an accelerating voltage of 80 kV and beam current of 500 uA. The HVL is the thickness of the aluminum sheet that absorbs half of the x-ray photons emitted from the x-ray source. We purchased aluminum standards made for such testing (RDP Inc. cat. #115-500) and used an ion detecting chamber and an electrometer (660, Victoreen, Cleveland, OH) to record the number of photons that penetrated through different thicknesses of aluminum.

Table A.4: Areal ion density detected by the ion detector through different thickness of aluminum. NOTE: These tests were all done in scout view.

Thickness of Al Plate(mm)	Mean (Ions/Area)
0	6.70 ±0.43
0.1	4.35±0.14
0.2	3.57±0.19
0.3	3.35±0.14
1	2.07±0.08
2	2.01±0.03

We plotted the data (Ions/Area vs. plate thickness) and curve fit it to a decaying exponential

$$II = AA + BBee^{-uuww} \qquad EEEEuuEEEEEEEEE\ AA. 3$$

The HVL was then found from the fit by identifying the x-axis value at which the y-axis value was ½ the value with no aluminum plate in the beam path. The HVL for this machine under the operating conditions we used is about 0.24 mm of aluminum.

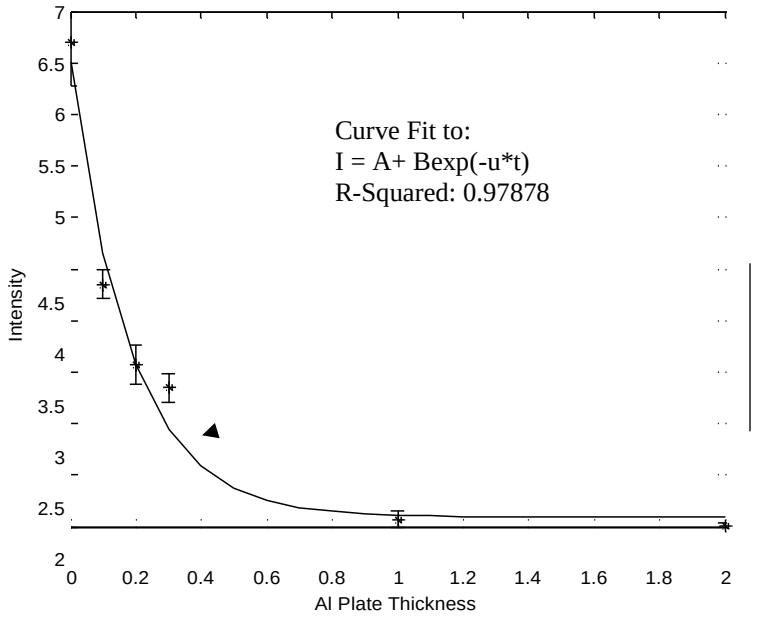


Figure A.3: This plot generated using Matlab, show a curve plotted using the equation obtained from the curve fit using Origin.

To find the effective energy, we used the standard x-ray absorption equation

$$II = II_0 ee^{-uuww} \; EEEEuuEEEEEEEEEE \; AA. 4$$

We set t=HVL, $I/I_0=1/2$, and then solved for μ. Using the NIST website for x-ray absorption coefficients, we looked up the attenuation values for aluminum as a function of energy. The energy that corresponded to the attenuation value computed for μ was 31kV. This is the effective energy of the Inveon scanner when the accelerating voltage is set to 80 kV and the beam current is set to 500 uA.

Note: if other users wish to use different accelerating voltages and/or different beam currents, they will need to reproduce this measurement at their desired X-ray source settings. The aluminum plates used to make this measurement are in a cabinet in the room housing the Inveon PET-CT scanner.

Choosing phantom material:
Due to its high x-ray density, iodine is commonly used as a contrast agent in CT scans. Therefore, we first thought of using water/iodine phantoms for our experiment. However, after looking at the plot of attenuation coefficient as a function of x-ray photon energy (Figure 4), we observed non-uniformity due to core shell transitions that appear a bit below the accelerating voltage, 80kV. Since bremsstrahlung radiation peaks that describe the shape of the X-ray source are generally strongly asymmetric with a long high energy tail, we were concerned that this hitch in the iodine spectrum might be strongly sampled by the machine's x-ray source. Therefore, we decided to make our phantoms with lead nitrate, which contains a more uniform attenuation decline in the relevant energy range.

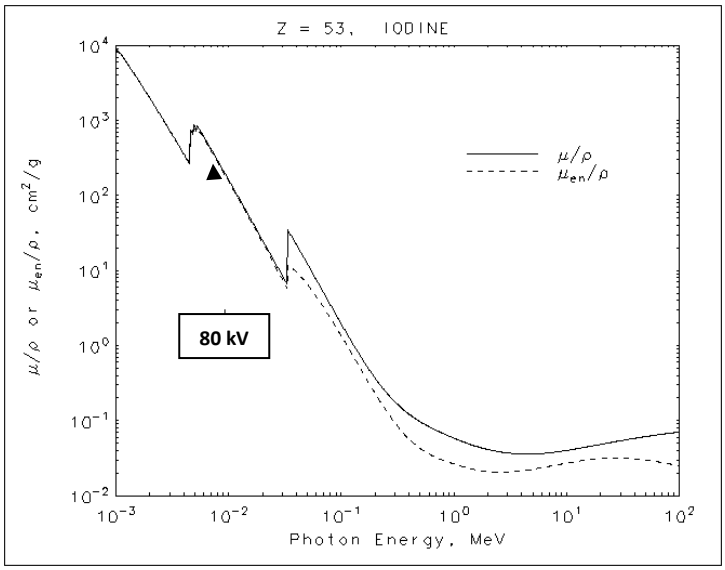


Figure A.4: plot of attenuation coefficient as a function of x-ray photon energy for Iodine

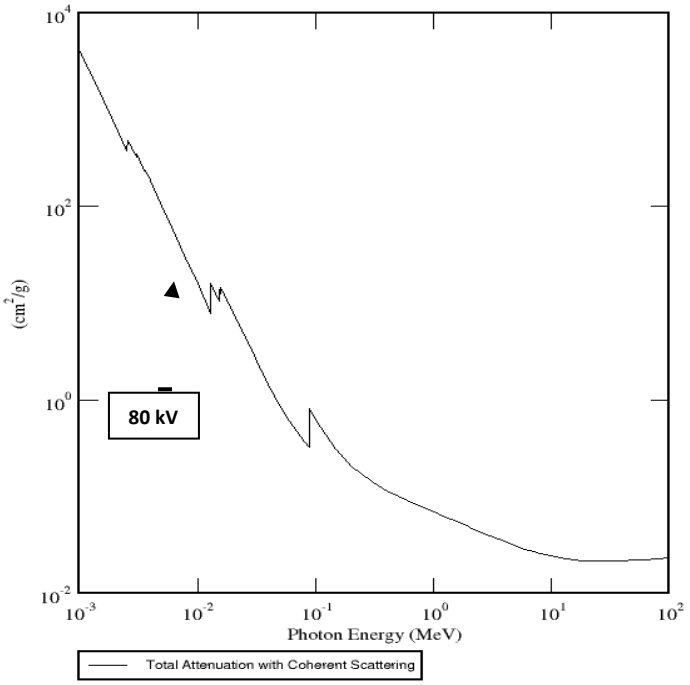


Figure A.5: plot of attenuation coefficient as a function of x-ray photon energy for Lead Nitrate

We then imaged four phantoms that contained different weight percentage of lead nitrate that correspond to the relevant HU range for bone. Table 5 shows each phantom, the measured gray scale value, the HU output using the Inveon rescale slope and intercept, the expected HU value obtained from the NIST website. Each phantom was imaged only once. The ± values in the gray value column arise because the values reported are mean averages through a volume of interest within the phantom.

Table A.5: Gray Scale values and HU values outputted by the machine for four different standards. Each scan was done with the sample placed in the same location in the scanner.

% by Mass of Lead Nitrate	Gray Value	Expected HU (NIST)	Inveon HU	Percent Error %
1.5	-7,920±159	741	2,515	239
3.5	14,382±312	1,755	4285	144
6.5	-1,849±77	3,339	6,625	98
10.5	24,634±235	5,576	8,530	53

We then imaged the four phantoms in a single field of view (FOV). Table 6 shows each phantom, the measured gray scale value, the HU output using the Inveon rescale slope and intercept, the expected HU value obtained from the NIST website, and finally, the HU value obtained when we generated our own rescale slope and intercept using the measured gray values and the NIST HU values to plot a best-fit line. Each phantom was imaged only once.

Table A.6: Gray Scale values and HU values outputted by the machine as well as HU values outputted by our linear calibration for four different standards in the same field of view.

% by Mass of Lead Nitrate	Gray Value	Expected HU (NIST)	Inveon HU	Percent Error %	Our HU	Percent Error %
1.5	-15,867± 136	741	2254	204	390	47
3.5	-12448± 387	1,755	3582	104	2,099	20
6.5	-8971± 760	3,339	4931	48	3,837	15
10.5	-6476 ± 1036	5,576	5900	6	5,085	9

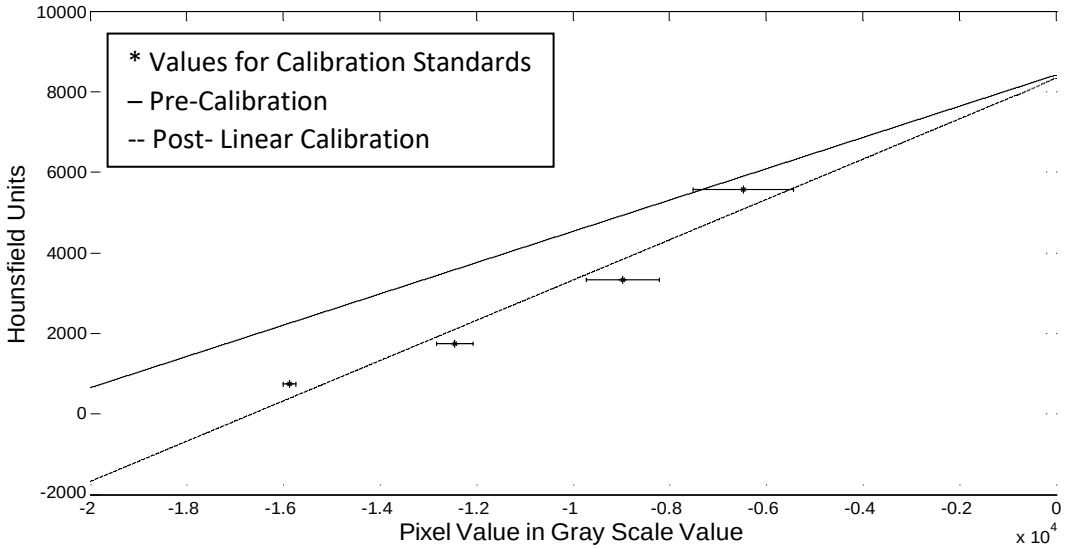


Figure A.6: This plot shows the Gray Scale Values for the four Lead Nitrate corresponding with the expected HU for the four standards. The two linear line represent the conversion between gray scale value and HU using the rescale slope and intercept, 0.3882 and 8,414 respectively, provided by the machine (Solid) and rescale slope and intercept, 0.4999 and 8,322 respectively, defined by the line of best fit to the four standards (Dashed).

Table A.5 shows a substantial improvement in the recovery of NIST HU, for the more dilute phantoms, compared to the Inveon system's default output. The average percent error over the bone range drops from 139% to 23%. However, there are still fairly large errors when trying to calibrate over this entire range.

Gray values are related to attenuation coefficients through the standard exponential x-ray absorption equation. The use of a straight line model to convert from gray values to HU is based on a 1st order Taylor series expansion of this exponential. The question arises as to how large a range of gray values can be accommodated when expanding about a given attenuation coefficient. It would appear from the quality of the fit in Figure A.6 and the percent errors of our HU in Table A.5, that the range of attenuation coefficients that spans from cancellous to cortical bone exceeds the linear range of the expansion. Due to our results in Table A.5, we have developed a fitting procedure that uses the full exponential function.

Table A.7: Gray Scale values and HU values outputted by the machine as well as HU values outputted by our exponential calibration for four different standards in the same field of view.

% by Mass of Lead Nitrate	Gray Value	Expected HU (NIST)	Inveon HU	Percent Error %	Our HU	Percent Error %
1.5	-15,958 ±174	741	2,219	199	811	9
3.5	-12,530 ±506	1,755	3,549	102	1,705	-3
6.5	-9,499 ±914	3,339	4,726	42	3,321	-1
10.5	-7,113 ±1166	5,576	5,652	1	5,630	1

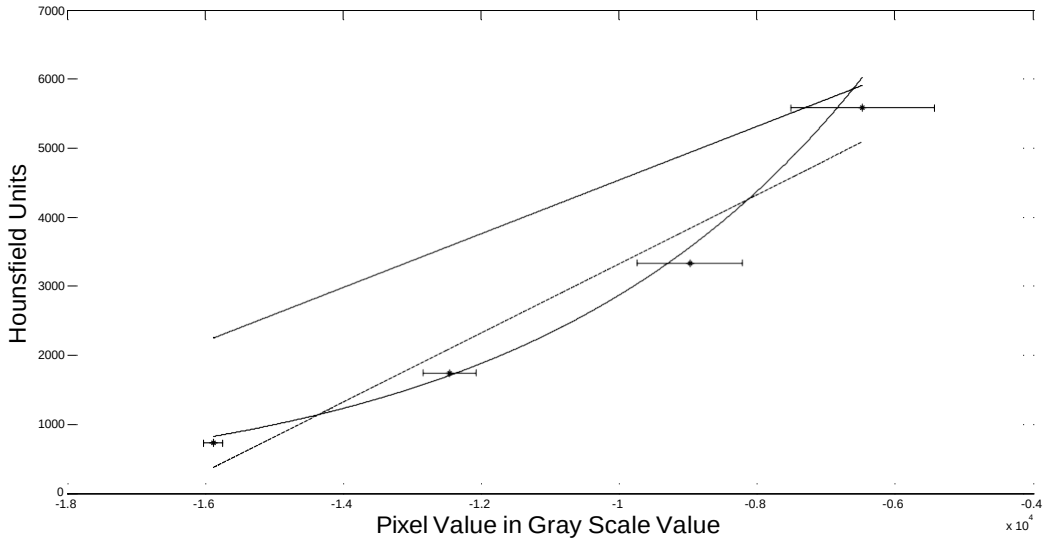


*Figure A.7: This plot shows the Gray Scale Values for the four Lead Nitrate corresponding with the expected HU for the four standards. The two linear line are similar as the lines in Fig. 6. The exponential line represents the exponential function used to fit the data using the equation HU = a*exp (b*GSV).*

Problem Statement #2: Eliminate Beam Hardening using Matlab algorithm.

We are currently collaborating with Dr. Wilson's lab to develop a Matlab algorithm, which corrects for beam hardening generated by components inside the Field of View (FOV). This should substantially reduce the +/- values reported for HU in all the above tables since the main source of that error is from beam hardening. One of Dr. Wilson's PhD. students has been developing a Matlab function that corrects for beam hardening in a clinical fan beam CT scan. He has agreed to work with us to alter his function to correct for beam hardening using the CBCT scans of our Lead Nitrate phantoms.

Problem Statement #3: Eliminate Exomass Problem

We have worked to design and produce a set of standards small enough so that multiple standards can fit along the region of interest (ROI). Lining the ROI with a small set of standards will allow us to generate calibration plots for each ROI and correct any offsets due to exomass (x-ray dense regions outside the field of view).

We are in the process of producing our own set of ceramic-polymer phantoms with densities ranging from trabecular to cortical bone. After producing these phantoms, we plan to fix multiple sets of these standards in the field of view (FOV) so that calibration curves can be produced for ROI's near each set of standards. These standards will effectively replace the lead nitrate standards used above. By virtue of their size and solid nature, they should be able to easily be used in every scan so that sample specific rescale slope and intercept values can be determined with each scan. Since the standards can be placed close to the volume of interest, exomass attenuation should affect the standards similarly to the unknown sample. Thus, calibration using the small phantoms should allow the user to overcome the exomass problem.

Appendix B:

Start up: Run Analysistype

```
analysistype = input('Please indicate what plane you would like to use for analysis (Axial = 1),
(Sagittal = 2), (Coronal = 3 \n');
%This script allows the user to first indicate at which orientation they
%want to open the image in.
if analysistype == 1
    DICOMViewer_Front_Axial
elseif analysistype == 2
    DICOMViewer_Side_Sagittal
elseif analysistype ==3
    DICOMViewer_Top_Coronal
end
```

Published with MATLAB® R2017a

Interface: DICOMViewer

There are three versions of DICOMViewer. Each one opens the DICOM files in a particular orientation.

```
function DICOMViewer_Top_Coronal
```

```
%This code actually changes the rescale slop and intercept using either a
%linear or an exponential fit. -- KV
clear global
close all
%Set all of your global variables. -- KV
firstDir = cd;
%evaltype is to determine which kind of calibration is going to be done.
%The function is written so everytime it is ran it automatically functions
%at a evaltype of 3 which is just for making measurements. -- KV
evaltype = 3;
%mark1 and mark2 are used to indicate the first and last slice you want to
%make measurements in between. -- KV
global mark1
mark1 = 1;
global mark2
mark2 = 1;
n = 1;
%n reperesent the slice you are viewing, while radius and is the radius of
%the cylinderical volume of interest. cordinates1 and cordinates3 mark the
%center of the region of interest in mark1 slice and mark2 slice
%respectively. -- KV
radius = 10;
cordinates1 = 0;
cordinates3 = 0;
%RS and RI is the rescale slope and rescale intercept for the linear
```

```
%calibration. RS and RI are obtained from generating a calibration curve
%from imaging standards with known HU values. -- KVV
global RS
RS = [];
global RI
RI = [];
%Just in case we choose to calibrate using a exponential curve we have
%introduced coeffa and coeffb as the rescale slopes and intercept for the
%exponential calibration using the equation coeffa(exp(coeffb*PV)).
%coeffa and coeffb are obtained from generating a calibration curve
%from imaging standards with known HU values. -- KV.
global coeffa
coeffa = [];
global coeffb
coeffb = [];
[dirname] = uigetdir('Please choose dicom directory');
%ginfo1 is used here to store the dicominfo information for each slice in
%the Generate3dMatrix -- KV
global ginfo1
global calibtype
%ginfo1 is used in regular generate3dMatrix, ginfo is used in
%Generate3dMatrixbrandon
%generate3dMatrix function is used for measurements, while
%Generate3dMatrixbrandon is written for calibrations.
%(Generate3dMatrixbrandon is name so because the code was written for
%Brandon W. project) -- KV
matrix = Generate3dMatrixCBCT(dirname);
```

This function starts the gui and asks for what kind of evaluation you want to do -- KV

```
    function start(hObject, event)
        evaltype = input('Air/Water Calibration(1), HA-HDPE Calibration (2), Measurement (3)\n');
        if evaltype == 1;
            disp('1)Please start with chosing a mark1. 2)Then press the radius button to choose a
center point for the ROI of air and a second point to measure the distance between the center
point and the boarder of the ROI. 3) Choose a mark2 and press the pixel button to choose a center
point for the ROI in mark2. 4)Press Run, repeat steps 2 and 3 for a ROI containing water.')
        elseif evaltype == 2;
            disp('1)Please start with chosing a mark1. 2)Then press the radius button to choose a
center point for the ROI in the lowest dense sample and a second point to measure the distance
between the center point and the boarder of the ROI. 3) Choose a mark2 and press the pixel button
to choose a center point for the ROI in mark2. 4)Press Run, repeat the steps for a ROI containing
next standard.')
        elseif evaltype == 3;
            disp('1)Please start with chosing a mark1. 2)Then press the radius button to choose a
center point for the ROI and a second point to measure the distance between the center point and
the boarder of the ROI. 3) Choose a mark2 and press the pixel button to choose a center point for
the ROI in mark2.')
        end
    end
```

This function is updating the image we see as we scroll through the z slices

```
%note here the 'n' is placed in the 2nd column of the matrix so that the slices
%analyzed are the Coronal slices-- KV
    function updateImage(hObject,event)
        n = uint16(get(hObject,'Value'))
        imshow(squeeze(matrix(n,:,:)),[]);
        drawnow;
    end
```

This function is used to go back and forth between mark1 and mark2 for

```
%the water/air calibration to ensure that the number of slices measured for
%air and water is constant. -- KV
    function updateImage1(hObject,event)
        n = mark1;
        imshow(squeeze(matrix(n,:,:)),[]);
        drawnow;
    end
    function  updateImage2(hObject,event)
        n = mark2;

        imshow(squeeze(matrix(n,:,:)),[]);
        drawnow;
    end
```

This function is indicating the Z slice we choose for Mark1 -- KV

```
    function setmark1(hObject,event)
        mark1 = n;
        set(btn1, 'string', strcat('Mark1: ',num2str(n)));
    end
```

This function is indicating the Z slice we choose for Mark2 -- KV

```
    function setmark2(hObject,event)
        mark2 = n;
        set(btn2, 'string', strcat('Mark2: ',num2str(n)));
    end
```

Inserting the x and y coordinates for the first and second point we choose to indicate the radius

```
 %cordinates1 is the center of the of the standard at Mark1. cordinates2
 %is the outer border of the standard. cordinates3 below is the
 %center of the of the standard at Mark2. -- KV
    function getradius(hObject,event)
        cordinates1 = ginput(1)
        cordinates2 = ginput(1);
        X1 = cordinates1(1);
```

```
        Y1 = cordinates1(2);
        X2 = cordinates2(1);
        Y2 = cordinates2(2);
        radius = sqrt((X2-X1)^2 + (Y2-Y1)^2)
        set(getRadius, 'string', strcat('Radius: ',num2str(radius)));
    end
```

Getting the pixel value from the ginput for the center of the ROI in

```
%in the mark2 slice-- KV
    function getpoint(hObject,event)
        cordinates3 = ginput(1);
        xvalue = cordinates3(1)
        yvalue = cordinates3(2)
        set(infobtn, 'string', strcat('X= ', num2str(xvalue), 'Y =',num2str(yvalue)))
    end
```

Choosing the center point and storing the x and y coordinates and finding the center axis of the standard

```
    %We make the user choose two center points, CenterM1 and CenterM2, just in case the sample is
off axis.
    %By doing this the center of the ROI will shift as a function of the
    %slice number. -- KV

    function run(hObject, event)
        cd(firstDir);
             cordinates = cordinates1;
        locationX = round(cordinates(1))
        locationY = round(cordinates(2))
        set(initRun, 'string', strcat('Center: ',num2str(cordinates)));
        CenterM1 = cordinates1;
        CenterM2 = cordinates3;
        deltay = double(CenterM2(2))-double(CenterM1(2))
        deltaz = mark2 - mark1
        my = double(deltay)/double(deltaz)
        by = double(CenterM1(2)) -double(my)*double(mark1)
        deltax = double(CenterM2(1))-double(CenterM1(1))
        mx = double(deltax)/double(deltaz)
        bx = double(CenterM1(1))- double(mx)*double(mark1)
    %Once the center of the ROI has been determined in each slice we get to
    %calibrating. evaltype == 1 is for a air/water calibration
    %calib indicates which portion of the calibration the code is at.
    %calib = 1 means that the air is being calibrated while calib = 2 means
    %the water is being calibrated. -- KV
        if evaltype == 1;
            calib = 1;
            for calib = [1:2]
               if calib == 2;
                   updateImage1(hObject,event)
                   getradius(hObject,event)
                   updateImage2(hObject,event)
```

```
                    getpoint(hObject,event)
                    cordinates = cordinates1;
                    locationX = round(cordinates(1))
                    locationY = round(cordinates(2))
                    set(initRun, 'string', strcat('Center: ',num2str(cordinates)));
                    %Here we are again determining the center points for the
                    %water ROI and aligning the ROI so the center point
                    %varies as a function of the slice number. -- KV
                    CenterM1 = cordinates1;
                    CenterM2 = cordinates3;
                    deltay = double(CenterM2(2))-double(CenterM1(2))
                    deltaz = mark2 - mark1
                    my = double(deltay)/double(deltaz)
                    by = double(CenterM1(2)) -double(my)*double(mark1)
                    deltax = double(CenterM2(1))-double(CenterM1(1))
                    mx = double(deltax)/double(deltaz)
                    bx = double(CenterM1(1))- double(mx)*double(mark1)
                end


                radius = input('Please specify what radius you want to start with\n');
                %ensures mark1 is less than mark2
                if(mark1>mark2)
                    tempVar = mark1;
                    mark1=mark2;
                    mark2 = tempVar;
                end
                count = int16(mark2-mark1);
                struct=[count];
                struct1=size(matrix);
                struct2 = [struct1(1), struct1(2)];
                for slicenumber = mark1:mark2
                    locationX = (double(slicenumber)*double(mx))+bx;
                    locationY = (double(slicenumber)*double(my))+by;
                    Center = [double(locationX), double(locationY)];
                    %CircularAVG is a minor function written by our group
                    %which finds the marked center pixel of the ROI and
                    %calculates the average pixel value for all of the pixel
                    %within a circular area with a radius inputed by the
                    %user. -- KV
                    tempStruct = CircularAVG(squeeze(matrix(slicenumber,:,:)), radius, Center(2),
Center(1));
                    struct(slicenumber - mark1 + 1) = tempStruct;
                end
                totalAverage = mean2(struct)
                STD =  std(double(struct))
                if calib == 1;
                    PVa = totalAverage;
                elseif calib == 2;
                    PVw = totalAverage;
                end
                calib = calib +1;
            end

            close all
            HUa = -1000;
            HUw = 0;
```

```
            HounsfieldUnitmat = [HUa;HUw];
            Dmat = [PVa;PVw];
            %Here is where we use a linear fit to relate the grey level
            %pixel values of the air and water with their known
            %Hounsfield Unit. -- KV
            rescale = polyfit(Dmat, HounsfieldUnitmat, 1);
            RS = rescale(1);
            RI = rescale(2);
            matrix = Generate3dMatrixbrandon(dirname);
            disp('please run Analysistype again for the new set of data');

        elseif evaltype == 2;
             %calibration using HA-HDPE Samples
            calib = 1;
            %Here calib indicates where standard is in use. calib = 1
            %corresponds to measuring the pixel value for the first, least
            %dense standard and calib = 4 corresponds to measuring the
            %pixel value for the fourth, highest denst standard.
            for calib = [1:4]
                if calib > 1
                    sprintf('please indicate Mark1, Mark2 and center of standard #%d',calib)
                    %Here we have introduced a pause, because we want the
                    %user to have a chance to go back and change the mark1
                    %and mark2 just in case the standards are not all
                    %visible in the same slices. -- KV
                    pause
                    updateImage1(hObject,event)
                    getradius(hObject,event)
                    updateImage2(hObject,event)
                    getpoint(hObject,event)
                    cordinates = cordinates1;
                    set(initRun, 'string', strcat('Center: ',num2str(cordinates)));
                    CenterM1 = cordinates1;
                    CenterM2 = cordinates3;
                    deltay = double(CenterM2(2))-double(CenterM1(2))
                    deltaz = mark2 - mark1
                    my = double(deltay)/double(deltaz)
                    by = double(CenterM1(2)) -double(my)*double(mark1)
                    deltax = double(CenterM2(1))-double(CenterM1(1))
                    mx = double(deltax)/double(deltaz)
                    bx = double(CenterM1(1))- double(mx)*double(mark1)
                end
                %The radius button used above
                %use just used to measure the radius here is where the
                %radius value is inputed by the user. -- KV
                radius = input('Please specify what radius you want to start with\n');
                %ensures mark1 is less than mark2
                if(mark1>mark2)
                    tempVar = mark1;
                    mark1=mark2;
                    mark2 = tempVar;
                end
                count = int16(mark2-mark1);
                struct=[count];
```

```
                struct1=size(matrix);
                struct2 = [struct1(1), struct1(2)];
                for slicenumber = mark1:mark2
                    locationX = (double(slicenumber)*double(mx))+bx;
                    locationY = (double(slicenumber)*double(my))+by;
                    Center = [double(locationX), double(locationY)];
                    tempStruct = CircularAVG(squeeze(matrix(slicenumber,:,:)), radius, Center(2),
Center(1));
                    struct(slicenumber - mark1 + 1) = tempStruct;
                end
                totalAverage = mean2(struct);
                STD  = std(double(struct));
                if calib == 1;
                    PV1 = totalAverage
                    PV1std = STD
                elseif calib == 2;
                    PV2 = totalAverage
                    PV2std = STD
                elseif calib == 3
                    PV3 = totalAverage
                    PV3std = STD
                elseif calib == 4
                    PV4= totalAverage
                    PV4std = STD
                end
            end
            close all
            HU1 = 723;
            HU2 = 1447;
            HU3 = 2728;
            HU4 = 4640;
            HounsfieldUnitmat = [HU1;HU2;HU3; HU4];
            Dmat = [PV1; PV2; PV3; PV4];
            %Here we are solving for the rescale slope and intercept for
            %a linear calibration curve. Here we also solve a rescale coeffs
            %for an exponential calibration curve Aexp(PV*B). -- KV
            rescale = polyfit(Dmat, HounsfieldUnitmat, 1);
            f1 = fit(Dmat, HounsfieldUnitmat, 'exp1');
            RS = rescale(1);
            RI = rescale(2);
            coeffa = f1.a;
            coeffb = f1.b;
            FixHU = (Dmat*RS) +RI;
            fixHU = coeffa*exp(coeffb*Dmat);
            figure
            plot(Dmat, FixHU, 'r--')
            hold on
            plot(Dmat, fixHU, 'b--')
            plot(Dmat, HounsfieldUnitmat, 'k+', 'MarkerSize', 15)
            hold off
            %Here we generate a plot for the user to look at which
            %calibration curve seems to fit their data best, an exponential
            %vs a linear. The calibtype indicates which calibration curve
            %the user chose to use for the calibration. -- KV
```

```
            calibtype = input('Please indicate if you want a linear (1) or an exponential (2)
calibration curve \n');
            matrix = Generate3dMatrixbrandon2(dirname);
            disp('please run DICOMViewerSlider5calibration again for the new set of data');

       elseif evaltype==3;
           %When evaltype = 3 then the interface is only used to make
           %measurements of the ROI. These measurements are usually made
           %after calibration has been done. -- KV
           %Moreover, you can choose to calculate the avg HU for cylinders
           %with a variety of radii. -- KV
            numbofradi = input('please insert how many radi you want to evalute\n');
            if numbofradi>1;
                firstradius = input('Please specify what radius you want to start with\n');
                delta = input('Please specify the incrament you want to increase the radius
by\n');
                lastradius = input('Please specify what radius you want to finish with\n');
                radius = zeros(((lastradius-firstradius)/delta)+1,1);
                tempradius = firstradius;
                for i = 1:length(radius)
                    radius(i) = tempradius;
                    tempradius = tempradius + delta;
                end
            else
                radius = input('Please specify what radius you want to start with\n');
            end
            %ensures mark1 is less than mark2
            if(mark1>mark2)
                tempVar = mark1;
                mark1=mark2;
                mark2 = tempVar;
            end
            count = int16(mark2-mark1);
            struct=[count];
            struct1=size(matrix);
            struct2 = [struct1(1), struct1(2)];
            m = 0;
            %Here we ask the user if they calibrated using a linear or
            %exponential calibration just so the right equation is used to
            %calculated the corrected HU from the grey level pixel values.
            %-- KV
            calibtype = input('Please indicate if you chose a linear (or air/water calibration)
(1) or exponential (2) calibration above \n');
            for p = 1:length(radius)    %CHANGE THIS WHEN YOU WANT DIFFERENT RADIUS.-- KV
                m = m+1;
                for slicenumber = mark1:mark2
                    locationX = (double(slicenumber)*double(mx))+bx;
                    locationY = (double(slicenumber)*double(my))+by;
                    Center = [double(locationX), double(locationY)];
                    tempStruct = CircularAVG(squeeze(matrix(slicenumber,:,:)), radius(p),
Center(2), Center(1));
                    struct(slicenumber - mark1 + 1) = tempStruct;
                end
                HUstruct = [];
```

```
                %HU calculation for a linear calibration. -- KV
                if calibtype == 1
                for structnumber = 1:length(struct)
                    rescaleint(structnumber)= ginfo1{structnumber-1+mark1}.RescaleIntercept;
                    rescaleslope(structnumber)= ginfo1{structnumber-1+mark1}.RescaleSlope;
                    struct = double(struct);
                    HUstruct(structnumber) =
(rescaleslope(structnumber)*struct(structnumber))+rescaleint(structnumber);
                end
                %HU calculation for a exponential calibration. -- KV
                elseif calibtype == 2
                    for structnumber = 1:length(struct)
                    rescaleint(structnumber)= ginfo1{structnumber-1+mark1}.RescaleIntercept;
                    rescaleslope(structnumber)= ginfo1{structnumber-1+mark1}.RescaleSlope;
                    struct = double(struct);
                    HUstruct(structnumber) =
rescaleint(structnumber)*exp(rescaleslope(structnumber)*struct(structnumber));
                    end
                end
               figure
               plot(HUstruct);
               xlabel('Number of Slices')
               ylabel('PV in HU')
               sixstruct = int16(struct);
               sixstruct = sixstruct +32767;
               total(m) = mean2(sixstruct)
               totalAverage(m) = mean2(struct)
               STD(m) = std(double(struct))
               HU(m) = mean2(HUstruct)
           end


           pp = radius(length(radius));   %CHANGE THIS WHEN YOU WANT DIFFERENT RADIUS.-- KV
           tempStruct = Blackingradius(squeeze(matrix(mark2,:,:)), pp, Center(2), Center(1));
           struct2 = tempStruct;
           figure
           imshow(squeeze(struct2(:,:)),[]);
        end

       if evaltype == 3;
           figure
           plot(HU, 'bo')
           ylabel('Total Avg Pixel Value (HU)')
           xlebel('Radius number')
       end

    end

f=figure(1);

slider = uicontrol('Parent',f,'Style','slider','Position',[81,114,420,23],'min',0,
'max',size(matrix,1));

btn1 = uicontrol('Style', 'pushbutton', 'String', 'Mark 1','Position', [81,34,210,20],'Callback',
@(hObject, event) setmark1(hObject, event));
```

```
btn2 = uicontrol('Style', 'pushbutton', 'String', 'Mark 2','Position',
[291,34,210,20],'Callback', @(hObject, event) setmark2(hObject, event));

initRun = uicontrol('Style', 'pushbutton', 'String', 'Run','Position', [81,14,420,20],'Callback',
@(hObject, event) run(hObject, event));
initStart = uicontrol('Style', 'pushbutton', 'String', 'Start','Position',
[81,54,420,20],'Callback', @(hObject, event) start(hObject, event));

getRadius = uicontrol('Style', 'pushbutton', 'String', 'Radius','Position',
[81,74,420,20],'Callback', @(hObject, event) getradius(hObject, event));

mTextBox = uicontrol('style','text','Position', [81,0,420,14])
infobtn = uicontrol('Style', 'pushbutton', 'String', 'Pixel', 'Position', [81,92,210,20],
'Callback', @(hObject, event) getpoint(hObject, event));

addlistener(slider,'ContinuousValueChange',@(hObject, event) updateImage(hObject, event));

%display%
ax1=axes('parent',f,'position',[0.13 0.39  0.77 0.54]);
imshow(squeeze(matrix(n,:,:)),[]);

global matrix2
matrix2 = matrix;

end
```

Published with MATLAB® R2017a

```
function DICOMViewer_Side_Sagittal
```

```
%This code actually changes the rescale slop and intercept using either a
%linear or an exponential fit. -- KV
clear global
close all
%Set all of your global variables. Global variables are capable of being
%accessed by any function, not just this one. -- KV
firstDir = cd;
%evaltype is to determine which kind of calibration is going to be done.
%The function is written so everytime it is ran it automatically functions
%at a evaltype of 3 which is just for making measurements. -- KV
evaltype = 3;
%mark1 and mark2 are used to indicate the first and last slice you want to
%make measurements in between. -- KV
global mark1
mark1 = 1;
global mark2
mark2 = 1;
%n reperesent the slice you are viewing, while radius and is the radius of
%the cylinderical volume of interest. cordinates1 and cordinates3 mark the
%center of the region of interest in mark1 slice and mark2 slice
%respectively. -- KV
n = 1;
radius = 10;
cordinates1 = 0;
cordinates3 = 0;
%RS and RI is the rescale slope and rescale intercept for the linear
%calibration. RS and RI are obtained from generating a calibration curve
%from imaging standards with known HU values. -- KV
global RS
RS = [];
global RI
RI = [];
%Just in case we choose to calibrate using a exponential curve we have
%introduced coeffa and coeffb as the rescale slopes and intercept for the
%exponential calibration using the equation coeffa(exp(coeffb*PV)).
%coeffa and coeffb are obtained from generating a calibration curve
%from imaging standards with known HU values. -- KV.
global coeffa
coeffa = [];
global coeffb
coeffb = [];
[dirname] = uigetdir('Please choose dicom directory');
%ginfo1 is used here to store the dicominfo information for each slice in
%the Generate3dMatrix -- KV
global ginfo1
global calibtype
%ginfo1 is used in regular generate3dMatrix, ginfo is used in
%Generate3dMatrixbrandon
%generate3dMatrix function is used for measurements, while
%Generate3dMatrixbrandon is written for calibrations.
%(Generate3dMatrixbrandon is name so because the code was written for
```

```
%Brandon W. project) -- KV
matrix = Generate3dMatrixCBCT(dirname);
```

This function starts the gui and asks for what kind of evaluation you want to do -- KV

```
    function start(hObject, event)
        evaltype = input('Air/Water Calibration(1), HA-HDPE Calibration (2), Measurement (3)\n');
        if evaltype == 1;
            disp('1)Please start with chosing a mark1. 2)Then press the radius button to choose a
center point for the ROI of air and a second point to measure the distance between the center
point and the boarder of the ROI. 3) Choose a mark2 and press the pixel button to choose a center
point for the ROI in mark2. 4)Press Run, repeat steps 2 and 3 for a ROI containing water.')
        elseif evaltype == 2;
            disp('1)Please start with chosing a mark1. 2)Then press the radius button to choose a
center point for the ROI in the lowest dense sample and a second point to measure the distance
between the center point and the boarder of the ROI. 3) Choose a mark2 and press the pixel button
to choose a center point for the ROI in mark2. 4)Press Run, repeat the steps for a ROI containing
next standard.')
        elseif evaltype == 3;
            disp('1)Please start with chosing a mark1. 2)Then press the radius button to choose a
center point for the ROI and a second point to measure the distance between the center point and
the boarder of the ROI. 3) Choose a mark2 and press the pixel button to choose a center point for
the ROI in mark2.')
        end
    end
```

This function is updating the image we see as we scroll through the z slices

```
%note here the 'n' is placed in the 2nd column of the matrix so that the slices
%analyzed are the Sagittal slices-- KV
    function updateImage(hObject,event)
        n = uint16(get(hObject,'Value'))
        imshow(squeeze(matrix(:,n,:)),[]);
        drawnow;
    end
```

This function is used to go back and forth between mark1 and mark2 for

```
%the water/air calibration to ensure that the number of slices measured for
%air and water is constant. -- KV
    function updateImage1(hObject,event)
        n = mark1;
        imshow(squeeze(matrix(:,n,:)),[]);
        drawnow;
    end
    function  updateImage2(hObject,event)
        n = mark2;

        imshow(squeeze(matrix(:,n,:)),[]);
```

```
        drawnow;
    end
```

This function is indicating the Z slice we choose for Mark1 -- KV

```
    function setmark1(hObject,event)
        mark1 = n;
        set(btn1, 'string', strcat('Mark1: ',num2str(n)));
    end
```

This function is indicating the Z slice we choose for Mark2 -- KV

```
    function setmark2(hObject,event)
        mark2 = n;
        set(btn2, 'string', strcat('Mark2: ',num2str(n)));
    end
```

Inserting the x and y cordinates for the first and second point we choose to indicate the radius

```
 %cordinates1 is the center of the of the standard at Mark1. cordinates2
 %is the outer border of the standard. cordinates3 below is the
 %center of the of the standard at Mark2. -- KV
    function getradius(hObject,event)
        cordinates1 = ginput(1)
        cordinates2 = ginput(1);
        X1 = cordinates1(1);
        Y1 = cordinates1(2);
        X2 = cordinates2(1);
        Y2 = cordinates2(2);
        radius = sqrt((X2-X1)^2 + (Y2-Y1)^2)
        set(getRadius, 'string', strcat('Radius: ',num2str(radius)));
    end
```

Getting the pixel value from the ginput for the center of the ROI in

```
%in the mark2 slice-- KV
    function getpoint(hObject,event)
        cordinates3 = ginput(1);
        xvalue = cordinates3(1)
        yvalue = cordinates3(2)
        set(infobtn, 'string', strcat('X= ', num2str(xvalue), 'Y =',num2str(yvalue)))
    end
```

Choosing the center point and storing the x and y cordinates and finding the center axis of the standard

```
    %We make the user choose two center points, CenterM1 and CenterM2, just in case the sample is
off axis.
    %By doing this the center of the ROI will shift as a function of the
```

```
%slice number. -- KV

function run(hObject, event)
    cd(firstDir);
         cordinates = cordinates1;
    locationX = round(cordinates(1))
    locationY = round(cordinates(2))
    set(initRun, 'string', strcat('Center: ',num2str(cordinates)));
    CenterM1 = cordinates1;
    CenterM2 = cordinates3;
    deltay = double(CenterM2(2))-double(CenterM1(2))
    deltaz = mark2 - mark1
    my = double(deltay)/double(deltaz)
    by = double(CenterM1(2)) -double(my)*double(mark1)
    deltax = double(CenterM2(1))-double(CenterM1(1))
    mx = double(deltax)/double(deltaz)
    bx = double(CenterM1(1))- double(mx)*double(mark1)
%Once the center of the ROI has been determined in each slice we get to
%calibrating. evaltype == 1 is for a air/water calibration
%calib indicates which portion of the calibration the code is at.
%calib = 1 means that the air is being calibrated while calib = 2 means
%the water is being calibrated. -- KV
    if evaltype == 1;
        calib = 1;
        for calib = [1:2]
           if calib == 2;
               updateImage1(hObject,event)
               getradius(hObject,event)
               updateImage2(hObject,event)
               getpoint(hObject,event)
               cordinates = cordinates1;
               locationX = round(cordinates(1))
               locationY = round(cordinates(2))
               set(initRun, 'string', strcat('Center: ',num2str(cordinates)));
               %Here we are again determining the center points for the
               %water ROI and aligning the ROI so the center point
               %varies as a function of the slice number. -- KV
               CenterM1 = cordinates1;
               CenterM2 = cordinates3;
               deltay = double(CenterM2(2))-double(CenterM1(2))
               deltaz = mark2 - mark1
               my = double(deltay)/double(deltaz)
               by = double(CenterM1(2)) -double(my)*double(mark1)
               deltax = double(CenterM2(1))-double(CenterM1(1))
               mx = double(deltax)/double(deltaz)
               bx = double(CenterM1(1))- double(mx)*double(mark1)
           end
           radius = input('Please specify what radius you want to start with\n');
           %ensures mark1 is less than mark2
           if(mark1>mark2)
               tempVar = mark1;
               mark1=mark2;
               mark2 = tempVar;
           end
```

```
                count = int16(mark2-mark1);
                struct=[count];
                struct1=size(matrix);
                struct2 = [struct1(1), struct1(2)];
                for slicenumber = mark1:mark2
                    locationX = (double(slicenumber)*double(mx))+bx;
                    locationY = (double(slicenumber)*double(my))+by;
                    Center = [double(locationX), double(locationY)];
                    %CircularAVG is a minor function written by our group
                    %which finds the marked center pixel of the ROI and
                    %calculates the average pixel value for all of the pixel
                    %within a circular area with a radius inputed by the
                    %user. -- KV
                    tempStruct = CircularAVG(squeeze(matrix(:,slicenumber,:)), radius, Center(2),
Center(1));
                    struct(slicenumber - mark1 + 1) = tempStruct;

                end
                totalAverage = mean2(struct)
                STD =  std(double(struct))
                if calib == 1;
                    PVa = totalAverage;
                elseif calib == 2;
                    PVw = totalAverage;
                end
                calib = calib +1;
            end

            close all
            HUa = -1000;
            HUw = 0;
            HounsfieldUnitmat = [HUa;HUw];
            Dmat = [PVa;PVw];
            %Here is where we use a linear fit to relate the grey level
            %pixel values of the air and water with their known
            %Hounsfield Unit. -- KV
            rescale = polyfit(Dmat, HounsfieldUnitmat, 1);
            RS = rescale(1);
            RI = rescale(2);
            matrix = Generate3dMatrixbrandon(dirname);
            disp('please run Analysistype again for the new set of data');

        elseif evaltype == 2;
             %calibration using HA-HDPE Samples
            calib = 1;
            %Here calib indicates where standard is in use. calib = 1
            %corresponds to measuring the pixel value for the first, least
            %dense standard and calib = 4 corresponds to measuring the
            %pixel value for the fourth, highest denst standard.
            for calib = [1:4]
                if calib > 1
                    sprintf('please indicate Mark1, Mark2 and center of standard #%d',calib)
                    %Here we have introduced a pause, because we want the
                    %user to have a chance to go back and change the mark1
                    %and mark2 just in case the standards are not all
                    %visible in the same slices. -- KV
```

```
                    pause
                    updateImage1(hObject,event)
                    getradius(hObject,event)
                    updateImage2(hObject,event)
                    getpoint(hObject,event)
                    cordinates = cordinates1;
                    set(initRun, 'string', strcat('Center: ',num2str(cordinates)));
                    CenterM1 = cordinates1;
                    CenterM2 = cordinates3;
                    deltay = double(CenterM2(2))-double(CenterM1(2))
                    deltaz = mark2 - mark1
                    my = double(deltay)/double(deltaz)
                    by = double(CenterM1(2)) -double(my)*double(mark1)
                    deltax = double(CenterM2(1))-double(CenterM1(1))
                    mx = double(deltax)/double(deltaz)
                    bx = double(CenterM1(1))- double(mx)*double(mark1)
                end


                %The radius button used above
                %use just used to measure the radius here is where the
                %radius value is inputed by the user. -- KV
                radius = input('Please specify what radius you want to start with\n');
                %ensures mark1 is less than mark2
                if(mark1>mark2)
                    tempVar = mark1;
                    mark1=mark2;
                    mark2 = tempVar;
                end
                count = int16(mark2-mark1);
                struct=[count];
                struct1=size(matrix);
                struct2 = [struct1(1), struct1(2)];
                for slicenumber = mark1:mark2
                    locationX = (double(slicenumber)*double(mx))+bx;
                    locationY = (double(slicenumber)*double(my))+by;
                    Center = [double(locationX), double(locationY)];
                    tempStruct = CircularAVG(squeeze(matrix(:,slicenumber,:)), radius, Center(2),
Center(1));
                    struct(slicenumber - mark1 + 1) = tempStruct;
                end
                totalAverage = mean2(struct);
                STD  = std(double(struct));
                if calib == 1;
                    PV1 = totalAverage
                    PV1std = STD
                elseif calib == 2;
                    PV2 = totalAverage
                    PV2std = STD
                elseif calib == 3
                    PV3 = totalAverage
                    PV3std = STD
                elseif calib == 4
                    PV4= totalAverage
                    PV4std = STD
                end
```

```
            end
            close all
            HU1 = 723;
            HU2 = 1447;
            HU3 = 2728;
            HU4 = 4640;
            HounsfieldUnitmat = [HU1;HU2;HU3; HU4];
            Dmat = [PV1; PV2; PV3; PV4];
            %Here we are solving for the rescale slope and intercept for
            %a linear calibration curve. Here we also solve a rescale coeffs
            %for an exponential calibration curve Aexp(PV*B). -- KV
            rescale = polyfit(Dmat, HounsfieldUnitmat, 1);
            f1 = fit(Dmat, HounsfieldUnitmat, 'exp1');
            RS = rescale(1);
            RI = rescale(2);
            coeffa = f1.a;
            coeffb = f1.b;
            FixHU = (Dmat*RS) +RI;
            fixHU = coeffa*exp(coeffb*Dmat);
            figure
            plot(Dmat, FixHU, 'r--')
            hold on
            plot(Dmat, fixHU, 'b--')
            plot(Dmat, HounsfieldUnitmat, 'k+', 'MarkerSize', 15)
            hold off
            %Here we generate a plot for the user to look at which
            %calibration curve seems to fit their data best, an exponential
            %vs a linear. The calibtype indicates which calibration curve
            %the user chose to use for the calibration. -- KV
            calibtype = input('Please indicate if you want a linear (1) or an exponential (2)
calibration curve \n');
            matrix = Generate3dMatrixbrandon2(dirname);
            disp('please run DICOMViewerSlider5calibration again for the new set of data');

       elseif evaltype==3;
           %When evaltype = 3 then the interface is only used to make
           %measurements of the ROI. These measurements are usually made
           %after calibration has been done. -- KV
           %Moreover, you can choose to calculate the avg HU for cylinders
           %with a variety of radii. -- KV
            numbofradi = input('please insert how many radi you want to evalute\n');
            if numbofradi>1;
                firstradius = input('Please specify what radius you want to start with\n');
                delta = input('Please specify the incrament you want to increase the radius
by\n');
                lastradius = input('Please specify what radius you want to finish with\n');
                radius = zeros(((lastradius-firstradius)/delta)+1,1);
                tempradius = firstradius;
                for i = 1:length(radius)
                    radius(i) = tempradius;
                    tempradius = tempradius + delta;
                end
            else
                radius = input('Please specify what radius you want to start with\n');
```

```
            end
            %ensures mark1 is less than mark2
            if(mark1>mark2)
                tempVar = mark1;
                mark1=mark2;
                mark2 = tempVar;
            end
            count = int16(mark2-mark1);
            struct=[count];
            struct1=size(matrix);
            struct2 = [struct1(1), struct1(2)];
            m = 0;
            %Here we ask the user if they calibrated using a linear or
            %exponential calibration just so the right equation is used to
            %calculated the corrected HU from the grey level pixel values.
            %-- KV
            calibtype = input('Please indicate if you chose a linear (or air/water calibration)
(1) or exponential (2) calibration above \n');
            for p = 1:length(radius)    %CHANGE THIS WHEN YOU WANT DIFFERENT RADIUS.-- KV
                m = m+1;
                for slicenumber = mark1:mark2
                    locationX = (double(slicenumber)*double(mx))+bx;
                    locationY = (double(slicenumber)*double(my))+by;
                    Center = [double(locationX), double(locationY)];
                    tempStruct = CircularAVG(squeeze(matrix(:,slicenumber,:)), radius(p),
Center(2), Center(1));
                    struct(slicenumber - mark1 + 1) = tempStruct;
                end
                HUstruct = [];
                %HU calculation for a linear calibration. -- KV
                if calibtype == 1
                for structnumber = 1:length(struct)
                    rescaleint(structnumber)= ginfo1{structnumber-1+mark1}.RescaleIntercept;
                    rescaleslope(structnumber)= ginfo1{structnumber-1+mark1}.RescaleSlope;
                    struct = double(struct);
                    HUstruct(structnumber) =
(rescaleslope(structnumber)*struct(structnumber))+rescaleint(structnumber);
                end
                %HU calculation for a exponential calibration. -- KV
                elseif calibtype == 2
                     for structnumber = 1:length(struct)
                         rescaleint(structnumber)= ginfo1{structnumber-1+mark1}.RescaleIntercept;
                         rescaleslope(structnumber)= ginfo1{structnumber-1+mark1}.RescaleSlope;
                         struct = double(struct);
                         HUstruct(structnumber) =
rescaleint(structnumber)*exp(rescaleslope(structnumber)*struct(structnumber));
                     end
                end
               figure
               plot(HUstruct);
               xlabel('Number of Slices')
               ylabel('PV in HU')
               sixstruct = int16(struct);
               sixstruct = sixstruct +32767;
```

```
                total(m) = mean2(sixstruct)
                totalAverage(m) = mean2(struct)
                STD(m) = std(double(struct))
                HU(m) = mean2(HUstruct)
             end
            pp = radius(length(radius));   %CHANGE THIS WHEN YOU WANT DIFFERENT RADIUS.-- KV
            tempStruct = Blackingradius(squeeze(matrix(:,mark2,:)), pp, Center(2), Center(1));
            struct2 = tempStruct;
            figure
            imshow(squeeze(struct2(:,:)),[]);
         end

        if evaltype == 3;
            figure
            plot(HU, 'bo')
            ylabel('Total Avg Pixel Value (HU)')
            xlebel('Radius number')
        end

    end

f=figure(1);

slider = uicontrol('Parent',f,'Style','slider','Position',[81,114,420,23],'min',0,
'max',size(matrix,2));

btn1 = uicontrol('Style', 'pushbutton', 'String', 'Mark 1','Position', [81,34,210,20],'Callback',
@(hObject, event) setmark1(hObject, event));

btn2 = uicontrol('Style', 'pushbutton', 'String', 'Mark 2','Position',
[291,34,210,20],'Callback', @(hObject, event) setmark2(hObject, event));

initRun = uicontrol('Style', 'pushbutton', 'String', 'Run','Position', [81,14,420,20],'Callback',
@(hObject, event) run(hObject, event));
initStart = uicontrol('Style', 'pushbutton', 'String', 'Start','Position',
[81,54,420,20],'Callback', @(hObject, event) start(hObject, event));

getRadius = uicontrol('Style', 'pushbutton', 'String', 'Radius','Position',
[81,74,420,20],'Callback', @(hObject, event) getradius(hObject, event));

mTextBox = uicontrol('style','text','Position', [81,0,420,14])
infobtn = uicontrol('Style', 'pushbutton', 'String', 'Pixel', 'Position', [81,92,210,20],
'Callback', @(hObject, event) getpoint(hObject, event));

addlistener(slider,'ContinuousValueChange',@(hObject, event) updateImage(hObject, event));

%display%
ax1=axes('parent',f,'position',[0.13 0.39  0.77 0.54]);
imshow(squeeze(matrix(:,n,:)),[]);

global matrix2
matrix2 = matrix;
```

```
end
```

Published with MATLAB® R2017a

```
function DICOMViewer_Front_Axial
```

```
%This code actually changes the rescale slop and intercept using either a
%linear or an exponential fit. -- KV
clear global
close all
%Set all of your global variables. -- KV
firstDir = cd;
%evaltype is to determine which kind of calibration is going to be done.
%The function is written so everytime it is ran it automatically functions
%at a evaltype of 3 which is just for making measurements. -- KV
evaltype = 3;
%mark1 and mark2 are used to indicate the first and last slice you want to
%make measurements in between. -- KV
global mark1
mark1 = 1;
global mark2
mark2 = 1;
%n reperesent the slice you are viewing, while radius and is the radius of
%the cylinderical volume of interest. cordinates1 and cordinates3 mark the
%center of the region of interest in mark1 slice and mark2 slice
%respectively. -- KV
n = 1;
radius = 10;
cordinates1 = 0;
cordinates3 = 0;
%RS and RI is the rescale slope and rescale intercept for the linear
%calibration. RS and RI are obtained from generating a calibration curve
%from imaging standards with known HU values. -- KV
global RS
RS = [];
global RI
RI = [];
%Just in case we choose to calibrate using a exponential curve we have
%introduced coeffa and coeffb as the rescale slopes and intercept for the
%exponential calibration using the equation coeffa(exp(coeffb*PV)).
%coeffa and coeffb are obtained from generating a calibration curve
%from imaging standards with known HU values. -- KV.
global coeffa
coeffa = [];
global coeffb
coeffb = [];
[dirname] = uigetdir('Please choose dicom directory');
%ginfo1 is used here to store the dicominfo information for each slice in
%the Generate3dMatrix -- KV
global ginfo1
global calibtype
%ginfo1 is used in regular generate3dMatrix, ginfo is used in
```

```
%Generate3dMatrixbrandon
%generate3dMatrix function is used for measurements, while
%Generate3dMatrixbrandon is written for calibrations.
%(Generate3dMatrixbrandon is name so because the code was written for
%Brandon W. project) -- KV
matrix = Generate3dMatrixCBCT(dirname);
```

This function starts the gui and asks for what kind of evaluation you want to do -- KV

```
    function start(hObject, event)
        evaltype = input('Air/Water Calibration(1), HA-HDPE Calibration (2), Measurement (3)\n');
        if evaltype == 1;
            disp('1)Please start with chosing a mark1. 2)Then press the radius button to choose a
center point for the ROI of air and a second point to measure the distance between the center
point and the boarder of the ROI. 3) Choose a mark2 and press the pixel button to choose a center
point for the ROI in mark2. 4)Press Run, repeat steps 2 and 3 for a ROI containing water.')
        elseif evaltype == 2;
            disp('1)Please start with chosing a mark1. 2)Then press the radius button to choose a
center point for the ROI in the lowest dense sample and a second point to measure the distance
between the center point and the boarder of the ROI. 3) Choose a mark2 and press the pixel button
to choose a center point for the ROI in mark2. 4)Press Run, repeat the steps for a ROI containing
next standard.')
        elseif evaltype == 3;
            disp('1)Please start with chosing a mark1. 2)Then press the radius button to choose a
center point for the ROI and a second point to measure the distance between the center point and
the boarder of the ROI. 3) Choose a mark2 and press the pixel button to choose a center point for
the ROI in mark2.')
        end
    end
```

This function is updating the image we see as we scroll through the z slices

```
%note here the 'n' is placed in the 2nd column of the matrix so that the slices
%analyzed are the Axial slices-- KV
    function updateImage(hObject,event)
        n = uint16(get(hObject,'Value'))
        imshow(squeeze(matrix(:,:,n)),[]);
        drawnow;
    end
```

This function is used to go back and forth between mark1 and mark2 for

```
%the water/air calibration to ensure that the number of slices measured for
%air and water is constant. -- KV
    function updateImage1(hObject,event)
        n = mark1;
        imshow(squeeze(matrix(:,:,n)),[]);
        drawnow;
    end
    function updateImage2(hObject,event)
```

```
        n = mark2;

        imshow(squeeze(matrix(:,:,n)),[]);
        drawnow;
    end
```

This function is indicating the Z slice we choose for Mark1 -- KV

```
    function setmark1(hObject,event)
        mark1 = n;
        set(btn1, 'string', strcat('Mark1: ',num2str(n)));
    end
```

This function is indicating the Z slice we choose for Mark2 -- KV

```
    function setmark2(hObject,event)
        mark2 = n;
        set(btn2, 'string', strcat('Mark2: ',num2str(n)));
    end
```

Inserting the x and y cordinates for the first and second point we choose to indicate the radius

```
 %cordinates1 is the center of the of the standard at Mark1. cordinates2
 %is the outer border of the standard. cordinates3 below is the
 %center of the of the standard at Mark2. -- KV
    function getradius(hObject,event)
        cordinates1 = ginput(1)
        cordinates2 = ginput(1);
        X1 = cordinates1(1);
        Y1 = cordinates1(2);
        X2 = cordinates2(1);
        Y2 = cordinates2(2);
        radius = sqrt((X2-X1)^2 + (Y2-Y1)^2)
        set(getRadius, 'string', strcat('Radius: ',num2str(radius)));
    end
```

Getting the pixel value from the ginput for the center of the ROI in

```
%in the mark2 slice-- KV
    function getpoint(hObject,event)
        cordinates3 = ginput(1);
        xvalue = cordinates3(1)
        yvalue = cordinates3(2)
        set(infobtn, 'string', strcat('X= ', num2str(xvalue), 'Y =',num2str(yvalue)))
    end
```

Choosing the center point and storing the x and y cordinates and finding the center axis of the standard

```
    %We make the user choose two center points, CenterM1 and CenterM2, just in case the sample is
off axis.
    %By doing this the center of the ROI will shift as a function of the
    %slice number. -- KV

    function run(hObject, event)
        cd(firstDir);
             cordinates = cordinates1;
        locationX = round(cordinates(1))
        locationY = round(cordinates(2))
        set(initRun, 'string', strcat('Center: ',num2str(cordinates)));
        CenterM1 = cordinates1;
        CenterM2 = cordinates3;
        deltay = double(CenterM2(2))-double(CenterM1(2))
        deltaz = mark2 - mark1
        my = double(deltay)/double(deltaz)
        by = double(CenterM1(2)) -double(my)*double(mark1)
        deltax = double(CenterM2(1))-double(CenterM1(1))
        mx = double(deltax)/double(deltaz)
        bx = double(CenterM1(1))- double(mx)*double(mark1)
    %Once the center of the ROI has been determined in each slice we get to
    %calibrating. evaltype == 1 is for a air/water calibration
    %calib indicates which portion of the calibration the code is at.
    %calib = 1 means that the air is being calibrated while calib = 2 means
    %the water is being calibrated. -- KV
        if evaltype == 1;
            calib = 1;
            for calib = [1:2]
               if calib == 2;
                   updateImage1(hObject,event)
                   getradius(hObject,event)
                   updateImage2(hObject,event)
                   getpoint(hObject,event)
                   cordinates = cordinates1;
                   locationX = round(cordinates(1))
                   locationY = round(cordinates(2))
                   set(initRun, 'string', strcat('Center: ',num2str(cordinates)));
                   %Here we are again determining the center points for the
                   %water ROI and aligning the ROI so the center point
                   %varies as a function of the slice number. -- KV
                   CenterM1 = cordinates1;
                   CenterM2 = cordinates3;
                   deltay = double(CenterM2(2))-double(CenterM1(2))
                   deltaz = mark2 - mark1
                   my = double(deltay)/double(deltaz)
                   by = double(CenterM1(2)) -double(my)*double(mark1)
                   deltax = double(CenterM2(1))-double(CenterM1(1))
                   mx = double(deltax)/double(deltaz)
                   bx = double(CenterM1(1))- double(mx)*double(mark1)
               end
```

```
                radius = input('Please specify what radius you want to start with\n');
                %ensures mark1 is less than mark2
                if(mark1>mark2)
                    tempVar = mark1;
                    mark1=mark2;
                    mark2 = tempVar;
                end
                count = int16(mark2-mark1);
                struct=[count];
                struct1=size(matrix);
                struct2 = [struct1(1), struct1(2)];
                for slicenumber = mark1:mark2
                    locationX = (double(slicenumber)*double(mx))+bx;
                    locationY = (double(slicenumber)*double(my))+by;
                    Center = [double(locationX), double(locationY)];
                    %CircularAVG is a minor function written by our group
                    %which finds the marked center pixel of the ROI and
                    %calculates the average pixel value for all of the pixel
                    %within a circular area with a radius inputed by the
                    %user. -- KV
                    tempStruct = CircularAVG(squeeze(matrix(:,:,slicenumber)), radius, Center(2),
Center(1));
                    struct(slicenumber - mark1 + 1) = tempStruct;
                end
                totalAverage = mean2(struct)
                STD =  std(double(struct))
                if calib == 1;
                    PVa = totalAverage;
                elseif calib == 2;
                    PVw = totalAverage;
                end
                calib = calib +1;
            end

            close all
            HUa = -1000;
            HUw = 0;
            HounsfieldUnitmat = [HUa;HUw];
            Dmat = [PVa;PVw];
            %Here is where we use a linear fit to relate the grey level
            %pixel values of the air and water with their known
            %Hounsfield Unit. -- KV
            rescale = polyfit(Dmat, HounsfieldUnitmat, 1);
            RS = rescale(1);
            RI = rescale(2);
            matrix = Generate3dMatrixbrandon(dirname);
            disp('please run Analysistype again for the new set of data');

        elseif evaltype == 2;
             %calibration using HA-HDPE Samples
            calib = 1;
            %Here calib indicates where standard is in use. calib = 1
            %corresponds to measuring the pixel value for the first, least
            %dense standard and calib = 4 corresponds to measuring the
            %pixel value for the fourth, highest denst standard.
```

```
            for calib = [1:4]
                if calib > 1
                    sprintf('please indicate Mark1, Mark2 and center of standard #%d',calib)
                    %Here we have introduced a pause, because we want the
                    %user to have a chance to go back and change the mark1
                    %and mark2 just in case the standards are not all
                    %visible in the same slices. -- KV
                    pause
                    updateImage1(hObject,event)
                    getradius(hObject,event)
                    updateImage2(hObject,event)
                    getpoint(hObject,event)
                    cordinates = cordinates1;
                    set(initRun, 'string', strcat('Center: ',num2str(cordinates)));
                    CenterM1 = cordinates1;
                    CenterM2 = cordinates3;
                    deltay = double(CenterM2(2))-double(CenterM1(2))
                    deltaz = mark2 - mark1
                    my = double(deltay)/double(deltaz)
                    by = double(CenterM1(2)) -double(my)*double(mark1)
                    deltax = double(CenterM2(1))-double(CenterM1(1))
                    mx = double(deltax)/double(deltaz)
                    bx = double(CenterM1(1))- double(mx)*double(mark1)
                end
                %The radius button used above
                %use just used to measure the radius here is where the
                %radius value is inputed by the user. -- KV
                radius = input('Please specify what radius you want to start with\n');
                %ensures mark1 is less than mark2
                if(mark1>mark2)
                    tempVar = mark1;
                    mark1=mark2;
                    mark2 = tempVar;
                end
                count = int16(mark2-mark1);
                struct=[count];
                struct1=size(matrix);
                struct2 = [struct1(1), struct1(2)];
                for slicenumber = mark1:mark2
                    locationX = (double(slicenumber)*double(mx))+bx;
                    locationY = (double(slicenumber)*double(my))+by;
                    Center = [double(locationX), double(locationY)];
                    tempStruct = CircularAVG(squeeze(matrix(:,:,slicenumber)), radius, Center(2),
Center(1));
                    struct(slicenumber - mark1 + 1) = tempStruct;
                end
                totalAverage = mean2(struct);
                STD  = std(double(struct));
                if calib == 1;
                    PV1 = totalAverage
                    PV1std = STD
                elseif calib == 2;
                    PV2 = totalAverage
                    PV2std = STD
```

```
                elseif calib == 3
                    PV3 = totalAverage
                    PV3std = STD
                elseif calib == 4
                    PV4= totalAverage
                    PV4std = STD
                end
            end
            close all
            HU1 = 723;
            HU2 = 1447;
            HU3 = 2728;
            HU4 = 4640;
            HounsfieldUnitmat = [HU1;HU2;HU3; HU4];
            Dmat = [PV1; PV2; PV3; PV4];
            %Here we are solving for the rescale slope and intercept for
            %a linear calibration curve. Here we also solve a rescale coeffs
            %for an exponential calibration curve Aexp(PV*B). -- KV
            rescale = polyfit(Dmat, HounsfieldUnitmat, 1);
            f1 = fit(Dmat, HounsfieldUnitmat, 'exp1');
            RS = rescale(1);
            RI = rescale(2);
            coeffa = f1.a;
            coeffb = f1.b;
            FixHU = (Dmat*RS) +RI;
            fixHU = coeffa*exp(coeffb*Dmat);
            figure
            plot(Dmat, FixHU, 'r--')
            hold on
            plot(Dmat, fixHU, 'b--')
            plot(Dmat, HounsfieldUnitmat, 'k+', 'MarkerSize', 15)
            hold off
            %Here we generate a plot for the user to look at which
            %calibration curve seems to fit their data best, an exponential
            %vs a linear. The calibtype indicates which calibration curve
            %the user chose to use for the calibration. -- KV
            calibtype = input('Please indicate if you want a linear (1) or an exponential (2)
calibration curve \n');
            matrix = Generate3dMatrixbrandon2(dirname);
            disp('please run DICOMViewerSlider5calibration again for the new set of data');

       elseif evaltype==3;
           %When evaltype = 3 then the interface is only used to make
           %measurements of the ROI. These measurements are usually made
           %after calibration has been done. -- KV
           %Moreover, you can choose to calculate the avg HU for cylinders
           %with a variety of radii. -- KV
            numbofradi = input('please insert how many radi you want to evalute\n');
            if numbofradi>1;
                firstradius = input('Please specify what radius you want to start with\n');
                delta = input('Please specify the incrament you want to increase the radius
by\n');
                lastradius = input('Please specify what radius you want to finish with\n');
                radius = zeros(((lastradius-firstradius)/delta)+1,1);
```

```
                tempradius = firstradius;
                for i = 1:length(radius)
                    radius(i) = tempradius;
                    tempradius = tempradius + delta;
                end
            else
                radius = input('Please specify what radius you want to start with\n');
            end
            %ensures mark1 is less than mark2
            if(mark1>mark2)
                tempVar = mark1;
                mark1=mark2;
                mark2 = tempVar;
            end
            count = int16(mark2-mark1);
            struct=[count];
            struct1=size(matrix);
            struct2 = [struct1(1), struct1(2)];
            m = 0;
            %Here we ask the user if they calibrated using a linear or
            %exponential calibration just so the right equation is used to
            %calculated the corrected HU from the grey level pixel values.
            %-- KV

            calibtype = input('Please indicate if you chose a linear (or air/water calibration)
(1) or exponential (2) calibration above \n');
            for p = 1:length(radius)    %CHANGE THIS WHEN YOU WANT DIFFERENT RADIUS.-- KV
                m = m+1;
                for slicenumber = mark1:mark2
                    locationX = (double(slicenumber)*double(mx))+bx;
                    locationY = (double(slicenumber)*double(my))+by;
                    Center = [double(locationX), double(locationY)];
                    tempStruct = CircularAVG(squeeze(matrix(:,:,slicenumber)), radius(p),
Center(2), Center(1));
                    struct(slicenumber - mark1 + 1) = tempStruct;
                end
                HUstruct = [];
                %HU calculation for a linear calibration. -- KV
                if calibtype == 1
                for structnumber = 1:length(struct)
                    rescaleint(structnumber)= ginfo1{structnumber-1+mark1}.RescaleIntercept;
                    rescaleslope(structnumber)= ginfo1{structnumber-1+mark1}.RescaleSlope;
                    struct = double(struct);
                    HUstruct(structnumber) =
(rescaleslope(structnumber)*struct(structnumber))+rescaleint(structnumber);
                end
                %HU calculation for a exponential calibration. -- KV
                elseif calibtype == 2
                    for structnumber = 1:length(struct)
                    rescaleint(structnumber)= ginfo1{structnumber-1+mark1}.RescaleIntercept;
                    rescaleslope(structnumber)= ginfo1{structnumber-1+mark1}.RescaleSlope;
                    struct = double(struct);
                    HUstruct(structnumber) =
rescaleint(structnumber)*exp(rescaleslope(structnumber)*struct(structnumber));
```

```
                    end
                end
                figure
                plot(HUstruct);
                xlabel('Number of Slices')
                ylabel('PV in HU')
                sixstruct = int16(struct);
                sixstruct = sixstruct +32767;
                total(m) = mean2(sixstruct)
                totalAverage(m) = mean2(struct)
                STD(m) = std(double(struct))
                HU(m) = mean2(HUstruct)

            end
            pp = radius(length(radius));   %CHANGE THIS WHEN YOU WANT DIFFERENT RADIUS.-kv
            tempStruct = Blackingradius(squeeze(matrix(:,:,mark2)), pp, Center(2), Center(1));
            struct2 = tempStruct;
            figure
            imshow(squeeze(struct2(:,:)),[]);
        end

       if evaltype == 3;
            figure
            plot(HU, 'bo')
            ylabel('Total Avg Pixel Value (HU)')
            xlebel('Radius number')
       end

    end

f=figure(1);

slider = uicontrol('Parent',f,'Style','slider','Position',[81,114,420,23],'min',0,
'max',size(matrix,3));

btn1 = uicontrol('Style', 'pushbutton', 'String', 'Mark 1','Position', [81,34,210,20],'Callback',
@(hObject, event) setmark1(hObject, event));

btn2 = uicontrol('Style', 'pushbutton', 'String', 'Mark 2','Position',
[291,34,210,20],'Callback', @(hObject, event) setmark2(hObject, event));

initRun = uicontrol('Style', 'pushbutton', 'String', 'Run','Position', [81,14,420,20],'Callback',
@(hObject, event) run(hObject, event));
initStart = uicontrol('Style', 'pushbutton', 'String', 'Start','Position',
[81,54,420,20],'Callback', @(hObject, event) start(hObject, event));

getRadius = uicontrol('Style', 'pushbutton', 'String', 'Radius','Position',
[81,74,420,20],'Callback', @(hObject, event) getradius(hObject, event));

mTextBox = uicontrol('style','text','Position', [81,0,420,14])
infobtn = uicontrol('Style', 'pushbutton', 'String', 'Pixel', 'Position', [81,92,210,20],
'Callback', @(hObject, event) getpoint(hObject, event));

addlistener(slider,'ContinuousValueChange',@(hObject, event) updateImage(hObject, event));
```

```
%display%
ax1=axes('parent',f,'position',[0.13 0.39  0.77 0.54]);
imshow(squeeze(matrix(:,:,n)),[]);

global matrix2
matrix2 = matrix;

end
```

Published with MATLAB® R2017a

Dicom2Volume: Here we show two versions of the Dicom2Volume function. One is used during calibration and the other is used during measurements.

```
%Generates an ordered volume of DiCOM images based on given directory.
%Returns an array of DiCOM file names in order.
function volume = DICOM2VolumeCBCT(directory)

%Changes directory.
cd(directory);

%Creates array of acquisition numbers based on unordered DiCOM files in
%directory.
if (directory ~= 0)
    loadingbar = waitbar(0,'Generating volume...');
    %dir gets all of the elements of the folder directory
    d = dir(directory);
    steps = length(d);
    step = 1;
    slices = [];


    for k = 1:size(d,1)
        %Checks if DiCOM file is present
        if strfind(d(k).name,'.dcm')>0
            %Here we are looking to see if any part of d(k).name has '.dcm'
            info = dicominfo(d(k).name);
            slice_num = info.InstanceNumber;
            %We're using the acquisitionNumber in the dicominfo to sort the
            %slices.
            waitbar(step / steps)
            step = step + 1;
            %List of file name along with acquisition number
            slices = [slices; slice_num];
        end
    end
    close(loadingbar)
```

```
    loadingbar = waitbar(0,'Sorting...');

    %Sorts file names by acquisition numbers
    slices = sort(slices);
    vol = cell([1,length(slices)]);
    %Here the goal is to make the cellstr array 'vol' with all of the names
    %of the slices in order.
    for k = 1:size(d,1)
        if strfind(d(k).name,'.dcm')>0
            inf1 = dicominfo(d(k).name);
            ind = find(slices==inf1.InstanceNumber);
            vol(ind)= cellstr(d(k).name);
            waitbar(info.InstanceNumber / steps)
        end
    end
    close(loadingbar)

    %Volume to be returned
    volume = vol;
end
```

Published with MATLAB® R2017a

```
%Generates an ordered volume of DiCOM images based on given directory.
%Returns an array of DiCOM file names in order.
function volume = DICOM2Volumebrandon(directory)

%Changes directory.
cd(directory);

%Creates array of acquisition numbers based on unordered DiCOM files in
%directory.
if (directory ~= 0)
    loadingbar = waitbar(0,'Generating volume...');
    d = dir(directory);
    steps = length(d);
    step = 1;
    slices = [];


    for k = 1:size(d,1)
        %Checks if DiCOM file is present
        if strfind(d(k).name,'.dcm')>0
            info = dicominfo(d(k).name);
            slice_num = info.AcquisitionNumber;
            waitbar(step / steps)
            step = step + 1;
            %List of file name along with acquisition number
            slices = [slices; slice_num];
        end
    end
    close(loadingbar)
```

```
    loadingbar = waitbar(0,'Sorting...');

    %Sorts file names by acquisition numbers
    slices = sort(slices);
    vol = cell([1,length(slices)]);
    inf = struct([]);
    for k = 1:size(d,1)
        if strfind(d(k).name,'.dcm')>0
            info = dicominfo(d(k).name);
            ind = find(slices==info.AcquisitionNumber);
            vol(ind)= cellstr(d(k).name);
            inf{ind} = info;
            waitbar(info.AcquisitionNumber / steps)
        end
    end
    close(loadingbar)

    %Volume to be returned
    volume = vol;
    global ginfo
    ginfo = inf;
end
```

Published with MATLAB® R2017a

Generate3dMatrix: In this section we show two versions of the generate3dmatrix function. One is used during calibration and the other used during measurments.

```
%Generates of 3-dimenional array that represents scanned image in 3-d space
%Returns matrix of Grayscale value.

function matrix = Generate3dMatrixCBCT(dirnameOriginal);
global ginfo1
%Creates volume using DICOM2Volume
vol = DICOM2VolumeCBCT(dirnameOriginal);

%intializes empty array to store images in,
%the dimensions are (imagesize, imagesize, volume size)
width = length(dicomread(char(vol(1))));

%Ensures data type of unsigned integer.
dimensionalRep = zeros(width,width,length(vol),'int16');

loadingbar = waitbar(0,'Creating 3-D space...');

%Reads in one image, copies content (Grayscale Values) to 3-d matrix
for imageNumber = 1:length(vol)
    waitbar(imageNumber/length(vol));
    cd(dirnameOriginal);
```

```
    %Read in image
    currentImage = dicomread(char(vol(imageNumber)));
    %copy in the dicominfo data so that we can extract the Rescale slope
    %and intercept for HU conversion.
    ginfo1{imageNumber} = dicominfo(char(vol(imageNumber)));
    %Copy X,Y values to X,Y value of 3-d matrix.
    for x = 1:length(currentImage)
        for y = 1:length(currentImage);
%             dimensionalRep(y,x,imageNumber) = uint16(currentImage(y,x));
            dimensionalRep(y,x,imageNumber) = (currentImage(y,x));
        end
    end
end

close(loadingbar);

%Ensures unsigned integer datatype
dimensionalRep1 = dimensionalRep;
%Addition by KV, tryig to figure out why some of the images aren't showing
%up. Maybe has to do with the uint16 command on the bottom turning all of
%the negative variables into zero.
im2uint16(dimensionalRep);

%return
matrix = dimensionalRep;
end
```

Published with MATLAB® R2017a

```
%Generates of 3-dimenional array that represents scanned image in 3-d space
%Returns matrix of Grayscale value.

function matrix = Generate3dMatrixbrandon(dirnameOriginal);

%Creates volume using DICOM2Volume
vol = DICOM2Volumebrandon(dirnameOriginal);

%intializes empty array to store images in,
%the dimensions are (imagesize, imagesize, volume size)
width = length(dicomread(char(vol(1))));

olddirectory=(char(cd))
for i = 1:length(vol)
    X{i} = dicomread(char(vol(i)));
end
mkdir('NewRescaleCoeff')
cd('NewRescaleCoeff')
for i = 1:length(vol)
    if length(vol) <= 512
```

```
        if i <10;
           nameofdicom = sprintf('slice_00%d.dcm',i);
        elseif i>= 10 && i <100;
            nameofdicom = sprintf('slice_0%d.dcm',i);
        elseif i>=100;
            nameofdicom = sprintf('slice_%d.dcm',i);
        end
    end
    if length(vol) <= 1024
        if i <10;
           nameofdicom = sprintf('slice_000%d.dcm',i);
        elseif i>= 10 && i <100;
            nameofdicom = sprintf('slice_00%d.dcm',i);
        elseif i>=100 && i <1000;
            nameofdicom = sprintf('slice_0%d.dcm',i);
        elseif i>=1000;
            nameofdicom = sprintf('slice_%d.dcm',i);
        end
    end
    global ginfo
    global RS
    global RI
    ginfo{i}.RescaleSlope = RS;
    ginfo{i}.RescaleIntercept = RI;
    S = ginfo{i};
    x{i} = X{i};
    dicomwrite(x{i},nameofdicom, S, 'CreateMode', 'copy');
end

cd(olddirectory)


%return
matrix = 1;
end
```

Published with MATLAB® R2017a

Minor Functions: CircularAVG function is a minor function written to measure the average pixel value in a circular area, and Blackingradius is written to mark the circular area that was measured to allow the user to visualize their chosen ROI.

```
%Averages GS values by location for specified volume / slice
function sliceAverage = CircularAVG(slice, radius, locationX, locationY)
slice = int32(slice);
sizerow = size(slice,1);
```

```
sizecol = size(slice,2);
sizeL=1;
if(sizerow > sizecol)
    sizeL = sizerow;
else
    sizeL = sizecol;
end
%Calculates distance at each index
distanceMatrix = zeros(sizerow, sizecol);

for i = 1:sizerow
    for j = 1:sizecol

        distanceMatrix(i,j) = sqrt(double(((locationX + .5) -i).^2 + ((locationY + .5 )-j).^2));
    end
end



valueCount = int32(zeros(sizeL, 3));

currentIndex = 1;
continueOn = true;
tolerance = 2;
%%Checking if the point chosen is less than the radius
for i = 1:sizerow
    for j = 1:sizecol
        r = 1;
        continueOn = true;

        if (distanceMatrix(i,j) < radius) %includes images within radius

            while r < length(valueCount) && continueOn==true

                %checks if there a new entry
                if valueCount(r,1) == 0
                    valueCount(currentIndex,1) = distanceMatrix(i,j);
                    valueCount(currentIndex,2) = 1;
                    valueCount(currentIndex,3) = slice(i,j);
                    currentIndex = currentIndex + 1;
                    continueOn = false;
                    %checks if it is valid to add to entry
                elseif abs(valueCount(r,1) - distanceMatrix(i,j)) < tolerance
                    valueCount(r,2) =  valueCount(r,2) + 1;
                    valueCount(r,3) = (valueCount(r,3) + slice(i,j)) / 2;
                    continueOn = false;
                    %increments to next entry
                else
                    r = r+1;
                end
            end
        end
    end


end
```

```
%%%%%%Prepares data for Analysis%%%%%%%%%%%%
%counts all non-zero data
count = 0;
for p = 1: length(valueCount)
    if valueCount(p,1) ~= 0
        count = count + 1;
    end
end

%copy calculated values into new plots up to calculated count
numOccurances = [count];
distances = [count];
averages = [count];
averages = int32(averages);
for p = 1:count
    numOccurances(p) = valueCount(p,2); %num occurances
    distances(p) = valueCount(p,1); %distances
    averages(p) = valueCount(p,3); %average
end
global valuCount
valuCount = valueCount;
sliceAverage = mean2(averages);

end
```

Published with MATLAB® R2017a

```
%Averages GS values by location for specified volume / slice
function newmat = Blackingradius(slice, radius, locationX, locationY)
slice = int32(slice);
sizeX = size(slice,1);
sizeY = size(slice,2);
sizeL=1;
if(sizeX > sizeY)
    sizeL = sizeX;
else
    sizeL = sizeY;
end
%Calculates distance at each index
distanceMatrix = zeros(sizeX, sizeY);

for i = 1:sizeX
    for j = 1:sizeY

        distanceMatrix(i,j) = sqrt(double(((locationX + .5) -i).^2 + ((locationY + .5 )-j).^2));
    end
end

valueCount = int32(zeros(sizeL, 3));
```

```
currentIndex = 1;
continueOn = true;
tolerance = 2;
%%Checking if the point chosen is less than the radius
for i = 1:sizeX
    for j = 1:sizeY
        r = 1;
        continueOn = true;

        if (distanceMatrix(i,j) < radius) %includes images within radius
           newmat(i,j) = 0;
        else
           newmat(i,j) = slice(i,j);
        end

    end
end
```

Published with MATLAB® R2017a

Appendix C:

This section of the thesis, lays out all of the protocols attempted to make the HA-HDPE standards before the working protocol was utilized. In this section the protocols are listed in whole and the bolded regions specify the specific section of the protocol which was modified from the previous protocol attempted.

Protocol Version #1

1. Sonicate Hydroxyapatite (HA) in 30 ml of Ethanol for 5 minutes (5 second cycles at an amplitude of 30%)
2. Add high density polyethylene (HDPE) and sonicate for 10 minutes (5 second cycles at an amplitude of 30%)
3. Centrifuge the particles to the bottom of a centrifuge tube and syringe the solvent out
4. Dry overnight in oven
5. Weigh out the amount of powder desired and press the powder under 150^{o}C and 45 MPa for 30 minutes
    a. Press using two metal plates a circular metal mold and a Teflon plate placed in between the top metal plate and the mold plate
6. Cool Down the press to 80^{o}Cand press for 10 more minutes
7. Take the sample out of the press and allow to cool to room temperature

Potential Pitfall: HA particle are not dispersing into <200 nm particles as we are assuming

Resolve by: Using Dynamic Light Scattering (DLS) to measure particle size of: non-sonicated HA-Solvent mixture and sonicated HA-Solvent mixture

Potential Pitfall: Centrifuging could be speeding up the process of gravity and maybe helping the phase separation by pushing the more dense HA lower than the less dense HDPE

Resolve by: Boil away the solvent or filter the particle mixture out of the solvent

Protocol Version #2

1. Sonicate Hydroxyapatite (HA) in 30 ml of Ethanol for 5 minutes (5 second cycles at an amplitude of 30%)
2. Add high density polyethylene (HDPE) and sonicate for 10 minutes (5 second cycles at an amplitude of 30%)
3. **Using a hot plate, boil away the ethanol while mixing the solution with a magnetic stir bar**
4. Dry overnight in oven
5. Weigh out the amount of powder desired and press the powder under 150^{o}C and 45 MPa for 30 minutes

    a. Press using two metal plates a circular metal mold and a Teflon plate placed in between the top metal plate and the mold plate
6. Cool Down the press to 80°Cand press for 10 more minutes
7. Take the sample out of the press and allow to cool to room temperature

Potential Pitfall: By heating and mixing, we might be allowing the hA and HDPE to aggregate to themselves

Resolve by: Mixing and sonicating while boiling away the solvent. Potentially use a water bath sonicator and a rotator.

Protocol Version #3

1. Sonicate Hydroxyapatite (HA) in 30 ml of **Methanol (lower boiling point than Ethanol)** for 5 minutes (5 second cycles at an amplitude of 30%)
2. Add high density polyethylene (HDPE) and sonicate for 10 minutes (5 second cycles at an amplitude of 30%)
3. **Using a water bath sonicator, heat and sonicate to evaporate the solvent, all while also mixing the solvent with a rotator controlled by an ARC speed controller (Pine Instrument Company, Grove City, PA).**
4. Dry overnight in oven
5. Weigh out the amount of powder desired and press the powder under 150°C and 45 MPa for 30 minutes
    a. Press using two metal plates a circular metal mold and a Teflon plate placed in between the top metal plate and the mold plate
6. Cool Down the press to 80°Cand press for 10 more minutes
7. Take the sample out of the press and allow to cool to room temperature

Potential Pitfall: Teflon sheet gives under the pressure and heat. We may be applying the pressure onto the Teflon sheet and not enough to our sample

Resolve by: Use a different press/mold

Protocol Version #4

1. Sonicate Hydroxyapatite (HA) in 30 ml of Methanol for 5 minutes (5 second cycles at an amplitude of 30%)
2. Add high density polyethylene (HDPE) and sonicate for 10 minutes (5 second cycles at an amplitude of 30%)
3. Using a water bath sonicator, heat and sonicate to evaporate the solvent, all while also mixing the solvent with a rotator controlled by an ARC speed controller (Pine Instrument Company, Grove City, PA).

4. Dry overnight in oven
5. Weigh out the amount of powder desired and press the powder under 150°C and 45 MPa for 30 minutes
    a. **Press using a cylindrical piston mold diagramed below**
6. Cool Down the press to 80°Cand press for 10 more minutes
7. Take the sample out of the press and allow to cool to room temperature

Potential Pitfall: We are not allowing enough time for the polymer melt to flow and mix with the HA particles.

Resolve by: Heat and press for a longer duration

Protocol Version #5

1. Sonicate Hydroxyapatite (HA) in 30 ml of Methanol for 5 minutes (5 second cycles at an amplitude of 30%)
2. Add high density polyethylene (HDPE) and sonicate for 10 minutes (5 second cycles at an amplitude of 30%)
3. Using a water bath sonicator, heat and sonicate to evaporate the solvent, all while also mixing the solvent with a rotator controlled by an ARC speed controller (Pine Instrument Company, Grove City, PA).
4. Dry overnight in oven
5. **Weigh out the amount of powder desired and press the powder under 150°C and 45 MPa for 3 hours**
    a. Press using a cylindrical piston mold diagramed below
6. Cool Down the press to 80°C and press for 10 more minutes
7. Take the sample out of the press and allow to cool to room temperature

Potential Pitfall: The slow rate of cooling of the sample may be leading to the seperationg of the hydrophobic HDPE and more hydrophilic HA.

Resolve by: Cooling the sample by quenching

Protocol Version #6

1. Sonicate Hydroxyapatite (HA) in 30 ml of Methanol for 5 minutes (5 second cycles at an amplitude of 30%)
2. Add high density polyethylene (HDPE) and sonicate for 10 minutes (5 second cycles at an amplitude of 30%)
3. Using a water bath sonicator, heat and sonicate to evaporate the solvent, all while also mixing the solvent with a rotator controlled by an ARC speed controller (Pine Instrument Company, Grove City, PA).

4. Dry overnight in oven
5. Weigh out the amount of powder desired and press the powder under 150°C and 45 MPa for 3 hours
    a. Press using a cylindrical piston mold diagramed below
6. **Quench the sample by quickly transferring the mold into an ice bath**

Potential Pitfall: Quenching does not allow for the air to escape from the sample creating a bunch of air bubbles.

Resolve by: Using a surfactant to connect the hydrophobic and hydrophilic material in solution

Protocol Version #7

1. **Prepare surfactant solution by diluting Tween-20 solution into methanol solution**
    a. **1.62×10^{-9} ml of Tween 20 into 1 ml of methanol/Tween-20 solution**
2. Sonicate Hydroxyapatite (HA) in 30 ml of Methanol for 5 minutes (5 seconds cycles at an amplitude of 30%)
3. **Add 0.2 ml of surfactant solution and sonicate for 5 more minutes (5 seconds cycles at an amplitude of 30%)**
4. Add high density polyethylene (HDPE) and sonicate for 10 minutes (5 second cycles at an amplitude of 30%)
5. Using a water bath sonicator, heat and sonicate to evaporate the solvent, all while also mixing the solvent with a rotator controlled by an ARC speed controller (Pine Instrument Company, Grove City, PA).
6. Dry overnight in oven
7. Weigh out the amount of powder desired and press the powder under 150°C and 45 MPa for 30 minutes
    a. Press using a cylindrical piston mold diagramed below
8. Cool Down the press to 80°Cand press for 10 more minutes
9. Take the sample out of the press and allow to cool to room temperature

Potential Pitfall: Attempting to make these standards with smaller presses have proven to be difficult due to the high pressure involved.

Resolve by: Try using twin screws extruder. The shear force introduced by the screws may reduce the viscosity of the polymer melt and allow for better mixing. The mold on the Minilab machine is also small enough to extrude standards with desired dimensions.

Bibliography

G. Giro *et al.*, "Impact of osteoporosis in dental implants: A systematic review," *World J. Orthop.*, vol. 6, no. 2, pp. 311–315, Mar. 2015.

R. Adell, U. Lekholm, B. Rockler, and P.-I. Brånemark, "A 15-year study of osseointegrated implants in the treatment of the edentulous jaw," *Int. J. Oral Surg.*, vol. 10, no. 6, pp. 387–416, Jan. 1981.

T. Albrektsson, P. I. Brånemark, H. A. Hansson, and J. Lindström, "Osseointegrated titanium implants. Requirements for ensuring a long-lasting, direct bone-to-implant anchorage in man," *Acta Orthop. Scand.*, vol. 52, no. 2, pp. 155–170, 1981.

G. A. Zarb and A. Schmitt, "The longitudinal clinical effectiveness of osseointegrated dental implants: the Toronto study. Part III: Problems and complications encountered," *J. Prosthet. Dent.*, vol. 64, no. 2, pp. 185–194, Aug. 1990.

E. Bałczewska, L. Klimek, and A. Palatyńska-Ulatowska, "SEM analysis of dental enamel morphological structures on the basis of their replicas in patients with systemic calcium deficiency," *Cent. Eur. J. Biol.*, vol. 4, no. 2, pp. 190–195, Jun. 2009.

A. H. FRIEDLANDER, "The physiology, medical management and oral implications of menopause," *J. Am. Dent. Assoc.*, vol. 133, no. 1, pp. 73–81, Jan. 2002.

M. R. Norton and C. Gamble, "Bone classification: an objective scale of bone density using the computerized tomography scan," *Clin. Oral Implants Res.*, vol. 12, no. 1, pp. 79–84, Feb. 2001.

M. Yamazaki *et al.*, "Bone reactions to titanium screw implants in ovariectomized animals," *Oral Surg. Oral Med. Oral Pathol. Oral Radiol. Endodontology*, vol. 87, no. 4, pp. 411–418, Apr. 1999.

R. Wang, S. J. Eppell, C. Nguyen, and N. Morris, "Relative Contribution of Trabecular and Cortical Bone to Primary Implant Stability: An In Vitro Model Study," *J. Oral Implantol.*, vol. 42, no. 2, pp. 145–152, Jun. 2015.

P. Mah, T. E. Reeves, and W. D. McDavid, "Deriving Hounsfield units using grey levels in cone beam computed tomography," *Dentomaxillofacial Radiol.*, vol. 39, no. 6, pp. 323–335, Sep. 2010.

T. Reeves, P. Mah, and W. McDavid, "Deriving Hounsfield units using grey levels in cone beam CT: a clinical application," *Dentomaxillofacial Radiol.*, vol. 41, no. 6, pp. 500–508, Sep. 2012.

"CT artifacts: Causes and reduction techniques - Semantic Scholar." [Online]. Available: /paper/CT-artifacts-Causes-and-reduction-techniques-Boas-Fleischmann/916896f8af9acae46c5a9a4c28e51b31d061c0c5. [Accessed: 10-Aug-2017].

R. K. Roeder, M. M. Sproul, and C. H. Turner, "Hydroxyapatite whiskers provide improved mechanical properties in reinforced polymer composites," *J. Biomed. Mater. Res. A*, vol. 67A, no. 3, pp. 801–812, Dec. 2003.

I. M. de C. C. Silva, D. Q. de Freitas, G. M. B. Ambrosano, F. N. Bóscolo, and S. M. Almeida, "Bone density: comparative evaluation of Hounsfield units in multislice and cone-beam computed tomography," *Braz. Oral Res.*, vol. 26, no. 6, pp. 550–556, Dec. 2012.

"Hounsfield units - scale of HU, CT numbers | Classifications, online calculators, and tables in radiology." [Online]. Available: http://radclass.mudr.org/content/hounsfield-units-scale-hu-ct-numbers. [Accessed: 11-Aug-2017].

C. Suplee, “X-Ray Mass Attenuation Coefficients,” *NIST*, 17-Sep-2009. [Online]. Available: https://www.nist.gov/pml/x-ray-mass-attenuation-coefficients. [Accessed: 10-Aug-2017].
J. B. Frey, “An experimental and theoretical study of the dark current and x-ray sensitivity of amorphous selenium x-ray photoconductors,” Jan. 2013.
S. Schweizer *et al.*, “Preparation and characterization of calibration standards for bone density determination by micro-computed tomography,” *Analyst*, vol. 132, no. 10, pp. 1040–1045, Sep. 2007.
T. Razi, M. Niknami, and F. Alavi Ghazani, “Relationship between Hounsfield Unit in CT Scan and Gray Scale in CBCT,” *J. Dent. Res. Dent. Clin. Dent. Prospects*, vol. 8, no. 2, pp. 107–110, 2014.