

# Automated Quality Assurance System for Clinical CT Systems

BME 400

December 14, 2016

Team Leader: Heather Shumaker

Communicator: Connor Ford

BSAC: Sam Brenny

BPAG, BWIG: Rachel Reiter

**Client:**

Prof. Timothy Szczykutowicz

Dept. of Radiology & Medical Physics

**Advisor:**

Prof. John Webster

Dept. of Biomedical Engineering

**Abstract**

Computed tomography (CT) machines are tested regularly to ensure the machines are calibrated and functioning properly. Each time a scanner is tested, a medical physicist must conduct numerous tests to assess each component of the machine and the images it produces. The physicist must then record all testing results by hand and generate a report based on the results. The report outlines each testing procedure used by the physicist and the results of each test. These reports are sent to service technicians who replicate the tests to determine which adjustments need be made to the machine. Currently, there are no standard protocols for CT quality assurance testing. Due to the inconsistency in quality assurance reports, miscommunication between the technicians and the physicists is common. Any misinterpretation of the reports can delay CT adjustments, creating a problem for the entire facility. In order to expedite and standardize CT quality assurance testing, a software program was created to accept user input, automate calculations and CT image analysis, and generate testing reports. The program consists of a graphical user interface created in MATLAB and will help reduce communication issues as well as significantly decrease the time and effort involved in CT quality assurance testing.

# Table of Contents

<b>Introduction .....</b>	<b>4</b>
Motivation .....	4
Competing Designs .....	4
Problem Statement .....	4
<b>Client Information .....</b>	<b>5</b>
<b>Background .....</b>	<b>5</b>
Computed Tomography .....	5
CT Quality Assurance Tests .....	5
Design Specifications .....	6
<b>Preliminary Designs.....</b>	<b>7</b>
Design 1: Multi-GUI .....	7
Design 2: Text Document.....	7
Design 3: Master GUI.....	8
<b>Preliminary Design Evaluation .....</b>	<b>8</b>
Design Criteria.....	9
Ease of Use.....	9
Degree of User Interaction .....	10
Modularity .....	10
Speed .....	10
Safety .....	11
Cost .....	11
Proposed Final Design .....	11
<b>Fabrication &amp; Development.....</b>	<b>12</b>
Materials .....	12
Methods .....	12
<b>Final Prototype.....</b>	<b>13</b>
Basic Information .....	13
Safety .....	13
Artifacts .....	13
Noise .....	14
LCD .....	14
CT Number.....	14
CT Uniformity.....	15
Monitor .....	15
Beam Width .....	16
Protocol Review .....	17

Gantry Tilt.....	17
Slice Width.....	18
Dose .....	18
LaTeX.....	19
<b>Testing .....</b>	<b>19</b>
<b>Results .....</b>	<b>20</b>
Algorithm Design.....	20
ROI Evaluation.....	21
Pixel to Distance Calibration.....	21
Image Angle Calculation .....	21
ROI Center to Isocenter Distance Calculation .....	22
<b>Discussion .....</b>	<b>22</b>
Challenges .....	22
Relevance .....	22
Ethics .....	23
Future Work.....	23
<b>Conclusion .....</b>	<b>24</b>
<b>References .....</b>	<b>25</b>
<b>Glossary .....</b>	<b>26</b>
<b>Appendices .....</b>	<b>28</b>
A. Product Design Specification .....	28
B. Materials.....	31
C. Semester Schedule .....	31
D. Project Schedule & Responsibilities .....	32
E. Panel Layouts .....	33
F. Source Code .....	40

# Introduction

## Motivation

*Quality assurance*<sup>1</sup> (QA) of *computed tomography* (CT) machines is a very important process for the radiologist and medical physicists who use the machines. Physicians make diagnoses based on the images they see, and they rely heavily on the fact that they are actually seeing the regions the machine indicates they are seeing. Our motivation involves streamlining the testing and reporting process to save time for both the medical physicist and service technician. This goal of this project is to develop a computer program that will be used for *quality assurance* testing. This program will allow for a universal and standard reporting system as opposed to varying report formats and tests depending on the specialists, machines, and facilities involved.

Additionally, our program will have the ability to look back on results of previous scans and include trend lines to see if certain characteristics of the machine are declining below desired values. This feature can help the medical physicist decide which parameters to pay attention to and can prevent scanner issues from becoming too serious. Lastly, our program will greatly improve communication between medical physicists and service technicians. This will ultimately reduce the time it takes to fix any issues with the CT machine once the medical physicist sends out a report.

## Competing Designs

Two commercially available CT QA software programs include Image Owl and PIPSPRO. Both of these programs highlight their database and trending capabilities, along with other advanced features such as built-in test types and cloud-based services [1,2]. However, their complexity comes with a trade-off in the form of reduced flexibility and higher cost. More details about these products can be found in the Product Design Specification in Appendix A.

## Problem Statement

CT machines are carefully tested on a daily, weekly, monthly, and annual basis. Each time a CT machine is tested, many components of the machine are analyzed to ensure the machine is properly calibrated and working correctly. The complexity of the testing procedures makes CT quality assurance testing and reporting an extremely time consuming task. The results of each test are recorded manually and entered into spreadsheet-based reporting tools.

---

<sup>1</sup> Note: All italicized terms are further defined in the glossary at the end of the report.

The reports and testing procedures often vary between medical physicists making it difficult for the results to be replicated by CT service technicians. The two main goals of this project are to create standardized testing protocols for use within the facility and to automate the reporting process. The client would like a software program capable of reading *DICOM images* from various *quality assurance* tests, evaluating the images with minimal user interaction, generating a report from the results, and writing the results to a database to track scanner performance over time.

## Client Information

Dr. Szczykutowicz is an Assistant Professor in the University of Wisconsin School of Medicine and Public Health Departments of Radiology, Medical Physics, and Biomedical Engineering. He received his undergraduate degree in physics and earned his Master's and Ph.D. in medical physics at the University of Wisconsin-Madison. Dr. Szczykutowicz is involved in several clinical and research activities including optimizing CT scan protocols, patient *dose* monitoring, and developing protocol management methodologies. We will be assisting him in creating a program for optimizing the reporting process for CT quality assurance testing and standardizing the format and protocols used for these reports [3].

## Background

This design project aims to expedite the testing and reporting process currently used for testing the quality and performance of computed tomography machines.

## Computed Tomography

*Computed tomography* (CT) scans combine X-ray images accumulated from multiple angles to create cross sectional images of a target object through digital computer processing [4]. This type of scan has “revolutionized diagnostic radiology over the past three decades” [5]. CT scans provide physicians with valuable information regarding the anatomy and structure of human tissue and organs without the need to make incisions. A disadvantage of CT scans includes the *dose* of radiation the machine sends through the patient. The radiation can potentially be harmful if the *dose* is too large or if a patient receives a large number of scans.

## CT Quality Assurance Tests

*Quality assurance* tests are performed on CT machines to validate whether they are functioning properly or if a certain part of the machine requires repair. The machines have

various tests that are required at different time intervals [6]. There are simpler tests performed daily, more complicated tests performed weekly and monthly, and rather extensive tests performed on an annual basis. Daily tests, for example, may include multiple series of helical and axial scan evaluations with parameters such as detector coverage, speed, rotation time, and slice thickness. Annual tests may include *artifact* testing, noise and *CT number* uniformity, laser consistency, couch movement and levelness, *dose*, beam width, and *gantry tilt*, among others [7].

Image *phantoms* are objects used to evaluate CT machine performance that can be designed to mimic human or animal tissue. *Phantoms* can be used to test for position verification, *slice width*, and scan incrementation, among other characteristics [8]. *Phantoms* can be developed with different design interests in mind. These can include designs optimized for tests to ensure accurate and proper scanning for human scans, and also phantoms optimally designed for scans of small animals such as mice. This is important because human diseases can be modeled in small rodents and research studies are supplemented with CT scans of these disease-ridden animals [9].

There are a variety of different *quality assurance* manuals that outline the different tests that can be performed and their significance. The main goal for CT scans is to produce quality diagnostic images at the lowest possible radiation *dose* [10]. This can only be achieved if *quality assurance* tests are performed to check whether the expected radiation *doses* are actually observed.

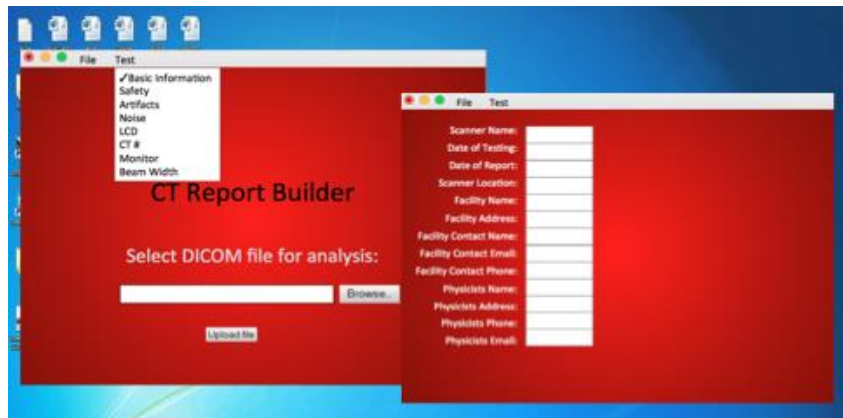
## Design Specifications

The client would like a software program capable of reading *DICOM images* from various *quality assurance* tests, evaluating the images with minimal user interaction, generating a report from the results, and writing the results to a database to track scanner performance over time. Ideally, the program will be packaged as an *executable* for distribution and will be capable of displaying machine trends in a chart. For further explanation of the design specifications, please refer to the Product Design Specifications in Appendix A.

# Preliminary Designs

## Design 1: Multi-GUI

Using *MATLAB's graphical user interface (GUI)* system, a program will be created capable of receiving user input, performing automatic calculations, and analyzing images. The code will be compiled into a single *executable* that is universally compatible with any operating system. The "Home Page" of the CT

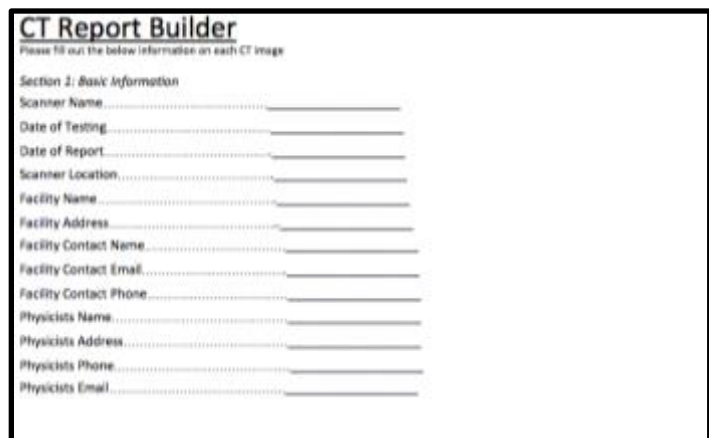


**Figure 1.** Representation of Design 1 showing multiple GUIs open at once.

Report Builder will allow the user to select a *DICOM image* or a specific test to analyze the data received from a CT system. Once a test is selected, a new *GUI* will appear with the input parameters for the specified test, as seen in Figure 1. After all desired tests and calculations are conducted based on the input parameters, the *GUI* will create a formatted text document with a full report on a specific CT scan.

## Design 2: Text Document

Design 2 features a PDF or Word document template. The template will have a section for each test that is performed and will have a fill-in-the-blank format. This design may also feature checkboxes and other options for user input, as seen in Figure 2. Once, the template is filled in, it will be used to generate a text or word document containing the testing protocols and results. The document can alternatively be printed and used in hard-copy format.



**Figure 2.** Design 2 featuring a PDF or Word format document of a CT Report.



## Design 3: Master GUI

Design 3 will have the same functionality as Design 1. However, all the testing protocols and information will be contained in a single master *GUI*. Each test will remain in an individual *panel* that will become visible and editable when selected from the drop down bar in the top left corner, as seen in Figure 3. Alternatively, the user can select another test at any time and the *GUI* will automatically update its input parameters and save previous test data. The drop down bar, save, and export button will





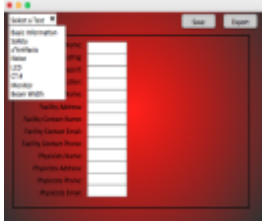
**Figure 3.** A representation of the dynamic functionalities of the Master *GUI* in Design 3.

always be visible at the top of the *GUI*. The export button will allow the user to generate a report at any time and the save button will allow the user to save data during each step. This software design will allow the user to enter all testing information in a single program with a simple and user-friendly interface. Finally, all the testing results are saved to a database where the information can be accessed to display testing trends of certain parameters from a CT machine over time.

## Preliminary Design Evaluation

To assess the feasibility and effectiveness of each of our three designs, we developed a design matrix to evaluate each design on its ease of use, degree of user interaction, modularity, speed, safety, and cost. The evaluation and description of each criterion are detailed in Figure 4.

**Figure 4. Design Matrix**

Design	Design 1		Design 2		Design 3	
<b>Criteria (Weight)</b>	 Multi-GUI		 Text Document		 Master GUI	
Ease of Use (30)	4/5	24	3/5	18	5/5	30
Degree of User Interaction (25)	5/5	25	0/5	0	5/5	25
Modularity (20)	2/5	8	0/5	0	4/5	15
Speed (15)	3/5	9	0/5	0	5/5	15
Safety (5)	5/5	5	5/5	5	5/5	5
Cost (5)	5/5	5	5/5	5	5/5	5
<b>Total (100)</b>	76		28		95	

## Design Criteria

The design criteria used to assess the feasibility and effectiveness of our designs includes ease of use, degree of user interaction, modularity, speed, safety, and cost. Safety and cost were required criteria and the other four were created to assess the most important features of our designs.

### Ease of Use

A logical and intuitive program is crucial to the success of providing effective *quality assurance* to the CT machine and images. The user conducting the various tests requires a sleek, yet simple environment to input and manipulate data and ultimately create a concise final report. The program must be visibly pleasing and contain a structured workflow that allows variability for different test sets. The final product must be easily obtainable and should require minimal effort to download and install, packaged with all necessary libraries in order to offer standalone functionality on all types of computers.

*Rationale:* Design 2 is bulky and tedious to use, requiring the user to complete the full report. Design 1 and 3 are far easier to use, but Design 3 has all the information, tests, and functionalities contained in a single interface, which greatly increases the ease of use.

## Degree of User Interaction

The overall goal of this project is to decrease the time and effort it takes to generate testing reports. We would like this program to be automatic with minimal user interaction. The user will need to enter all parameters from the test and then, ideally, the program should be capable of processing all data and performing the necessary calculations automatically.

*Rationale:* Using *MATLAB* to calculate data based on user input makes Designs 1 and 3 exponentially easier to use compared to Design 2; this significantly reduces the degree of user interaction. In addition, automatically creating and storing reports in a database will decrease the overall time and effort of testing and reporting.

## Modularity

The code format is important, not only for ease of development and debugging, but also for future alterations. As new testing requirements and methods change, certain portions of the code must be easily accessible for modifications. This aspect of the design is essential for implementation of an *open-source application* that can be modified by the user.

*Rationale:* Unlike the single program featured in Design 3, Design 1 would require multiple files, figures, and *GUIs* for each test, which could cause issues when debugging and modifying code. Additionally, there may be complications in packaging Design 1 into a universal *executable* due to the multiple program files.

## Speed

It is important that CT machines are tested on a regular basis to ensure all functions are working properly for research and patient imaging. Because these tests are performed so often, it is important for the program to process data and generate testing reports very quickly. A quick turnaround is essential in order for the CT machines to be adjusted before further use.

*Rationale:* *MATLAB's* capability to perform complex calculations and the simple user interface of Design 3 makes this design the most ergonomic and effective method to create CT scan reports.

## Safety

Care should be taken to minimize visual strain, such as using sufficiently large font size and bright colors. Additionally, the overall accuracy and reliability of the program affects the calibration of the CT scanner, which ultimately contributes to patient safety.

*Rationale:* Safety is a required design criterion.

## Cost

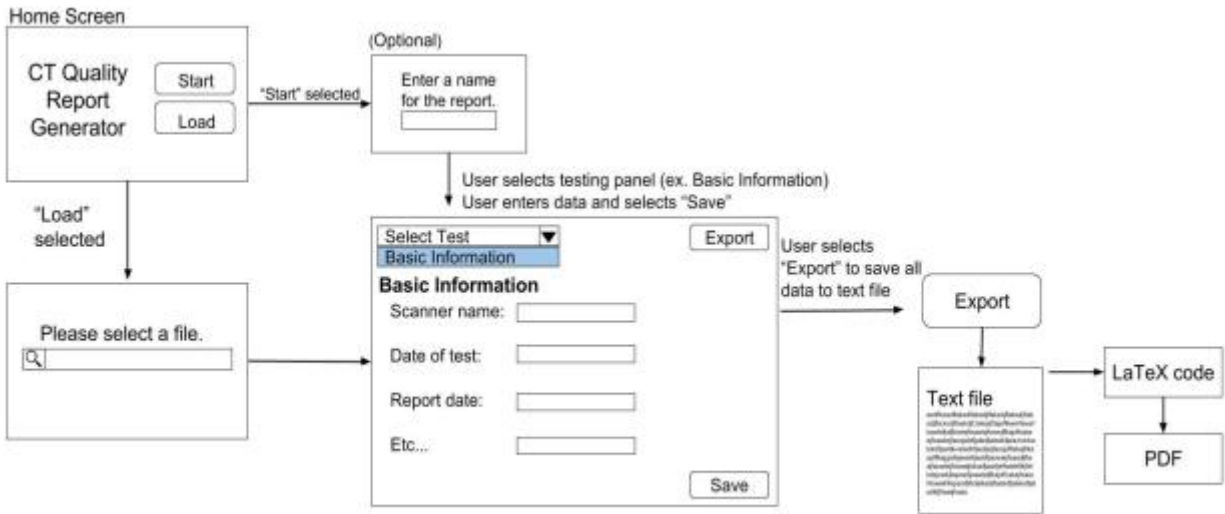
Many CT scans are conducted daily and in rapid succession, so the efficiency at which quality assurance tests can be completed greatly influences cost of use. The speed at which CT scans can be analyzed will cut into cost. In addition, the designs require third-party software, which can vary in cost but is free for UW students and staff.

*Rationale:* Cost is a required design criterion.

## Proposed Final Design

Our team chose to pursue Design 3, which consists of a single *GUI* featuring multiple *panels* and a drop-down bar to select the type of test. This design was chosen for its compactness, user-friendly interface, and low degree of user interaction. This design was chosen over the multi-*GUI* design for its increased speed, modularity, and ease of distribution.

We concluded that this design would be faster than the multi-*GUI* because, unlike the multi-*GUI* design, the single *GUI* is contained in one program and will not generate additional windows or programs. It will also be easier to distribute and implement future modifications because all of the code is contained within a single program. If changes need to be made to the multi-*GUI* it is likely that the user would have to search through several programs to make the desired change. Furthermore, with multiple *GUIs* such as in Design 1, data would have to be shared between multiple programs, adding an additional challenge to the software fabrication. Overall, Design 3 meets all of the design specifications provided by the client and will feature a seamless user interface, automated calculations and image analysis, and historical scanner trending. A block diagram of the proposed final design can be seen in Figure 5.



**Figure 5.** A block diagram of the final proposed design. This includes a main menu with the option to create a new test or load previous testing parameters, a drop-down menu to select the type of test, and multiple panels that become visible when selected by the drop-down menu.

## Fabrication & Development

### Materials

The materials used in the fabrication of this design project include *MATLAB*, a software program licensed by MathWorks, and *LaTeX*, a software system for document formatting. Each team member used the University’s computer labs or his or her own computer to develop the code for the final program. The materials, in further detail, can be found in Appendix B.

### Methods

To create the framework for the program, we met with the client to discuss the desired capabilities of the program. The client created a spreadsheet detailing each section of the code to correspond to each type of *quality assurance* test performed. We used the spreadsheet to make a skeleton for the program by creating “*panels*” for each section of the code. The sections of code were divided among three team members to match each member’s skillset. The remaining team member was assigned to the *LaTeX* code specifically. Each team member worked on his or her sections of the code independently. Throughout the semester, we met with the client to discuss our questions and receive feedback on our existing work. When all sections of the *GUI* were completed individually, the sections were seamlessly combined in the “master” *GUI* into their respective *panels*. To view the project schedule, please refer to Appendix C.

# Final Prototype

The final prototype consists of a single *graphical user interface* developed in *MATLAB*. The program functionality is broken up into individual *panels*, each capable of accepting user input, performing calculations, or analyzing testing results. The program currently has thirteen *panels*. Following are detailed descriptions of each *panel's* features and functionality. Please see Appendix D for individual responsibilities regarding the *panels* and Appendix E for images of the *panels* during program use.

## Basic Information

The basic information *panel* requires user input relating to the basic information of the test. Examples of the information include the testing date, scanner type and location, the facility contact information, and the physicist contact information. If information is stored in the *DICOM* header of a CT image, then the program will acquire that information automatically.

## Safety

This *panel* consists of a series of questions regarding the room set-up safety precautions. Some of these questions include the functionality of the “X-Ray On” light in various locations, the ability to disable couch movement, and proper intercom performance and volume. Each question requires a Yes/No/NA selection with an optional comment box for additional observations to be noted.

## Artifacts

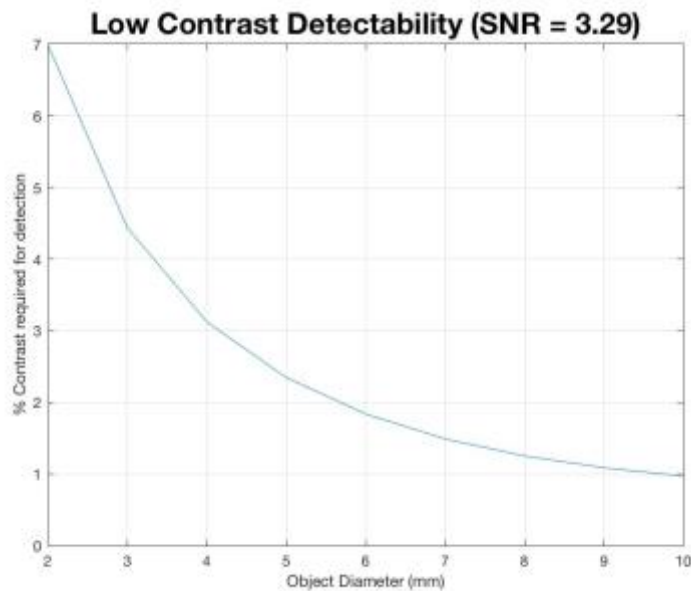
*Artifacts* are often present in clinical CT scans due to unforeseen circumstances that occur during the initial acquisition of data. Analyzing a CT image for *artifacts* is essential in the *quality assurance* process. Calculating the standard deviation of the region where the *artifact* is present is useful for diagnosing and fixing the problem. The artifacts *panel* allows the user to draw a circular *region of interest (ROI)* on a selected *DICOM* image and then calculates the pixel values and the standard deviation of these pixels for each *ROI*. When an image is selected, the code automatically obtains *DICOM* header information and is utilized for calculations. For example, the slice thickness is used to determine pixel width in millimeters and then calculates the distance of the *ROI* from the *isocenter*.

## Noise

Noise is always present and is an uncontrollable source of error in CT data acquisition. As a result, calculations must be completed to correct and properly document the magnitude of this source of error. This *panel* performs calculations of the standard deviation are conducted from a uniform data set to quantitatively determine the noisiness of the data. A *DICOM* image must be uploaded in order to document noise information. These calculations are saved and documented in the final report for further analysis.

## LCD

The low contrast detectability (LCD) *panel* of the code allows the user to select two images. The rest of the calculations are executed through code provided by the client. Figure 6 shows that the code produces a plot featuring the low contrast detectability of the selected images.



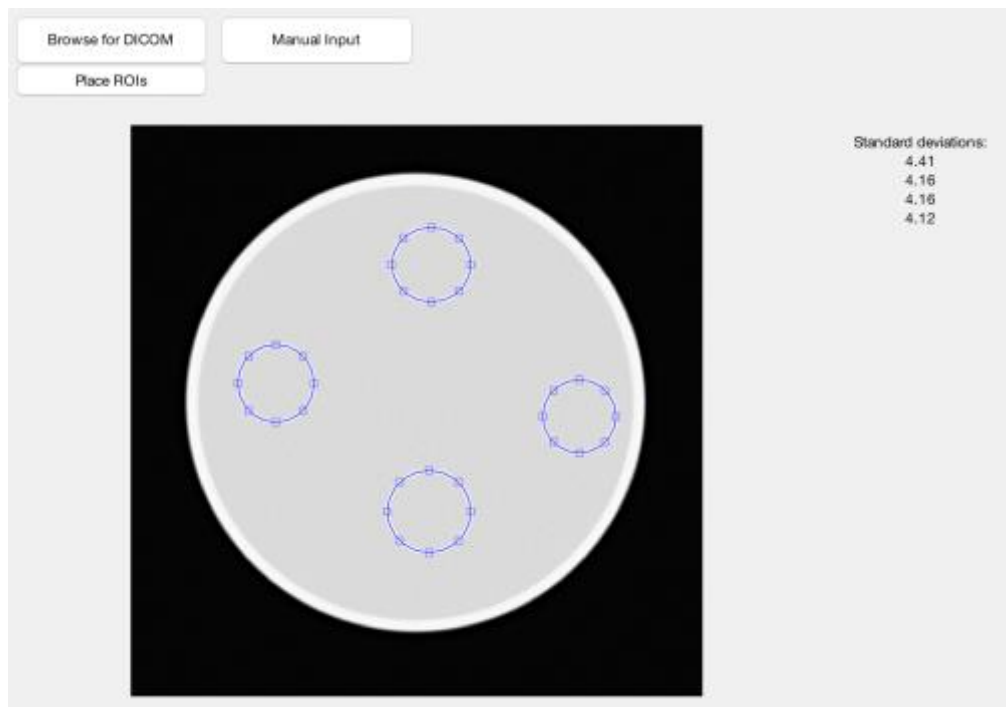
**Figure 6.** The LCD panel produces a plot of the low contrast detectability of the images. This plot was produced using two example images and does not accurately represent typical data.

## CT Number

This portion of the code allows the user to browse for a *DICOM image* and draw a circular *ROI* over a portion of the image. The program automatically computes the *CT number* of the pixels inside the *ROI*, displays it to the user, and saves it to the program. This *panel* functions similarly to the CT Uniformity *panel* in Figure 7 and the Artifacts and Noise *panel* mentioned above.

## CT Uniformity

The CT Uniformity *panel* functions similar to the Artifacts, Noise, and CT Number *panels* and allows the user to browse for a *DICOM image* and then draw a circular *ROI* around a certain region of the image. The program computes the standard deviation of the pixels inside each *ROI* and displays these to the user while saving them to the program. The CT uniformity *panel* is displayed in Figure 7.



**Figure 7.** The CT Uniformity panel of the program has buttons to allow the user to browse for an image and place ROIs on certain parts of image. The program calculates the standard deviation of the pixel values inside each ROI and displays them to the right. Note that this image does not have any inserts and was used for demonstration purposes.

## Monitor

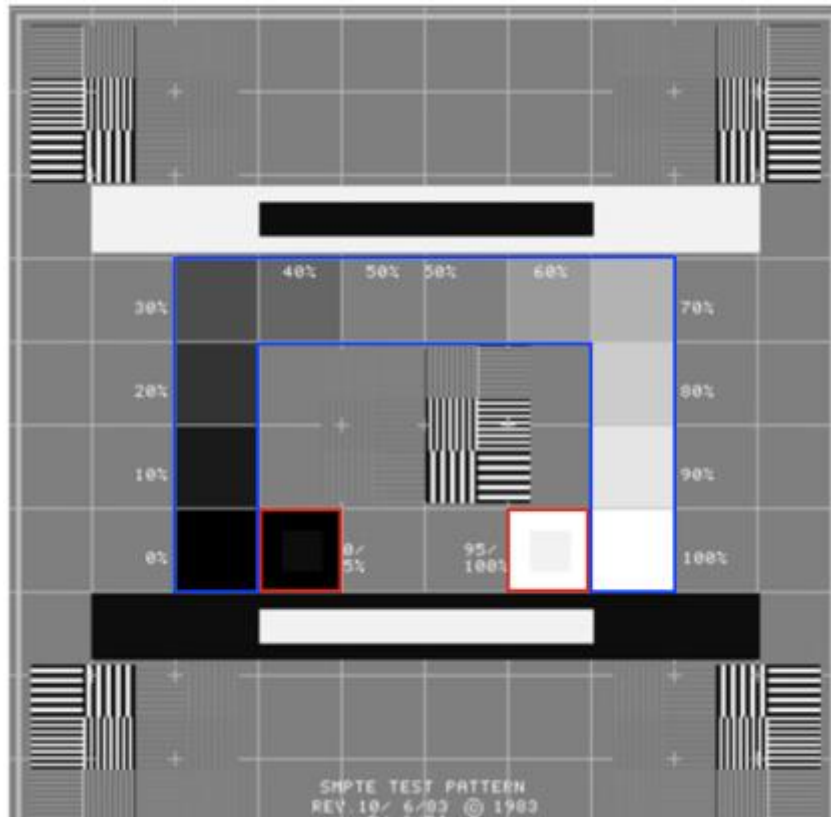
The monitor *panel* is used to assess the quality of the screen's display. First, a test pattern must be selected from the drop-down menu. Options include, 'SMPTE Pattern', 'All Black Image', 'All White Image', or 'Other'. If 'Other' is selected, a new field will appear for the user to specify the test pattern name. A series of questions about the test pattern image must then be answered with a Yes/No/NA selection and optional comment. For example, using the *SMPTE test pattern* pictured in Figure 8, the user must assess whether the 0%-100% patches of grayscale (outlined in blue in Figure 8), are all distinguishable. Additionally, the 0/5% and 95/100% patches (outlined in red in Figure 8), must each contain a discernible square within the patch. Lastly, to assess screen luminance, the



user must adjust the screen brightness to its minimum level, enter the screen luminance at five locations (the center of the screen and the four corners), and average the recorded values. This process is then repeated with the screen brightness at its maximum level. The program uses the average minimum and maximum luminance values to compute the nonuniformity of the display brightness with Equation 1.

$$\% \text{ Difference} = 200 * (L_{max} - L_{min}) / (L_{max} + L_{min})$$

**Equation 1.** Nonuniformity of display brightness.  $L_{max}$  and  $L_{min}$  represent the average maximum and minimum screen luminance, respectively.

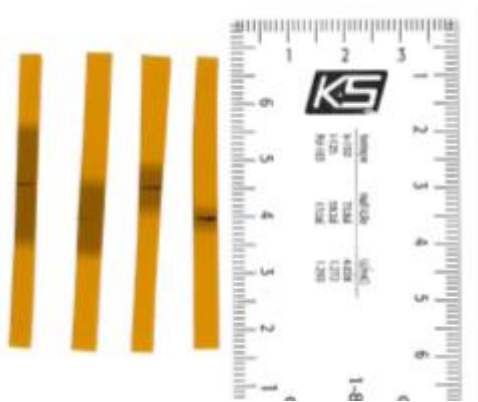


**Figure 8.** SMPTE Test Pattern. Widely used testing pattern for the evaluation of display systems for medical diagnostic imaging. Most important pattern components for CT QA include the 0-100% patches outlined in blue in addition to the 0/5% and 95/100% patches outlined in red.

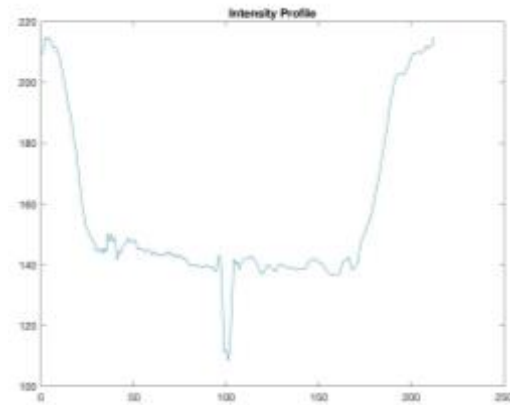
## Beam Width

The beam width *panel* allows the user to calculate the width of the x-ray beams. Strips of paper are exposed to x-ray beams, which create a darkened region on each strip. The width of each darkened region is calculated by 1) drawing a line over a known distance on a ruler as seen in Figure 9.1, 2) typing the known distance in a pop-up window, 3) asking the user to draw a line over the darkened region of each strip of paper, 4) calculating the intensity profile of the region covered by the line as seen in Figure 9.2, 5) calculating the points of the maximum intensity and half of the maximum intensity, 6)

calculating the beam widths based on the corresponding intensities and their location on the image, and 6) relating the pixel widths to millimeters using the calibration factor.



**Figure 9.1.** The width of the darkened portion of each strip represents the beam width. The user calibrates the pixels



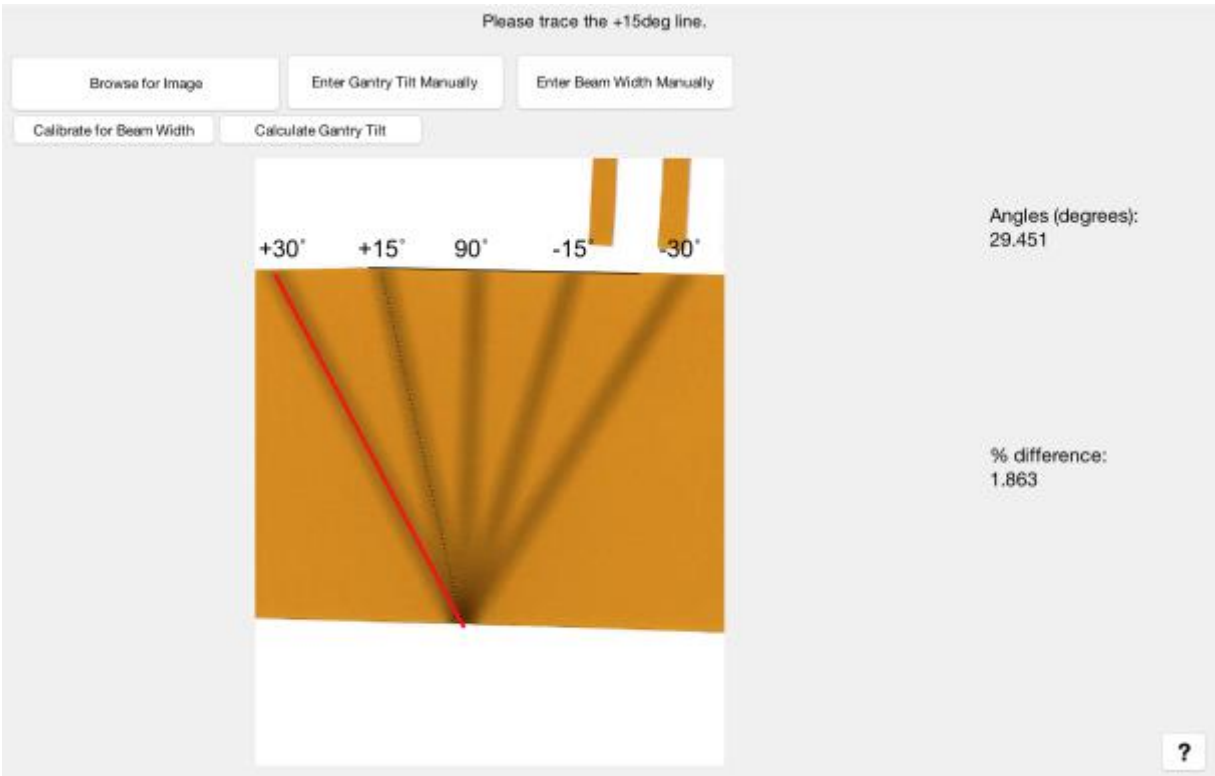
**Figure 9.2.** The intensity profile of the beam width of the first strip. This prompts the user to select the max intensity and min intensity.

## Protocol Review

Protocol review is required to ensure the diagnostic image quality is achieved with the lowest *dose* of radiation possible. Examples of protocols that require annual review include pediatric head, pediatric abdomen, adult head, adult abdomen, high resolution chest, and brain perfusion [11]. Protocol parameters are programmed into the machine and subject to adjustment whenever under review. This *panel* is designed to document these changes by allowing the user to enter the protocol name, flag it as 'Okay', 'Needs Attention', or 'Changed during Survey', and indicate the changes implemented and/or needed in the corresponding comment section. This can be performed for up to six protocols.

## Gantry Tilt

This portion of the code computes the *gantry tilt* by calculating the angles between five lines that were created on paper by the x-ray beams. The program prompts the user to draw a line parallel to the top of the paper to calibrate for zero degrees, as seen in Figure 10. The program calculates the 90° angle based on the user-selected 0° angle. The user then draws a line over each beam to calculate the angle relative to the 90° line. Lastly, the program displays the angles to the user, calculates the percent difference between the calculated and nominal angle values, and saves the data to the program.



**Figure 10.** The user draws a line over the top of the piece of paper. This represents the  $0^\circ$  line. Then the user snaps a line over each beam to calculate the angle between the beam and the  $90^\circ$  line. The angles are displayed on the right and the percent difference between the calculated angle and nominal angle is also displayed.

## Slice Width

This portion of the program allows the user to browse for a CT image and gets the *slice width* by extracting information from the *DICOM* image header. The *slice width* is the width of the portion of the patient that was imaged during the scan. The *slice width* value is returned to the user and saved to the program.

## Dose

*Dose* is evaluated by measuring the radiation exposure with a dosimeter probe. The probe is placed in or on the phantom, a scan is performed, the exposure is then measured by the dosimeter and entered into the program by the user. This process is performed three times with the probe in the center of the phantom and three more times with the probe on the phantom's surface. The center and surface values are averaged separately and used to calculate the *CT Dose Index* ( $CTDI_{100}$ ) from the averaged exposure values using Equation 2 and user entered beam width. Both  $CTDI_{100,center}$  and  $CTDI_{100,surface}$  are used in the program's calculation of the weighted *CTDI* ( $CTDI_w$ ) with Equation 3. Lastly,

the user enters the  $CTDI_w$  displayed on the CT scanner's screen so the percent difference between the scanner and dosimeter *dose* measurements can be determined.

Note: If the user already has the  $CTDI_w$  measurement from the dosimeter, therefore not needing the intermediate calculations, this value can be entered directly.

$$CTDI_{100} = (\text{Average Dose} \times 100) / \text{Beam Width}$$

**Equation 2.**  $CTDI_{100}$  calculation. The average *dose* is multiplied by the CTDI subscript, in this case 100, and divided by the beam width to determine CTDI.

$$CTDI_w = \frac{1}{3}CTDI_{\text{center}} + \frac{2}{3}CTDI_{\text{surface}}$$

**Equation 3.**  $CTDI_w$  calculation. The center and surface CTDI values are weighted differently and summed to determine the weighted CTDI.

## LaTeX

*LaTeX* is a document preparation system that creates richly formatted PDF files. When the user decides to export their results from the report builder program, *MATLAB* writes all of the user-inputted information and test results into a text file that is properly formatted for *LaTeX*. This process involves taking in all of the variables from the various *panels* and placing them into the text file using proper *LaTeX* syntax so that when the text file is sent through *LaTeX*, the PDF can be created correctly. If a certain image is desired to be included in the report, the *MATLAB* code will find the full link to where that image is stored in the user's computer and will print the link in correct syntax to the text file. The benefit of using this system to create the final report is that it will always be formatted in the same way. This creates a standardized format for the report which will help service technicians working on the CT machines. They will always know where to look for the results of a given test, and they will be more familiar with the full set of testing processes as the same procedures will be performed for each test and will be documented in a consistent manner.

## Testing

The program will be tested by having volunteer testers with knowledge in the field of *quality assurance* for CT systems use the program and fill out a qualitative and quantitative evaluation of both the functionality and ease of use of the program. The protocol will have specific questions about the various *panels* regarding how intuitive they

are to use and will include questions where the testers have to rate specific aspects of the program on a scale from zero to ten. In addition to the numeric rankings, the testers will have the option to make specific comments on any aspects of the program as they see fit. All of the numeric rankings will be put through statistical analysis to determine which aspects of the program need the most improvement and which aspects performed well. In addition, the tester will write down how long it took for them to create their report using the program and compare it to the time it would take them to create it manually, as one goal of the project is to vastly expedite the test report building process.

The results are expected to inform us on how intuitive the program is to use as a full *quality assurance* test report builder. The results will give insight into the potential time savings that using the program will introduce to the process as well as the potential convenience of having standardized reports regardless of who performed the testing or who used the program. Another goal of the testing protocol is to determine if there are any errors in the functionality or glitches in the program that would result in failure to reliably create an accurate report. Based on the results received, we will be able to improve the program and talk with the testers to decide on which changes and additions should be made to increase functionality and ease of use.

## Results

The program we created will significantly improve the *quality assurance* reporting process by improving communication between the physicist and service technicians and by reducing the time it takes to create these reports. Prior to our program, the medical physicist who conducts the *quality assurance* test would perform all calculations manually or through several different software programs including ImageJ and *ROI* contouring software programs. Our program has combined the capabilities of these programs into a single *GUI* by developing algorithms for these functions from scratch. Our program also improves upon these capabilities by performing calculations automatically within the program while requiring minimal user interaction. The program will be distributed throughout research groups in the Wisconsin Institute for Medical Research and will eventually be distributed to other medical physicists outside of UW - Madison. Through the program developed by our team, we hope to significantly reduce the time and effort it takes to perform QA tests on CT machines and also aim to standardize QA reporting to improve the communication between the people involved in testing.

## Algorithm Design

There were four main algorithms designed for this program from scratch by our team members. One for the beam width *panel*, one for the *gantry tilt panel*, one for the *artifacts*

*panel*, and one used in the *CT number*, CT uniformity, noise, and *artifacts panels*. The source code for these algorithms can be found in Appendix F.

1. ROI evaluation
2. Pixel to distance calibration
3. Image angle calculation
4. ROI center to isocenter distance calculation

## ROI Evaluation

The *ROI* evaluation component of the program allows the user to load a *DICOM image* and draw a circular *ROI* on the image. The code extracts the locations of the image surrounded by the *ROI* and correlates the locations to pixel values of the original image. The code then calculates either the mean or standard deviation of the pixel values inside each *ROI*. This algorithm is used for computation in the noise, *artifacts*, *CT number*, and CT uniformity *panels*. Previously, the client performed these calculations through an external software program - the need for this *ROI* program is eliminated through this *ROI* evaluation algorithm.

## Pixel to Distance Calibration

The pixel to distance calibration algorithm is used in the beam width section of the program. This part of the code relates a known distance in millimeters to a distance in pixels of an image and creates a calibration factor to convert pixel distance to distance in millimeters. The images used for the beam width *panel* must include a ruler. The program prompts the user to draw a line over a known distance (e.g. from 1 to 2 cm on the ruler) and enter the known distance in centimeters. The program correlates this known distance to the number of pixels on the drawn line and creates the calibration factor for pixel to millimeter calculations. Previously, ImageJ was used for this test and all calibration calculations had to be done manually. This algorithm completely eliminates the use of ImageJ in the beam width testing protocol.

## Image Angle Calculation

The image angle calculation works by allowing the user to draw a reference line on an image. The code converts the line drawn into a vector. Then the user can place a line over the angle of interest and the code will convert this line to a vector and then compute the angle between the two vectors using dot and cross product calculations. This algorithm is used in the *gantry tilt panel* to calculate the angle of the *gantry* (the torus-shaped portion of the CT machine) relative to 0° and 90°.

## ROI Center to Isocenter Distance Calculation

This algorithm is used in the *artifacts panel* to calculate the distance from a selected *artifact* to the center of the image. The code for this algorithm uses the *ROI* evaluation functionality mentioned above to determine the pixels in the *ROI*. Then the code determines the center pixel of each *ROI* and calculates the distance from that pixel to the center of the entire *DICOM image*.

## Discussion

### Challenges

Throughout development of the program, we faced various challenges and setbacks. The decision to stick with using one universal *GUI* for the program led to logistical issues within *MATLAB*. There was a lot of variable name matching that needed to be accounted for as well as making sure all *panels* were visible on the screen at the right time and invisible otherwise. Through rigorous iterations of running through the code and checking variable names, we were able to match all variables between the *GUI* and the written *MATLAB* code to create a fully functional program. A challenge that we will have in the future involves other interested parties that may want the ability to add a different *panel* to the program. As the program is structured currently, it would be difficult to add in a new *panel* without previous knowledge of how the program is structured and how variables need to be named to interact with each other. This problem can be solved by making a video tutorial and user manual outlining how to create a new *panel* and integrate it into the existing program.

Another challenge we have is receiving *DICOM image* info from tags that come along with a *DICOM image*. They are not labeled in an obvious manner so it will take additional research and comparison to see which data is available for us to input into the program automatically versus which information the user will have to manually input. Finally, the greatest challenge will be the distribution and implementation of the design to those in the relevant field and ensuring universal compatibility.

### Relevance

Traditionally, it can take a lot of time for a machine with a functional problem to be tested by a physicist and repaired by a service technician. Creating a program that automates and accelerates this process is highly valued as it reduces the amount of time a machine spends in repair. This can be the difference between a patient being able to receive a CT scan on Tuesday instead of Thursday during a week, which can speed up the patient's diagnosis and recovery process. Additionally, the ability for this program to standardize

the reporting process helps service technicians and physicists alike have a better and more transparent understanding of how the tests are conducted. Instead of varying from physicist to physicist and machine to machine, the reports are consistent across machines and facilities allowing for smoother communication from all pertinent parties trying to fix a machine. Finally, there is no universal *quality assurance* system in medical imaging and each physicist, service technician, and doctor uses different methods and procedures for CT calibration and repair. The program that was developed will allow professionals in the medical field to report diagnostic information of medical images in a standardized format. This will create a more uniform procedure and allow doctors across the country to share information faster.

## Ethics

The final software program will be used for research and clinical purposes. With that in mind, it is imperative that the program performs accurate calculations and image analysis. The values and test results exported by the program must be verified to ensure they are not a result of faulty program function. Failure to do so could lead to inaccurate assessment of the CT machine, resulting in improper diagnoses and/or additional testing, which can increase patient exposure to harmful x-rays. Furthermore, there will be no malicious code in the downloadable software package. Overall, the code will perform the functions it cites accurately without harming the computer. This plays a vital role in proper calibration on the CT scanner and ultimately influences patient safety.

## Future Work

Next semester, our team would like to focus mainly on testing and improving the program. We intend to develop a user-experience survey to distribute with the program. Our client offered to present our product for beta testing at a radiology conference he attends annually to obtain valuable feedback from prospective users. After forming a testing protocol and collecting the testing data, it will need to be analyzed to determine the necessary improvements for the next generation of the program. Additionally, the updated program will include more tests and functionality. A key supplement in functionality includes a database trending addition to the program. This function would allow users to bring up results from previous tests on the same machine, and graph those results over time. It will be able to show if a certain parameter of the machine is trending towards failure, and would allow service technicians to maintain that aspect of the machine before it actually becomes a problem.

We also plan on working closely with the client to develop a comprehensive user manual to not only outline the program's operation, but also document standardized procedures



for collecting the data entered into to the program. The final goal is to publish the details of our program and findings in a scientific journal.

## Conclusion

The overall goal of this design project is to develop a software program that aids in CT *quality assurance* testing by decreasing the time and effort involved and by developing standardized testing protocols to eliminate communication issues. The program was developed using *MATLAB* and *LaTeX* to accept user input, perform calculations and tests, and ultimately create a PDF file. The file can then be sent to service technicians so they can make any necessary repairs to the CT machine in question. The user interface allows for a highly intuitive ability to input information about the machine that is significantly faster than the traditional methods of creating the PDF file manually. As with any open source software development, the program will always be available for improvement and updates as other parties desire to add functionality to the program.

# References

- [1] "Comprehensive QA Services in the Cloud," Image Owl, Inc. [Online]. Available: <http://www.imageowl.com/>. [Accessed: 09-Oct-2016].
- [2] "PIPSpro Software," Standard Imaging, Inc.. [Online]. Available: <http://www.standardimaging.com/qa-software/pipspro-software/>. [Accessed: 09-Oct-2016].
- [3] "Faculty and Staff," University of Wisconsin School of Medicine and Public Health. [Online]. Available: <https://www.radiology.wisc.edu/people/facultyContent.php?vaultID=552>
- [4] "CT Scan," Mayo Clinic [Online]. Available: <http://www.mayoclinic.org/tests-procedures/ct-scan/basics/definition/prc-20014610>
- [5] G. T. German, "Fundamentals of Computerized Tomography: Image reconstruction from projection" Springer 2nd edition, 2009.
- [6] S. Mutic *et al.*, "Quality assurance for computed-tomography simulators and the computed tomography-simulation process," *Medical Physics* 30 (10). Oct. 2003.
- [7] T.P. Szczykutowicz. "CT Scanner Annual Testing: East Clinic UWMC DHO," UW-Madison Dept. of Radiology. Madison, WI. July, 2016.
- [8] D. J. Goodenough, "Catphan 500 and 600 Manual" Salem, NY. 2006.
- [9] L. Y. Du *et al.*, "A quality assurance phantom for the performance evaluation of volumetric micro-CT systems," *Phys. Med. Biol.* 52 7087. Nov. 2007.
- [10] D. D. Cody *et al.*, "Computed Tomography: Quality Control Manual," ACR, 2012.
- [11] D. Cody, "CT Protocol Review: Practical Tips for the Imaging Physicist," in *The American Association of Physicists in Medicine*, 2013. [Online]. Available: <http://amos3.aapm.org/abstracts/pdf/72-20324-242393-90804.pdf>. Accessed: Dec. 5, 2016.

# Glossary

**Artifact:** an aberration in a CT image often caused by poor calibration

**CT dose index (CTDI):** standardized measure of radiation dose of a CT scanner which allows for comparison between different scanner doses, a number subscript indicates the length of the ionization chamber in mm (e.g. CTDI<sub>100</sub> is the CTDI measurement for a 100mm ionization tube)

**CT number:** a variable that describes the radiodensity of a pixel in a DICOM image

**DICOM images:** DICOM stands for “Digital Imaging and Communications in Medicine” and is the standard image type for storing and transmitting medical imaging information and scans. DICOM images store information pertinent to the scan such as scan location and many other important parameters.

**Dose:** the concentration of energy deposited in the tissue from exposure to ionizing radiation

**Executable:** a file that can be run by any computer without downloading other software

**Gantry:** the torus-shaped component of the CT machine that the patient slides through

**Gantry tilt:** the angle between the vertical plane and the x-ray beam of the gantry

**Graphical user interface (GUI):** a user interface that allows user to interact with a program by buttons, graphs, and text

**Isocenter:** the point in an image that is in the exact center both horizontally and vertically

**LaTeX:** a document-preparation system for typesetting and formatting text

**MATLAB®:** a high-level programming language for technical computing

**Open-source application:** software with the source code available to the public for editing

**Panel:** a MATLAB GUI component that allows code with different functionalities to be compartmentalized into panels that can be made visible or invisible with the click of a button

**Phantom:** a cylinder filled with water or another liquid that is made to mimic the human body used for scanner testing

**Slice width:** the thickness of the part of the body/phantom that is being imaged

**SMPTE test pattern:** a standard developed by the Society of Motion Picture and Television Engineers used for testing and evaluating the display quality of medical imaging devices

# Appendices

## A. Product Design Specification

**Title:** Automated Quality Assurance System for Clinical CT Systems

**Client:** Prof. Timothy Szczykutowicz

**Advisor:** Prof. John Webster

**Team Leader:** Heather Shumaker

**Communicator:** Connor Ford

**BPAG & BWIG:** Rachel Reiter

**BSAC:** Sam Brenny

### **Function:**

A software program will be designed and built to aid in computed tomography (CT) quality assurance testing and reporting. The software will process testing results and export them to a report analyzing the results and reporting corrections that must be made to the CT system. The report will also specify how the tests are conducted.

### **Client requirements:**

The client would like the software to be capable of:

- Processing DICOM (Digital Imaging and Communication in Medicine) images that represent quality assurance test scans
- Automatically analyze images
- Create reports from the test outputs into easy to read report using LaTeX [1]
- Write test results to a database
- Ability to pull up past results in trends chart

Preferably, the program will consist of a graphical user interface (GUI) with a user-friendly interface. Ideally, the program will be capable of doing several automated calculations for the client.

### **Design requirements:**

The program will be developed in MATLAB and be exported to LaTeX and then to a PDF. The client prefers that the program be capable of allowing user input of test values in whichever order the user chooses.

## **1. Physical and Operational Characteristics**

- a. **Performance requirements:** The program should have capabilities for a variety of tests, including daily, monthly, and annual tests. All uploaded images and/or data should have the ability to be saved in separate subfolders in reference to the report. The text file needs to be accessible to accommodate the addition of alternative tests for specialized scanners.

- b. **Safety:** N/A
- c. **Accuracy and Reliability:** The software must be reliable in the sense that the program functions as designed during each use. The reports must be generated consistently throughout the use of the program and the program must function without crashes or bugs. The calculations computed by the program must be consistent and accurate. A pop-up window should appear as the calculations are being done for analysis by the user to ensure sanity of the results before compilation in the PDF file.
- d. **Life in Service:** The program will be used indefinitely with the potential for modifications and improvements in the future.
- e. **Shelf Life:** The program should be able to run indefinitely.
- f. **Operating Environment:** The program will mainly be used by radiologist and physicists at the WIMR. However, the software may be shared in the future with other radiologists via forum boards.
- g. **Ergonomics:** The software should have a user-friendly interface that makes sense to the user. All text within the program and the PDF output must be well organized and readable.
- h. **Size:** N/A
- i. **Weight:** N/A
- j. **Materials:**
  - MATLAB
  - LaTeX
  - Sample testing data and reports will be provided
  - CT scanner available
- k. **Aesthetics, Appearance, and Finish:**

The finished software package should have a clean and pleasing interface for the user. The software may be packaged into an executable for users without MATLAB.

## 2. Production Characteristics

- a. **Quantity:** One software program will be created.
- b. **Target Product Cost:** \$0 or cost of MATLAB licensing fees

## 3. Miscellaneous

- a. **Standards and Specifications:** The tests outlined in the exported PDF will outline the testing procedures and the testing results. The goal of this project is to automate the testing report to increase the consistency of the CT quality assurance testing reports in the department.

- b. **Customer:** The customer requests for the code to be well commented and easily modulated so others can easily understand and modify for their own use. Additionally, the user should be able to enter testing data in any order they choose.
- c. **Patient-related concerns:** In order to achieve an accurate CT scan with proper dosing, the CT scan must be well tested prior to use. This program will help analyze CT system testing results and compile them in a report detailing the testing procedures and results. This report will be sent to technicians to fix the CT scanner.
- d. **Competition:** There are two software programs on the market that have many of the design specifications. These programs include ImageOwl and PIPSpro.

Image Owl is a cloud based system, which facilitates the retrieval of data and tracking trends over time, along with other features such as mobile apps [2]. While these features are convenient, they also greatly increase the price. Customization is another source of expense. Given their data analyses are specialized for Catphan® and Tomophan® phantoms, their more comprehensive and customizable testing options are more expensive [2].

PIPSpro, created by Standard Imaging Inc., provides quantitative analysis of scanner performance on a variety of phantoms sold by the same company [3]. Additionally, complexity of the program itself requires training to use properly [3]. As with Image Owl, the program does not lend itself to alterations and testing protocols are not included in the report.

## References

[1] T. Szczykutowicz. "CT Scanner Annual Testing: East Clinic UWHC DHO (GE LS16 Pro)" Department of Radiology, University of Wisconsin-Madison. Jul. 2016.

[2] "Comprehensive QA Services in the Cloud," Image Owl, Inc. [Online]. Available: <http://www.imageowl.com/>. [Accessed: 09-Oct-2016].

[3] "PIPSpro Software," Standard Imaging, Inc.. [Online]. Available: <http://www.standardimaging.com/qa-software/pipspro-software/>. [Accessed: 09-Oct-2016].

## B. Materials

Description	Supplier	Part/Model #	Link to Part	Qty	Date	Price	TOTAL
MATLAB	MathWorks	UW-Madison License	<a href="https://www.mathworks.com/products/matlab/">https://www.mathworks.com/products/matlab/</a>	NA	NA	\$0.00	\$0.00
LaTeX	LaTeX Project	LaTeX Project Public License	<a href="https://www.latex-project.org">https://www.latex-project.org</a>	NA	NA	\$0.00	\$0.00
TOTAL							\$0.00

## C. Semester Schedule

Task	Sept		Oct				Nov				Dec		
	23	30	14	19	21	28	4	11	18	25	2	9	15
<b>Project R&amp;D</b>													
Research	X	X											
Design Alternatives and Matrix	X	X											
Decide Final Design					X								
Design Development						X	X	X	X	X			
<b>Deliverables</b>													
Progress Reports	X	X	X	X	X	X	X		X	X	X	X	X
PDS	X												
Preliminary Presentations			X										
Final Deliverables												X	
<b>Meetings</b>													
Client	X	X		X							X		
Team	X	X	X	X	X	X			X	X	X	X	X
Advisor	X	X	X	X	X	X	X		X	X	X	X	X
<b>Website</b>													
Update	X	X	X	X	X	X	X	X	X	X	X	X	X



## D. Project Schedule & Responsibilities

Below is the project schedule for the last month of the semester.

Date	Priorities
11/18	Client Meeting
11/21-11/23	Fix panels with client feedback
11/30	Have all panels complete
11/30	Team Meeting - 8pm Put all panels into program Start making program look nice
12/2	Team meeting - generate export text file section
12/5	Have program complete Begin Paper Begin Poster
12/5	Have poster complete → send to printing
12/9	Poster Due
12/12	Have paper complete - final edits begin
12/14	Final Deliverables Notebooks and final paper and peer reviews

The GUI was divided into 13 panels. Here is the list of individual contributions to the project.

Panel	Team Member Responsible
Basic Information	Team
Safety	
Artifacts	Sam
Noise	
Monitor	Rachel
Dose	
Protocol Review	

Panel	Team Member Responsible
LCD	Heather
CT Number	
CT Number Uniformity	
Beam Width	
Gantry Tilt	
Slice Width	
LaTeX Code	Connor

## E. Panel Layouts

### Main Menu



### Basic Information

The screenshot shows the 'Basic Information' form within the 'CT\_QA\_Report\_Builder' application. The window title is 'CT\_QA\_Report\_Builder'. At the top left, there is a 'Basic Information' dropdown menu. At the top right, there is an 'Export' button. The form is titled 'Basic Information' and contains the following fields:

- Scanner Name:
- Date of testing:
- Date of report:
- Scanner Location:
- Facility Name:
- Facility Address:
- Facility Contact Name:
- Facility Contact Email:
- Facility Contact Phone:
- Physicist's Name:
- Physicist's Address:
- Physicist's Phone:
- Physicist's Email:

There is also a 'Comments:' section with a large text area for input. At the bottom right of the form, there is a 'Save' button.

## Safety

CT\_QA\_Report\_Builder

Safety

Export

Safety

Do the scanner speakers allow you to hear the patient lying on couch?	N/A
Does the intercom work and have a volume set to allow patients to hear coaching instructions?	N/A
Are the X-ray warning labels present in control room?	N/A
Is the X-ray warning label present on scanner?	N/A
Does the X-ray "on" light work on control panel?	N/A
Does the X-ray "on" light work on scanner?	N/A
Does the X-ray "on" light works outside room?	N/A
Is the scatter information available to staff working with scanner?	N/A
Does the disable couch movement button work?	N/A
When disable couch movement button is pressed, is the couch left in a state movable by hand?	N/A

Comments:

Save

## Artifacts

CT\_QA\_Report\_Builder

Artifacts

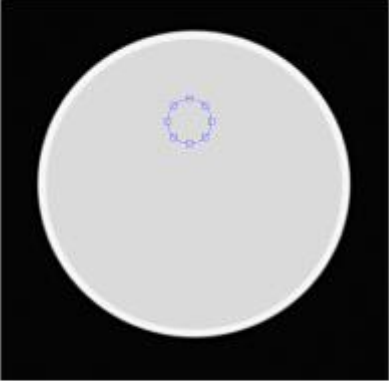
Export

Artifacts

Pitch:	
Scan mode:	
mA:	
Rotation time:	
Effective mAs:	
Denoising level:	
Kernel:	
SFOV:	
RFOV:	
Slice thickness:	

Comments:

Standard deviations: 4.90

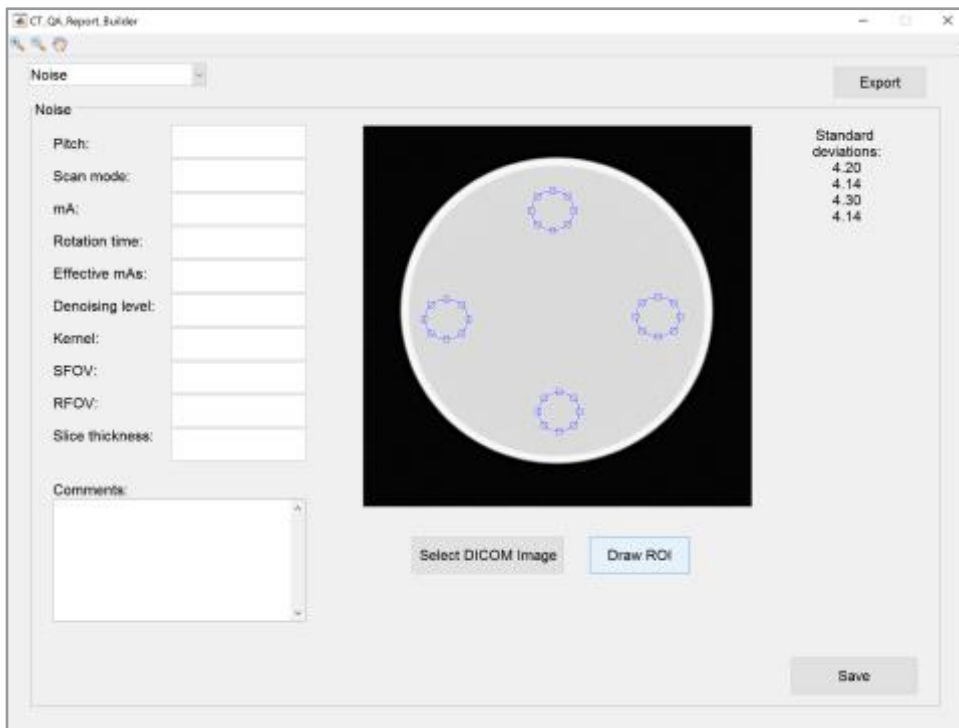


Select DICOM Image Draw ROI

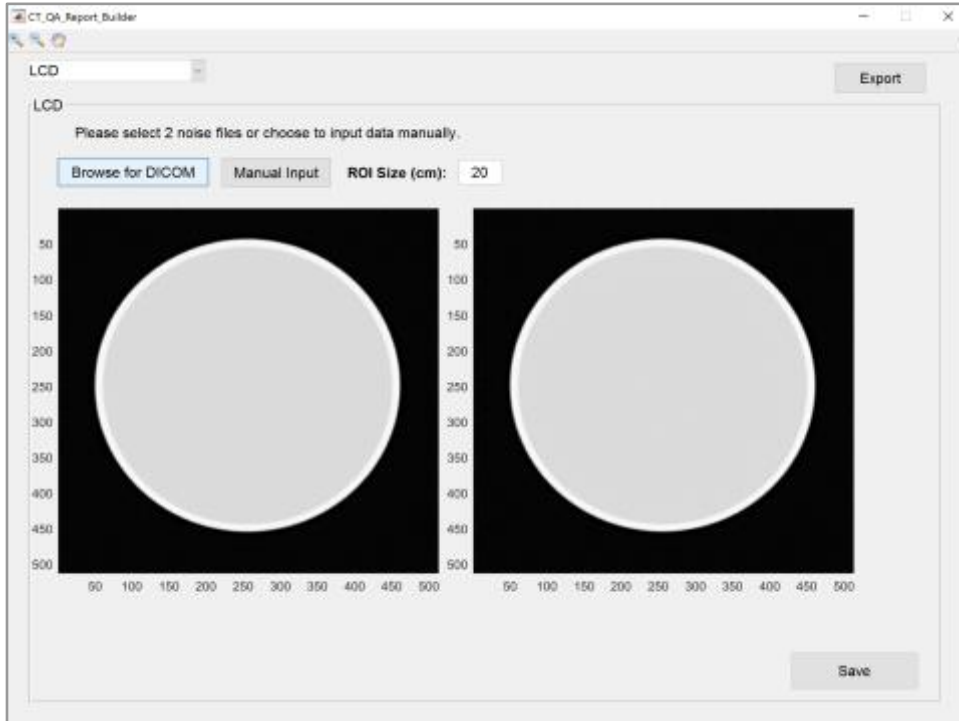
Artifacts Present?

Save

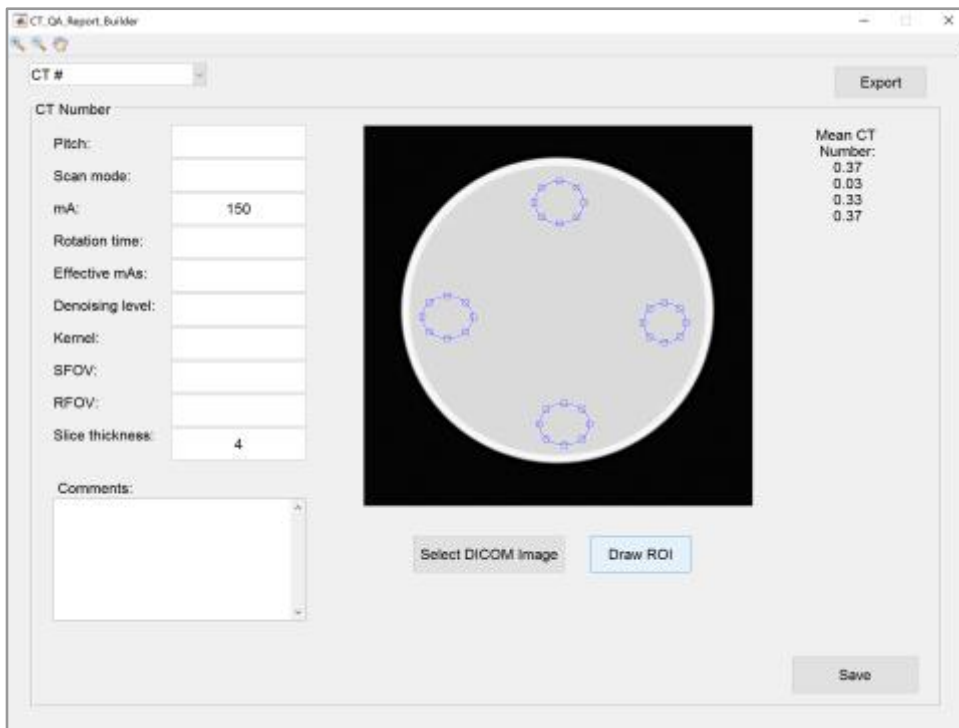
## Noise



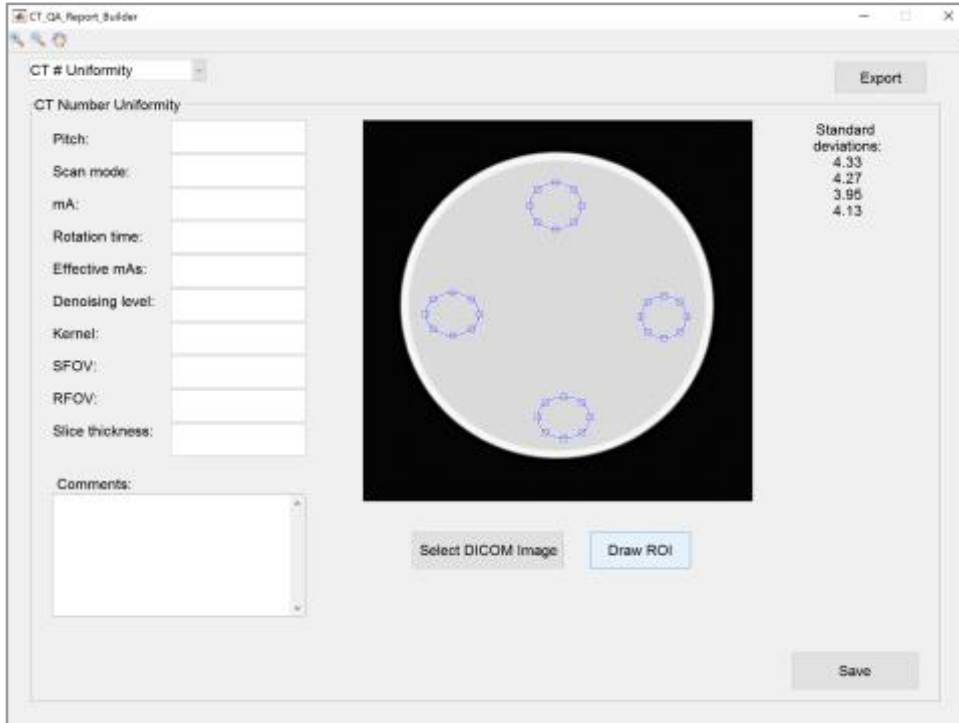
## LCD (Low contrast detectability)



## CT Number



## CT Uniformity



## Monitor

CT\_QA\_Report\_Builder

Monitor Export

Monitor

Test Pattern: SMPTE Pattern Comments

Is 0/5% visible? N/A

Is 95/100% visible? N/A

Are 0-100% steps all distinguishable? N/A

Any artifacts present? N/A

Max Screen Luminance: Center:   
Corner 1:   
Corner 2:   
Corner 3:   
Corner 4:

Min Screen Luminance: Center:   
Corner 1:   
Corner 2:   
Corner 3:   
Corner 4:

Calculate Monitor Uniformity Save

## Beam Width

CT\_QA\_Report\_Builder

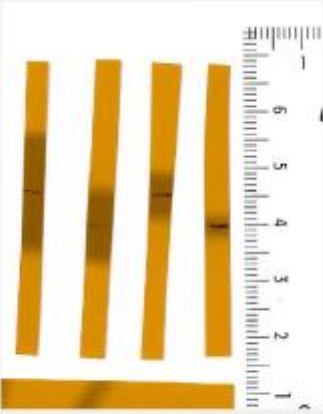
Beam Width Export

Beam Width

For calibration: position the image so that the ruler is up close & visible.

Browse for Image Enter Beam Width Manually

Calibrate for Beam Width



Save

## Protocol Review

CT\_QA\_Report\_Builder

Protocol Review

Export

Protocol Review

Number of Protocols to Review: 6

Protocol	Status	Comments
<input type="text"/>	SELECT STATUS	<input type="text"/>
<input type="text"/>	SELECT STATUS	<input type="text"/>
<input type="text"/>	SELECT STATUS	<input type="text"/>
<input type="text"/>	SELECT STATUS	<input type="text"/>
<input type="text"/>	SELECT STATUS	<input type="text"/>
<input type="text"/>	SELECT STATUS	<input type="text"/>

Save

## Gantry Tilt

CT\_QA\_Report\_Builder

Gantry Tilt


Export

Gantry Tilt

Please draw a line over the top of the film for the 0 degree mark. Right click to end line.

Browse for image    Enter Gantry Tilt Manually

Calculate Gantry Tilt



?    Save

## Slice Width

The screenshot shows the 'CT\_QA\_Report\_Builder' window with the 'Slice Width' section selected in the top-left dropdown menu. The main content area is titled 'Slice Width' and contains a 'Browse for DICOM' button on the left and a 'Slice Width: 4' input field on the right. An 'Export' button is located in the top-right corner, and a 'Save' button is in the bottom-right corner.

## Dose

The screenshot shows the 'CT\_QA\_Report\_Builder' window with the 'Dose' section selected in the top-left dropdown menu. The main content area is titled 'Dose' and contains several input fields and buttons. At the top, there is a 'Calculate CTDIw' dropdown menu. Below it is a 'Beam Width:' input field. The 'Center Measure:' and 'Surface Measure:' sections each have three stacked input fields and an 'Average' button. Below these are 'CTDI100 Center' and 'CTDI100 Surface' buttons. At the bottom, there is a 'Calculate CTDIw' button, a 'Displayed on Scanner:' input field, and a '% Difference' button. An 'Export' button is in the top-right corner, and a 'Save' button is in the bottom-right corner.



## F. Source Code

*Note: only the code that performs calculations/functions is displayed below*

- Loading Logo %%
- Basic Information Panel
- Safety Panel
- Artifacts Panel
- Fill in parameters
- Loop code for creating ROIs
- LCD Panel
- CT Number Panel
- Loop code for creating ROIs
- CT Number Uniformity Panel
- Loop code for creating ROIs
- Noise Panel
- Loop code for creating ROIs
- Monitor Panel
- Dose Panel
- hide fields to calculate CTDIw %%%%%%%%%%
- hide option to enter CTDIw %%%%%%%%%%
- Slice Width Panel
- Beam Width Panel
- Receive user input for line & intensity profile
- Full-width half max
- Gantry Tilt Panel
- Get +30 angle
- Get +15 angle
- Get -15 angle
- Get -30 angle
- Protocol Review Panel
- Basic Information
- Safety
- Artifacts
- LCD
- Noise
- CT Number
- CT Uniformity
- Slice Width

```
function varargout = CT_QA_Report_Builder(varargin)
% CT_QA_REPORT_BUILDER MATLAB code for CT_QA_Report_Builder.fig
% CT_QA_REPORT_BUILDER, by itself, creates a new CT_QA_REPORT_BUILDER or raises the existing
% singleton*.
%
% H = CT_QA_REPORT_BUILDER returns the handle to a new CT_QA_REPORT_BUILDER or the handle to
% the existing singleton*.
%
% CT_QA_REPORT_BUILDER('CALLBACK', hObject,eventData,handles,...) calls the local
% function named CALLBACK in CT_QA_REPORT_BUILDER.M with the given input arguments.
%
% CT_QA_REPORT_BUILDER('Property','Value',...) creates a new CT_QA_REPORT_BUILDER or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before CT_QA_Report_Builder_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to CT_QA_Report_Builder_OpeningFcn via varargin.
```

```

%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help CT_QA_Report_Builder

% Last Modified by GUIDE v2.5 09-Dec-2016 00:12:51

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @CT_QA_Report_Builder_OpeningFcn, ...
                  'gui_OutputFcn',  @CT_QA_Report_Builder_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before CT_QA_Report_Builder is made visible.
function CT_QA_Report_Builder_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

Loading Logo %%
axes(handles.axes_logo);
imshow('CTQALogo.png');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes CT_QA_Report_Builder wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

```



```
function varargout = CT_QA_Report_Builder_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on selection change in main_dropbar.
```

```
function main_dropbar_Callback(hObject, eventdata, handles)
val = get(handles.main_dropbar, 'value');
switch(val)
```

```
case 1 % Do nothing, "Select test" option
    set(handles.logoPanel, 'visible', 'on');
    set(handles.basicInformationPanel, 'visible', 'off');
    set(handles.safetyPanel, 'visible', 'off');
    set(handles.artifactsPanel, 'visible', 'off');
    set(handles.LCDPanel, 'visible', 'off');
    set(handles.CTNumPanel, 'visible', 'off');
    set(handles.CTUniformityPanel, 'visible', 'off');
    set(handles.noisePanel, 'visible', 'off');
    set(handles.monitorPanel, 'visible', 'off');
    set(handles.sliceWidthPanel, 'visible', 'off');
    set(handles.dosePanel, 'visible', 'off');
    set(handles.gantryPanel, 'visible', 'off');
    set(handles.protocolPanel, 'visible', 'off');
    set(handles.beamWidthPanel, 'visible', 'off');
case 2 % Display general info tab, hide rest
    set(handles.logoPanel, 'visible', 'off');
```

```

set(handles.basicInformationPanel,'visible','on');
set(handles.safetyPanel,'visible','off');
set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','off');
set(handles.CTNumPanel,'visible','off');
set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','off');
case 3 % Display safety tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','on');
set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','off');
set(handles.CTNumPanel,'visible','off');
set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','off');
case 4 % Display artifacts tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','off');
set(handles.artifactsPanel,'visible','on');
set(handles.LCDPanel,'visible','off');
set(handles.CTNumPanel,'visible','off');
set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','off');
case 5 % Display noise tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','off');
set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','off');
set(handles.CTNumPanel,'visible','off');
set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','on');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','off');
case 6 % Display LCD tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','off');

```

```

set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','on');
set(handles.CTNumPanel,'visible','off');
set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','off');
case 7 % Display CT # tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','off');
set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','off');
set(handles.CTNumPanel,'visible','on');
set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','off');
case 8 % Display CT # uniformity tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','off');
set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','off');
set(handles.CTNumPanel,'visible','off');
set(handles.CTUniformityPanel,'visible','on');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','off');
case 9 % Display slice width tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','off');
set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','off');
set(handles.CTNumPanel,'visible','off');
set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','on');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','off');
case 10 % Display gantry tilt tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','off');
set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','off');

```

```

set(handles.CTNumPanel,'visible','off');
set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','on');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','off');
case 11 % Display protocol review tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','off');
set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','off');
set(handles.CTNumPanel,'visible','off');
set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','on');
set(handles.beamWidthPanel,'visible','off');

case 12 % Display beam width tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','off');
set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','off');
set(handles.CTNumPanel,'visible','off');
set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','on');
case 13 % Display monitor tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','off');
set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','off');
set(handles.CTNumPanel,'visible','off');
set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','on');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','off');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','off');
case 14 % Display dose tab, hide rest
set(handles.logoPanel,'visible','off');
set(handles.basicInformationPanel,'visible','off');
set(handles.safetyPanel,'visible','off');
set(handles.artifactsPanel,'visible','off');
set(handles.LCDPanel,'visible','off');
set(handles.CTNumPanel,'visible','off');

```

```

set(handles.CTUniformityPanel,'visible','off');
set(handles.noisePanel,'visible','off');
set(handles.monitorPanel,'visible','off');
set(handles.sliceWidthPanel,'visible','off');
set(handles.dosePanel,'visible','on');
set(handles.gantryPanel,'visible','off');
set(handles.protocolPanel,'visible','off');
set(handles.beamWidthPanel,'visible','off');
end

```

## Artifacts Panel

```

% --- Executes on button press in artifactsImageSelect.
function artifactsImageSelect_Callback(hObject, eventdata, handles)

```

```

%ask user to select DICOM image
[ImgName,ImgPath] = uigetfile(*.dicom');
%convert into full file name
ImgFullFile = fullfile(ImgPath,ImgName);
%make sure user selected file and updates GUI with Image selected
if isempty(ImgName)==0;
    info = dicominfo([ImgPath ImgName]);
    handles.pix = double(dicomread([ImgPath ImgName]));
end

```

Fill in parameters

```

setappdata(gcf,'ImgWidth',info.Width);
setappdata(gcf,'ImgHeight',info.Height);
setappdata(gcf,'PixSpace',info.PixelSpacing);

```

```

set(handles.axes_artifacts,'visible','on');
set(handles.artifactsDrawROI,'visible','on');
set(handles.artifactsPresentText,'visible','on');
set(handles.artifactsYN,'visible','on');
axes(handles.axes_artifacts)
imshow(handles.pix, []);

```

```

guidata(hObject,handles);

```

```

% --- Executes on button press in artifactsDrawROI.
function artifactsDrawROI_Callback(hObject, eventdata, handles)

```

```

%get image info loaded from previous function
ImgSave=getappdata(gca,'ImgLoad');
%Select region of interest on image, MUST right-click on ROI and press
%"create mask"

```

```

waitfor(msgbox('Drag mouse to draw an ROI.'));
string{1,1} = 'Standard deviations: ';
set(handles.artifactsStdevText, 'string',string);

```

Loop code for creating ROIs  
Clear variables

```
circle = [];  
roi = [];  
positions = [];  
pixels = [];  
  
circle = imellipse; % allows user to draw circle  
roi = createMask(circle); % matrix with 0s and 1s (1s are where user selected)  
k = 1;  
for j = 1:length(roi)  
    v = find(roi(:,j) == 1);  
    if isempty(v) == 0 %if vector is not empty  
        l = length(v);  
        positions(k:k+l-1,1) = v;  
        positions(k:k+l-1,2) = j;  
        k = k+l;  
    end  
end  
% Get actual pixel values  
for m = 1:length(positions)  
    pixels(m) = handles.pix(positions(m,1),positions(m,2));  
end  
%handles.roi(1:length(pixels),i) = pixels;  
handles.artifactsStdev = std(pixels);  
string{2,1} = num2str(handles.artifactsStdev,'%2f');  
set(handles.artifactsStdevText, 'string', string);  
  
waitfor(msgbox('Calculations complete.'));  
ImgWidth= getappdata(gcf,'ImgWidth');  
ImgHeight= getappdata(gcf,'ImgHeight');  
PixSpace = getappdata(gcf,'PixSpace');  
ImgXCenter= ImgWidth/2;  
ImgYCenter= ImgHeight/2;  
ROly= round(mean(positions(:,1)));  
ROlx = round(mean(positions(:,2)));  
yDiff = abs(ImgYCenter-ROly);  
xDiff = abs(ImgXCenter-ROlx);  
yCal = double(PixSpace(2)*yDiff);  
xCal = double(PixSpace(1)*xDiff);  
  
IsoDist = sqrt((yCal^2)+(xCal^2));  
  
guidata(hObject,handles);
```

### LCD Panel

```
% --- Executes on button press in LCDBrowse.  
function LCDBrowse_Callback(hObject, eventdata, handles)  
[filename pathname] = uigetfile('*.dicom','Select 2 noise files.', 'multiselect','on');  
  
if iscell(filename) == 0  
    h1 = waitfor(msgbox('Please select 2 files.'));  
    set(h1,'Deletefcn',@closeMsg);  
    [filename pathname] = uigetfile('*.dicom','Select 2 noise files.', 'multiselect','on');  
end  
% Load image information  
if iscell(pathname) == 0  
    info = dicominfo([pathname filename{1}]);  
    image1 = double(dicomread([pathname filename{1}]));  
    image2 = double(dicomread([pathname filename{2}]));
```



```

else
    info = dicominfo([pathname{1} filename{1}]);
    image1 = double(dicomread([pathname{1} filename{1}]));
    image2 = double(dicomread([pathname{2} filename{2}]));
end

% Show images
set(handles.axes_LCD1,'visible','on');
set(handles.axes_LCD2,'visible','on');
set(handles.axes_LCD1,'xtick',[], 'ytick',[])
set(handles.axes_LCD2,'xtick',[], 'ytick',[])
imagesc(handles.axes_LCD1, image1);
colormap('gray');
imagesc(handles.axes_LCD2, image2);

% Transformation from HU to mu
image1 = (image1+info.RescaleIntercept);
image2 = (image2+info.RescaleIntercept);

diffIm = image1 - image2;

%crop down to the middle part of the image
iminCropp = diffIm(128:389,128:389);
figure; imagesc(iminCropp);colormap('gray');

countt = 1;
for i=2:10 %object diameter in mm
    iminCroppedBlurr = imfilter(iminCropp,fspecial('average',[round(i/info.PixelSpacing(1))
round(i/info.PixelSpacing(1))]),'same');
    plotter(countt) = (1*3.29*std2(iminCroppedBlurr(30:end-30,30:end-30)))/sqrt(2);
    countt = countt +1;
end
figure;plot(2:10,plotter);
grid on;
ylabel('% Contrast required for detection');
xlabel('Object Diameter (mm)');
title('Low Contrast Detectability (SNR = 3.29)','FontSize',20);
guidata(hObject,handles);

CT Number Panel
% --- Executes on button press in CTNumImageSelect.
function CTNumImageSelect_Callback(hObject, eventdata, handles)
[filename pathname] = uigetfile('*.*dicom','Please select DICOM file.');
```

```

set(handles.CTNumDrawROI,'visible','on');
set(handles.axes_CTNum,'visible','on');
set(handles.CTNum_meanCT, 'string', "");
info = dicominfo([pathname filename]);
%handles.pix = double(dicomread([pathname filename]));
handles.pix = double(dicomread([pathname filename])) + info.RescaleIntercept ;
% Fill in parameters
set(handles.CTNumMA, 'string',info.XrayTubeCurrent);
set(handles.CTNumSlice,'string',info.SliceThickness);
axes(handles.axes_CTNum)
imshow(handles.pix, []);
guidata(hObject,handles);

% --- Executes on button press in CTNumDrawROI.
function CTNumDrawROI_Callback(hObject, eventdata, handles)

```

```

% Ask number of inserts so that user can draw that number of ROIs
num = inputdlg('Number of ROIs:');
num = str2double(num{1});
waitfor(msgbox('Drag mouse to draw an ROI.'));
string{1,1} = 'Mean CT Number: ';
set(handles.CTNum_meanCT, 'string', string);

Loop code for creating ROIs
for i = 1:num
    % Clear variables
    circle = [];
    roi = [];
    positions = [];
    pixels = [];

    circle = imellipse; % allows user to draw circle
    roi = createMask(circle); % matrix with 0s and 1s (1s are where user selected)
    k = 1;
    for j = 1:length(roi)
        v = find(roi(:,j) == 1);
        if isempty(v) == 0 %if vector is not empty
            l = length(v);
            positions(k:k+l-1,1) = v;
            positions(k:k+l-1,2) = j;
            k = k+l;
        end
    end
    % Get actual pixel values
    for m = 1:length(positions)
        pixels(m) = handles.pix(positions(m,1),positions(m,2));
    end

    %handles.CTNum_roi(1:length(pixels),i) = pixels;
    handles.CTNum_mean(i,1) = mean(pixels);
    string{i+1,1} = num2str(handles.CTNum_mean(i), '%.2f');
    set(handles.CTNum_meanCT, 'string', string);

end
waitfor(msgbox('Calculations complete.'));
% handles.stdev are the standard deviations
guidata(hObject,handles);

```

### CT Number Uniformity Panel

```

% --- Executes on button press in CTUniformityImageSelect.
function CTUniformityImageSelect_Callback(hObject, eventdata, handles)
[filename pathname] = uigetfile('*.dicom','Please select DICOM file. ');
set(handles.CTUniformity_stdev, 'string', '');
set(handles.axes_CTUniformity, 'visible', 'on');
set(handles.CTUniformityDrawROI, 'visible', 'on');
info = dicominfo([pathname filename]);
handles.pix = double(dicomread([pathname filename])) + info.RescaleIntercept ;
% Fill in parameters
axes(handles.axes_CTUniformity)
imshow(handles.pix, []);
guidata(hObject,handles);

```

```
% --- Executes on button press in CTUniformityDrawROI.
function CTUniformityDrawROI_Callback(hObject, eventdata, handles)
```

```
% Ask number of inserts so that user can draw that number of ROIs
num = inputdlg('Number of inserts:');
num = str2double(num{1});
waitfor(msgbox('Drag mouse to draw an ROI.'));
string{1,1} = 'Standard deviations: ';
set(handles.CTUniformity_stdev, 'string',string);
```

Loop code for creating ROIs

```
for i = 1:num
    % Clear variables
    circle = [];
    roi = [];
    positions = [];
    pixels = [];

    circle = imellipse; % allows user to draw circle
    roi = createMask(circle); % matrix with 0s and 1s (1s are where user selected)
    k = 1;
    for j = 1:length(roi)
        v = find(roi(:,j) == 1);
        if isempty(v) == 0 %if vector is not empty
            l = length(v);
            positions(k:k+l-1,1) = v;
            positions(k:k+l-1,2) = j;
            k = k+l;
        end
    end
    % Get actual pixel values
    for m = 1:length(positions)
        pixels(m) = handles.pix(positions(m,1),positions(m,2));
    end
    handles.roi(1:length(pixels),i) = pixels;
    handles.CTUniformity_standDeviation(i,1) = std(pixels);
    string{i+1,1} = num2str(handles.CTUniformity_standDeviation(i), '%.2f');
    set(handles.CTUniformity_stdev, 'string', string);
end
waitfor(msgbox('Calculations complete.'));
% handles.stdev are the standard deviations
guidata(hObject,handles);
```

### Noise Panel

```
% --- Executes on button press in NoiseImageSelect.
function NoiseImageSelect_Callback(hObject, eventdata, handles)
[filename pathname] = uigetfile('*.dicom','Please select DICOM file. ');
set(handles.Noise_stdev, 'string','');
set(handles.axes_Noise,'visible','on');
set(handles.NoiseDrawROI,'visible','on');
info = dicominfo([pathname filename]);
handles.pix = double(dicomread([pathname filename])) + info.RescaleIntercept ;
% Fill in parameters
axes(handles.axes_Noise)
imshow(handles.pix, []);
guidata(hObject,handles);
```

```
% --- Executes on button press in NoiseDrawROI.
function NoiseDrawROI_Callback(hObject, eventdata, handles)
```

```
% Ask number of inserts so that user can draw that number of ROIs
num = inputdlg('Number of ROIs:');
num = str2double(num{1});
waitfor(msgbox('Drag mouse to draw an ROI.'));
string{1,1} = 'Standard deviations:';
set(handles.Noise_stdev, 'string',string);
```

Loop code for creating ROIs

```
for i = 1:num
    % Clear variables
    circle = [];
    roi = [];
    positions = [];
    pixels = [];

    circle = imellipse; % allows user to draw circle
    roi = createMask(circle); % matrix with 0s and 1s (1s are where user selected)
    k = 1;
    for j = 1:length(roi)
        v = find(roi(:,j) == 1);
        if isempty(v) == 0 %if vector is not empty
            l = length(v);
            positions(k:k+l-1,1) = v;
            positions(k:k+l-1,2) = j;
            k = k+l;
        end
    end
    % Get actual pixel values
    for m = 1:length(positions)
        pixels(m) = handles.pix(positions(m,1),positions(m,2));
    end
    handles.roi(1:length(pixels),i) = pixels;
    handles.Noise_standDeviation(i,1) = std(pixels);
    string{i+1,1} = num2str(handles.Noise_standDeviation(i),'%.2f');
    set(handles.Noise_stdev, 'string', string);
end
waitfor(msgbox('Calculations complete.'));
% handles.stdev are the standard deviations
guidata(hObject,handles);
```

### Monitor Panel

```
% --- Executes on selection change in monitorTestPatternMenu.
function monitorTestPatternMenu_Callback(hObject, eventdata, handles)
contents = cellstr(get(hObject,'String'));
testPattern = contents(get(hObject,'Value'));

% allow the user to enter test pattern if 'Other' is selected from dropdown
if strcmp(testPattern,'Other') == 1;
    set(handles.monitorOtherTestPattern, 'visible', 'on');
    set(handles.monitorOtherTestPatternText, 'visible', 'on');
else
    set(handles.monitorOtherTestPattern, 'visible', 'off');
    set(handles.monitorOtherTestPatternText, 'visible', 'off');
end
```

```
guidata(hObject,handles);
```

```
% --- Executes on button press in monitorUniformity.
```

```
function monitorUniformity_Callback(hObject, eventdata, handles)
```

```
% convert the max luminance enteries to doubles
```

```
maxCenter = str2double(get(handles.monitorQ5_center,'String'));
```

```
maxCorner1 = str2double(get(handles.monitorQ5_corner1,'String'));
```

```
maxCorner2 = str2double(get(handles.monitorQ5_corner2,'String'));
```

```
maxCorner3 = str2double(get(handles.monitorQ5_corner3,'String'));
```

```
maxCorner4 = str2double(get(handles.monitorQ5_corner4,'String'));
```

```
% convert the min luminance enteries to doubles
```

```
minCenter = str2double(get(handles.monitorQ6_center,'String'));
```

```
minCorner1 = str2double(get(handles.monitorQ6_corner1,'String'));
```

```
minCorner2 = str2double(get(handles.monitorQ6_corner2,'String'));
```

```
minCorner3 = str2double(get(handles.monitorQ6_corner3,'String'));
```

```
minCorner4 = str2double(get(handles.monitorQ6_corner4,'String'));
```

```
% combine the values into vectors for max amd min luminance, respectively
```

```
maxLum = [maxCenter maxCorner1 maxCorner2 maxCorner3 maxCorner4];
```

```
minLum = [minCenter minCorner1 minCorner2 minCorner3 minCorner4];
```

```
% calculate the uniformity percent difference
```

```
handles.monitorUniformPercentDiff = 200 * (max(maxLum) - min(minLum)) / (max(maxLum) + min(minLum));
```

```
% display the uniformity percent difference to 2 decimal places
```

```
set(handles.monitorUniformityText, 'String', ['% difference = ', num2str(handles.monitorUniformPercentDiff, '%0.2f'), '%']);
```

```
guidata(hObject, handles);
```

## Dose Panel

```
% --- Executes on selection change in dose_ctdiwOption.
```

```
function dose_ctdiwOption_Callback(hObject, eventdata, handles)
```

```
contents = cellstr(get(hObject,'String'));
```

```
ctdiOption = contents{get(hObject,'Value')};
```

```
% selection determines which fields are visible
```

```
if strcmp(ctdiOption,'Enter CTDIw') == 1;
```

```
hide fields to calculate CTDIw
```

```
set(handles.doseWidthText, 'Visible', 'Off');
```

```
set(handles.doseWidth, 'Visible', 'Off');
```

```
set(handles.doseCenterMeasureText, 'Visible', 'Off');
```

```
set(handles.doseCenterMeasure1, 'Visible', 'Off');
```

```
set(handles.doseCenterMeasure2, 'Visible', 'Off');
```

```
set(handles.doseCenterMeasure3, 'Visible', 'Off');
```

```
set(handles.doseAvgCenterButton, 'Visible', 'Off');
```

```
set(handles.doseAvgCenter, 'Visible', 'Off');
```

```
set(handles.dose_ctdi100CenterButton, 'Visible', 'Off');
```

```
set(handles.dose_ctdi100Center, 'Visible', 'Off');
```

```
set(handles.doseSurfaceMeasureText, 'Visible', 'Off');
```

```
set(handles.doseSurfaceMeasure1, 'Visible', 'Off');
```

```
set(handles.doseSurfaceMeasure2, 'Visible', 'Off');
```

```
set(handles.doseSurfaceMeasure3, 'Visible', 'Off');
```

```
set(handles.doseAvgSurfaceButton, 'Visible', 'Off');
```

```
set(handles.doseAvgSurface, 'Visible', 'Off');
```

```
set(handles.dose_ctdi100SurfaceButton, 'Visible', 'Off');
```

```
set(handles.dose_ctdi100Surface, 'Visible', 'Off');
```

```

set(handles.dose_ctdiw_calculated, 'Visible' , 'Off');
set(handles.dose_ctdiwButton, 'Visible' , 'Off');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% allow user to ENTER value for ctdiw
set(handles.dose_ctdiw_enteredText, 'Visible' , 'On');
set(handles.dose_ctdiw_entered, 'Visible' , 'On');

set(handles.doseDisplayedOnScannerText, 'Visible' , 'On');
set(handles.doseDisplayedOnScanner, 'Visible' , 'On');

set(handles.dosePercentDiffButton, 'Visible' , 'On');
set(handles.dose_ctdiPercentDiff, 'Visible' , 'On');

elseif strcmp(ctdiOption, 'Calculate CTDIw') == 1;

% make fields available for user to CALCULATE ctdiw
set(handles.doseWidthText, 'Visible' , 'On');
set(handles.doseWidth, 'Visible' , 'On');

set(handles.doseCenterMeasureText, 'Visible' , 'On');
set(handles.doseCenterMeasure1, 'Visible' , 'On');
set(handles.doseCenterMeasure2, 'Visible' , 'On');
set(handles.doseCenterMeasure3, 'Visible' , 'On');
set(handles.doseAvgCenterButton, 'Visible' , 'On');
set(handles.doseAvgCenter, 'Visible' , 'On');
set(handles.dose_ctdi100CenterButton, 'Visible' , 'On');
set(handles.dose_ctdi100Center, 'Visible' , 'On');

set(handles.doseSurfaceMeasureText, 'Visible' , 'On');
set(handles.doseSurfaceMeasure1, 'Visible' , 'On');
set(handles.doseSurfaceMeasure2, 'Visible' , 'On');
set(handles.doseSurfaceMeasure3, 'Visible' , 'On');
set(handles.doseAvgSurfaceButton, 'Visible' , 'On');
set(handles.doseAvgSurface, 'Visible' , 'On');
set(handles.dose_ctdi100SurfaceButton, 'Visible' , 'On');
set(handles.dose_ctdi100Surface, 'Visible' , 'On');

set(handles.dose_ctdiw_calculated, 'Visible' , 'On');
set(handles.dose_ctdiwButton, 'Visible' , 'On');

hide option to enter CTDIw
set(handles.dose_ctdiw_enteredText, 'Visible' , 'Off');
set(handles.dose_ctdiw_entered, 'Visible' , 'Off');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

set(handles.doseDisplayedOnScannerText, 'Visible' , 'On');
set(handles.doseDisplayedOnScanner, 'Visible' , 'On');

set(handles.dosePercentDiffButton, 'Visible' , 'On');
set(handles.dose_ctdiPercentDiff, 'Visible' , 'On');
else
% all fields are invisible if neither option is selected
set(handles.doseWidthText, 'Visible' , 'Off');
set(handles.doseWidth, 'Visible' , 'Off');

set(handles.doseCenterMeasureText, 'Visible' , 'Off');
set(handles.doseCenterMeasure1, 'Visible' , 'Off');
set(handles.doseCenterMeasure2, 'Visible' , 'Off');

```

```

set(handles.doseCenterMeasure3, 'Visible', 'Off');
set(handles.doseAvgCenterButton, 'Visible', 'Off');
set(handles.doseAvgCenter, 'Visible', 'Off');
set(handles.dose_ctdi100CenterButton, 'Visible', 'Off');
set(handles.dose_ctdi100Center, 'Visible', 'Off');

set(handles.doseSurfaceMeasureText, 'Visible', 'Off');
set(handles.doseSurfaceMeasure1, 'Visible', 'Off');
set(handles.doseSurfaceMeasure2, 'Visible', 'Off');
set(handles.doseSurfaceMeasure3, 'Visible', 'Off');
set(handles.doseAvgSurfaceButton, 'Visible', 'Off');
set(handles.doseAvgSurface, 'Visible', 'Off');
set(handles.dose_ctdi100SurfaceButton, 'Visible', 'Off');
set(handles.dose_ctdi100Surface, 'Visible', 'Off');

set(handles.dose_ctdiw_calculated, 'Visible', 'Off');
set(handles.dose_ctdiwButton, 'Visible', 'Off');

set(handles.dose_ctdiw_enteredText, 'Visible', 'Off');
set(handles.dose_ctdiw_entered, 'Visible', 'Off');

set(handles.doseDisplayedOnScannerText, 'Visible', 'Off');
set(handles.doseDisplayedOnScanner, 'Visible', 'Off');

set(handles.dosePercentDiffButton, 'Visible', 'Off');
set(handles.dose_ctdiPercentDiff, 'Visible', 'Off');
end
guidata(hObject, handles);

% --- Executes on button press in doseAvgCenterButton.
function doseAvgCenterButton_Callback(hObject, eventdata, handles)
% convert each center measurement to a double
center1 = str2double(get(handles.doseCenterMeasure1, 'String'));
center2 = str2double(get(handles.doseCenterMeasure2, 'String'));
center3 = str2double(get(handles.doseCenterMeasure3, 'String'));
% average the center measurement values
avgCenter = mean([ center1 center2 center3 ]);
% display the average value
set(handles.doseAvgCenter, 'String', num2str(avgCenter, '%0.2f' ))
guidata(hObject, handles);
% --- Executes on button press in doseAvgSurfaceButton.
function doseAvgSurfaceButton_Callback(hObject, eventdata, handles)
surface1 = str2double(get(handles.doseSurfaceMeasure1, 'String'));
surface2 = str2double(get(handles.doseSurfaceMeasure2, 'String'));
surface3 = str2double(get(handles.doseSurfaceMeasure3, 'String'));
% average the surface measurement values
avgSurface = mean([ surface1 surface2 surface3 ]);
% display the average value
set(handles.doseAvgSurface, 'String', num2str(avgSurface, '%0.2f' ))
guidata(hObject, handles);
% --- Executes on button press in dose_ctdi100CenterButton.
function dose_ctdi100CenterButton_Callback(hObject, eventdata, handles)
% access the beam width and average center measurement
beamWidth = str2double(get(handles.doseWidth, 'String'));
avgCenter = str2double(get(handles.doseAvgCenter, 'String'));
% calculate the CTDI100 center value
ctdiCenter = avgCenter*100/beamWidth;
% display the CTDI100 center value
set(handles.dose_ctdi100Center, 'String', num2str(ctdiCenter, '%0.2f' ))
guidata(hObject, handles);

% --- Executes on button press in dose_ctdi100SurfaceButton.

```

```

function dose_ctdi100SurfaceButton_Callback(hObject, eventdata, handles)
% access the beam width and average surface measurement
beamWidth = str2double(get(handles.doseWidth,'String'));
avgSurface = str2double(get(handles.doseAvgSurface,'String'));
% calculate the CTDI100 surface value
ctdiSurface = avgSurface*100/beamWidth;
% display the CTDI100 surface value
set(handles.dose_ctdi100Surface, 'String', num2str(ctdiSurface, '%0.2f' ))
guidata(hObject, handles);

```

% --- Executes on button press in dose\_ctdiwButton.

```

function dose_ctdiwButton_Callback(hObject, eventdata, handles)
% access both CTDI100 values
surface = str2double(get(handles.dose_ctdi100Surface,'String'));
center = str2double(get(handles.dose_ctdi100Center,'String'));
% calculate the CTDIw value
ctdiw = (2/3)*surface + (1/3)*center;
% display the CTDIw value
set(handles.dose_ctdiw_calculated, 'String', num2str(ctdiw, '%0.2f' ))
guidata(hObject, handles);

```

% --- Executes on button press in dosePercentDiffButton.

```

function dosePercentDiffButton_Callback(hObject, eventdata, handles)
% access the correct CTDIw value based on option selected
if strcmp(get(handles.dose_ctdiw_calculated, 'visible'), 'on') == 1
% use the CALCULATED CTDIw value if 'Calculate CTDIw' is selected
ctdiw = str2double(get(handles.dose_ctdiw_calculated,'String'));
else
% use the ENTERED CTDIw value if 'Enter CTDIw' is selected
ctdiw = str2double(get(handles.dose_ctdiw_entered,'String'));
end
% access the value displayed on scanner
display = str2double(get(handles.doseDisplayedOnScanner,'String'));
% calculate the percent difference
percentDiff = (ctdiw - display)/display*100;
% display the percent difference
set(handles.dose_ctdiPercentDiff, 'String', [num2str(percentDiff, '%0.2f' ), '%'])
guidata(hObject, handles);

```

% --- Executes on button press in doseSave.

```

function doseSave_Callback(hObject, eventdata, handles)

```

### Slice Width Panel

% --- Executes on button press in sliceWidthBrowse.

```

function sliceWidthBrowse_Callback(hObject, eventdata, handles)
%browse for file
[filename pathname] = uigetfile(*.dicom','Please select DICOM file. ');
%load dicom info
info = dicominfo([pathname filename]);
data = dicomread(info);

```

% If slice width = slice thickness

```

handles.sliceWidth_width=info.SliceThickness;

```

```

set(handles.sliceWidthInput, 'string', handles.sliceWidth_width);
guidata(hObject,handles);

```

% --- Executes on button press in sliceWidthSave.

```

function sliceWidthSave_Callback(hObject, eventdata, handles)

```



```
handles.sliceWidth_width = get(handles.sliceWidthInput, 'string');
guidata(hObject,handles);
```

### Beam Width Panel

```
% --- Executes on button press in beamWidthBrowse.
```

```
function beamWidthBrowse_Callback(hObject, eventdata, handles)
[filename pathname ] = uigetfile('*.png','Please select an image file for beam width'); %get image
set(handles.axes_beamWidth,'visible','on');
axes(handles.axes_beamWidth);
handles.beamWidth_image = [pathname filename];
I = imread(handles.beamWidth_image);
imshow(I, 'InitialMagnification','fit'); %display image on axis
set(handles.beamWidthTextBox,'visible','on'); %Makes instruction text visible
set(handles.beamWidthCalibrate,'visible','on'); %makes calibrate button visible
guidata(hObject,handles);
```

```
% --- Executes on button press in beamWidthCalibrate.
```

```
function beamWidthCalibrate_Callback(hObject, eventdata, handles)
zoom reset
set(handles.beamWidthTextBox,'string','Please draw a line from two points of a known distance. ');
waitfor(msgbox('Draw a line from the first point and then right-click to finish the line at the second point. '));
[x,y] = getline(handles.axes_beamWidth); %get user input from two points
pixels = sqrt((x(2)-x(1))^2 + ((y(2)-y(1))^2)); %calculate distance between two points
known_distance = cell2mat(inputdlg('Please enter the known distance (cm).'));
handles.beamWidth_calibrationFactor = str2double(known_distance) * 10 / pixels; %get calibration factor
set(handles.beamWidthTextBox,'string','Calibration factor calculated. Please reposition image and then select Calculate Beam Width');
set(handles.beamWidthCalibrate,'string','Redo Calibration');
% Make next options visible
set(handles.beamWidthCalculate, 'visible','on');
```

```
% replot unzoomed image
```

```
axes(handles.axes_beamWidth);
image = imread(handles.beamWidth_image);
imshow(image, 'InitialMagnification','fit'); %display image on axis
%%%% handles.beam_calibrate is calibration factor %%%%
guidata(hObject, handles);
```

```
% --- Executes on button press in beamwidthCalculate.
```

```
function beamWidthCalculate_Callback(hObject, eventdata, handles)
zoom reset
set(handles.beamWidthText,'string','');
set(handles.beamWidthTextBox,'string','');
num_widths = cell2mat(inputdlg('Please enter how many widths you will be calculating. '));
string{1,1} = 'Beam widths (mm)';
set(handles.beamWidthText,'string',string)
```

```
for i = 1:str2double(num_widths)
```

```
% Clear variables
```

```
minLoc = []; maxLoc = []; x = []; y = []; c = []; locFirstHalf = [];
locSecondHalf = []; minsSecondHalf = []; minsFirstHalf = [];
```

```
Receive user input for line & intensity profile
```

```
set(handles.beamWidthTextBox,'string',['Please draw a line for the width of beam ', num2str(i)]);
% Get image profile
[x,y,c] = improfile;
```

```

% Smooth with lowpass filter to make calculations easier
c = smooth(c(:,1));
% Plot intensity profile
figure
plot(c(:,1))
title('Please select the max y & average min y values of the intensity curve.')
pause(0.000001)
% Get user chosen max
[~,maxIntensity] = ginput(2);
close %close figure
% Find half max
halfMax = round((maxIntensity(1)+maxIntensity(2))/ 2);
% Find locations of halfMax
vec = c(:,1) - halfMax;
[a,~] = find(abs(vec) < 10);
% separate vec into 2 portions for down slope & up slope
p = diff(a);
[loc,~]= find(p>3);

```

Full-width half max

Take averages of the two portions to find average 2 locations for

```

firstHalf = length(a(1:loc));
secondHalf = length(a(loc+1:end));
for m = 1:firstHalf
    minsFirstHalf(m,1) = abs(vec(a(m)));
    minsFirstHalf(m,2) = a(m);
end

for k = 1:secondHalf
    minsSecondHalf(k,1) = abs(vec(a(firstHalf+k)));
    minsSecondHalf(k,2) = a(firstHalf+k);
end

% Get right side value & location
mLoc1 = min(minsFirstHalf(:,1));
l1 = find(minsFirstHalf(:,1) == mLoc1)
locFirstHalf = minsFirstHalf(l1, 2);
% Get left side location
mLoc2 = min(minsSecondHalf(:,1));
l2 = find(minsSecondHalf(:,1) == mLoc2)
locSecondHalf = minsSecondHalf(l2,2);

% Convert locations to actual coordinates
xCoord(1) = x(locFirstHalf);
yCoord(1) = y(locFirstHalf);
xCoord(2) = x(locSecondHalf);
yCoord(2) = y(locSecondHalf);
% Plot points
hold on
plot(handles.axes_beamWidth, xCoord(1),yCoord(1),'k+', 'linewidth', 2);
plot(handles.axes_beamWidth, xCoord(2),yCoord(2),'k+', 'linewidth', 2);

% Calculate distance between two points
pixels = sqrt((xCoord(2)-xCoord(1))^2 + ((yCoord(2)-yCoord(1))^2));
handles.beamWidth_beamWidths(i) = pixels*handles.beamWidth_calibrationFactor;
% Update text box with beam widths
string{i+1,1} = num2str(handles.beamWidth_beamWidths(i), '%.3f');

```

```

set(handles.beamWidthText,'string',string)

end
set(handles.beamWidthTextBox,'string','Beam widths acquired. ');
pause(0.001);
% replot unzoomed image
axes(handles.axes_beamWidth);
imshow(handles.beamWidth_image, 'InitialMagnification','fit'); %display image on axis
%%%%%%%% handles.beam_beamWidths is the beam widths %%%%%%%%%%
guidata(hObject,handles);

% --- Executes on button press in beamWidthManual.
function beamWidthManual_Callback(hObject, eventdata, handles)
prompt = {'Please enter the nominal beam widths.',' ',' ',' '};
num_lines = 1;
default_ans = {' ',' ',' '};
nominal = inputdlg(prompt,'Manual Beam Width',1,default_ans);
prompt2 = {'Please enter the vendor supplied beam widths.',' ',' ',' '};
vendor = inputdlg(prompt2,'Manual Beam Width',1,default_ans);
prompt3 = {'Please enter the measured beam widths.',' ',' ',' '};
measured = inputdlg(prompt3,'Manual Beam Width',1,default_ans);

for i = 1:4
    nom = str2num(nominal{i});
    vend = str2num(vendor{1});
    meas = str2num(measured{i});
    %measured beam widths
    handles.beamWidths_beamWidths(i,1) = meas;
    %percent diff between nominal and measured
    handles.beamWidth_beamWidths(i,2) = (abs((nom-meas)/meas)*100);
    % percent diff between vendor supplied and measured
    handles.beamWidth_beamWidths(i,3) = (abs((vend-meas)/meas)*100);

end

%%%%%%%% handles.beamWidth_beamWidths has all data %%%%%%%%%%
guidata(hObject,handles);

% --- Executes on button press in beamWidthSave.
function beamWidthSave_Callback(hObject, eventdata, handles)

```

### Gantry Tilt Panel

```

% --- Executes on button press in gantryTiltBrowse.
function gantryTiltBrowse_Callback(hObject, eventdata, handles)
[filename pathname ] = uigetfile('*.png','Please select an image file for beam width'); %get image
set(handles.axes_gantryTilt,'visible','on');
axes(handles.axes_gantryTilt);
handles.gantryTilt_image = [pathname filename];
I = imread(handles.gantryTilt_image);
imshow(I, 'InitialMagnification','fit'); %display image on axis
set(handles.gantryTiltTextBox,'visible','on'); %Makes instruction text visible
set(handles.gantryTiltCalculate,'visible','on');
guidata(hObject, handles);

```

```

% --- Executes on button press in gantryTiltManual.
function gantryTiltManual_Callback(hObject, eventdata, handles)

% --- Executes on button press in gantryTiltCalculate.
function gantryTiltCalculate_Callback(hObject, eventdata, handles)

```

```

zoom reset
set(handles.gantryTiltHelp,'visible','on')
set(handles.gantryTiltPercentDiffText,'string','');
set(handles.gantryTiltAngleText,'string','');

set(handles.gantryTiltTextBox,'string','Please draw a line over the top of the film for the 0 degree mark. Right click to end line. ');
[x,y] = getline(handles.axes_gantryTilt);
hold on
plot(handles.axes_gantryTilt,x,y,'k','linewidth',1)

v_0 = [(x(1)-x(2)) (y(1)-y(2)) 0]; %vector for 0 degrees
v_90 = [-(y(1)-y(2)) (x(1)-x(2)) 0]; %Find perpendicular line
string{1} = 'Angles (degrees):';
string2{1} = '% difference: ';

```

```

Get +30 angle
set(handles.gantryTiltTextBox,'string','Please trace the +30deg line. ');
[x,y] = getline(handles.axes_gantryTilt);
hold on
plot(handles.axes_gantryTilt,x,y,'r','linewidth',3)
v = [(x(1)- x(2)) (y(1)- y(2)) 0];
angle = atan2d(norm(cross(v,v_90)),dot(v,v_90));
handles.gantryTilt_angles(1,1) = angle;
string{2} = num2str(angle,'%3f');
set(handles.gantryTiltAngleText,'string',string)

% Calculate percent difference
handles.gantryTilt_angles(1,2) = (abs((30-angle)/angle)*100);
string2{2} = num2str(handles.gantryTilt_angles(1,2), '%3f');
set(handles.gantryTiltPercentDiffText,'string',string2)

```

```

Get +15 angle
set(handles.gantryTiltTextBox,'string','Please trace the +15deg line. ');
[x,y] = getline(handles.axes_gantryTilt);
hold on
plot(handles.axes_gantryTilt,x,y,'r','linewidth',3)
v = [(x(1)- x(2)) (y(1)- y(2)) 0];
angle = atan2d(norm(cross(v,v_90)),dot(v,v_90)); %calc angle
handles.gantryTilt_angles(2) = angle;
string{3} = num2str(angle,'%3f');
set(handles.gantryTiltAngleText,'string',string)

```

```

% Calculate percent difference
handles.gantryTilt_angles(2,2) = (abs((15-angle)/angle)*100);
string2{3} = num2str(handles.gantryTilt_angles(2,2), '%3f');
set(handles.gantryTiltPercentDiffText,'string',string2)

```

```

Get -15 angle
set(handles.gantryTiltTextBox,'string','Please trace the -15deg line. ');
[x,y] = getline(handles.axes_gantryTilt);
hold on
plot(handles.axes_gantryTilt,x,y,'r','linewidth',3)
v = [(x(1)- x(2)) (y(1)- y(2)) 0];
angle = -atan2d(norm(cross(v,v_90)),dot(v,v_90)); %calc angle
handles.gantryTilt_angles(3) = angle;
string{4} = num2str(angle,'%3f');
set(handles.gantryTiltAngleText,'string',string)

% Calculate percent difference
handles.gantryTilt_angles(3,2) = (abs((-15-angle)/angle)*100);
string2{4} = num2str(handles.gantryTilt_angles(3,2), '%3f');
set(handles.gantryTiltPercentDiffText,'string',string2)

```

```

Get -30 angle
set(handles.text2,'string','Please trace the +15deg line. ');
[x,y] = getline(handles.axes_gantryTilt);
hold on
plot(handles.axes_gantryTilt,x,y,'r','linewidth',3)
v = [(x(1)- x(2)) (y(1)- y(2)) 0];
angle = -atan2d(norm(cross(v,v_90)),dot(v,v_90)); %calc angle
handles.gantryTilt_angles(4) = angle;
string{5} = num2str(angle,'%3f');
set(handles.gantryTiltAngleText,'string',string)

% Calculate percent difference
handles.gantryTilt_angles(4,2) = (abs((-30-angle)/angle)*100);
string2{5} = num2str(handles.gantryTilt_angles(4,2), '%3f');
set(handles.gantryTiltPercentDiffText,'string',string2);

```

```

set(handles.gantryTiltTextBox,'string','Angles acquired. ');

```

```

% replot unzoomed image
axes(handles.axes_gantryTilt);
I = imread(handles.gantryTilt_image);
imshow(I, 'InitialMagnification','fit'); %display image on axis
%%%% handles.gantryTilt_angles is the gantry angles %%%%
guidata(hObject,handles);

```

```

% --- Executes on button press in gantryTiltHelp.

```

```

function gantryTiltHelp_Callback(hObject, eventdata, handles)
figure
imshow('gantryTilt.png');
guidata(hObject,handles);

```

### Protocol Review Panel

```

% --- Executes on selection change in protocolNum.
function protocolNum_Callback(hObject, eventdata, handles)

```

```

contents = cellstr(get(hObject,'String'));
numProtocols = contents{get(hObject,'Value')};

% selection determines which fields are visible
switch numProtocols
case '1'
    set(handles.protocolText, 'Visible', 'On');
    set(handles.protocolStatusText, 'Visible', 'On');
    set(handles.protocolCommentsText, 'Visible', 'On');
    set(handles.protocolName1, 'Visible', 'On');
    set(handles.protocolStatusMenu1, 'Visible', 'On');
    set(handles.protocolComments1, 'Visible', 'On');

    set(handles.protocolName2, 'Visible', 'Off');
    set(handles.protocolStatusMenu2, 'Visible', 'Off');
    set(handles.protocolComments2, 'Visible', 'Off');
    set(handles.protocolName3, 'Visible', 'Off');
    set(handles.protocolStatusMenu3, 'Visible', 'Off');
    set(handles.protocolComments3, 'Visible', 'Off');
    set(handles.protocolName4, 'Visible', 'Off');
    set(handles.protocolStatusMenu4, 'Visible', 'Off');
    set(handles.protocolComments4, 'Visible', 'Off');
    set(handles.protocolName5, 'Visible', 'Off');
    set(handles.protocolStatusMenu5, 'Visible', 'Off');
    set(handles.protocolComments5, 'Visible', 'Off');
    set(handles.protocolName6, 'Visible', 'Off');
    set(handles.protocolStatusMenu6, 'Visible', 'Off');
    set(handles.protocolComments6, 'Visible', 'Off');
case '2'
    set(handles.protocolText, 'Visible', 'On');
    set(handles.protocolStatusText, 'Visible', 'On');
    set(handles.protocolCommentsText, 'Visible', 'On');
    set(handles.protocolName1, 'Visible', 'On');
    set(handles.protocolStatusMenu1, 'Visible', 'On');
    set(handles.protocolComments1, 'Visible', 'On');
    set(handles.protocolName2, 'Visible', 'On');
    set(handles.protocolStatusMenu2, 'Visible', 'On');
    set(handles.protocolComments2, 'Visible', 'On');

    set(handles.protocolName3, 'Visible', 'Off');
    set(handles.protocolStatusMenu3, 'Visible', 'Off');
    set(handles.protocolComments3, 'Visible', 'Off');
    set(handles.protocolName4, 'Visible', 'Off');
    set(handles.protocolStatusMenu4, 'Visible', 'Off');
    set(handles.protocolComments4, 'Visible', 'Off');
    set(handles.protocolName5, 'Visible', 'Off');
    set(handles.protocolStatusMenu5, 'Visible', 'Off');
    set(handles.protocolComments5, 'Visible', 'Off');
    set(handles.protocolName6, 'Visible', 'Off');
    set(handles.protocolStatusMenu6, 'Visible', 'Off');
    set(handles.protocolComments6, 'Visible', 'Off');
case '3'
    set(handles.protocolText, 'Visible', 'On');
    set(handles.protocolStatusText, 'Visible', 'On');
    set(handles.protocolCommentsText, 'Visible', 'On');
    set(handles.protocolName1, 'Visible', 'On');
    set(handles.protocolStatusMenu1, 'Visible', 'On');
    set(handles.protocolComments1, 'Visible', 'On');
    set(handles.protocolName2, 'Visible', 'On');
    set(handles.protocolStatusMenu2, 'Visible', 'On');
    set(handles.protocolComments2, 'Visible', 'On');
    set(handles.protocolName3, 'Visible', 'On');

```



```

set(handles.protocolName1, 'Visible', 'On');
set(handles.protocolStatusMenu1, 'Visible', 'On');
set(handles.protocolComments1, 'Visible', 'On');
set(handles.protocolName2, 'Visible', 'On');
set(handles.protocolStatusMenu2, 'Visible', 'On');
set(handles.protocolComments2, 'Visible', 'On');
set(handles.protocolName3, 'Visible', 'On');
set(handles.protocolStatusMenu3, 'Visible', 'On');
set(handles.protocolComments3, 'Visible', 'On');
set(handles.protocolName4, 'Visible', 'On');
set(handles.protocolStatusMenu4, 'Visible', 'On');
set(handles.protocolComments4, 'Visible', 'On');
set(handles.protocolName5, 'Visible', 'On');
set(handles.protocolStatusMenu5, 'Visible', 'On');
set(handles.protocolComments5, 'Visible', 'On');
set(handles.protocolName6, 'Visible', 'On');
set(handles.protocolStatusMenu6, 'Visible', 'On');
set(handles.protocolComments6, 'Visible', 'On');

```

otherwise

```

set(handles.protocolText, 'Visible', 'Off');
set(handles.protocolStatusText, 'Visible', 'Off');
set(handles.protocolCommentsText, 'Visible', 'Off');
set(handles.protocolName1, 'Visible', 'Off');
set(handles.protocolStatusMenu1, 'Visible', 'Off');
set(handles.protocolComments1, 'Visible', 'Off');
set(handles.protocolName2, 'Visible', 'Off');
set(handles.protocolStatusMenu2, 'Visible', 'Off');
set(handles.protocolComments2, 'Visible', 'Off');
set(handles.protocolName3, 'Visible', 'Off');
set(handles.protocolStatusMenu3, 'Visible', 'Off');
set(handles.protocolComments3, 'Visible', 'Off');
set(handles.protocolName4, 'Visible', 'Off');
set(handles.protocolStatusMenu4, 'Visible', 'Off');
set(handles.protocolComments4, 'Visible', 'Off');
set(handles.protocolName5, 'Visible', 'Off');
set(handles.protocolStatusMenu5, 'Visible', 'Off');
set(handles.protocolComments5, 'Visible', 'Off');
set(handles.protocolName6, 'Visible', 'Off');
set(handles.protocolStatusMenu6, 'Visible', 'Off');
set(handles.protocolComments6, 'Visible', 'Off');

```

end

```
guidata(hObject,handles);
```

Basic Information

```

BI.testingDate = get(handles.BI_testingDate, 'string');
BI.reportDate = get(handles.BI_reportDate, 'string');
BI.scannerLocation = get(handles.BI_scannerLocation, 'string');
BI.facilityName = get(handles.BI_facilityName, 'string');
BI.facilityAddress = get(handles.BI_facilityAddress, 'string');
BI.facilityContactName = get(handles.BI_facilityContactName, 'string');
BI.facilityContactEmail = get(handles.BI_facilityContactEmail, 'string');
BI.facilityContactPhone = get(handles.BI_facilityContactPhone, 'string');
BI.physicistName = get(handles.BI_physicistName, 'string');
BI.physicistAddress = get(handles.BI_physicistAddress, 'string');
BI.physicistPhone = get(handles.BI_physicistPhone, 'string');
BI.physicistEmail = get(handles.BI_physicistEmail, 'string');
BI.comments = get(handles.BI_comments, 'string');

```



Safety  
Question 1

```
val = get(handles.safeQ1,'value');
switch(val)
  case 1
    safety.Q1 = 'N/A';
  case 2
    safety.Q1 = 'Yes';
  case 3
    safety.Q1 = 'No';
end
% Question 2
val = get(handles.safeQ2,'value');
switch(val)
  case 1
    safety.Q2 = 'N/A';
  case 2
    safety.Q2 = 'Yes';
  case 3
    safety.Q2 = 'No';
end
% Question 3
val = get(handles.safeQ3,'value');
switch(val)
  case 1
    safety.Q3 = 'N/A';
  case 2
    safety.Q3 = 'Yes';
  case 3
    safety.Q3 = 'No';
end
% Question 4
val = get(handles.safeQ4,'value');
switch(val)
  case 1
    safety.Q4 = 'N/A';
  case 2
    safety.Q4 = 'Yes';
  case 3
    safety.Q4 = 'No';
end
% Question 5
val = get(handles.safeQ5,'value');
switch(val)
  case 1
    safety.Q5 = 'N/A';
  case 2
    safety.Q5 = 'Yes';
  case 3
    safety.Q5 = 'No';
end
% Question 6
val = get(handles.safeQ6,'value');
switch(val)
  case 1
    safety.Q6 = 'N/A';
  case 2
    safety.Q6 = 'Yes';
  case 3
    safety.Q6 = 'No';
end
```

```

% Question 7
val = get(handles.safeQ1,'value');
switch(val)
    case 1
        safety.Q7 = 'N/A';
    case 2
        safety.Q7 = 'Yes';
    case 3
        safety.Q7 = 'No';
end
% Question 8
val = get(handles.safeQ8,'value');
switch(val)
    case 1
        safety.Q8 = 'N/A';
    case 2
        safety.Q8 = 'Yes';
    case 3
        safety.Q8 = 'No';
end
% Question 9
val = get(handles.safeQ9,'value');
switch(val)
    case 1
        safety.Q9 = 'N/A';
    case 2
        safety.Q9 = 'Yes';
    case 3
        safety.Q9 = 'No';
end
% Question 10
val = get(handles.safeQ10,'value');
switch(val)
    case 1
        safety.Q10 = 'N/A';
    case 2
        safety.Q10 = 'Yes';
    case 3
        safety.Q10 = 'No';
end

```

#### Artifacts

```

artifacts.Pitch = get(handles.artifactsPitch,'string');
artifacts.ScanMode= get(handles.artifactsScanMode,'string');
artifacts.MA = get(handles.artifactsMA,'string');
artifacts.RotationTime= get(handles.artifactsRotationTime,'string');
artifacts.Effective= get(handles.artifactsEffective,'string');
artifacts.Denoising= get(handles.artifactsDenoising,'string');
artifacts.Kernel= get(handles.artifactsKernel,'string');
artifacts.SFOV= get(handles.artifactsSFOV,'string');
artifacts.RFOV= get(handles.artifactsRFOV,'string');
artifacts.Slice= get(handles.artifactsSlice,'string');
val = get(handles.artifactsYN,'value');
switch(val)
    case 1
        artifacts.ArtifactsPresent = '';
    case 2
        artifacts.ArtifactsPresent = 'Yes';
    case 3
        artifacts.ArtifactsPresent = 'No';
end

```

```
end
artifacts.Comments= get(handles.artifactsComments,'string');
%artifacts.Stdev= get(handles.artifactsStdev,'string');
```

#### LCD

Not sure what he wants from LCD other than the images

#### Noise

```
Noise.Pitch = get(handles.NoisePitch,'string');
Noise.ScanMode= get(handles.artifactsScanMode,'string');
Noise.MA = get(handles.artifactsMA,'string');
Noise.RotationTime= get(handles.NoiseRotTime,'string');
Noise.Effective= get(handles.NoiseEffective,'string');
Noise.Denoising= get(handles.NoiseDenoising,'string');
Noise.Kernel= get(handles.NoiseKernel,'string');
Noise.SFOV= get(handles.NoiseSFOV,'string');
Noise.RFOV= get(handles.NoiseRFOV,'string');
Noise.Slice= get(handles.NoiseSlice,'string');
Noise.Comments= get(handles.NoiseComments,'string');
%Noise.StandDeviation = get(handles.Noise_standDeviation,'string');
```

#### CT Number

```
CTNum.Pitch = get(handles.CTNumPitch,'string');
CTNum.ScanMode= get(handles.CTNumScanMode,'string');
CTNum.MA = get(handles.CTNumMA,'string');
CTNum.RotationTime= get(handles.CTNumRotTime,'string');
CTNum.Effective= get(handles.CTNumEffective,'string');
CTNum.Denoising= get(handles.CTNumDenoising,'string');
CTNum.Kernel= get(handles.CTNumKernel,'string');
CTNum.SFOV= get(handles.CTNumSFOV,'string');
CTNum.RFOV= get(handles.CTNumRFOV,'string');
CTNum.Slice= get(handles.CTNumSlice,'string');
CTNum.Comments= get(handles.CTNumComments,'string');
%CTNum.MeanCTNum = get(handles.CTNum_mean,'string');
```

#### CT Uniformity

```
CTUniformity.Pitch = get(handles.CTUniformityPitch,'string');
CTUniformity.ScanMode= get(handles.CTUniformityScanMode,'string');
CTUniformity.MA = get(handles.CTUniformityMA,'string');
CTUniformity.RotationTime= get(handles.CTUniformityRotTime,'string');
CTUniformity.Effective= get(handles.CTUniformityEffective,'string');
CTUniformity.Denoising= get(handles.CTUniformityDenoising,'string');
CTUniformity.Kernel= get(handles.CTUniformityKernel,'string');
CTUniformity.SFOV= get(handles.CTUniformitySFOV,'string');
CTUniformity.RFOV= get(handles.CTUniformityRFOV,'string');
CTUniformity.Slice= get(handles.CTUniformitySlice,'string');
CTUniformity.Comments= get(handles.CTUniformityComments,'string');
%CTUniformity.StandDeviation = get(handles.CTUniformity_standDeviation,'string');
```

#### Slice Width

```

%handles.sliceWidth_width;
h = msgbox('Your text file has been exported.');
```

```

title = 'Example Title';
fid=fopen('CTReport.txt','w');
fprintf(fid, '\');
fprintf(fid,
'documentclass[{}]{spie}\n\usepackage{graphicx}\n\usepackage{float}\n\usepackage{caption}\n\usepackage{enumite
m}\n');
fprintf(fid, '\');
fprintf(fid,
'DeclareGraphicsExtensions{.pdf,.png,.jpg,.eps}\n\usepackage{subfigure}\n\usepackage{lipsum}\n\usepackage{am
smath}\n\usepackage{cite}\n\usepackage{color}\n\usepackage{wrapfig}\n\usepackage{amsmath}\n\usepackage{b
ooktabs}\n\newcommand{\ben}{\begin{eqnarray}});
fprintf(fid, '\');
fprintf(fid, 'displaystyle\n\newcommand{');
fprintf(fid, '\');
fprintf(fid, 'een}{');
fprintf(fid, '\');
fprintf(fid, 'end{eqnarray}}\n\newcommand{');
fprintf(fid, '\');
fprintf(fid, 'et}{');
fprintf(fid, '\');
fprintf(fid, 'emph{et al }}\n\newcommand{\refb}[1]{(\ref{#1})}\n\newcommand{\lignore}[1]{\n\n');

fprintf(fid, '\title{*TITLE VARIABLE HERE*}\n\n\n\n');

fprintf(fid, '\author{');
fprintf(fid, BI.physicistName);
fprintf(fid, '\n\skiplinehalf\nReport Date: ');
fprintf(fid, BI.reportDate);
fprintf(fid, '\n\skiplinehalf\nTesting Date: ');
fprintf(fid, BI.testingDate);
fprintf(fid, '\n\skiplinehalf\n');
fprintf(fid, BI.facilityName);
fprintf(fid, '\n');
fprintf(fid, BI.scannerLocation);
fprintf(fid, '\n\n');
fprintf(fid, BI.facilityAddress);
fprintf(fid, '\n\n');
fprintf(fid, BI.facilityContactName);
fprintf(fid, '\n\n');
fprintf(fid, BI.facilityContactEmail);
fprintf(fid, '\n\n');
fprintf(fid, BI.facilityContactPhone);
fprintf(fid, '\n\n\n');

%Background of scanner and report section
fprintf(fid, '\begin{document}\n\n\maketitle\n\n\section*(Background of this scanner and Report)\n\n');
fprintf(fid, '*BI_COMMENTS VARIABLE*\n\n');

%Overview and recommendations section
fprintf(fid, '\section{Overview and recommendations}\n\n');
fprintf(fid, 'Table \ref{tbl:overview} lists all tests performed and the outcome.\n\n');

fprintf(fid, '\begin{table}[htb]begin{footnotesize}\n');
fprintf(fid, '\begin{center}\n\caption{Overview of tests performed}\n\label{tbl:overview}\n\');
fprintf(fid, 'resizebox{0.5\');
fprintf(fid, 'textwidth}{!}\n\');
fprintf(fid, 'begin{tabular}{l l}\n\toprule\nTest & Result\n\n\midrule\n');

fprintf(fid, 'Room Safety Features & ');

```

```

fprintf(fid, '*ROOMSAFETY P/F variable*\\n\\n');

fprintf(fid, 'Scanner Safety Features & ');
fprintf(fid, '*SCANNER_SAFETY P/F variable*\\n\\n');

fprintf(fid, 'Artifact Check & ');
fprintf(fid, '*ARTIFACT_CHECK P/F variable*\\n\\n');

fprintf(fid, 'Low contrast detection & ');
fprintf(fid, '*LOW_CONTRAST_DETECT P/F variable*\\n\\n');

fprintf(fid, 'Spatial Resolution & ');
fprintf(fid, '*SPATIAL_RESOLUTION P/F variable*\\n\\n');

fprintf(fid, 'Noise & ');
fprintf(fid, '*NOISE P/F variable*\\n\\n');

fprintf(fid, 'CT Number Check & ');
fprintf(fid, '*CT# P/F variable*\\n\\n');

fprintf(fid, 'Dose (Scanner Output) & ');
fprintf(fid, '*DOSE P/F variable*\\n\\n');

fprintf(fid, 'Actual Beam Width & ');
fprintf(fid, '*BEAMWIDTH P/F variable*\\n\\n');

fprintf(fid, 'Slice Thickness & ');
fprintf(fid, '*SLICE_THICKNESS P/F variable*\\n\\n');

fprintf(fid, 'Lasers & ');
fprintf(fid, '*LASERS P/F variable*\\n\\n');

fprintf(fid, 'Table Movement & ');
fprintf(fid, '*TABLE_MOVEMENT P/F variable*\\n\\n');

fprintf(fid, 'Lasers to Scout to Image Consistency & ');
fprintf(fid, '*LASER_SCOUT_IMAGE P/F variable*\\n\\n');

fprintf(fid, 'Monitor Luminance/Uniformity & ');
fprintf(fid, '*Monitor Lumin/Unif P/F variable*\\n\\n');

fprintf(fid, 'Geometric Distortion & ');
fprintf(fid, '*GeometricDistortion P/F variable*\\n\\n');

fprintf(fid, 'Spatial Accuracy & ');
fprintf(fid, '*SPATIAL+ACCURACY P/F variable*\\n\\n');

fprintf(fid, 'Gantry Tilt & ');
fprintf(fid, '*GANTRY_TILT P/F variable*\\n\\n');

fprintf(fid, 'Protocol Review & ');
fprintf(fid, '*Protocol_REVIEW P/F variable*\\n\\n');

fprintf(fid, 'Review QA Logs & ');
fprintf(fid, '*Review_QA_LOGS P/F variable*\\n\\n');

fprintf(fid, '\\bottomrule\\n\\n');
fprintf(fid, 'end{tabular}\\n\\n');
fprintf(fid, 'end{center}\\n\\n');
fprintf(fid, 'end{footnotesize}\\n\\n');
fprintf(fid, 'end{table}\\n\\n');

```

```

%Artifacts section

%Code to see if they filled out the artifacts test needed here
fprintf(fid, '\\section{Artifacts}\n');
%Code to print either pass or fail depending on if it passed or failed

%if passed
fprintf(fid, '\\');
fprintf(fid, 'textcolor{green}{PASS}\n\n');

%if failed
fprintf(fid, '\\');
fprintf(fid, 'textcolor{red}{FAIL}\n\n');

%Artifact comments
fprintf(fid, 'INSERT ARTIFACT COMMENT VARIABLE HERE*\n\n');

%INSERT ARTIFACT TABLE CODE HERE IF APPROPRIATE

%Noise and CT Number Uniformity Section
%Code here to only post this if these tests were done
fprintf(fid, '\\section{Noise and CT Number Uniformity}\n\\');

%if passed
fprintf(fid, '\\');
fprintf(fid, 'textcolor{green}{PASS}\n\n');

%if failed
fprintf(fid, '\\');
fprintf(fid, 'textcolor{red}{FAIL}\n\n');

%NOISE/CT Number Uniformity comments
fprintf(fid, 'INSERT NOISE/CT# Uniformity COMMENT VARIABLE HERE*\n\n');

fprintf(fid, '\\');
fprintf(fid, 'end{document}');
fclose(fid);

```