A LabVIEW® based generic CT scanner control software platform.

Dierick M., Van Loo D., Masschaele B., Boone M., Van Hoorebeke L.

UGCT, Centre for X-ray Tomography, Department of Physics and Astronomy, Ghent University, Proeftuinstraat 86, B-9000 Ghent, Belgium

Abstract : UGCT, the Centre for X-ray tomography at Ghent University (Belgium) does research on X-ray tomography and its applications. This includes the development and construction of state-of-the-art CT scanners for scientific research. Because these scanners are built for very different purposes they differ considerably in their physical implementations. However, they all share common principle functionality. In this context a generic software platform was developed using LabVIEW® in order to provide the same interface and functionality on all scanners. This article describes the concept and features of this software, and its potential for tomography in a research setting. The core concept is to rigorously separate the abstract operation of a CT scanner from its actual physical configuration. This separation is achieved by implementing a sender-listener architecture. The advantages are that the resulting software platform is generic, scalable, highly efficient, easy to develop and to extend, and that it can be deployed on future scanners with minimal effort.

Corresponding author : manuel.dierick@UGent.be , Tel: +32/2646611, Fax: +32/2646697

Keywords : X-ray tomography, software, LabVIEW

1. Introduction:

X-ray micro-CT is becoming a well-established technique for non-destructive 3D imaging of samples, both in medical and non-medical research. Based on a series of 2D X-ray radiographs it allows to reconstruct the internal structure of an object in 3D by means of a suitable reconstruction algorithm [1]. Micro-CT not only offers valuable morphological information, but based on the 3D data one can extract useful data such as surface meshes that can serve as input for finite element simulations (e.g. for simulating fluid flow in blood vessels), or porosity distributions (e.g. for studying osteoporosis).

UGCT, the Centre for X-ray Tomography at Ghent University (Belgium), specializes in micro and nano-CT on non-living samples. Crucially, all scanners and software tools used at UGCT are developed in-house to gain maximum control over the data processing. UGCT's research covers everything from the design and construction of the actual CT scanners, over data acquisition, processing, reconstruction and visualization up to and including 3D analysis [2]. Besides doing research on tomography itself, UGCT also operates as a user facility. Because of the large number of different available X-ray sources, detectors and motors that have to be controlled in the different scanners, a generic scanner control software was designed that is independent of a particular hardware configuration and which is used on all UGCT scanners for operation and data-acquisition. This article will give an overview of this generic scanner control software. The resulting raw scanner data is subsequently reconstructed using the software package Octopus [3] which provides access to all relevant reconstruction parameters, including misalignment corrections, making it ideal for CT in a research setting.

2. Materials and Methods:

2.1 The problem:

UGCT has two CT scanners already in operation, two others in the commisisiong stage and a fifth one in the design stage. One scanner has two separate X-ray sources and two detectors, which can be selected on the fly. Another scanner has some parts rotating on a gantry while others are static, and therefore two control PC's are required. For all scanners together, a total of more than 20 components need to be controlled. This includes 3 Hamamatsu X-ray sources (HM_L10711, HM_L6622 and HM_L9181) each controlled through RS-232, a Feinfocus FXE-160 dual head microfocus source, 2 PCI-based CCD–cameras from Photonic Science (PS-VHR 12 MPixel and a PS-VHR Air 15 MPixel) that come with a dll supplied by the manufacturer, 2 Rad-Icon flat panel CMOS sensors (RadEye EV and RadEye HR with Gadox scintillators) which are read out by means of a frame grabber (Imagenation PXD-1000). There are 2 Varian Paxscan 2520V aSi flat panel detectors with CsI scintillator which are controlled through an Ethernetcard and a dll supplied by the manufacturer. Yet another detector is the PerkinElmer XRD 1620 which comes with a dll and a PCI card. Further equipment includes 2 high-precision air-bearing rotation stages (Micos UPR-160 Air controlled through a serial port and an Aerotech ABRS-150MP controlled using a .NET library supplied by the manufacturer) and multiple RS-485 controlled linear motor stages, serially controlled piezo XY stages from PI Motion (M-662 PILine), an Active-X controlled webcam to monitor the setup and to take optical images of the sample, etc.

2.2 The solution:

It is clear that it is quite a challenge to develop a scanner control software that is user friendly and at the same time independent from the underlying hardware. It was decided to develop the scanner control software in LabVIEW® (version 8.6), a graphical programming language which is ideally suited for integrating different types of hardware and for speedy development in a research environment. The use of LabVIEW for controlling micro-CT systems has been reported before [4,5,6,7]. In most cases however this involves code specifically written around one specific combination of hardware or one specific

component such as a camera driver [8]. They were not designed from the start to independent from underlying hardware. The core concept of the control software reported in this article is to completely separate the abstract scanner operation from its physical implementation and actual configuration. To separate the abstract parts of the code from the physical parts of the code that address the actual hardware components, a custom LabVIEW® design template was implemented based on so called 'user events', something similar to an interrupt but with embedded data. In this case the user event was defined as a cluster containing the name of the addressed component (tube, camera, mag_obj etc.), the command to be executed (set exposure, xrayon, etc), arguments and feedback. Four event types were defined, tube events, motor events, camera events and other events. A standardized set of commands was defined for each category. For example, tube commands include xrayon, set current, etc. Basically, abstract commands are broadcast and are picked up to be executed by the corresponding physical component. This sender-listener concept effectively decouples the abstract operation of a scanner and the physical actions to be taken on a lower level. More generally, not only the main program but also individual components or add-on programs can broadcast or receive commands, resulting in a very flexible framework. A schematic representation of this framework is shown in figure 1. Commands are broadcast into the 'ether', and all components that registered for that particular type of event receive the command when fired and process it as soon as possible.

2.3 The component programs:

Each component is implemented as a standalone program (or VI as it is called in LabView) that runs separately from the main program. It is always based on the same template, written around an event structure which registers for the type of event corresponding to that component. When the event structure receives a command event it first checks the component field. If it is addressed to another component it ignores the command, else it confirms to the outside world that it received the command

(by firing a RCV event) and starts executing it. During execution it may occasionally fire feedback (FDB) events (for example motor positions during a movement). Finally the component acknowledges to the outside world that the command has been executed (ACK) or if an error occurred (ERR). In case a command event cannot be handled immediately the event structure automatically queues it to be processed as soon as the previously received event has been handled. This ensures that no commands get lost. Typically, if a component is idle, a command event is heard almost instantaneously after being fired (<1ms). Finally it should be noted that a component program is more that just a driver. It has its own interface from which it can be controlled directly (although typically this is only used for actions that are very specific for that particular component). It can run as a standalone program, without being part of the CT scanner framework.

2.3 The main program:

The main program, referred to as the scannerGUI, controls a complete CT scanner. The code behind it only contains abstract functionality that holds for any particular physical implementation of a CT scanner. An abstract scanner configuration is supposed to contain one or more X-ray sources, one or more X-ray cameras and a number of motion axes. All X-ray sources have some common properties and actions, like switching the X-ray beam on or off, setting the high voltage and current, setting the focusing mode etc. Similarly, all X-ray cameras have a certain pixel pitch, pixel count, exposure time, binning modes etc. Every motion axis has a certain position, speed, lower and upper limit etc. The motion axes can not only move the sample, but also a tube or a detector. When combined with four direction categories, namely rotation, magnification, translation and vertical movements, this results in an abstract motor naming convention of the nature direction_subject (e.g. mag_tube, rot_obj, ver_det, etc.). When operating a scanner, the user normally only interacts with the scannerGUI. The scannerGUI contains four independent loops running in parallel. The first is the user event loop. This handles all

events related to user interaction with the interface. For example, when the user clicks on the X-ray ON button the command ‘tube xrayon’ is sent out by the user event loop. The second loop is the polling loop which processes feedback from all components and adjusts the interface and status globals accordingly. For example, when it receives acknowledgment that mag_obj has moved to position X it will recalculate the geometry, voxel size, magnification etc and update the 3D drawing accordingly. To ensure a good responsiveness of the polling loop, time-consuming operations like image corrections and saving to harddisk are handed over to the user event loop which will block user interactions for the time necessary. A third independent loop is the script loop. A script is a text based series of commands, generated by the script preparation tool or manually, and represents a CT scan (or other types of scans like tiled radiographies). The script is executed in a serial way. Before each command all component status flags are checked (this way, execution of a scan is automatically paused when for example someone enters the bunker room, and it continues as soon as the door closes again). Then the command is sent out. A ‘wait’ flag with each command determines if the script waits until the command finished execution (default), or to proceed as soon as the component confirmed having received the command. This means that, although the script is executed sequentially, that certain actions can run in parallel if desired. For example, in some cases the ‘wait’ flag for the ‘save image’ command is set to FALSE so that the command to rotate to the next angular position can already start executing while the image is being saved, thus speeding up the acquisition. Additionally, during exposures the X-ray status is monitored and if necessary the image is taken again to ensure that all images were properly exposed (for example, operating an X-ray source close to its highest voltage can occasionally give discharges causing the X-ray output to drop temporarily). The fourth and final loop implements the abstract component ‘scanner’ in such a way that scripts or external routines can call commands that relate to the abstract concept of a ‘scanner’, like ‘scanner set_output_directory’.

3. Results

A short overview of the discerning features of the scanner control software is given below:

3.1 The main interface:

When doing CT research with self built setups one needs control over all components: the tube, the camera, sample positioning stage etc. Only actions and properties which are common to at least most of the components of that type are added to the main scannerGUI interface. Actions or properties that are very specific to a particular component are not included in the main interface, but only on the interface of that specific component. When the software is started, an ini file is read which lists all the components that make up that particular scanner. These components are all launched and the scannerGUI is updated accordingly, listing the available components and their appropriate settings like possible binning modes, exposure time range etc. Figure 2 shows a screenshot of the main window.

Figure 2 : The main user interface.

Tube control: from the GUI one can select one of the available tubes. One can control typical settings like the high voltage, tube current and focusing mode (from a drop-down list that lists the available modes for that particular tube), autocentering, warmup, etc. but also more advanced settings like manual adjustment of the focusing coils (if available on that particular tube of course). If a filter is used to modify the spectrum it can be selected from a drop-down list or be filled in. UGCT currently only operates continuous sources, not pulsed ones.

Camera control: this allows one to select one of the available cameras, to set exposure time, binning mode, a region of interest (ROI) etc. For cameras that do not support hardware binning or ROI modes these were implemented in the component software of that camera, so that from an operator standpoint any camera has these options. One can take single images, grab images constantly (with a

selectable interval), and one can record time lapse radiography sequences to disk. It is possible to operate the scanner with raw projection images, or to use an inline normalization using prerecorded flat field and/or dark field images. Lag or ghosting corrections are not applied because they proved not to be problematic in the typical circumstances at the UGCT scanners. When some lag is to be expected the exposure is made with multiple frame averaging to reduce the effect. It is possible to apply a spot filter during acquisition to suppress noise or defective pixels. This was implemented as a selective median filter with a user selectable threshold to choose the level of suppression. Based on the known geometry and camera settings a horizontal line is drawn on top of the image to precisely indicate the central plane of the setup. This is important in setups where one can interactively switch between source and/or detector. When a sample is larger than the field of view one can enable extended FOV. Each 'take image' command is then replaced by a small script (a series of detector movements and 'take image' commands) so that a bigger image can be assembled from the different positions. These are calculated automatically based on the set ROI, binning, magnification etc. This of course requires precise calibration of the setup. To minimize motor movements the shortest 'route' is always taken based on the current position of the detector and the central position around which to extend the FOV.

Motor control: this allows to move the different axes to position the sample, but also to control their speed and acceleration, to set the limits, etc. There are 4 buttons to move the rotation stage directly to 0°, 90°, 180° and 270°. The operator can fill in a desired magnification or voxel size, source detector or source object distance and the motor positions will be adjusted accordingly. Since the voxel size in a reconstructed volume differs slightly from the voxel size in radiography (due to the conical shape of the beam and the way the reconstruction software was implemented) the user gets to choose which of both he or she wants to set.

3.2 Scripting:

The scannerGUI supports a multitude of scanning scripts. A selection is described here:

The most common is a **standard CT** scan which is basically a sequence of recording a projection, saving it, and proceed to the next projection angle. This is repeated until the desired angular range is covered. Optionally one can choose to acquire a number of dark field images (detector offset and noise contributions) and flat fields (beam intensity profile and detector response), to include occasional autocentering of the tube for longer scans and to include occasional reference images to monitor possible movements in the scanner geometry due to thermal expansion or sample movement (only relevant in very high resolution applications). If desired, the script can explicitly include settings like exposure time, binning mode, ROI, output path, high voltage, current or even voxel size, although by default the script does not set these, but rather takes the active scanner configuration at the moment the script is started.

When faster scanning is required the step-and-shoot strategy is abandoned for a **continuous CT** acquisition. The full rotational movement is initiated and then images are grabbed at a preset rate. To exclude effects of the acceleration and deceleration of the rotation an overshoot angle can be set. In this case, the rotation starts at -10 degrees for example, and then acquisition starts when the rotational position crosses the starting angle (typically 0 degrees). A special command was therefore implemented which is included in the script: [scanner, wait until position >=, arg]. One possible issue with this type of scan is that, when using frame grabber based detectors (like many flat panels detectors), there can be occasional buffer refreshes during which no images are recorded, resulting in missing data at certain angles. Other issues can be harddisk transfer delays or temporary instabilities in the tube output. UGCT therefore adjusted its iterative reconstruction algorithm in Octopus to allow reconstruction of datasets with non-uniform angular sampling. This requires the angular position to be recorded with each projection so that it can be read during the reconstruction.

For scanners where a vertical sample axis is installed a **helical acquisition** script was implemented. Due to the large cone angle the standard CT scan exhibits strong so called 'cone beam artefacts'. This is a common problem in cone beam CT where the information of different slices is mixed in the vertical direction [9]. This is caused by the incomplete sampling of the Fourier space representation of the sample and especially critical near the top and bottom of the scanned region. Helical scanning requires an additional vertical movement or pitch with each rotational increment along a vertical axis, or more generally, along an axis parallel to the rotational axis. This acquisition sequence has the advantage that the resulting reconstruction does not suffer from cone beam artefacts. Figure 3 shows a comparison between a scan made with standard cone beam acquisition and with helical acquisition.

To allow radiography of objects that are too large for a given detector a **tiled radiography** script was implemented to record a large object by translating the detector vertically and/or horizontally to record a complete view. This of course requires the detector to be mounted on a vertical and/or horizontal axis. It also requires pixel-perfect alignment of the detector with the axes, as the sub-images are recorded without overlap. No registration procedures are used to fit the sub-images together, they are seamlessly assembled in one bigger image. Subsequently a **tiled tomography** script was implemented to allow tomography scanning of objects that are larger than the detector. This can be done in two ways. By default, the detector is positioned and all projection angles are recorded and this is repeated for all required detector positions (this can be more than two, and in both horizontal and/or vertical direction if the axes are available). When sample or setup movement become an issue (typically in high resolution applications) the detector is moved to the different positions for each angular position. This results in more motor movements and is thus slower, but it ensures that the different parts of each image align precisely. The scan shown in Figure 4 is the result of a standard tiled scan where the left half was scanned first, and the right one after moving the detector.

To enable batch scanning, multiple scripts can be prepared, even different types of scans in one batch, each with its own sample description, sample scanning position and data output path. This is typically used to scan multiple samples stacked on top of each other without operator intervention between scans. But it is also used for comparative studies where the same sample is for example first scanned using a standard cone beam acquisition and immediately after using a helical acquisition. Finally, scripts can be edited manually from the main program or loaded from text files that were saved earlier. When running a script, the progress and an estimated duration are shown to the operator.

3.3 Other features :

Essential parameters in a CT scanner setup include the source-object distance, source-detector distance, magnification, detector pixel pitch, voxel size in the scanned object etc. An **automated calibration** procedure was implemented to relate the source position, detector plane and sample position in space to the actual motor positions. A routine was written to image a strongly absorbing object at different magnifications. Using image processing routines the projected size is extracted. Using a fitting procedure this allows to derive the magnification, voxel size and other relevant parameters with great accuracy. These geometry parameters are critical for a proper reconstruction and are logged with each scan in a text file which is read by the reconstruction software Octopus. This way, minimal user input is required to obtain a high quality reconstruction.

In high resolution CT, especially below 1 micrometer resolutions, typical sample sizes are below 1 mm. To make optimal use of the detector area it is crucial that the sample is well centered on the rotational axis. A piezo XY stage is therefore mounted on top of the rotational axis. The sample is placed on this XY stage. To facilitate sample positioning an **automated sample centering** routine was implemented. The automatic centering routine records images of the sample at 0°, 90°, 180° and 270° and adjusts the

position of the sample until it is perfectly centered on top of the rotational axis, thus allowing for maximal magnification while keeping the desired volume of interest in the field of view at all time.

Finally, the scannerGUI supports different users with different user levels (supervisor, administrator, operator). When logging on, the interface gives access to only those features that correspond to the user level. The operator does not have access to advanced tube settings such as for example manual adjustment of the centering coils. An administrator has access to all functions of the program except the user management. Supervisors finally have access to all functionality, including user management.

4. Discussion:

4.1 Precautions:

When executing a series of commands, typically when running a scan script, the decoupling implies one should take explicit care of the execution flow. Commands are not necessarily executed in the same order they are broadcast, because of the inherent parallel nature of the sender-listener concept. Therefore a timing protocol was implemented in the template mentioned before. When a command is sent out, the receiver first acknowledges receiving the command (RCV). While executing feedback can be sent out (FDB), and when execution has finished, the command is acknowledged (ACK) to the outside world. This way one can choose to wait until a certain command is executed before proceeding to the next, or only to wait for acknowledgment from the component that it received the command. By default the wait flag is enabled. But there are cases where one does not want to wait until a command has finished execution, for example in the case of a continuous acquisition where the rotation is started and one proceeds with taking images as soon as the rotation motor confirmed it received the command to start rotating. This provides the necessary control over the precise execution flow. At this point it should be noted that different cameras behave differently in terms of timing. CCD cameras typically start exposing when an image is requested, and return the image after the exposure time plus a certain

readout time, whereas flat panel detectors typically use frame grabbers that take images from the sensor at a preset rate. When the software requests an image from a framegrabber, it either immediately returns the last image it received from the sensor, or in some cases it returns the first image that comes from the sensor after the request was made. In the latter case it can return anywhere between almost immediately and the time between successive frames. In both cases this means that the exposure could (partially) overlap with the execution of a previous command (typically rotation of the sample). Frame-grabber based cameras were therefore all implemented in such a way that the first frame can optionally be discarded in order to make sure that the frame that is returned to the scannerGUI is one that did not start exposure before the command to take an image was issued.

All status information of a scanner is stored in four status variables (scanner status, tube status, camera status and motors status). These are global variables, making them accessible by all components at any time. However, one should be aware that using global variables in LabView holds some risks as a global variable is not locked when accessed. This can lead to ill-defined behavior when reading and writing from different parts of the code at the same time. Precautions were therefore taken to ensure the correct execution flow. Only the scannerGUI is allowed to write into the globals. Components and add-on routines can only read from them, or send a command to the scannerGUI to update their content. When reading from the globals a small delay is first executed to ensure that the scannerGUI had time to process any commands recently issued so that the globals are up to date when being read from.

4.2 Benefits:

Decoupling the abstract and physical operations allows for **minimal duplication of coding** efforts. For example, the scripting engine only makes use of abstract commands. As a result, the same scripts can be executed on completely different implementations of a CT scanner without any modification to the software. The decoupling achieved by the sender-listener concept also ensures the **scalability** of the

programming. At this point more than 20 different components have been implemented. New components can be added **without any modification to the main program** or any other add-on routine that executes abstract operations. Deploying the software on a newly built scanner requires minimal effort. Of course, the functionality of new hardware components first has to be implemented within the template structure. Besides that, only an ini-file should be written listing the components that make up this particular scanner. These components will all be launched at the startup of the main program and the GUI will be configured accordingly, listing the available tubes, cameras and motors and their possible settings.

In one particular scanner at UGCT, parts of the setup are mounted on a gantry, and other parts are stationary. Therefore, the components making up the scanner are effectively controlled by two different PC's. The tube, detector and magnification axis are controlled by a PC which rotates along with them on the gantry, whereas the gantry motor and vertical axis are controlled by a stationary PC outside of the gantry. A bi-directional **TCP/IP tunnel** was therefore written which registers for all event categories. Any command that is broadcast on one side is tunneled through a (wireless) TCP/IP connection to be broadcasted on the other side too. This provides a completely transparent solution in which components can run behind a TCP/IP connection. No modification to the main program nor to any code of the components is required, thus preserving the scaleability and generic nature of the platform. This can also be advantageous when testing third party components that come with their own control PC, or to reduce processor load on the main machine.

For test and development purposes **dummy components** were implemented (a dummy tube, a dummy camera and dummy motors) that behave exactly as their physical counterparts. This facilitates the off-line development of new acquisition sequences or the adoption of add-on modules that interact with a scanner such as routines to control peripheral equipment based on the status of the scanner.

For example, to record a pressure reading each time an image is taken simply requires programming a structure that listens for camera events. When a 'camera take_image' command is acknowledged this will be heard and the pressure reading can be taken. No knowledge of the physical implementation of the scanner is needed, and as such the same piece of code can be used at different CT scanners.

Overhead can be reduced to the absolute minimum thanks to the inherent parallel nature of the separate components running in parallel. The GUI is **optimally responsive** because updates are processed as the feedback is returning from the separate components (no sequential polling loops are required since the events behave as interrupts). The intrinsic overhead of the concept of sending commands and receiving commands using user events is negligible. A typical scan script comprises a few thousand commands. The total overhead was measured to be of the order of seconds. Even components that run behind a TCP/IP connection send and receive commands with minimal overhead of the order of milliseconds per command.

A final benefit of this approach is that the acquisition sequence(s) one wants to execute are not embedded in the code of the main program but only determined by the text-based script that is generated by the script editor. Other acquisition schemes besides the already extensive list of available schemes can easily be implemented in the script generator, and the scripts can even be edited manually.

5. Applications:

Figure 3 shows a comparison of a standard cone beam CT acquisition versus a helical acquisition of a soda can with liquid inside. These scans were made on UGCT's nano-CT scanner using the microfocus source at 100kV with a voxel size of (89 micrometer)^3 and a magnification of 2,85 times. A total of 1000 projections were recorded covering 360 degrees. The helical scan had a pitch of 0,1 mm. The reconstruction was made with Octopus. The part shown in the images is the top part of the reconstructed volume. Notice how the surface of the liquid is blurred near the centre, and how the

features at the top ‘radiate’ outward along the cone angle under which they were recorded. The image on the right shows the result of a scan of the same area with a helical acquisition scheme [10]. The surface of the liquid is much better defined, and no streaking is seen.

Figure 3 : Comparison of a cone beam tomography (left) and a helical tomography (right).

Figure 4 shows a result of a tiled tomography scan where the projection images are assembled from two exposures taken at different horizontal positions to extend the field of view and thus scan objects wider than the detector area. In this particular case a 30 cm large cast of the blood vessels in a liver was scanned using a 20cm wide Varian flat panel detector. Because it is a large sample and thus a low resolution scan the left half was first acquired, then the sample was moved once, and subsequently the right half was acquired. The images were stitched without any registration procedure. For high resolution scans where one may expect long term sample shift due to spot shift or thermal expansion of the setup, there is also a scan mode where the detector is shifted for each projection angle. This ensures that the two parts of each projections align accurately but takes longer due to the many detector movements.

Figure 4 : Example of a result of a tiled tomography scan of a 30 cm large liver scanned with a 20cm wide detector.

6. Conclusions:

The current software platform provides all the functionality needed for operating a CT scanner for research purposes. Thanks to the generic implementation, the software can be deployed with minimal effort on any CT scanner, regardless of its actual physical implementation. Future developments will mainly focus on implementing other acquisition schemes, including dual energy approaches. Add-on

functionality will continue to be implemented to allow interaction with peripheral equipment such as pressure stages, climate control chambers, DAQ boards etc.

A separate tool is currently used to predict the recorded spectrum using Monte Carlo simulated spectra for the available tubes together with the spectral sensitivity of the chosen detector to estimate the transmission through a sample or to decide which is the most appropriate filter to be used to scan a certain sample. This will be integrated in the scannerGUI and take the active settings as input.

Finally, the Octopus SDK (software development kit) will be used to integrate reconstruction functionality into the scanning environment, so that fast preview reconstructions can be made by the operator in order to quickly assess samples before executing a detailed scan.

Figure captions:

Figure 1 : Schematic representation of the programming concept.

Figure 2 : The main user interface.

Figure 3 : Comparison of a cone beam tomography (left) and a helical tomography (right) of a soda can.

Figure 4 : Example of a result of a tiled tomography scan of a 30 cm large liver scanned with a 20cm wide detector.

References:

[1] Kak A.C. and Slaney M; Principles of Computerized Tomographic Imaging, *Society of Industrial and Applied Mathematics*, 2001.

[2] Vlassenbroeck J, Masschaele BC, Dierick M, et al.; Recent developments in the field of X-ray nano- and micro-CT at the Centre for X-ray Tomography of the Ghent University; *Microscopy And Microanalysis,* **13**, 184-185, 2007.

[3] Vlassenbroeck J, Dierick M, Masschaele B, et al.; Software tools for quantification of X-ray microtomography, *Nuclear Instruments & Methods In Physics Research Section A-Accelerators Spectrometers Detectors And Associated Equipment*, **580**, Issue: 1, 442-445, 2007

[4] Ionita, Ciprian N; Cone-beam micro-CT system based on LabVIEW software, *Journal of digital imaging* 21 (3), 296-305, 2008

[5] Harrison H. Barrett et al; Compact CT/SPECT Small-Animal Imaging System, *Nuclear Inst. and Methods in Physics Research*, A, 584 (1), 135-148, 2008

[6] G Cao et al, A dynamic micro-CT scanner based on a carbon nanotube field emission x-ray source, *Physics in Medicine and Biology*, 54 (8), 2323-2340, 2009

[7] C T Badea et al, In vivo small-animal imaging using micro-CT and digital subtraction angiography,*Physics in Medicine and Biology*, 53 (19), R319-R350, 2008

[8] A LabVIEW driver for X-ray flat-panel detector, Journal of X-Ray Science and Technology, 16 (4), 261-268, 2008

[9] Feldkamp L A, Davis L C and Kress J W; Practical cone-beam algorithm, *J. Opt. Soc. Am. A*, **A6,** 612–19, 1984

[10] Kudo H, Noo F and Defrise M; Cone-beam filtered-backprojection algorithm for truncated helical data, *Phys. Med. Biol.*, **43**, 2885–909, 1998

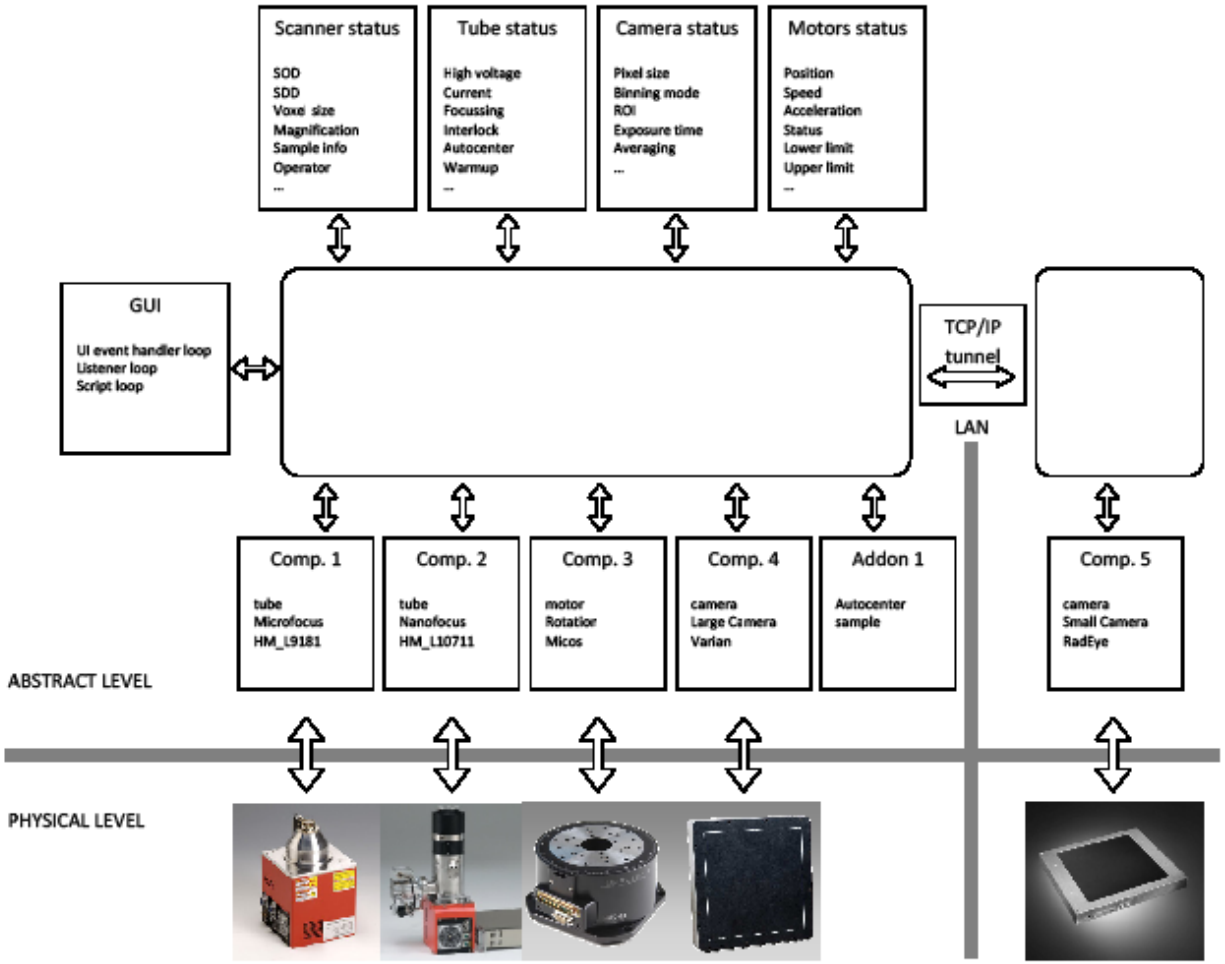

Figure 1 : Schematic representation of the programming concept.

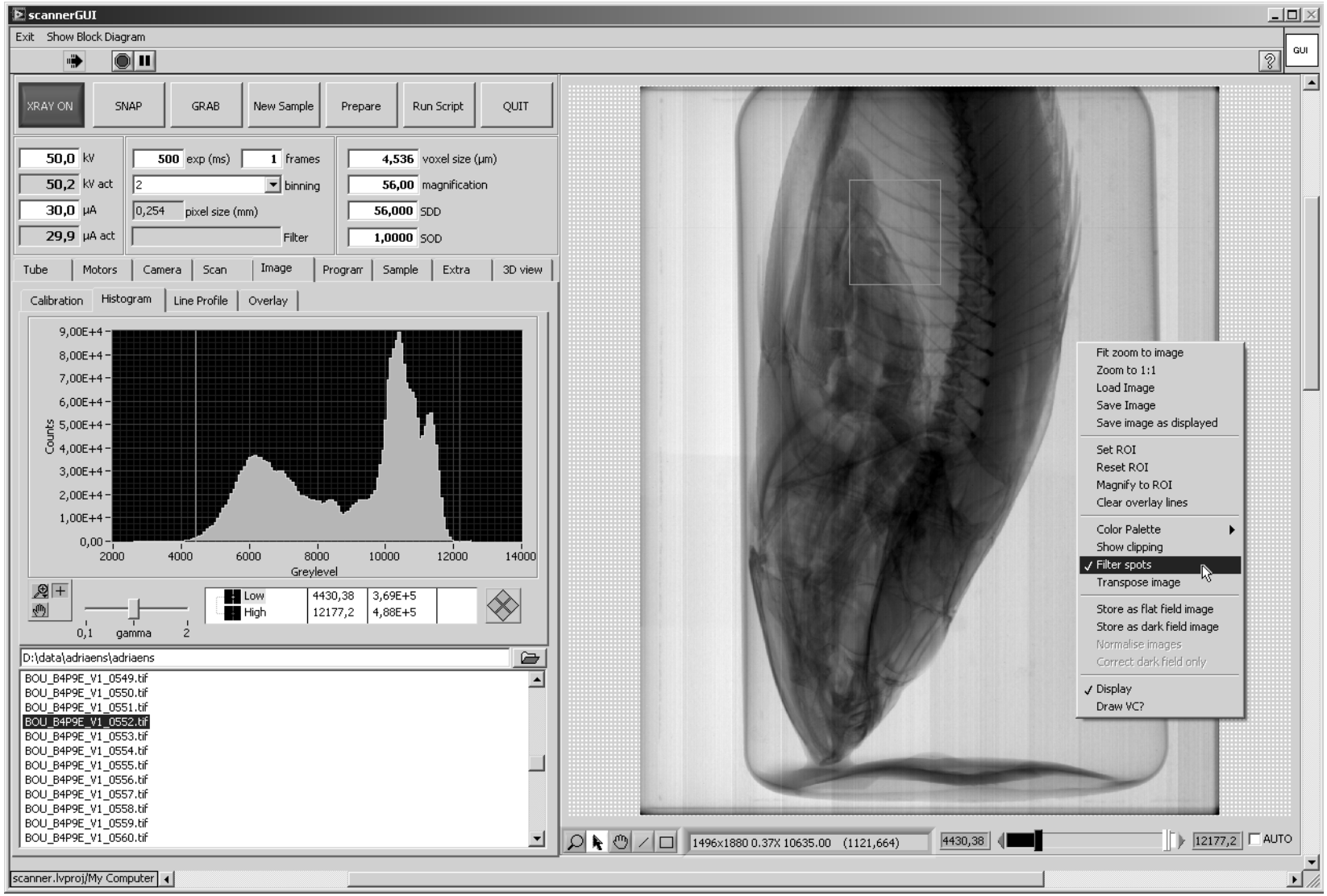

Figure 2 : The main user interface.

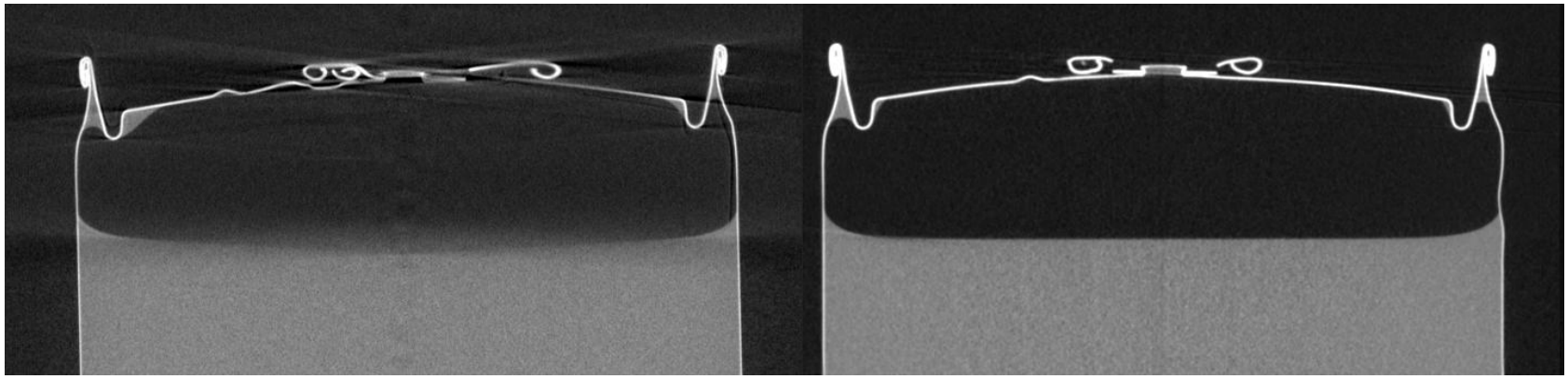

Figure 3 : Comparison of a cone beam tomography (left) and a helical tomography (right) of a soda can.

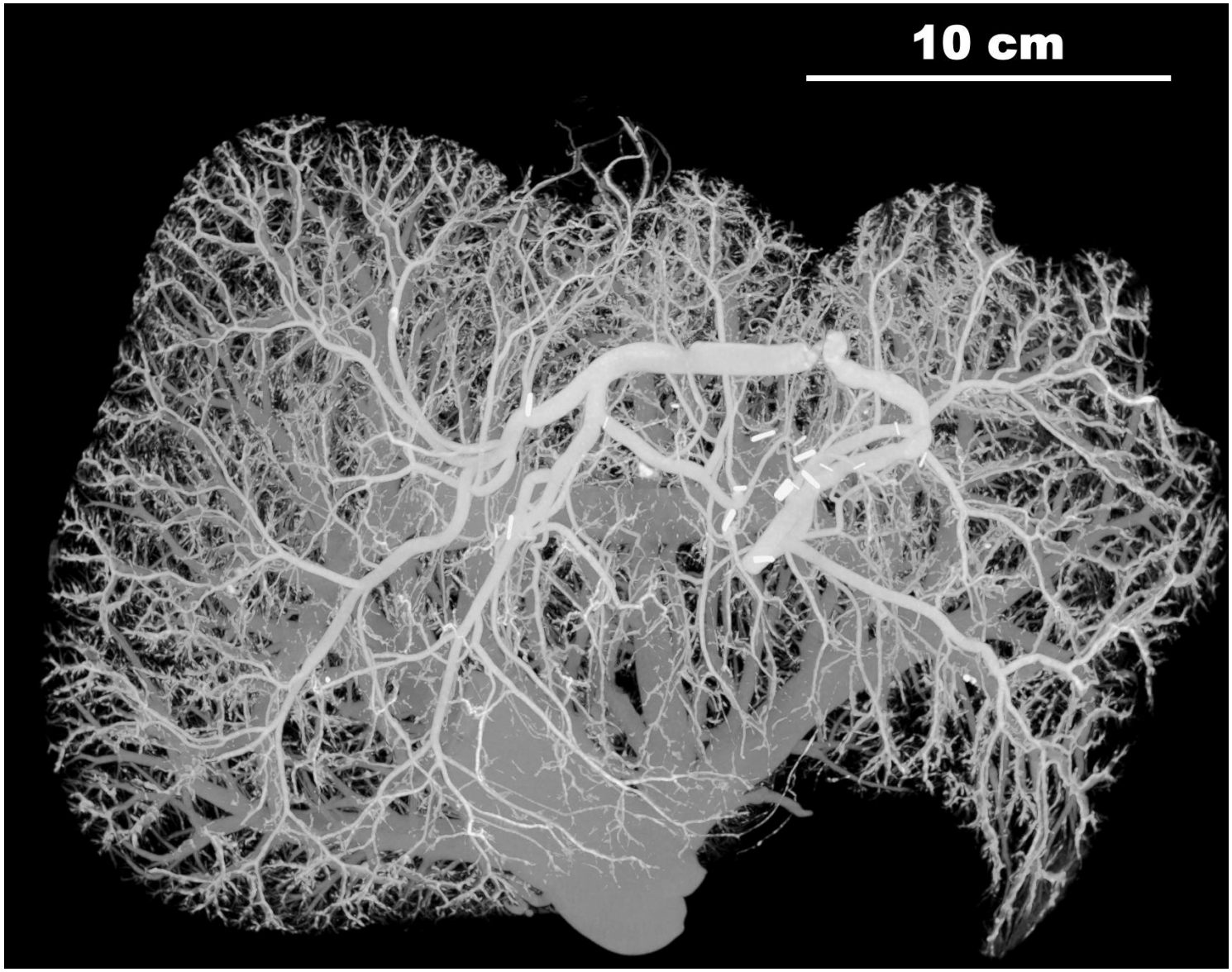

Figure 4 : Example of a tiled tomography scan of a 30 cm large liver cast scanned with a 20cm wide detector.