A MATLAB GUI for the analysis and reconstruction of signal and image data of a SAFT-based 3D Ultrasound Computer Tomograph

Torsten Hopp, Gregor F. Schwarzenberg, Michael Zapf, Nicole V. Ruiter
Institute of Data Processing and Electronics, Forschungszentrum Karlsruhe
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
e-mail: {torsten.hopp, gregor.schwarzenberg, michael.zapf, nicole.ruiter}@ipe.fzk.de

***Abstract*—At Forschungszentrum Karlsruhe, a new imaging system for early diagnosis of breast cancer is currently developed. The 3D Ultrasound Computer Tomograph (USCT) consists of approximately 2000 ultrasound transducers, which produce 3.5 million A-scans (amplitude scans) summing up to 20 GB of raw data for one image. The large number of A-scans, the large amount of data and the complex relationship between raw data and reconstructed image makes analysis, understanding and further development difficult for the scientists and especially for new employees and students. For this reason, an interactive graphical user interface (GUI) was developed using MATLAB. It integrates existent analysis methods and is easily extendable with new functionality via a plugin concept. The software provides several visualization functions for the raw data, the reconstructed 3D images, the USCT aperture and the relationships between them. This approaches demonstrate that MATLAB is not only applicable as programming language for numerical problems, but also adequate for representing complex systems by a GUI. It has a large benefit for the working group as it is used as common development platform: The plugin concept is widely used to integrate new analysis methods and share them with the rest of the scientists. The GUI and the visualization of the complex relationships of the USCT reduces the training period for new employees and students. An evaluation of the usability shows that the users evaluated the user interface to be very helpful, clearly arranged and beneficial for a better understanding of the coherencies of the USCT system.**

MATLAB; Graphical User Interface; Ultrasound Computer Tomography; Analysis software

I. Introduction

At Forschungszentrum Karlsruhe, a 3D Ultrasound Computer Tomograph (USCT) is currently developed [1][2]. The long term goal of the system under construction is the development of a new imaging modality for early breast cancer detection. It produces images in significant higher quality than conventional sonography. In contrast to conventional (X-ray) computer tomography and mammography there is no radiation exposure for the patient.

The system consists of approximately 2000 transducers, arranged in layers on a cylindrical tank, which is filled with water. The tank can be rotated in small steps. For a measurement each emitter emits an unfocused ultrasound pulse, while all other receivers record the ultrasound reflections caused by the objects located in the cylinder. This procedure creates up to 3.5 million A-scans (amplitude scans), which is equal to more than 20 GB of raw data.

In the first step, the raw data is read out from the data acquisition hardware and stored in a MATLAB data format [3] on a file server. In the second step a reconstruction method based on SAFT (synthetic aperture focusing technique) [4] and implemented in MATLAB is used to calculate 3D images based on the raw data. This workflow including the three levels (USCT aperture, raw data and 3D images) is shown in Figure 1. The relationship between raw data and reconstructed 3D images is not intuitive due to the large number of A-scans and the 3D sensor geometry. Every A-scan as well as the reconstructed 3D images correspond to 3D positions in the USCT aperture. For the scientists it is necessary to get a comprehensive overview of this large amount of A-scans and the whole system.

So far most analysis tools are implemented as command line scripts using several parameters. These scripts are difficult to use especially for new employees and students joining the group.

There are several libraries and software tools implementing only parts of the required functionality. ImageJ is a Java based image editing and analysis software including a powerful plugin concept [5]. For 2D and 3D visualization several commercial software systems are available as e.g., Volume Graphics Studio [6]. For signal processing, MATLAB and LabVIEW [7] are commonly used. However, the different software systems are standalone applications and not aimed to work together. An approach using MATLAB for the implementation of a GUI for analysis of ocean acoustic propagation is described in [8], only focusing on this specific topic. To the knowledge of the authors, no software system satisfies the requirements of the USCT project for an integrated software for the analysis and exploration of signal and image data.

For this reason a software tool was developed which integrates the different levels of the USCT workflow into one interactive GUI [1]. Because of constant changes and additions during the development process of the new imaging method, the interactive GUI has to be adaptable and extendable to new functionality. As the scientists implement the USCT algorithms in MATLAB, the implementation of the interactive GUI was also carried out in MATLAB, despite its limited GUI

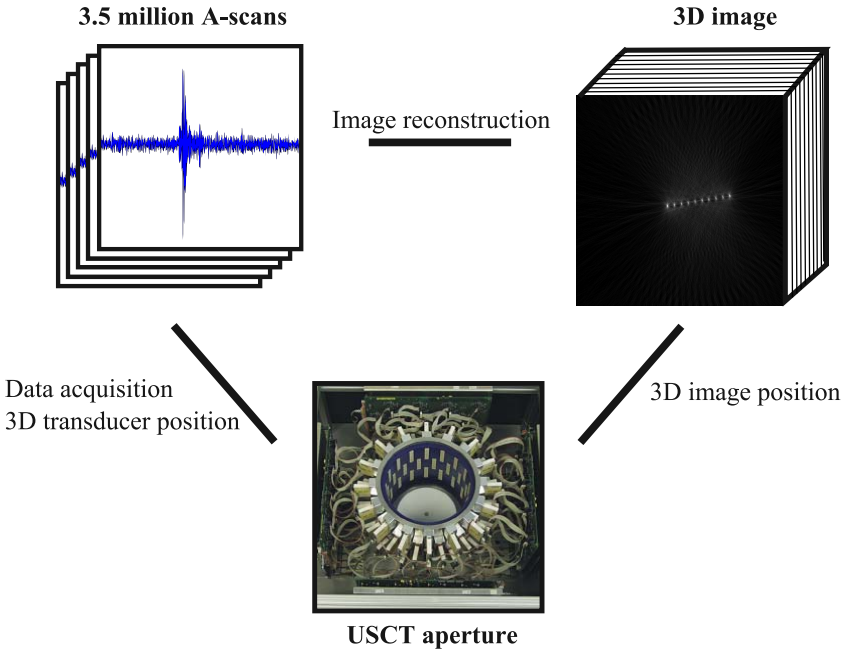

Figure 1. Workflow in the USCT: The USCT (bottom) produces A-scans (left), which are recorded by the data acquisition software. The reconstruction software creates 3D images (right) using the A-scans. The image corresponds to a 3D region in the USCT.

capabilities.

The aim of this paper is to give an outline of the design of the developed software and to show, how the limited GUI features of MATLAB can be used to create a complex interactive and adaptable GUI. In the following, an introduction to the ultrasound computer tomograph is given followed by an overview of the capabilities of the GUI elements that MATLAB can offer. After this, the design and structure of the developed software is described. Section IV details selected functionalities of the software. A user evaluation is shown in Section V and, finally, a conclusion is presented.

II. Basics

In this section, the basics on Ultrasound Computer Tomography are presented to motivate the need for a software system representing the workflow from raw data to reconstructed images. Furthermore, capabilities of MATLAB are shown, which form the framework used for the development of the software.

A. 3D Ultrasound Computer Tomography

Our first prototype for 3D Ultrasound Computer Tomography consists of a cylindrical tank with a diameter of 18 cm and a height of 15 cm. On the cylinder wall three layers of 16 Transducer Array Systems (TAS) [9] are mounted, each consisting of eight ultrasound emitters and 32 receivers resulting in 384 emitters and 1536 receivers (Figure 2). To achieve an uniform distribution of emitters and receivers, the cylinder can be rotated to six positions using a stepping motor. Applying six motor positions, the system consists of a virtual number of 2304 emitters and 9216 receivers, creating the 3.5 million A-scans. The emitters are virtually numbered in layers from 0 to 23 and elements from 0 to 95. The receivers are virtually numbered in layers from 0 to 47 and elements from 0 to 191.

A measurement is done by emitting an approximately spherically ultrasound pulse front (center frequency of 2.7 MHz) into the tank, which is filled with water as coupling medium. The ultrasound is absorbed, scattered and reflected, depending on the objects within the tank. Simultaneously, all receivers of the system record the amplitudes of the transmitted and reflected pulses. Afterwards the next emitter sends an ultrasound pulse while all other transducers receive. The data acquisition hardware of the system digitalizes at a sampling rate of 10 MHz, which results in A-scans of 3000 samples (each 300 μs). The complete procedure for all emitter-receiver-combinations or a subset of them is controlled by a data acquisition software (Andromeda) based on Java, which reads out the hardware memory via native libraries. The data is stored on a file server in a MATLAB data format. Additionally to the raw data, information about the measurement parameters (temperature, emitted pulse etc.) is stored in separate files.

The A-scans contain the transmitted pulse from emitter i at position $\vec{e}_i$ to receiver j at position $\vec{r}_j$ as well as the reflections from the object (Figure 3). With the information about the time the ultrasound needed from the emitter to the object and

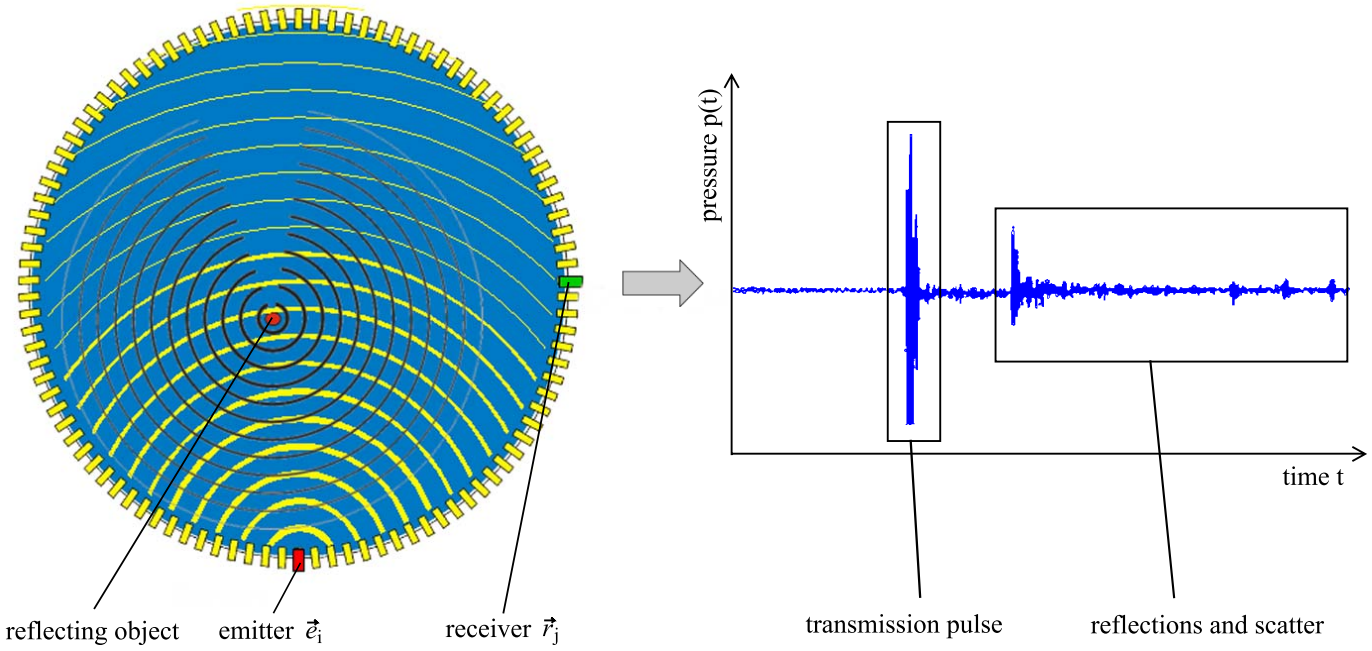

Figure 3. Dependence of A-scans from sensor geometry. Left: Principle of the data acquisition for USCT simplified in 2D. The emitter emits an ultrasound pulse, which is reflected by the object. Here an idealized point scatterer scatters the ultrasound in all directions. The reflected and transmitted signals are recorded by all the receivers. Right: A-scan recorded at green receiver as pressure over time. The first signal is the transmission pulse (directly transmitted ultrasound). The successive signals are reflections and scatter from the imaged object and the cylinder.

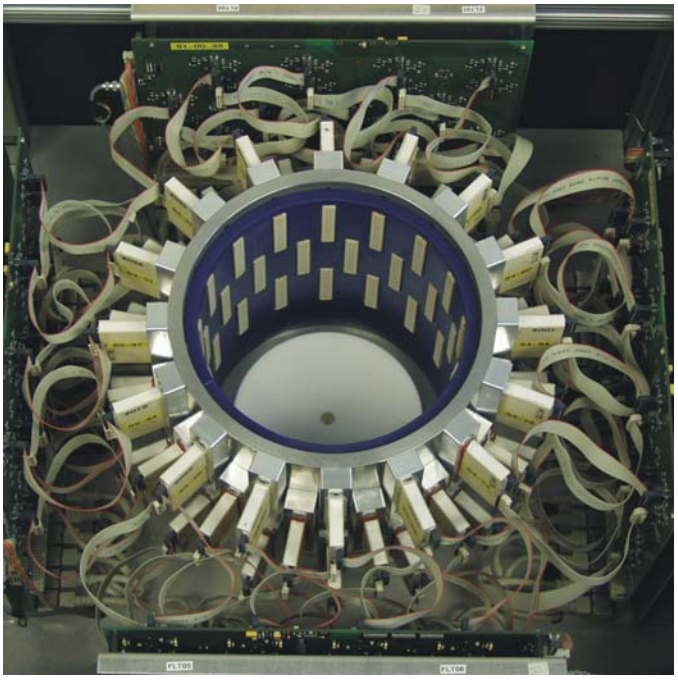

Figure 2. The prototype of a 3D Ultrasound Computer Tomograph: water tank (middle) equipped with 48 TAS which are connected by flat cables to four distribution boards (edges of the picture).

from the object to the receiver, a spheroidal shell with emitter and receiver as focal points representing all possible positions of the scattering object can be calculated. This assumes a constant speed of sound v throughout the water. The volume of interest $I(x)$ at the position $\vec{x}$ is then reconstructed by summing over each projection of each sample of an A-scan $A_{i,j}$ with its specific amplitude. Repeating this for all $m \cdot n$ emitter-receiver-combinations – hence for all A-scans – high values are summed up at points that are reflecting ultrasound at a high level (Figure 4, equation 1). This method is known amongst others as SAFT [4] and can be written as:

$$I(\vec{x}) = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{i,j}(t),\ t = \left(\frac{\|\vec{e}_i - \vec{x}\| + \|\vec{r}_j - \vec{x}\|}{v} \right) \quad (1)$$

The image reconstruction software is subjected to constant development. The code is mainly written in MATLAB. The kernel of the backprojection was implemented in assembler for speed up [10]. Depending on the resolution the reconstruction of a complete volume can take from a few hours up to several weeks on a standard PC. The reconstruction of two dimensional slice images takes some seconds to a few minutes.

Currently a second prototype of the 3D Ultrasound Computer Tomograph is being developed. The sensor distribution was optimized to increase the illumination level as well as the contrast and theoretical resolution [11]. While the first prototype consists of a cylindrical water tank, the second generation changes the shape to a half ellipsoid. In the course of this new development new Transducer Array Systems are built and the second generation of the acquisition hardware promises first clinical experiments.

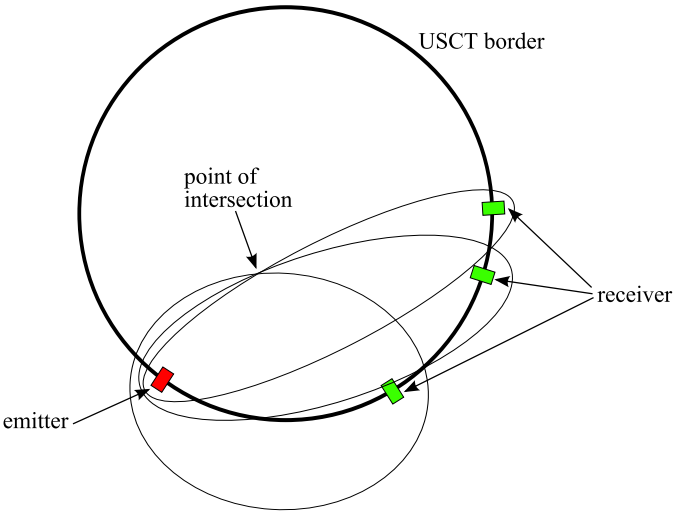

Figure 4. Principle of the SAFT. Ellipses for three emitter-receiver-combinations intersect in one point describing the position of the object.

B. MATLAB capabilities

MATLAB (the abbreviation stands for 'MATrix LABoratory') is a proprietary mathematical software developed by 'The Mathworks Inc.'. It provides a wide functionality for numerical calculations and allows the user to work with vectors and matrices in a straightforward way. The kernel of MATLAB is extendable with a large selection of toolboxes for different application domains. An overview of the available toolboxes can be found on the MATLAB website [12]. The main part of MATLAB interpreter, which interprets commands typed by the user. The commands derive from MATLAB's own programming language called M-Code. For rapid prototyping in signal and image processing, as necessary for improvement of the USCT reconstruction code, MATLAB provides a very handy platform due to the availability of many functions and toolboxes.

The MATLAB GUI functionality is not often used in scientific programming. Although the GUI functionality is not as powerful as e.g., the GUI functionality of Java (e.g., tabs are missing), it was the method of choice for this application, as the existing script software could easily be integrated. A complete reference for all interaction objects provided by MATLAB can be found in [13].

In MATLAB 6 only ten interaction objects were available to build GUIs:

- Textfields
- Edit Boxes
- Frames
- Pushbuttons
- Togglebuttons
- Checkboxes
- Radiobuttons
- Popup Menus
- List Boxes
- Sliders

Beside these interaction objects, MATLAB offers the powerful axes object used for data visualization. It can be extended by numerous functions for user interaction. Moreover every MATLAB GUI object can be appended by any desired data structure, which simplifies object-based interaction.

MATLAB offers several features for creating an interactive GUI, which kept the effort in development low. The extendability is based on the fact that MATLAB is an interpreter language. By just adding folders to the MATLAB path it is possible to access new functionality, even at runtime. For data visualization MATLAB uses a so-called *axes* object, which is able to visualize arbitrary 2D and 3D data. It offers the possibility to define click functions and context menus for all displayed graphical elements even in 3D. Moreover any form of data can be assigned individually to each graphics object. This simplifies object-based interaction and allows the easy design of interactive GUIs of any kind.

III. Software structure and design

In the following, the structure and design of the developed software is presented. Starting with the basic structure, the partition into three main parts is described. This is followed by presentation of the data mapping functions and the extendability of the software.

A. Basic structure

Due to the limited availability of MATLAB 7.x by the time of the development, MATLAB 6.1 was chosen as version for implementation. The downward compatibility and version checking of MATLAB ensures the usability of the interactive GUI with MATLAB 7.x.

To represent the imaging and image reconstruction in the USCT as concise as possible the functional requirements were divided into three parts:

1) *A-scan GUI*: contains all functions working directly on the A-scans.
2) *Images GUI*: contains all functionality for the reconstructed 3D images.
3) *USCT GUI*: contains the functionality for the representation of the USCT emitters and receivers for visualization purposes.

A main purpose of this representation in three parts is to show clearly what is done with the raw data when it is taken to reconstruct an image, to show how an image is built up by the raw data and where images and data sources are located in the USCT. The functionality covers recurrent procedures used to explore and analyze the experiments with the USCT.

The structure of the GUI for the system was designed to consider these requirements: the main GUI consists of three separate windows (MATLAB figures) related to the three parts of the functional requirements (Figure 5) This has the advantage that every part can also be used standalone. The challenge posed by the use of multi-figure GUIs is that they are complex to develop in MATLAB.

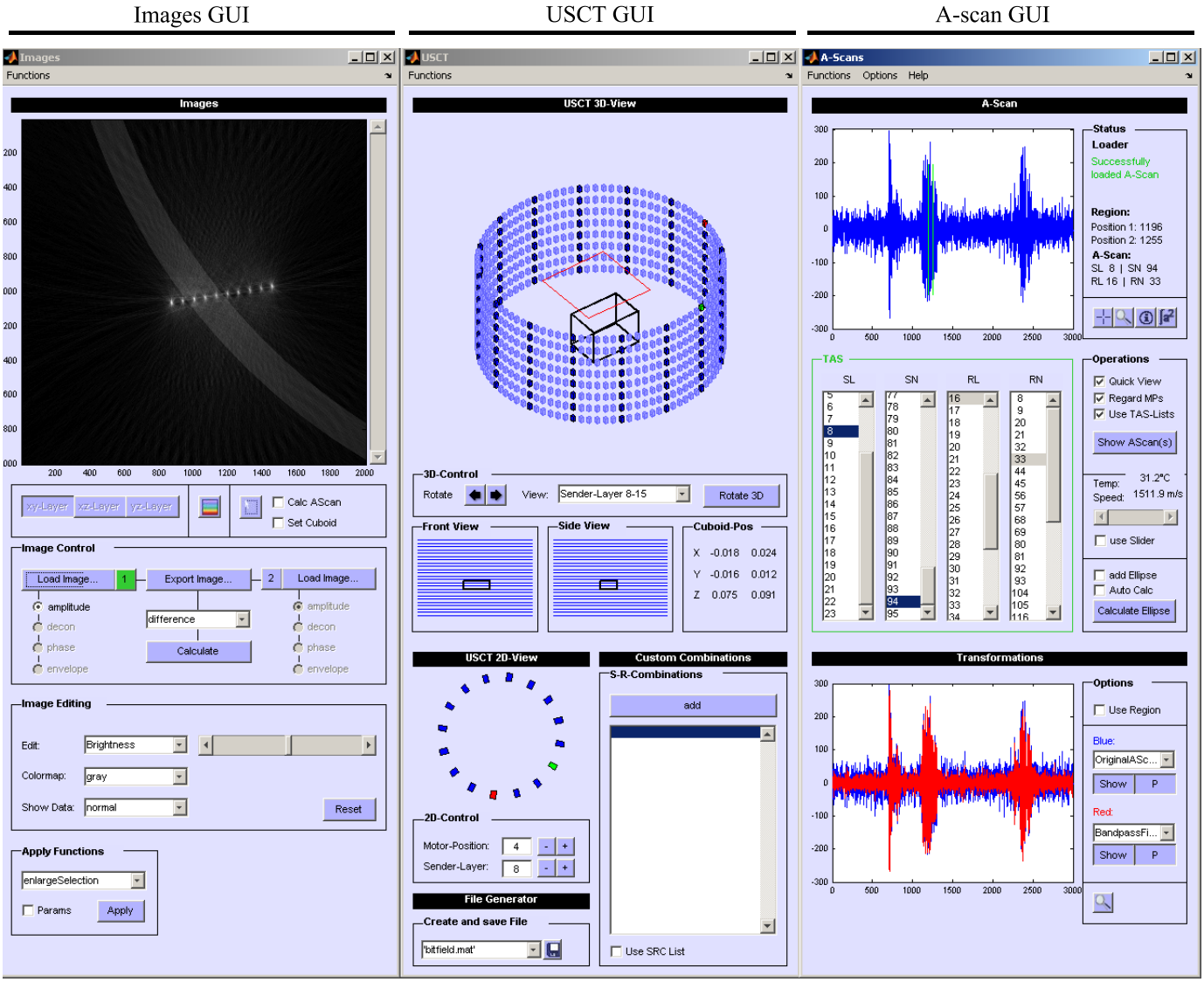

Figure 5. Graphical user interface with three separate windows: images related function at the left (*Images GUI*), USCT related functions in the middle (*USCT GUI*) and raw data related functions at the right (*A-scan GUI*).

B. A-scan GUI

The aim of the *A-scan GUI* is to give a quick overview of one or more A-scans from USCT experiments by selecting certain emitter-receiver-combinations. This is useful for quality control, signal-to-noise ratio and time-of-arrival analysis. This functionality is provided by four listboxes used to choose one or more emitter layers, emitter elements, receiver layers and receiver elements according to the USCT specification. Dependent on the number of selected A-scans the software plots the data to an *axes* object in the GUI or opens a scrollable dialog showing multiple plots for direct comparison (Figure 6). The scrollable dialog was done using a MATLAB script by Bjorn Gustavsson [14]. An additional functionality is to apply different signal processing functions to the A-scans without typing MATLAB commands. For this a second *axes* object is located at the lower half of the GUI. The user can choose from a long extendable list of transformations in a popup menu which are then applied to the currently loaded A-scan or a marked region of it. It is also possible to overlay a second graph for comparison of the original A-scan and transformed data. Measurement information that formerly had to be read out of separate files is now presented in a structured dialog reducing the effort of gathering information to a minimum of one click.

C. Images GUI

The *Images GUI* deals with large volume datasets. To give the user an overview, appropriate display functions are necessary. Therefore the *Images GUI* shows slices of 3D images in an *axes* object. The orientation of the slice can be chosen by the user as well as the slice number. A slider allows to browse through the volume. For comparison of images, which were reconstructed with different parameters, the user

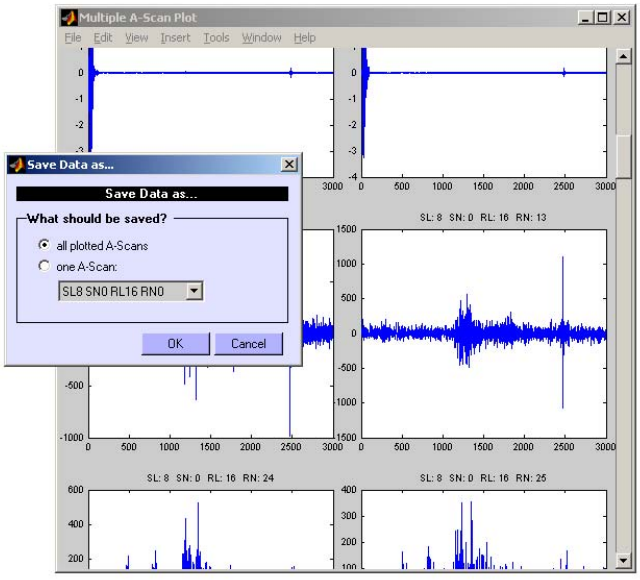

Figure 6. Scrollable dialog showing multiple plots of A-Scans for comparison.

can swap between two images and apply aggregation functions like calculating the difference of two images. This procedure is important for the analysis of different parameters for the reconstruction algorithm. To improve the appearance of the images, image processing functions are available. The images can be exported to the several graphical formats offered by MATLAB.

D. USCT GUI

A 3D model of the USCT aperture, i.e. emitter and receiver elements, is the main control element in the *USCT GUI*. It displays the connection between the *Images GUI* and the *A-scan GUI*, the interactive and color-coded visualization of selected transducer elements, and the location of currently loaded images. The model can be displayed in complete or in three parts according to the three TAS layers of the USCT. It can be rotated stepwise or freehand using the mouse.

Because of the usage of a parameter file the model can be adapted to all kinds of aperture geometries by simply exchanging this file. For the second generation of our 3D USCT prototype this will be of great advantage. For the the new developed geometry, a MATLAB file containing the cartesian transducer positions has to be built and copied into the specified folder of the USCT GUI.

Each element of the model represents an emitter and four receivers surrounding the emitter. By right-clicking an element a context menu offers functionality to mark the elements in different patterns as emitters or receivers. The left mouse button is programmed by the last chosen marking pattern and can be applied to other elements. Hence, the software is adaptive to click behavior of the user. The use of marking patterns and the programmable mouse button simplifies user interaction with the 384 elements shown in the model. An additional 2D model cutting out layers of the USCT also provides functionality for the marking of elements in a more precise way. The file generator uses the current marking pattern to create parameter files for the data acquisition and reconstruction software.

The markings in 3D are more intuitive for the scientists. They can be used to select a subset of A-scans in order to visualize them or calculate statistics.

E. Data mapping functions

For better understanding of the A-scan to image relationship, the system maps a region in a selected A-scan in the *A-scan GUI* to a currently loaded reconstructed image in the *Images GUI*. For the inverse process a region in the image can be calculated back to a time duration in the A-scan. Marked regions in an image are shown as a rectangle in the 3D model of the USCT giving the user an impression of the position of the currently shown slice image. The colored transducer elements in the *USCT GUI* and the illustration of the image position supports the understanding of the complex coherences. The data mapping functions allow the user a comfortable and easy to use in-depth analysis of the image formation process.

F. Extendability

The second fundamental purpose of the system was to provide extendability and adaptability. The interactive GUI serves as basis for the frequently used functions, which can be extended by new functionalities. Because of the ongoing development process modifications in all parts of the USCT workflow have to be introduced. New functions should be integrated into the system without major effort. Therefore a plugin concept was designed which allows the user to add new functionality by copying the source code – regarding the interface constraints – in defined folders. At startup the software checks the folders for new files and integrates them into pop-up menus or menus of the GUI for immediate access. The system offers interfaces for function integration. The interfaces can be grouped into:

- Standalone plugins without parameters and no return values. These plugins are used to integrate all kinds of tools into the software, as e.g., a standalone volume visualization tool . They are presented in the menu bar of each of the three windows. The GUI has to be designed by the user.
- Plugins using the currently loaded data of the software. They have input parameters and return defined data structures. Additionally to these defined parameters the programmer can add its own parameters for which a dynamically generated input dialog is created when calling the function. For these plugins the user does not need to program a GUI as it is generated automatically. This type of plugin is used for transformation functions on A-scans and for transformation and aggregation functions on images.

At program start the three parts of the GUI are arranged on the screen dependent on the adjusted screen resolution and all components including the computationally intensive 3D model are loaded in background while a status bar is displayed.

IV. Implementation of selected functions with respect to the MATLAB's GUI capabilities

Based on the basic overview on the software structure shown in the last section, the implementation of selected functions is presented in detail below in respect to MATLAB's GUI capabilities.

A. Main GUI

As described in the last section, the interactive GUI consists of three main windows which are realized as three MATLAB figures. The interaction between these independent figures is realized using MATLAB's global variables, which are available to all functions within a MATLAB session. At startup MATLAB stores all references to GUI objects in a specific variable. By declaring a global variable holding a copy of the references to GUI objects of the figure, the GUI objects can be accessed from every function or subfunction. To keep track of the numerous references every MATLAB figure (*A-scan GUI*, *Images GUI* and *USCT GUI*) creates its own global variable.

B. 3D USCT Model

A central interaction object of the system is the 3D model of the USCT. It is based on an USCT emitter geometry file which holds the position of every ultrasound emitter in cartesian coordinates. By plotting a voxel (created from six patches) at each position into the *axes* object the model is built up at startup of the software (Figure 7). This is done by use of the voxel function by Suresh Joel [15].

To restrict the number of voxels, one emitter and four receivers are represented as single voxel and each can be addressed by the corresponding emitter layer, emitter number and the according receiver layers and receiver numbers. These numbers are stored in each voxel using MATLAB's *UserData* variable, which is part of every GUI object. Within the assigned context menu of each voxel the user can select different options, e.g., setting the element as current emitter or receiver, setting the complete row or column as emitter or receiver and deselecting one, more or all elements. Depending on the users current choice the called routine reads its *UserData* variable via the voxel handle and sets the color of the voxel to red (emitter), green (receiver) or yellow (emitter and receiver). If the user sets an element as receiver an additional dialog is opened offering the selection of one to four of the receivers surrounding the selected element. The coloring routines are also called when selecting emitters or receivers in the *A-scan GUI*.

The name of the last callback routine is saved internally. By pressing the left mouse button which is pointing to a voxel this routine is called again. This results in an adaptive behavior of the left mouse button, which allows a fast marking of elements and avoids the repetitive use of the context menu. The 3D model can be rotated using the MATLAB rotating function for *axes* objects.

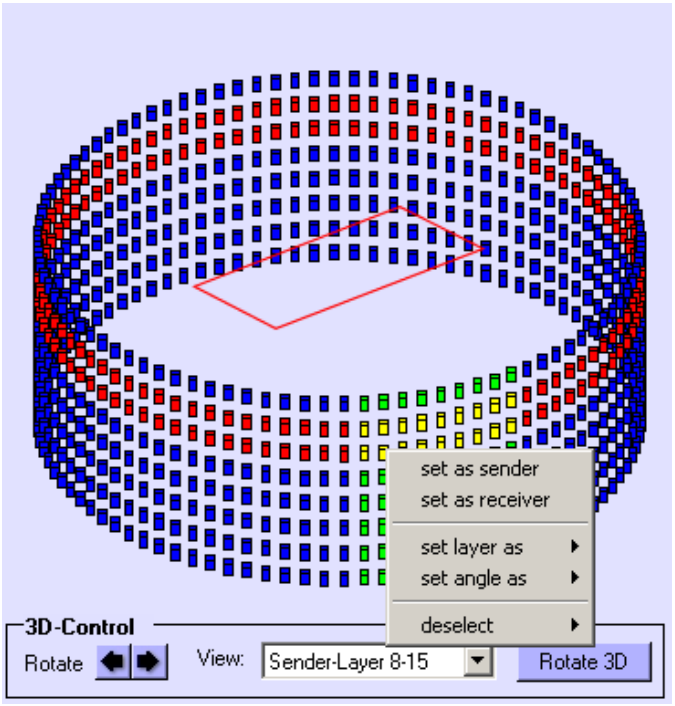

Figure 7. Interactive 3D model of the USCT with elements (blue) marked as emitter (red), receiver (green), emitter and receiver (yellow), the context menu called by clicking an element with the right mouse button and a red rectangle showing the position of a reconstructed image loaded in the *Images GUI*.

C. Data-image relationship

The relationship between raw data and reconstructed images is complex. For an in-depth analysis of reconstructed 3D images a method was needed to identify regions of one or more A-scans, which contribute to a specific part to the image. Also it is of interest which region of the image is concerned when selecting an area in an A-scan (e.g., a reflection). Therefore the data-image relationship can be visualized in both ways by functions marking a region in an A-scan by two points and in an image by a rectangle. This enables the user a detailed analysis of artifacts in the image or the spatial contribution of specific ultrasound pulses to the image.

Using MATLAB's *ginput* function two points restricting an A-scan to a relevant region can be chosen. The corresponding region in the currently displayed image slice is calculated by a routine based on the kernel of the reconstruction software. A slice image of a spheroidal hull, i.e. an ellipse, is calculated which is drawn into the reconstructed image. This can be done in two ways (Figure 8): The first method scales the pixel values of the calculated slice image to the colormap of the shown image and sums up both images. The second method creates an image by setting pixels within the ellipse to an opacity of 50 % and all other pixels to an opacity of 100 %. Moreover, these ellipses can be summed up, which demonstrates the process of the reconstruction for a number of A-scans.

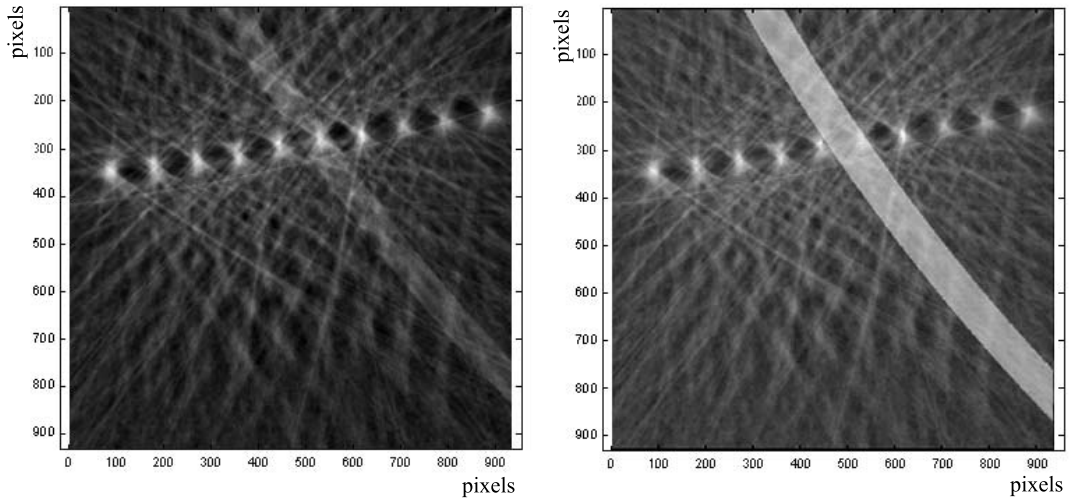

Figure 8. Slice of a reconstructed volume showing ten nylon threads with a diameter of 0.15 mm and spacing of 2 mm. Ellipses calculated by the system are drawn into the image - on the left by summing up the original and the ellipse image, on the right by using semi-transparent overlay.

Vice versa it can be of interest to know about the relationship of points or regions in the reconstructed images to A-scans. The system provides functionality to map an area R in the reconstructed image to a time duration in a chosen A-scan. For calculation of the two sample points t_{lower} and t_{upper} in the A-scan, which limit this region, the minimum and maximum distance from the chosen emitter to the chosen receiver via the object is calculated by the following terms regarding equation (1):

$$t_{lower} = \arg\min_{\vec{x}}(t), \vec{x} \in R \tag{2}$$

$$t_{upper} = \arg\max_{\vec{x}}(t), \vec{x} \in R \tag{3}$$

The resulting two sample points are drawn into the A-scan.

The calculation of the contributing part in an A-scan enables additional analysis functions such as the calculation of the N most contributing A-scans to a selected area in the reconstructed images. The routine searches for the highest amplitude in the computed region in each A-scan of a chosen subset and sorts them descending from the highest to the lowest peak. The resulting list is shown in a separate listfield, the emitters and receivers are marked in the USCT model. A-scans can be selected and chosen for further processing by A-scan analysis functions.

D. Extendability and plugins

The software has several interfaces for the extension with new functionality. When the software starts up the available functions are searched in predefined subfolders and entries are automatically generated in the according menus and pop-up menus of the GUI.

The plugins implementing the interface for standalone plugins can be called by accessing the menu entry in the particular figure. The software calls the specified function by the filename of the MATLAB source code file (.m-file). Information about the label of the plugin is stored in an identically named MATLAB data file (.mat-file) in the same folder.

The interfaces for non-standalone plugins have predefined input parameters and return a predefined data structure – e.g., an A-scan as input and return value, which is represented as an array of double values. If a function interface has more than these predefined parameters, an optional parameter can be passed as MATLAB struct which can contain arbitrary subvariables of different data types. The variable used as parameter is saved in a .mat-file identically named to the .m-file containing the function. When calling the function, the .mat-file reads its variable. Using the names of the subvariables within the struct and the default values defined for them, a dialog is created. The user can change these values which are then saved to the .mat-file (Figure 9). For better understanding of the parameters the .mat-file may contain an additional comment for each parameter which is shown to the user as tooltip. After saving modified parameters to the .mat-file the function is executed. The interface passes all parameters to it and passes back the result for visualization after execution.

This approach enables the encapsulation of any functionality by just two files where the .m-file contains the source code and the .mat-file holds the information for the GUI, the number of parameters, their default values and parameter descriptions. Due to the fact that MATLAB is an interpreter the .m-files can be easily accessed via adding them to the MATLAB path. It is possible to pass any data structure to the function, however, the dynamically created GUI can only process single numerical values so far.

The plugins implementing the interfaces can access all available data (e.g., the currently selected emitters and receivers)

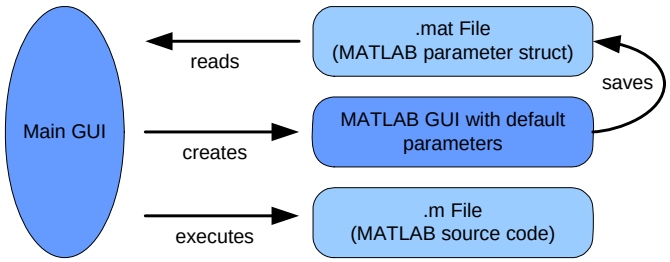

Figure 9. Files and dialogs used for the non-standalone plugins.

via a set of setter- and getter-functions, which encapsulate the access to the global variables and return the specific values.

E. Integration of the reconstruction software

Before the integration process, the reconstruction software was used as command line based tool and based on a simple text-file defining the various parameters for image reconstruction. In course of the integration, graphical user interfaces were developed giving the user a consistent view to all USCT software systems.

Therefore, the reconstruction software was integrated into the user interface by setting up a plugin calling the main function (*Image reconstruction GUI*). An additionally implemented GUI element in the *USCT GUI* offers selection of regions to be reconstructed (Figure 10). This was done using two axes objects for the representation of the emitter layers shown from two directions. Dragable lines refer to a cuboid drawn into the 3D USCT model shown in Section IV-B. The user can now select a region in 3D to retrieve the according cartesian coordinates of the edges, which will be used for definition of the region of interest to be reconstructed. They are automatically saved in the data structure of the reconstruction software, which defines all relevant parameters used for image reconstruction.

For changing parameters there is a dynamically generated GUI, the *Parameter GUI*, which reads out the parameter file of the reconstruction software. By specified types of input, it chooses interaction elements and arranges them in a MATLAB figure.

For a better overview, the inputs can be placed in groups according to their hierarchy level, which can be expanded and retracted by the use of push buttons. Since MATLAB does not offer a specific interaction element to manage such hierarchical structures, the rearrangement of the interaction elements, when expanding or retracting the parameter groups, has to be done via displacements of them. This is carried out by accessing the handles of following interaction elements in the parameter list and reset the position in the MATLAB figure window.

Another benefit of the *Parameter GUI* is an automatic validity check of the parameters. Also an effort was made to ensure downward compatibility for users working with the command line based tool. Therefore function wrappers were developed encapsulating the new functionality. Parsers had to be developed to read out the parameter text file and format it

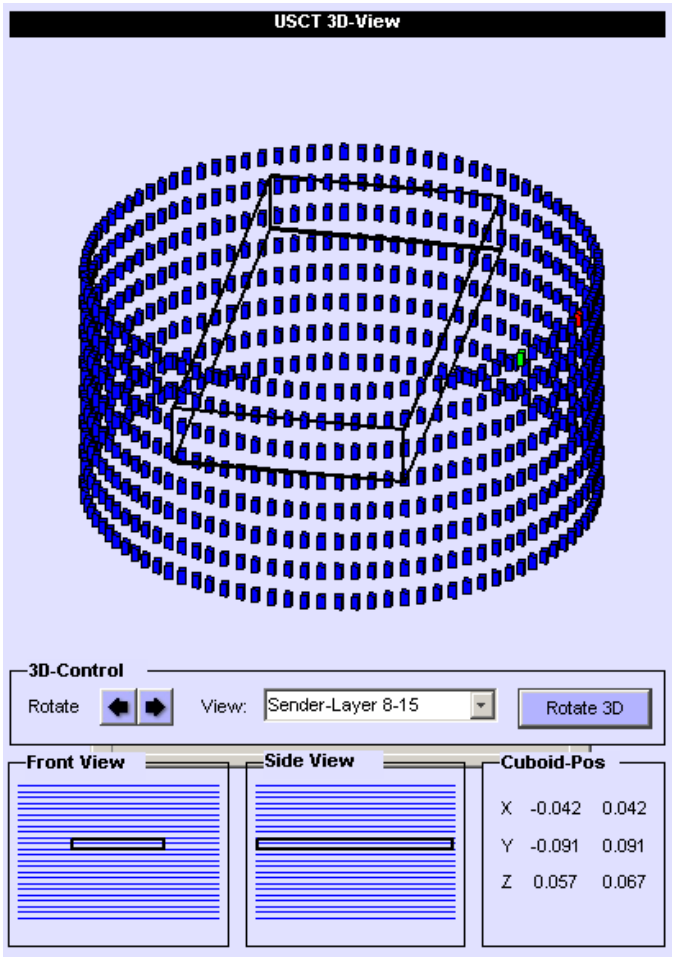

Figure 10. Selection of three dimensional regions to be reconstructed by the reconstruction software.

to an internal data structure used for the GUI generation. To stay consistent, values changed in the GUI have to be admitted to the text-based files by the use of text pattern matching.

Because of the dynamic generation of the GUI, it remains highly adaptable to changes in the reconstruction software.

Each parameter is stated by an importance value. When stating the category of the current user (beginners, intermediate and advanced) at program call, the importance value is applied to enable the appropriate parameters for the current user. E.g. beginners using the reconstruction software only get access to the most important parameters, while the rest of them are set by default values. This simplifies the complexity of the reconstruction for students and collaboration partners, which are just starting to use the reconstruction software and do their first image reconstructions. For experienced users all available parameters are shown. The visibility level can be set in the *Image reconstruction GUI* before calling the *Parameter GUI*.

V. Evaluation of Usability

An evaluation of the GUI was done using a usability questionnaire consisting of three parts. First, the user has to evaluate his know-how about ultrasound computer tomography

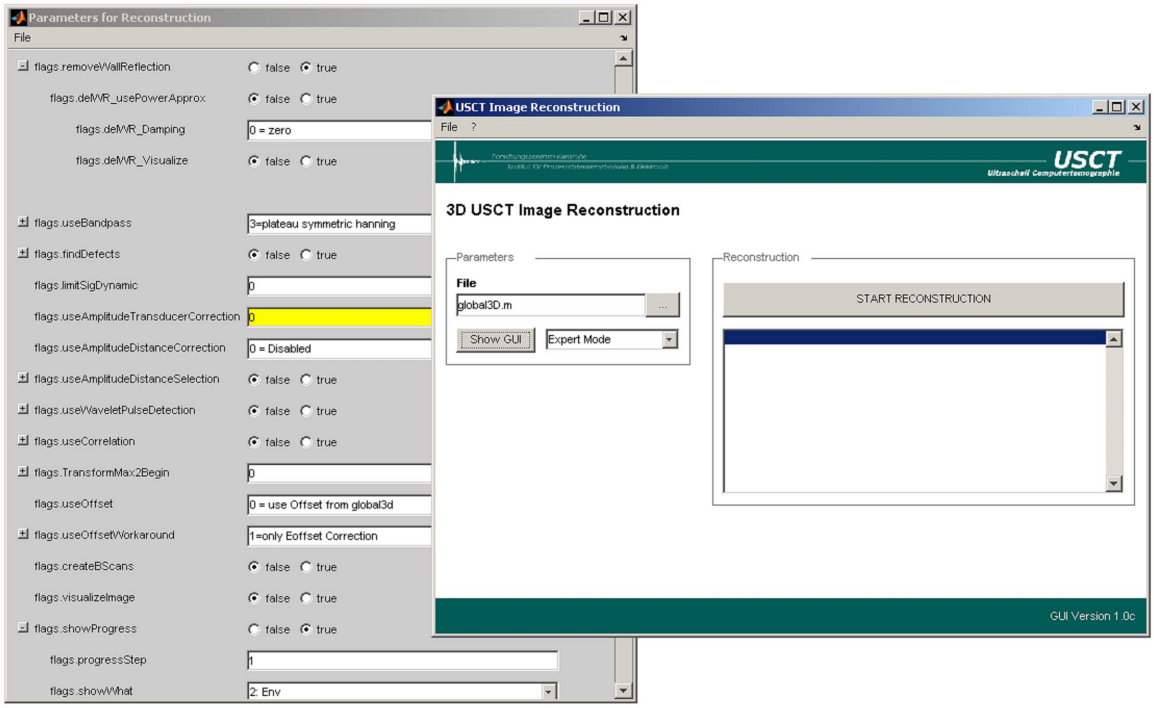

Figure 11. Dynamically generated GUI for changes in the parameter file of the reconstruction software.

and his capability level in software development. The second part investigates the appearance and functionality of the presented analysis software as well as the usability of specific functions. The third part of the questionnaire is based on the Software Usability Measurement Inventory (SUMI) [16]. The questions focus on the evaluation of efficiency, affect, helpfulness, control and learnability of the software. Until now, the questionnaire was completed by five scientists of the working group.

The results of the first part shows that all of the users in question develop software in MATLAB. They mainly create command line-based software tools, which they share with other scientists. The major part (four of five)of the users share the opinion, that training of students to get familiar with the USCT project is time-consuming. The results show that unexperienced users have problems to understand the coherency between A-scans, reconstructed images and the USCT aperture.

In the second part of the questionnaire, the users evaluated the user interface as clearly arranged. The users agreed on the simplicity of handling of the 3D model and selection of A-scans. They appreciated the partition into the three user interfaces, the presentation and browsing through 3D images and the extendability of the software. Another significant conclusion is the good applicability of the software for the training of students. Throughout all questions concerning the coherency of the A-scans, reconstructed images and the USCT aperture the users rated these function to be very beneficial.

The third part of the user evaluation was done using 50 questions following the SUMI questionnaire. For the analysis

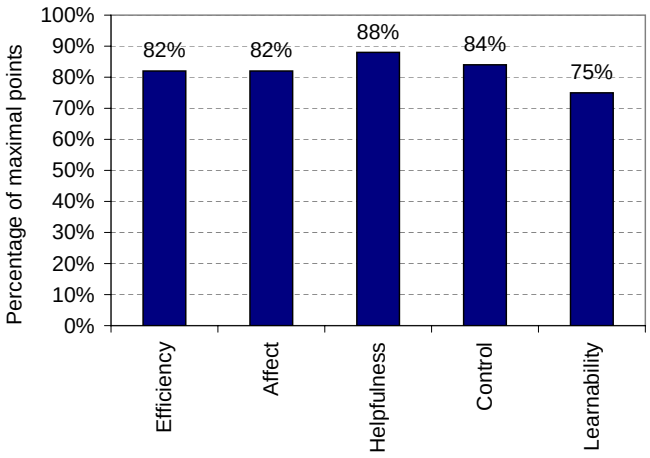

Figure 12. Results of the SUMI user evaluation questionnaire showing the attributes efficiency, affect, helpfulness, control and learnability.

of it, every question was assigned to one or more of the attributes efficiency, affect, helpfulness, control and learnability. The answers are rated with zero, one or two points, which are summed up for each attribute and related to the maximal number of points.

The overall averaged rating of the usability of the software is 81 % of the total number of points. The standard deviation is 12 %. Though the number of completed questionnaires is rather small, with a confidence coefficient of 0.95, the confidence interval can be acounted to $[65\,\%; 97\,\%]$, where the lowest value is still significantly higher than 50 %. This

encourages us to assume, that the developed software offers a good usability to the users.

Figure 12 shows the averaged result for the different attributes. The helpfulness of the software to process USCT problems was evaluated best (88 %), while there seem to be small difficulties in the learnability (75 %). Efficiency, affect and control are on the same level. Overall the results are on a quite satifying level of 75 % and up.

VI. Conclusion and Future Work

The described design and implementation shows the variability of MATLAB as programming language not only for numerical solutions but also for the design of complex software systems. With limited choice of interaction objects offered by MATLAB, a software system modeling the workflow of the USCT was created. The extendability is guaranteed by several interfaces leaving the programmer the choice of using already existing GUI elements or providing additional functions. By using parameter files whenever possible the system is adaptable to changes in the future – only these files have to be replaced.

As well as showing the capabilities of using MATLAB, the developed software also has a large benefit for the scientists. Before the software system was introduced to the USCT working group everyone implemented individual analysis tools for specific problems. The software now integrates most of these tools and provides a comprehensive GUI to access them instead of calling them by command line with multiple parameters. Together with the use of a subversion system everyone can now share analysis methods with the whole working group in a concise way. The integration of the reconstruction software was the next step to centralize all software systems used for the USCT in a comprehensive GUI.

The usability evaluation showed, that the software provides a good applicability for the training of students and new employees because of the new viewing method for the relationships between all parts of the USCT. The visualization makes the complex system more understandable than only reading documentation or even source code. By the integration of the reconstruction software the benefits of the analysis GUI can be ported to a second software used by the working group.

In the future the software will grow by adding new plugins and extending the basic functionality. Also the integration of the data acquisition software has to be improved. For the second generation of the USCT prototype the software can easily be adapted to the new transducer geometry by replacing the parameter files and revising the *A-scan GUI*. Furthermore the user evaluation will be extended to a higher number of attendants. The results of the user evaluation will be considered while extending and reengineering parts of the software.

References

[1] T. Hopp, G. Schwarzenberg, M. Zapf, and N. V. Ruiter. A MATLAB GUI for the Analysis and Exploration of Signal and Image Data of an Ultrasound Computer Tomograph. In *Proceedings of the First International Conference on Advances in Computer-Human Interaction 2008, ACHI 2008, IARIA*, pages 53–58. Published by IEEE Computer Society Press, 2008.

[2] H. Gemmeke and N.V. Ruiter. 3D Ultrasound Computer Tomography for Medical Imaging. *Nuclear Instruments & Methods in Physics Research*, 2:1057–1065, 2007.

[3] The MathWorks Inc. *MAT-File Format*. The MathWorks Inc., 2005.

[4] S.R. Doctor, T.E. Hall, and L.D. Reid. SAFT - the evolution of a signal processing technology for ultrasonic testing. *NDT International 19(3)*, pages 163–172, June 1986.

[5] M.D. Abramoff, P.J. Magelhaes, and S.J. Ram. Image Processing with ImageJ. *Biophotonics International*, 11(7):36–42, 2004.

[6] Volume Graphics Inc. Website. http://www.volumegraphics.com/, 2009.

[7] National Instruments Corporation. Website. http://www.ni.com/, 2009.

[8] C. Eggen, B. Howe, and B. Dushaw. A MATLAB GUI for ocean acoustic propagation. volume 3, pages 1415–1421, Oct. 2002.

[9] R. Stotzka, H. Widmann, T. Müller, and K. Schlote-Holubek. Prototype of a new 3D ultrasound computer tomography system: transducer design and data recording. In *SPIE's Internl. Symposium Medical Imaging 2004*, pages 70 – 79, 2004.

[10] M. Zapf, G. F. Schwarzenberg, and N. V. Ruiter. High throughput SAFT for an experimental USCT system as MATLAB implementation with use if SIMD CPU instructions. In Stephen A. McAleavey and Jan D'hooge, editors, *Medical Imaging 2008: Ultrasonic Imaging and Signal Processing*, volume 6920, page 692010. SPIE, 2008.

[11] G.F Schwarzenberg, M. Zapf, and N.V. Ruiter. Aperture Optimization for 3D Ultrasound Computer Tomography. In *Ultrasonics Symposium, 2007. IEEE*, pages 1820–1823, 2007.

[12] The MathWorks Inc. The MathWorks - MATLAB and Simulink for Technical Computing. Website, http://www.mathworks.com, 2007.

[13] The MathWorks Inc. *Creating Graphical User Interfaces - Version 6*. The MathWorks Inc., 2002.

[14] The MathWorks, Inc. Matlab file exchange: Scrollsubplot. Website, http://www.mathworks.com/matlabcentral/fileexchange/7730, 2008.

[15] The MathWorks, Inc. Matlab file exchange: Voxel. Website, http://www.mathworks.com/matlabcentral/fileexchange/3280, 2008.

[16] Drs. Erik P.W.M. van Veenendaal. Questionaire based usability testing. In *Conference Proceedings European Software Quality Week*, November 1998.