Building a user interface with MATLAB GUIDE for MRI data volumes in Imiomics

Anna Larsson

Institutionen för informationsteknologi
Department of Information Technology
Abstract

Building a user interface with MATLAB GUIDE for MRI data volumes in Imiomics

Anna Larsson

In this thesis project, a graphical user interface (UI) was built with the purpose of visualizing MRI data volumes that are used within Imiomics.

Imiomics is an ongoing research project that is a collaboration between several departments at Uppsala University. It involves handling a great number of medical imaging volumes which are whole-body MRI scans. In short, the work started within the Imiomics project can give disease understanding using scans and image analysis instead of invasive procedures.

The aim of this thesis project has been to create a UI that helps visualize and compare these image volumes to each other. The purpose of the UI is to enable quality control of the processed images and to facilitate medical interpretation of large cohort study findings.

The UI was built in MATLAB's development environment for graphical user interfaces, GUIDE. GUIDE is relatively easy to learn, fast to work with, and suitable for making prototypes for UIs containing data that is already handled in MATLAB.

Based on requirements from the users, the UI was divided into two modules with functionality that complement each other: One for studying correlation maps, and one for comparing image volumes before and after performing the image analysis operations in the Imiomics pipeline.

Towards the end of the project, a user test was organized. Members of the research group tested the UI and gave written feedback, and based on their suggestions several improvements were made. All user feedback is summarized in the report.
# Table of Contents

Populärvetenskaplig sammanfattning ................................................................. 3  
Acknowledgements .......................................................................................... 4  
Images ................................................................................................................. 4  
Abbreviations ..................................................................................................... 4  
1 Introduction ..................................................................................................... 5  
2 Background ..................................................................................................... 6  
   2.1 MRI and PET/MRI .................................................................................... 6  
   2.2 The anatomical planes – Coronal, Sagittal, Axial .................................... 6  
   2.3 About Imiomics ...................................................................................... 7  
      2.3.1 Data structures and output from the Imiomics pipeline ................. 9  
   2.4 Aim of this project ................................................................................ 13  
   2.5 Related work .......................................................................................... 13  
      2.5.1 3D Slicer ....................................................................................... 14  
      2.5.2 ENIGMA-Viewer ........................................................................... 15  
      2.5.3 The Nora medical imaging platform ............................................. 16  
3 Method ............................................................................................................. 17  
   3.1 Main requirements .................................................................................. 17  
   3.2 Software development model: Evolutionary prototyping ...................... 18  
   3.3 User test ................................................................................................ 18  
      3.3.1 Description of the test procedure ................................................... 19  
      3.3.2 The user test survey ...................................................................... 20  
   3.4 About MATLAB GUIDE ...................................................................... 24  
4 Result: Description of the user interface ......................................................... 27  
   4.1 A short user guide ................................................................................. 29  
      4.1.1 Part 1: Correlation Explorer ........................................................... 29  
      4.1.2 Part 2: Image Comparison ............................................................. 30  
   4.2 Detailed description ................................................................................ 31  
      4.2.1 Part 1: Correlation Explorer ........................................................... 32  
      4.2.2 Part 2: Image Comparison ............................................................. 36  
5 User test results and subsequent improvements ............................................. 40  
   5.1 Implemented improvements ................................................................... 40
## 5.1 Part 1: Correlation Explorer

- 5.1.1 Part 1: Correlation Explorer ................................................................. 42
- 5.1.2 Part 2: Image Comparison ................................................................. 44

## 5.2 User test survey: Answers ................................................................. 45

## 6 Implementation ................................................................................. 56

- 6.1 Processing image data ........................................................................ 56
  - 6.1.1 The image volume coordinate system ........................................... 56
  - 6.1.2 Creating 2D images from 3D volumes ........................................ 57
- 6.2 Common structures ........................................................................... 59
  - 6.2.1 The current point ........................................................................... 59
  - 6.2.2 Markers ......................................................................................... 59
  - 6.2.3 Drop-down menus ........................................................................ 60
  - 6.2.4 Colormaps ..................................................................................... 61
  - 6.2.5 Zoom in and Zoom out ................................................................. 62
  - 6.2.6 Scroll through volume .................................................................. 62
  - 6.2.7 Jacobian display range or limit values ......................................... 63
- 6.3 Part 1: Correlation Explorer ............................................................. 64
  - 6.3.1 Data structures: The “storage class” ............................................ 64
  - 6.3.2 The correlation map menu and parameter buttons .................... 66
  - 6.3.3 The “Scatter plot of data points” plot .......................................... 67
  - 6.3.4 The “Data cursor” replacement .................................................. 68
- 6.4 Part 2: Image Comparison ................................................................. 70
  - 6.4.1 The “Load saved ID” button ...................................................... 70

## 7 Discussion ......................................................................................... 71

- 7.1 Advantages and disadvantages of MATLAB GUIDE ......................... 71
- 7.2 Limitations of this work .................................................................... 72
- 7.3 Conclusion ......................................................................................... 73

## 8 Future work ....................................................................................... 74

- 8.1 Suggested improvements ................................................................... 74

## References ............................................................................................ 75

## Appendix: Code information ............................................................... 77
Populärvetenskaplig sammanfattning


Målet med examensarbetet har varit att skapa ett grafiskt användargränssnitt för att visualisera dessa bildvolymer och hur de förhåller sig till varandra. Syftet med användargränssnittet är dels kvalitetskontroll av de ändrade bildvolymerna och dels att underlätta en medicinsk tolkning av de beräkningar som har gjorts med bilddata från ett stort antal försökspersoner.

Gränssnittet implementerades i MATLABs inbyggda utvecklingsmiljö för grafiska användargränssnitt, GUIDE. GUIDE är relativt lätt att lära sig, snabbt att arbeta i och lämpat för att göra prototyper till gränssnitt som hanterar data som redan finns i MATLAB.

Baserat på användarnas behov delades gränssnittet upp i två moduler med funktionality som kompletterar varandra: En för att granska korrelationskartor och en för att jämföra bildvolymerna före och efter de steg av bildanalys som sker inom Imiomics.

I slutet av projektet genomfördes ett användartest där medlemmar i forskargruppen fick testa det färdiga gränssnittet. De gav skriftlig feedback och baserat på deras förslag gjordes ett antal förbättringar. Alla kommentarer från användarna finns sammanställda i rapporten.
Acknowledgements

I would like to thank everyone at the MRT-lab for help with user testing and for this opportunity to be a part of the research group.

I would especially like to thank my supervisor Robin Strand and my reviewer Joel Kullberg for great feedback and support during this thesis project.

Images

All images without specified sources are screenshots or drawings made by the author.

For data privacy reasons regarding the medical imaging volumes, some screenshots have been altered to remove identifying information.

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>GUIDE</td>
<td>Graphical User Interface Development Environment, a part of MATLAB</td>
</tr>
<tr>
<td>MRI</td>
<td>Magnetic Resonance Imaging</td>
</tr>
<tr>
<td>POEM</td>
<td>The Prospective investigation of Obesity, Energy and Metabolism, the source of the medical imaging volumes seen in this project</td>
</tr>
<tr>
<td>POEM ID</td>
<td>An ID number identifying a subject in the POEM study</td>
</tr>
<tr>
<td>Imiomics</td>
<td>“Imaging-omics”, a holistic analysis method for whole-body imaging data</td>
</tr>
</tbody>
</table>
1 Introduction

Medical imaging makes it possible to observe the inside of the human body. There are many techniques for this, and the “x-rays” from the beginning of the 20th century have evolved into sophisticated machines – scanners – that can scan a patient’s entire body and produce detailed 3D image volumes. In figure 1, a typical medical scanner is illustrated. Handling all this data requires advanced image analysis and significant resources in computing power and storage space. Thus, medical imaging is where IT and medicine come together. There is a lot of ongoing research in medical imaging, and one such research project is Imiomics.

![Illustrating the principle of a medical scanner](Image source: MRI clip art image, WorldArtsMe, [http://worldartsme.com/](http://worldartsme.com/))

The delivery of this thesis project is a user interface designed to handle image volumes used by or generated within the Imiomics pipeline.

At a deeper level, the purpose of this thesis project was solving a visualization problem. Imiomics provides a complex set of data that will be explained in the coming chapter. Is it possible to visualize this data? Is it possible to do it with the time and resources available? The user interface that has been developed shows one possibility for such a visualization.
2 Background

This chapter presents some necessary background information and clarifies the aims of this project. It ends with Related work, in which three existing applications for viewing medical data volumes are studied.

2.1 MRI and PET/MRI

Magnetic Resonance Imaging (MRI) is a common modality for medical imaging. Unlike Computed Tomography (CT) it does not require ionizing radiation and it excels at distinguishing between different kinds of soft tissue, like water, fat and muscle. A downside to MRI is that it involves very strong magnetic fields. It can be dangerous to bring magnetic objects in the vicinity of the scanner; thus, MRI cannot be used on patients with some types of implants or metal artefacts in their bodies.

Positron Emission Tomography (PET) works by injecting a tracer (often radiolabeled glucose) into the bloodstream. A scanning procedure is performed shortly thereafter to see where in the body the tracer has been absorbed. In this way, PET can give what is called “functional” information – it is possible to see which cells are absorbing the tracer. This is useful for, for example, finding cancerous cells as these cells absorb the tracer in greater amounts than other cells.

The disadvantage of PET is that it only gives functional information and does not give any information about the anatomical structures. Therefore, it is often combined with a MRI or CT procedure.

A combination of PET and MRI in the same procedure (referred to as PET/MRI) can give the advantages of both modalities, providing functional information in exact correlation to the anatomy. For this reason, the procedure has become widespread since it was introduced about two decades ago [1].

2.2 The anatomical planes – Coronal, Sagittal, Axial

Using medical imaging, we can view scans of the entire human body in 3D. However, studying image volumes directly is difficult in practice so 2D images are also used. It is easy to “cut” the data volume into planes, of any orientation, to produce 2D slices of the body.

There are three standard anatomical planes which are orthogonal to each other. These are three directions in which the human body can be observed as 2D projections, as shown in figure 2.
• **Coronal** or frontal planes divide the body into front and back sections.

• The **Sagittal** plane divides the body into left and right halves. Usually this refers to only one sagittal plane, the one in the middle, but any plane parallel to this one will be of interest in this report. Any such parallel plane will be referred to as a "sagittal slice".

• **Axial** planes or horizontal planes are parallel to the ground and divide the body into top and bottom parts.

---

**Fig 2: The anatomical planes in relation to the human body.**

---

### 2.3 About Imiomics

Imiomics is “a concept for holistic whole-body MRI data analysis” which was presented in a recently published article [2]. The word is derived from “imaging-omics”, analogous to other analysis methods like genomics. Here, the name Imiomics will be used to refer both to the analysis method and the ongoing research project in which it was invented. This research is a collaboration between several departments at Uppsala University spanning medical science and information technology. In short, the work started within the Imiomics project can give disease understanding using scans and image analysis. Future research in this field might lead to new ways to diagnose certain medical conditions without invasive procedures.
The Imiomics pipeline is a procedure involving a number of image analysis operations. Its input data is image volumes which are MRI or PET/MRI whole-body scans. Using image registration all this image data is transformed (or “deformed”) into a common coordinate system, which allows for comparisons between volumes. This can be used to compare scans of different people or scans of the same patient at different times, and it allows for integration with non-image data – any additional data associated with a subject aside from the MR image volumes. This non-image data can, for example, be measured numerical values like body weight or blood pressure. It can also be “true or false” values like subject is female (yes or no) and subject has a certain diagnosis (yes or no).

The calculations made within the Imiomics pipeline produce a variety of data, including statistical studies and correlation maps (see figure 3 for an example). This data is presented in further detail in subsection 2.3.1.

![Image of fat content comparison](image)

**Fig 3: Example of data generated within the Imiomics project. This is the mean and standard deviation of fat content in a large set of MRI volumes. (Image from [2])**

The Imiomics concept has been tested with sets (cohorts) of hundreds of data volumes. There are plans to test the pipeline with much larger data sets of hundreds of thousands of volumes. So far it has been possible to inspect all data sets manually but that will be much more difficult with a larger data set. Not all data volumes are
suitable for use in the pipeline, as different patient positioning during the scan causes a number of issues. The image registration can counter these issues to a degree, but there are still some data volumes that differ too much and must be discarded.

There was an idea that a graphical user interface could facilitate the future up-scaling of Imiomics. This interface should enable quality control – to help the researchers find any anomalous image volumes that are unsuitable for use, or where the image registration had unexpected results. Additionally, a good visualization of the data generated by the pipeline (the correlation maps for example) can facilitate a medical interpretation of the results. A student (the author) was assigned to design and build the user interface as a thesis project, which is described in this report.

The Imiomics project uses whole-body MR imaging data as well as non-imaging data collected from several different sources. One of these is the POEM study (Prospective investigation of Obesity, Energy and Metabolism) [3], where all data examples in this report originate from. All subjects in the POEM study are identified (pseudonymized) by a number which is called the POEM ID number.

For the subjects in the POEM study, total body fat and lean mass were measured by dual energy x-ray absorptiometry (DXA). This value is called the DXA fat value of the subject. The DXA fat value together with the ID number and gender of the subject are the only non-imaging data that were included in the user interface.

### 2.3.1 Data structures and output from the Imiomics pipeline

The user interface created during this project contains various data that is either used or generated within the Imiomics pipeline. Here this data will be presented in further detail. To clarify, no image analysis operations were performed as part of this thesis project, but data resulting from the Imiomics pipeline is handled and displayed in the user interface.

The input into the Imiomics pipeline is data volumes from MRI or PET/MRI scans, as well as data about various non-imaging parameters. The pipeline has many different steps of image analysis operations and is continually being improved upon. Most of the steps are done in MATLAB scripts, Python or C.

The output from the pipeline is a number of data volumes, modified MR image volumes as well as statistical data. For every non-imaging parameter, correlation studies are made which result in correlation maps (image volumes of R-values and P-values). The image volumes are stored as files in the \textit{vtk} format, which can be easily imported into MATLAB data. Non-imaging parameters are stored in Excel tables.

In List 1 presents the data that was handled in the user interface during this project. It is only a subset of all the data that is generated or used within Imiomics, but this limitation was chosen to make the data more manageable.
List 1: Data handled in the user interface

For every subject in the POEM study, the following data is included:

- Image volumes (MRI):
  - Deformed fat fraction (after image registration)
  - Undeformed fat fraction (before image registration)
  - Deformation field (Jacobian determinant) of the fat fraction

- Non-imaging parameters:
  - The DXA fat value
  - The gender of the subject (male/female)
  - POEM ID number

The following data is related to the entire study:

- Correlation maps (8) that were calculated from all subjects in the group:
  - R-values for fat fraction (male and female)
  - R-values for deformation field (male and female)
  - P-values for fat fraction (male and female)
  - P-values for deformation field (male and female)

- Two “reference volumes”, fat fractions of two subjects (1 male and 1 female) that were used as targets for the image registration.

List 1: The subset of data that was included in this project.

There are ~150 female (and a roughly equal number of male) subjects from the POEM study for which a DXA value for body fat has been measured and which were included in the calculation of the correlation maps. This is the subject group that will be visible in the user interface.

In the MRI scans the signals from water and fat have been separated. Thus, for every subject we have a “fat fraction” image. The image registration has created “deformed” images that are in the same coordinate system (based on a reference or target image) and that can be more easily compared to each other.

One of the correlation maps that have been generated within the Imiomics pipeline is that of the $R$-value – the linear relationship between the (local) fat fraction value and the DXA fat value belonging to the subject.
Fig 4: A scatter plot, where every data point represents a subject, with DXA fat values on the X-axis and fat fraction values (from the deformed fat fraction image volumes) on the Y-axis.

The R-value, or Pearson correlation coefficient, has values in the range [-1, 1]. Values close to 1 indicate a strong positive linear relationship between the two variables and values close to -1 indicate a strong negative relationship. An R-value close to 0 indicates that there is none or a weak relationship.

In a scatter plot of the two variables, a strong linear relationship means that the values will be close to forming a line. Figure 4 presents an example of such a scatter plot, but with a rather weak linear correlation.

Since the R-value is calculated automatically in the Imiomics pipeline, sometimes it can be misleading because of image artefacts or errors in the computation resulting in outliers. For this reason, it is necessary to inspect the scatter plot manually to verify whether the data points are close to forming a line or not.
Fig 5: Example of a correlation map (R-values). Only in the red areas is there a strong indication of a linear correlation.

Since there is a correlation value for every voxel in the image space, for every correlation study that is made the resulting correlation map is also a 3-dimensional image volume. Figure 5 shows an axial slice of such a correlation map.

For every R-value there is also a P-value. The P-value is calculated during the correlation study and has the range [0, 1]. When you are trying to prove a certain hypothesis, the P-value is the chance that you would have gotten the same statistical data if the opposite of the hypothesis were true (or if the correlation were 0). The P-value needs to be smaller than a certain number (usually 0.05) before you can say that the correlation is statistically significant. As with the R-values, there are P-value correlation maps for every non-imaging parameter.

To summarize: For every voxel in image space a scatter plot can be made showing the distribution of image data from the entire subject group. This scatter plot can be used to visually verify the correlation. Part of the aim of this thesis project was to find a convenient way to handle and inspect all these scatter plots and the related image volumes.
2.4 **Aim of this project**

The overall aim of this thesis project was to, by developing a graphical user interface, find a solution to the problem of visualizing the complex data that is used or generated within Imiomics.

The purpose of the visualization is to facilitate the future up-scaling of the Imiomics project – to make it easier to test and evaluate the methods in the Imiomics pipeline when used with very large sets of data. The users need tools to help identify anomalous data volumes in a large group, and to compare image volumes before and after they have been deformed.

The specific aim of this project is to make a visualization of the data specified in the previous section that fulfills the requirements made by the users.

To reach this aim, the following sub projects will be carried out:

- Implement suitable data processing methods
- Develop and implement a user interface
- Design, implement, and carry out a user test

The project is limited in the available time (6 months) and in the available resources (1 person).

2.5 **Related work**

There are many applications for viewing and manipulating medical imaging data in 2D and in 3D. The following are three examples of such software that inspired the work within this project.
2.5.1 3D Slicer

3D Slicer is “an open source software platform for medical image informatics, image processing, and three-dimensional visualization” [4]. It is very useful for viewing MR- and CT-volumes. It can handle DICOM data, which includes metadata such as subject identification and non-imaging data which is often included in medical imaging data.

When opening data volumes such as the ones seen in this project, the standard behavior of 3D Slicer is to divide the screen into smaller windows, where three slices of the volume are shown according to the standard anatomical planes – the coronal, sagittal and axial planes. An example of this is shown in figure 6.

The sliders or the mouse wheel can be used to scroll through the coronal, sagittal or axial slices when the cursor is in the corresponding window. The user can choose different “color maps” and a variety of settings for how the slices are displayed. 3D Slicer also has a lot of more advanced functionality.

Slicer was used as the main inspiration for how to display and navigate image volumes in this project: To display a 3D volume in 2D slices using the anatomical planes, showing all 3 orientations simultaneously, and the use of sliders and the mouse wheel to navigate between slices.

**Fig 6:** One of the Imiomics data volumes shown in 3D Slicer 4.8.1. Notice how the volume is displayed in slices based on the three anatomical planes.
2.5.2 ENIGMA-Viewer

ENIGMA (Enhancing Neuro Imaging Genetics through Meta-Analysis) is a global scale brain research collaboration that collects data in a large quantity.

The ENIGMA-Viewer was created in order for the results of the research collaboration to be communicated effectively among brain scientists. It is an open-source interactive web-based visualization tool that shows a 3D-model of the human brain. It is used to “compare statistics such as effect sizes from meta-analysis results on standardized ROIs (Regions-Of-Interest) across multiple studies” [5]. The brain is more viable for 3D modelling than the rest of the body since it is immobile and relatively similar between different individuals.

This exact type of visualization was deemed unsuitable for this project, but the use of the “bars” (on the right in figure 7) could be an inspiration for how to handle several non-imaging parameters at once. Also, the way specific brain regions are displayed could be useful for handling PET/MRI volumes where regions with strong PET-signal need to be highlighted.

Fig 7: Screenshot of the ENIGMA-Viewer opened in Chrome. A 3D model of the brain is shown. The directions “Superior” and “Anterior” are medical terminology for “up” and “front”. The tables on the right show the effect of some non-imaging variable on different parts of the brain. The bar for “Genu of corpus callosum” is selected, and this brain region is highlighted in the 3D model. The user can rotate the model by clicking and dragging.
2.5.3 The Nora medical imaging platform

The Nora medical imaging platform was developed by a team of researchers and clinicians at the University Medical Center Freiburg, Germany. It is a web-based toolbox for visualization and analysis of medical imaging. It “has been developed to bridge the gap between research and clinic, and to boost medical imaging research to the next level. It provides a high-level web-interface accessible from any web-browser to visualize, organize, process and share data in a very customizable way.” [6]

Nora has many features such as overlays for comparison of longitudinal data, lesion-segmentation, 3D surface rendering and fiber visualization. It can be used in scientifical studies to automatically analyze hundreds of images through a processing pipeline.

Like Slicer, Nora shows 2D slices of a 3D volume in accordance with the anatomical planes. The main inspiration taken from Nora in this project was the ability to select a voxel and mark this voxel using lines similar to crosshairs (the horizontal and vertical lines seen to the right in figure 8).

![Fig 8: Screenshot of a Nora demo opened in Chrome. An axial slice of a brain is shown. The user can click with the mouse to select a voxel, which is then marked by vertical and horizontal lines. This brain volume is segmented, and the segment containing the selected voxel has been highlighted. The mouse wheel can be used to scroll through the axial slices of the image volume.](image-url)
3 Method

A lot of the Imiomics image analysis pipeline is done in MATLAB scripts, and much of the data is either handled there or can be easily imported to MATLAB. Therefore, a convenient approach was to use MATLAB’s own built-in Graphical User Interface Development Environment, MATLAB GUIDE [7].

GUIDE is suitable for creating prototypes as it is relatively easy to learn and fast to work in. Section 3.4 provides a short introduction to GUIDE.

The UI that has been developed is presented in chapter 4, Result: Description of the user interface, and the technical implementation is explained in further detail in chapter 6, Implementation.

3.1 Main requirements

Based on the needs of the users, the following were the main requirements for the deliverable:

- Inspection of the data distribution: Make it possible to observe a correlation map and select a voxel to get a visualization of how the image data is distributed in that particular voxel (as a scatter plot). What does the distribution look like, and are there any extreme values that can affect the correlation?
- Quality control: Implement methods to inspect the quality of the result of the image registration for an individual image volume.

As the understanding of the Imiomics project and the data grew, the aim could be made more concrete and the requirements for the UI were fleshed out over time.

Based on the initial requirements, it was concluded that the UI could be divided into two parts or modules. The first part of the user interface is meant to be used by the researchers to facilitate the identification of outliers without having to look through all the hundreds of data volumes, and the second part should be focused on comparing an image volume before and after the image registration.

Among the most important desired features were the 2D representation of data volumes by showing slices according to the three anatomical planes (as in 3D Slicer), and the possibility to mark a voxel and see its coordinates in 3D-space (as in Nora). Several other features were also borrowed from the medical imaging software observed in section 2.5, such as sliders, mouse wheel scrolling, and different colormap options.
The challenge of this project consisted of both understanding the rather complex data within Imiomics and building a specialized UI that can provide the functionality asked for by the users.

3.2 **Software development model: Evolutionary prototyping**

The development process followed the evolutionary prototyping [8] model of software development, also called iterative development. The development process starts with initial requirements and a first design or prototype. This design or prototype is shown to the user or product owner who provides feedback. Unlike in “throw-away” prototyping, in iterative development the previous prototype is kept but it is continually changed and expanded upon – it “evolves” into the final product. In every iteration, the requirements and system design are refined based on the feedback.

During the development process in this project, the supervisor and reviewer represented the users. Feedback sessions were held weekly or every other week and features were continually added, altered or improved.

An advantage of the evolutionary prototyping model is that having a working prototype helps the users give concrete feedback. You can find out early in a project if the requirements need to be changed. A disadvantage is that iterative development can produce messy code. Therefore, it is important to structure the system carefully to make future changes as easy as possible.

3.3 **User test**

Towards the end of the project, a user test was organized. A document with instructions and questions was prepared, and several members of the research group volunteered as test users. Based on suggestions from this feedback, some final improvements were made to create the current version of the UI. A list of these improvements, and a comparison between the current version and the tested version, can be found in chapter 5.

Some additional suggested improvements that could not be implemented due to time constraints are presented in chapter 8, Future work.
3.3.1 Description of the test procedure

The testing procedure used was a participant-based evaluation using a written questionnaire. A participant-based evaluation is a test where a group of the intended end-users evaluate a prototype or a fully functional system in a structured manner and are asked to give feedback. [9]

Several members of the research group at the MRT-lab* volunteered as testers. They are representative of the end-users of the UI as they are all involved in projects related to Imiomics.

The questionnaire or the “user test survey” is presented in 3.2.2. Users were given a task to perform and a number of questions to answer in writing. Since the number of available testers was small (less than 10) the tests were designed for a qualitative approach rather than a quantitative one. There were a few quantitative questions included, the answers to which are summarized in chapter 5.

The author prepared the tests by starting up MATLAB GUIDE and launching the UI modules on a desktop computer. The users were given the test survey document containing instructions and questions, and were then left alone to complete the test at their own pace. The document was printed with plenty of blank space between the questions where feedback could be written down and the testers were provided with pencils. The average time spent was about 30 minutes.

There are several reasons written feedback was chosen above oral feedback – it makes it easier to preserve the user feedback in detail to use it for future improvements and it is more anonymous. Also, the use of printed instructions and questions helps ensure the test conditions are the same for all users.

* The “MRT-lab” is the locale where this thesis project was carried out. It is located one floor above of one of the MRI-scanners at “Akademiska” (the Uppsala University hospital) and it hosts a research group with members belonging to either the Department of Information Technology: Centre for Image Analysis or the Department of Surgical Sciences: Radiology. Imiomics is one of many research projects in medical image analysis conducted within this group.
3.3.2 The user test survey

(The survey began by asking for the testers’ consent in using their feedback in this report.)

Please mark the option that best fits you:

I have ___ knowledge of human anatomy
○ no
○ some
○ a lot of

I have ___ experience in using computer programs that show MR-volumes (like Slicer or similar)
○ no
○ some
○ a lot of

I have ___ experience in using Matlab
○ no
○ some
○ a lot of

(A short user guide followed, similar to section 4.1 of the report.)
Test description

For testing purposes, I have inserted cube artefacts into a couple of fat-fraction volumes, one male and one female. I have also let this artefact visibly affect the correlation maps. Use part 1, the Correlation Explorer, to try to identify the offending volumes.

Suggested steps

1. Look at the correlation map with filename correlation_analysis_female_rval_parametric_fat_dxa_n156.vtk.
2. Find the artefact (it looks like a square).
3. Drag the marker to the artefact and look in the plot for a data point that is an outlier (has a value very different from the others).
4. Click on the data point to select it. You can now see the POEM ID of the volume that the data point belongs to.
5. Move the marker around slightly to confirm that the selected volume has an outlier value in other places within the artefact.
6. Go into the Image Comparison module and press “Load saved ID”. Confirm that this is the subject with an artefact.
7. Look at the correlation map with filename correlation_analysis_male_rval_parametric_fat_dxa_n151.vtk and repeat the steps above.

Task questions

- Can you find the female fat-fraction volume where the artefact is located? What is the POEM ID number?
- Can you find the male fat-fraction volume where the artefact is located? What is the POEM ID number?
- Any comments?

Feedback questionnaire

Freely experiment with the UI modules for a while, and please answer the following questions. Any additional feedback you can think of is welcome.

On the last pages of this document there are screenshots that you can use as reference or to leave further comments directly onto the images. Make arrows, circle or mark things as needed.
Part 1: “Correlation Explorer”

1. What do you think about the green line in the scatter plot? Is it clear what it represents?
2. In the plot, how difficult is it to see and / or click on the data points?
3. How well do you think the plot works for finding data points that are outliers?
4. At the right side of the UI window, several values are shown: POEM ID, R-value, P-value and image volume value. How useful are they?
5. Is it clear how to switch between the 8 different correlation maps available (R-values and P-values)?
6. There are three markers, one for each of the Coronal, Sagittal and Axial views. What do you think about their functionality?
7. What do you think about the layout?

Part 2: “Image Comparison”

8. In this UI module, you only see one view at a time (Coronal, Sagittal or Axial). What do you think about this? Is it clear how to change the view?
9. What do you think about the setting to switch the left-most image between undeformed image and “difference image”?
10. Is it clear how to change the display range of the Jacobian image? What do you think about this functionality?
11. What do you think about the layout?

General questions

12. What do you think about the keyboard shortcuts? (In [L]oad, [L] means that the “L” key is a shortcut for using this button.)
13. There are menus for changing colormaps and marker color. What do you think about the colormaps and marker colors?
14. What do you think about the use of drop-down menus?
15. You can use the mouse wheel to scroll through the volumes. Is this function working as desired?
16. How is the response time of the applications? Did you experience any “lag”?
17. Is there any data or functionality that these UI modules are lacking, that you would like them to have?
18. What data or functionality in these UI modules seems unnecessary or redundant?
19. Did you find any bugs or strange behavior in the UI? Please describe it in as much detail as possible.
20. Any additional comments?

(Users were also given screenshots of the UI modules and encouraged to reference these when giving comments.)
3.4 About MATLAB GUIDE

MATLAB GUIDE is used to create graphic user interfaces. Using this environment, it is easy to place buttons, graphs and other pre-defined objects and move them around. GUIDE is included in the standard MATLAB installation and any data or function in MATLAB can be used in a GUIDE user interface.

To open GUIDE, simply write the command `guide` in MATLAB. Figure 9 shows a screenshot of the GUIDE interface.

![Figure 9: Screenshot of the GUIDE window while working on part 1 of the UI.](image)

For every created user interface GUIDE generates a fig-file and an m-file. All programming is done in the m-file. At the start, and every time a new object is added to the UI, skeleton code for a number of functions is added to the m-file.

One important type of graphics object in GUIDE are the so-called “axes”. These are the big white boxes in fig 9, and they can be filled with either images or graphs. They can be seen as “frames” that you place and then fill with content.

When an object is placed in the GUIDE window, the skeletons for a creation function and a callback function are generated at the end of the m-file. Most of the advanced functionality happens in the callback functions, which are activated when the user clicks a button or otherwise interacts with the object in the UI.
For example, when a button with the tag (object name) “buttonX” is created, the following functions will be generated in the m-file:

```matlab
function buttonX_Callback(hObject, eventdata, handles)
% ...
function buttonX_CreateFcn(hObject, eventdata, handles)
% ...
```

If the name of the button is changed at a later time, the names of the functions will be updated. It is inadvisable to change the auto generated function heads (name and arguments) in the m-file, but the function bodies can be filled with any desired functionality.

Another important GUIDE function is the opening function. It is located in the beginning of the m-file, and its function head begins like this:

```matlab
% --- Executes just before UIName is made visible.
function UIName_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% ...
```

In this function you want to, for example, populate your graphs (fill them with content) or load your images.

One important thing that happens in the opening function for the UI created in this project, is to run a prepared script to load all the data that is needed into the MATLAB workspace. After that, any variables that will be needed by the UI are added to the handles structure. Any variable that needs to be available to other functions in the UI must have a handle. Example:

```matlab
handles.xmax = xmax;
```

Every graphics object has a number of properties. There are “get”- and “set”-functions that can be used to retrieve or change the values of these properties as needed. Example:

```matlab
point = get(handles.axesData, 'CurrentPoint');
```

Here, `handles.axesData` is the object (the frame that holds a plot), and `CurrentPoint` is the extracted property which holds the coordinates of a point (in the 2D space of the plot). Figure 10 shows many of the common properties in an example of the “Property inspector” window.
Fig 10: Example of properties: The GUIDE “Property inspector”, showing the properties of figure1, or the base window of a UI. Every object (axes, buttons etc.) has its own set of properties.
4 Result: Description of the user interface

In this chapter, the user interface that was built during the project will be presented.

In accordance with the two goals defined in the beginning of the project, the UI was built as two parts or modules: Part 1, the “Correlation Explorer”, and part 2, the “Image Comparison”.

In part 1 the user is shown a correlation map along with a scatter plot where the corresponding data points from all subjects are shown (the local values of fat fraction or Jacobian determinant images are mapped against DXA fat values). The user can place a marker at any voxel in the correlation map and the scatter plot is updated accordingly.

In part 2 the user can study the effect of the image registration on a particular subject. The same slice of five different image volumes are shown next to each other for comparison: The undeformed, deformed, reference, difference and Jacobian determinant images.

For both parts of the UI, several options for colormaps are available. A colormap is a function that attributes a color to every value in the numeric range of an image. For example, if an image has the range $[0, 1]$, 0 can be displayed as black and 1 as white, with a grayscale for all values in between.

After the user has selected a certain subject ID in part 1 for closer inspection, the ID number is saved so the images related to that subject can be easily opened in part 2.

The UI modules run on top of MATLAB as two different applications. To access them, they must be opened in MATLAB GUIDE and run by pressing the “play” button. If the modules are minimized, they can be found by a mouse-over of the MATLAB symbol on the Windows task bar at the bottom of the screen, as seen in figure 11.

Fig 11: The Windows task bar when running the UI modules. Highlighted in green are the MATLAB symbol on the task bar, and the UI applications “Correlation Explorer” and “Image Comparison”.

27
In section 4.1, the UI will be presented through a short “user guide” that shows screenshots and points out the most important functions. Then, the functionality will be presented in further detail in 4.2. In chapter 5 there are images of what the UI looked like during testing (version 0.5) and all the improvements from that version to the current one (version 1.0) are listed.

Chapter 6 gives more information about the implementation or “back-end” of the UI.
4.1 **A short user guide**

Here the UI developed in this project will be introduced through a short “user guide” and a description of the most important functionality in its two modules.

### 4.1.1 Part 1: Correlation Explorer

Part 1, shown in figure 12, allows the user to study some different correlation maps and see the value distribution of all the subjects in the group (male or female subjects) in a selected voxel. The user can select any voxel in XYZ-space to study by clicking to place the black marker anywhere in the images on the left. These three images together illustrate a 3D volume by displaying different cuts or “slices” of the volume.

On the right-hand side of the UI, there is a scatter plot. Every data point in this scatter plot represents an image volume belonging to an individual subject. You can select a subject by clicking on one of the data points, and the POEM ID of the chosen subject is shown.

![Image of Correlation Explorer](image.png)

*Fig 12: In the Correlation Explorer the user can select an individual subject either by using the “Next” and “Previous” buttons, or by clicking one of the data points in the scatter plot.*

The green line in the scatter plot is generated by using a least-squares method to find the best possible line to approximate the data. The correlation value (R-value)
signifies how close the data points in this particular voxel are to be in alignment with this green line.

Instead of displaying correlation maps (that are based on data from many subjects), the UI can display the fat fraction or Jacobian determinant image volume of an individual subject. To do this, change the current display from “Correlation Map” to “Image Volume” in the drop-down menu in the top left of the screen.

Once you have selected a subject, go to part 2 of the UI – it is more suitable for studying all the image volumes related to a subject.

4.1.2 Part 2: Image Comparison

Once you have selected a subject (female number 1 is selected by default), the POEM ID number is automatically saved. You can then proceed to part 2 and press the “Load” button to open the image volumes related to this subject and study them more closely (the Load button can be seen in the top left of the UI). Figure 13 shows the Image Comparison when it is initially opened, and figure 14 shows it after the image volumes have been loaded.

![Image Comparison UI](image)

*Fig 13: The Image Comparison UI before the images associated with a subject have been loaded. Only a reference subject and a difference image are shown.*

After the image volumes are loaded, you can see the deformed and undeformed images as well as the Jacobian determinant image. The reference subject (male or
female) is also shown. Lastly, there is a volume (second from the right) that is the calculated “difference” between the deformed image and the reference. This can be used to see how well the image registration went, or if there are any extreme values. The Jacobian determinant and difference volumes use the same colormap (red, white and blue), but have slightly different ranges. They will be explained in further detail later on.

Fig 14: The Image Comparison UI after all the images related to a subject have been loaded.

4.2 Detailed description
This section contains a complete description of all functionality of the UI, first the Correlation Explorer and then the Image Comparison. Each subsection starts with a screenshot of the UI module where components are identified by numbers (figure 15 and figure 16) and the numbered sections on the following pages give a description of each component.
4.2.1 Part 1: Correlation Explorer

Fig 15: The Correlation Explorer UI module with components marked by numbers. See the following pages for more detailed descriptions of each component.
1) The “Currently Displaying” menu
A drop-down menu containing two alternatives – “Correlation Map” and “Image Volume”. This menu determines which type of image volume is shown in the images: One of the (8) correlation maps, or an image of an individual subject.

2) The correlation map filename menu
A drop-down menu containing 8 alternatives, the available correlation maps. The full filenames are shown in the menu. This menu determines which correlation map is displayed (if the “Currently Displaying” menu is set to Correlation Map). The current correlation map can be chosen either through this menu or by using the parameter buttons. When the user changes the filename, the parameters are updated, and vice versa.

3) The “Current correlation map” panel
The correlation map parameter buttons
Three pairs of radio buttons representing the parameters that signify the 8 available correlation maps: Female/male, P/R, and fat/Jacobian. These are an easier way of choosing the current correlation map. When the buttons are used and the parameters are changed, the filename is updated, and vice versa.

Voxel correlation values
Here the R- and P-values of the current voxel are given. They are shown regardless of whether a correlation map or an individual image volume are currently displayed in the image panel.

4) The image panels
Here are the 3 images that show the axial (4a), sagittal (4b) and coronal (4c) slices of the current image volume.

On top of the images are the markers. They can be moved using click-and-drag, or by clicking anywhere in the image. The 3 markers are all connected to the same point in 3D-space (the current point), so if the marker in one image is moved, that will affect the current point and the markers in the other images. The markers will also be moved by all other functions which change the current point.

The colorbar shows the current image display range and the corresponding colors.

The three sliders (one for each image panel) allows the user to scroll through the volume in the x, y or z-direction. It is also possible to scroll through the volume by using the mouse wheel – the direction is based on which slice is selected (the one that
was last clicked on by the user). A text (“Scroll through volume [MouseWheel]”) is shown above the image panel of the slice that is currently selected.

5) The “Marker position” panel

The number boxes

The boxes display the current point, which is the position in xyz-space of the three markers. The user can edit the numbers, but the numbers cannot go below 1 or above the size of the image volume (invalid input will be ignored). The keyboard shortcuts X, Y, Z are bound to these boxes.

The “increase value” buttons (+)

These buttons increase the value of x, y or z and updates the current point. The numbers cannot exceed the size of the image volume.

The “decrease value” buttons (-)

These buttons decrease the value of x, y or z and updates the current point. The numbers cannot go below 1.

6) The “Zoom in” and “Zoom out” buttons

The “Zoom in” button zooms in the image view around the current point (keyboard shortcut I). There are 3 pre-defined zoom-steps, thus this button can be used 3 times.

The “Zoom out” button zooms out the image view (keyboard shortcut O). “Zoom out” can only be used when the view is already zoomed in. This button goes back through the zoom-steps until the view is restored to normal.

7) The “Reset position” button

This button will restore the current point to the starting point, which is the middle voxel of the image volume. If the view was zoomed in, it will also zoom out to the normal view.

8) The “Colormap” menu

This drop-down menu allows the user to change the colormap of the images. There are 5 options: Jet, Gray, Parula, Hot and BlueWhiteRed. The first 4 are pre-defined colormaps in MATLAB, the last one is a custom colormap that is useful for viewing correlation maps and Jacobian determinant images.
9) **The “Marker color” menu**
This drop-down menu allows the user to change the color of the markers. There are 3 options: Black, white and magenta. This can be used as needed to make the markers easier to see against the underlying images.

10) **The “Scatter plot of data points” graph**
In this graph, a scatter plot is shown with DXA fat values on the X-axis, and either Fat fraction values or Jacobian determinant values on the Y-axis.

The user can change the selected subject by clicking any of the data points. A blue square marks the data point of the currently selected subject.

The scatter plot is changed when the current point is changed, or when the data type is changed (through the parameter buttons) from fat fraction to Jacobian determinant. The display range (the upper and lower limits of the Y-axis) can be changed by the user through the “Limit values for Jacobian determinant volumes” panel.

The green line is the best possible line that can fit onto the data, found by using a least-squares method. It is intended as a visual aid, since the correlation value (R) is a measure of how closely the data points follow a linear function.

11) **The “Selected subject” panel**
This panel displays the “voxel value”, the value in the “current point” voxel for the fat fraction or Jacobian determinant image volume of the current subject.

The POEM ID of the currently selected subject is also displayed.

The user can click the buttons “Previous subject” and “Next subject” (or use the left and right arrow buttons) to switch between subjects in the current subject group (male or female). The POEM ID changes accordingly, and the blue square moves to the data point of the selected subject.

12) **The “Limit values for Jacobian determinant volumes” panel**
The user can use the sliders or edit the numbers to change the upper and lower limits for the display range. This affects both the display range of the images (when displaying the Jacobian determinant) and the Y-axis of the scatter plot. The ranges of possible values are [1, 5] for max and [-3, 0.99] for min.

The “Find limits” button (keyboard shortcut F) automatically sets the limits to the highest and lowest values in the current slice (with a small margin added in the scatter plot). The “Set to (0, 2)” button resets the limits to 0 and 2.
4.2.2 Part 2: Image Comparison

Fig 16: The Image Comparison UI module with components marked by numbers. See the following pages for more detailed descriptions of each component.
1) The “Load saved ID” button
This opens the images for the subject that was previously selected in the part 1 module. It has the keyboard shortcut L.

2) The “POEM ID” number box
This shows the POEM ID number of the subject whose images are currently displayed in the UI. For all POEM IDs that can be selected from the part 1 module, all the five images exist. However, that does not cover all possible ID numbers. The user can type other numbers into the box, and if the images exist (in the file directory that is specified in the code) they will be loaded. If the images cannot be found, the UI will print an error message to the MATLAB command prompt.

3) The “Colormap” menu
This drop-down menu allows the user to change the colormap of the three left-most images (there are separate menus for the two right-most images). There are 4 options: Jet, Gray, Parula and Hot. These are pre-defined colormaps in MATLAB.

4) The “Marker color” menu
This drop-down menu allows the user to change the color of the markers. There are 3 options: Black, white and magenta. This can be used as needed to make the markers easier to see against the underlying images.

5) The “View” panel
Here the user can switch the view between Coronal, Sagittal and Axial slices. The keyboard shortcuts are C, S and A respectively. These changes affect all the images.

6) The “Marker position” number boxes
In part 2, there is a marker in every image but they all have the same position in the image space. If one is moved, the others move in the same way.

The number boxes display the current point, which is the position in xyz-space of the markers. The user can edit the numbers, but the numbers cannot go below 1 or above the size of the image volume (invalid input will be ignored). The keyboard shortcuts X, Y, Z are bound to these boxes.
7) The “Scroll through volume” slider
This slider can be used instead of the mouse wheel to scroll through the image volumes.

8) The “Zoom in” and “Zoom out” buttons
The “Zoom in” button zooms in the image view around the current point (keyboard shortcut I). There are 3 pre-defined zoom-steps, thus this button can be used 3 times.

The “Zoom out” button zooms out the image view (keyboard shortcut O). “Zoom out” can only be used when the view is already zoomed in. This button goes back through the zoom-steps until the view is restored to normal.

9) The “Reset position” button
This button will restore the current point to the starting point, which is the middle voxel of the image volume. If the view was zoomed in, it will also zoom out to the normal view.

10) The “Values” panel
Here the values of all the images (except the Difference image) are shown.

11) The image panels
All the images have the same position and orientation in xyz-space, so that the image volumes can easily be compared. Changing the view affects all images. So does changing the current point – the markers are always in the same position in the images.

Each image has a colormap that shows the display range.

The following images are displayed in the UI, from left to right:

11 a) Undeformed Image
The fat fraction image before the image registration (this volume cannot be viewed in part 1).

11 b) Deformed Image
The fat fraction image after the image registration (this is the volume that can be viewed in part 1).
11 c) Reference Subject
The fat fraction volume of the reference subject, male or female (the image registration used this image as a target).

11 d) Difference Image
The Difference volume is calculated as Deformed – Reference. As the display range is always [-2, 2], it can be used to determine how “good” the registration was.

11 e) Jacobian Determinant
The Jacobian determinant image volume, which is a measure of the size of the deformation during the image registration. The display range starts at [0, 2] as the normal value is 1 (no change). The display range can be changed using the controls in the “Jacobian display range” panel.

12) The “Jacobian display range” panel
The user can use the sliders or edit the numbers to change the upper and lower limits for the display range of the Jacobian determinant image. The ranges of possible values are [1, 5] for max and [-3, 0.99] for min.

The “Find limits” button (keyboard shortcut F) automatically sets the limits to the highest and lowest values in the current slice. The “Set to (0, 2)” button resets the limits to 0 and 2.

13) The “Difference colormap” menu
This drop-down menu allows the user to change the colormap of the Difference image. There are 3 options: BlueWhiteRed, Jet and Gray. BlueWhiteRed is a custom colormap that is useful for showing differences and Jacobian determinants, and the others are pre-defined colormaps in MATLAB.

14) The “Jacobian colormap” menu
This drop-down menu allows the user to change the colormap of the Jacobian determinant image. There are 3 options: BlueWhiteRed, Jet and Gray. BlueWhiteRed is a custom colormap that is useful for showing differences and Jacobian determinants, and the others are pre-defined colormaps in MATLAB.
There were 8 participants in the user tests. The feedback from the user test surveys was mostly positive, as several testers stated that the UI should be useful. There were also many good suggestions for improvements.

Section 5.1 summarizes all subsequently implemented improvements to the UI that were suggested in user feedback. In section 5.2, the complete results from the user test survey are listed, including all written comments. In chapter 8, Future work, there is a list of some additional suggested improvements that were not implemented due to time constraints.

5.1 Implemented improvements

In this section, version 0.5 of both parts of the UI – the version that was used during the testing phase – is shown to facilitate comparisons with the latest version, 1.0.

There were several changes that apply to both parts of the UI, these are listed in table 1 below. On the following pages, changes specific to the two modules are listed in tables 2 and 3 together with images (figure 17 and figure 18) showing version 0.5 of the respective module.
Table 1: Improvements from version 0.5 to 1.0, that apply to both modules of the UI

<table>
<thead>
<tr>
<th>In version 0.5</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>The markers could only be moved by click-and-drag, which could be difficult.</td>
<td>Now the user can click anywhere in the images and the marker will be moved there.</td>
</tr>
<tr>
<td>The XYZ-position numbers were in magenta (due to a previous version where marker colors were always in magenta).</td>
<td>The position numbers were changed to black.</td>
</tr>
<tr>
<td>The display range for Jacobian images started at [-1, 1], and the minimum value could not be raised above 0. Users pointed out this was erroneous, as the Jacobian should be centered around 1.</td>
<td>The starting display range was changed to [0, 2], centered around 1, and the minimum value can now be increased to 1.</td>
</tr>
<tr>
<td>The Jacobian determinant images used the same colormaps as other images, but it has a different range and behavior.</td>
<td>The custom colormap “BlueWhiteRed” was created. It is suitable for Jacobian determinants and also for images with a range of [-1, 1] (like the R-value correlation).</td>
</tr>
<tr>
<td>The keyboard shortcut “L” was used for different functions in part 1 and 2.</td>
<td>Some unnecessary keyboard shortcuts were removed, and some were changed so “L” is only used for one function.</td>
</tr>
</tbody>
</table>
5.1.1 Part 1: Correlation Explorer

Fig 17: Part 1 during the user test (version 0.5). The layout has since been changed.
### Table 2: Improvements from version 0.5 to 1.0: Part 1

<table>
<thead>
<tr>
<th>In version 0.5</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>The controls for the image volume and the scatter plot were close together and interleaved.</td>
<td>The layout has been changed to improve grouping – to put the controls associated with the image volumes closer to the images, and the controls for the scatter plot closer to the plot.</td>
</tr>
<tr>
<td>Finding a particular correlation map could only be done by finding the right filename in the drop-down menu, which was difficult.</td>
<td>3 radio buttons were added for the 3 parameters female/male, P/R and fat/Jacobian, to make it easier to find a correlation map.</td>
</tr>
<tr>
<td>Data points (for JD data) in the plot could be located on the border, making them hard to see.</td>
<td>When using “Find limits”, the function will add a small margin of space so that the limit values are well inside the border.</td>
</tr>
<tr>
<td>The R- and P-values, POEM ID and “Image volume value” were grouped together in a confusing way.</td>
<td>R- and P-values were separated into the “Voxel correlation values” panel (closer to the images). The others were moved into the “Selected subject” panel (closer to the scatter plot) and “Image volume value” was changed into “Voxel value”.</td>
</tr>
<tr>
<td>An “ID-number” was used for the selected subject (when looking at the female subject group, the first female subject had number 1 etc.) which could be confusing.</td>
<td>Only the POEM ID is shown to identify a subject (the other ID number still exists in the implementation, but the user does not need it).</td>
</tr>
<tr>
<td>The “Limits” settings for Jacobians needed to be reworked (see table 1).</td>
<td>The starting range was set to [0, 2] and a button was added to reset the range to this value.</td>
</tr>
</tbody>
</table>
5.1.2 Part 2: Image Comparison

**Fig 18**: Part 2 during the user test (version 0.5). The most visible change between versions 0.5 and 1.0 was showing 5 images simultaneously instead of 4.

**Table 3: Improvements from version 0.5 to 1.0: Part 2**

<table>
<thead>
<tr>
<th>In version 0.5</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only 4 images were shown simultaneously, and the user had to use</td>
<td>The layout was changed so that all 5 images can be shown simultaneously.</td>
</tr>
<tr>
<td>a drop-down menu to choose between the undeformed and difference images in</td>
<td>The drop-down menu was removed.</td>
</tr>
<tr>
<td>the left-most image.</td>
<td></td>
</tr>
<tr>
<td>The images were all tied to the same colormap option.</td>
<td>More controls were added so that the user can select colormaps for the difference image and Jacobian</td>
</tr>
<tr>
<td>Users thought it unclear what the difference image is.</td>
<td>determinant separately from the other images.</td>
</tr>
<tr>
<td>Users thought the max/min controls for the Jacobian were too far apart.</td>
<td>The difference image is now stated to be calculated as “deformed - reference”.</td>
</tr>
<tr>
<td>Users requested an “auto” button for the range (as there already was in part 1).</td>
<td>The controls were moved together at the top and the “Jacobian display range” panel was created.</td>
</tr>
<tr>
<td></td>
<td>The same buttons as in part 1 were added: “Find limits” and “Set to (0, 2)”.</td>
</tr>
</tbody>
</table>
5.2 User test survey: Answers

(There were 8 participants in the user test. In this section, user comments are presented below the questions. Some answers were originally written in Swedish and have been translated by the author.)

Please mark the option that best fits you:

I have ___ knowledge of human anatomy
○ no (0)
○ some (8)
○ a lot of (0)

I have ___ experience in using computer programs that show MR-volumes (like Slicer or similar)
○ no (0)
○ some (4)
○ a lot of (4)

I have ___ experience in using Matlab
○ no (0)
○ some (3)
○ a lot of (5)
Task questions

- Can you find the **female** fat-fraction volume where the artefact is located? What is the POEM ID number?

  (7 out of 8 testers were able to find the volume and provide the right ID-number.)

- Can you find the **male** fat-fraction volume where the artefact is located? What is the POEM ID number?

  (6 out of 8 testers were able to find the volume and provide the right ID-number.)

- Any comments?

  I think the task can be solved neatly. The only remark is that, at a certain point, it was a bit confusing to select the image to be displayed in the Correlation Explorer, it takes some concentration to find the right file among the ones in the drop-down menu.

  If possible, it would be easier if the volumes are updated as you pull on the scrollbars, not only when you let go of them.

  Very good with the scatter plot showing the outlier as the artefact can be quite subtle in the correlation map.

  Good workflow, easy to find outliers. I realized you could inspect the individual images even quicker by changing from “correlation map” to “Image volume” in correlation explorer, which is good.

  Instead of “grabbing” the marker to move it, is it possible to just click at the point you want to inspect? The marker is small, and finding where it is on the screen currently is an extra step...

  POEM ID field in Image Comparison module automatically resets to a random ID when pressing “[L]oad saved ID” and pressing enter.

  When pressing “Find [L]imits” in Correlation Explorer, data points may end up on the border of the plot, making them hard to see.

Feedback questionnaire

Freely experiment with the UI modules for a while, and please answer the following questions. Any additional feedback you can think of is welcome.

On the last pages of this document there are screenshots that you can use as reference or to leave further comments directly onto the images. Make arrows, circle or mark things as needed.
Part 1: “Correlation Explorer”

1. What do you think about the green line in the scatter plot? Is it clear what it represents?

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is quite intuitive it represents a regression line.</td>
</tr>
<tr>
<td>Makes sense, it’s clear what the green line represents.</td>
</tr>
<tr>
<td>Good, yes.</td>
</tr>
<tr>
<td>Yes, it is clear I think. But you could always put an explanatory text in green next to the scatter plot to make it more clear.</td>
</tr>
<tr>
<td>Yes, at least for a user with knowledge in statistics. Makes it easier to relate to the correlation map.</td>
</tr>
<tr>
<td>Yes, very clear.</td>
</tr>
<tr>
<td>Yes</td>
</tr>
</tbody>
</table>

2. In the plot, how difficult is it to see and / or click on the data points?

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>It’s not difficult to see, however it’s sometimes difficult to click on the data points.</td>
</tr>
<tr>
<td>It gets a bit hard when several points are close together. It would be nice to be able to zoom into the plot.</td>
</tr>
<tr>
<td>If two data points are close to each other it’s hard to see which one is selected. Maybe it should be possible to also zoom in the scatter plot or toggle between data points. To zoom out would also be nice to get a better visual overview of the data points.</td>
</tr>
<tr>
<td>I think it is relatively easy. Perhaps it would be nice to be able to zoom in the scatter plot but I don’t know if it would be worth it to implement.</td>
</tr>
<tr>
<td>They’re easy to see but sometimes a bit hard to select (click on).</td>
</tr>
<tr>
<td>Easy.</td>
</tr>
<tr>
<td>They could be a little bigger. Also, selecting points requires some precision. Is it possible to always select the closest point when you click in the plot?</td>
</tr>
<tr>
<td>Those on the figure borders are hard to see.</td>
</tr>
</tbody>
</table>

3. How well do you think the plot works for finding data points that are outliers?

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>It helps to find the outliers from looking at the plot.</td>
</tr>
<tr>
<td>It is very intuitive I think.</td>
</tr>
<tr>
<td>Works well.</td>
</tr>
<tr>
<td>Good.</td>
</tr>
<tr>
<td>It works very well.</td>
</tr>
<tr>
<td>Works very well.</td>
</tr>
<tr>
<td>OK. Might be easier if the points where a little larger.</td>
</tr>
<tr>
<td>“Find [L]imits” should leave a margin, so that no points are placed on the borders.</td>
</tr>
</tbody>
</table>
4. At the right side of the UI window, several values are shown: POEM ID, R-value, P-value and image volume value. How useful are they?

<table>
<thead>
<tr>
<th>It's useful to get a detail about a particular patient from POEM-ID.</th>
</tr>
</thead>
<tbody>
<tr>
<td>They are very useful, but it would be much easier to navigate the data if it was possible to see the values just having the cursor over the value on the plot (like a data probe).</td>
</tr>
<tr>
<td>Good to know. But the position could be different; Group them as [R- and P-value] and [POEM ID and Image value] as the former stays the same and the latter changes with selected data point.</td>
</tr>
<tr>
<td>Good</td>
</tr>
<tr>
<td>I think they are useful but I would put POEM ID directly underneath “Selected image [V]olume” as they belong together (see pge 13).</td>
</tr>
<tr>
<td>They're very useful, however, it's sometimes hard to interpret. Examples: Which values are per-subject (eg POEM ID) and which are per-group (R-value). This also makes the P-value a bit ambiguous. I thought it was from a test between subject and group at first.</td>
</tr>
<tr>
<td>Nice, especially the POEM ID.</td>
</tr>
<tr>
<td>Image volume value, what does it stand for?</td>
</tr>
</tbody>
</table>

5. Is it clear how to switch between the 8 different correlation maps available (R-values and P-values)?

<table>
<thead>
<tr>
<th>Yes, it's clear enough.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes, but it gets a bit hard having to look at the file names. Since there are 3 parameters in the choice of the map (p/r, Jac/DXA, male/female) it would be easy to select them separately (eg with 3 drop-down menues).</td>
</tr>
<tr>
<td>Yes.</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes.</td>
</tr>
<tr>
<td>Yes, but I assume that every R-value map will have an accompanying P-value map. It would maybe make more sense to group them somehow.</td>
</tr>
<tr>
<td>Is it possible to have multiple dropdown menus for this, instead of the long filenames? They are a little hard to parse...</td>
</tr>
<tr>
<td>For example: Male / female, r value / p value, fat / jacdet, 151 / 156</td>
</tr>
<tr>
<td>Yes</td>
</tr>
</tbody>
</table>
6. There are three markers, one for each of the Coronal, Sagittal and Axial views. What do you think about their functionality?

<table>
<thead>
<tr>
<th><strong>To get more detail image and understanding in the interested area.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>I think they are nice, I like this feature also on other volume visualization tools (eg the MINC display tool). It would be better if it were possible to move the marker with a left click, and not just by dragging.</td>
</tr>
<tr>
<td>I would prefer if I could also just click on a voxel to change the marker’s position. Now I think it’s only possible to drag.</td>
</tr>
<tr>
<td>Good, easier to navigate.</td>
</tr>
<tr>
<td>Very good</td>
</tr>
<tr>
<td>Very good. The fact that the two other views change while you drag the marker makes it very pleasant to work with.</td>
</tr>
<tr>
<td>I would prefer to just click at the point I want, rather than having to drag the marker.</td>
</tr>
<tr>
<td>Good</td>
</tr>
</tbody>
</table>

7. What do you think about the layout?

<table>
<thead>
<tr>
<th><strong>Simple, easy to use even for beginner. However, it can’t do the full screen for the application.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Very clean and intuitive. Maybe the buttons are a bit too big, so it feels like the controls are a little too far away.</td>
</tr>
<tr>
<td>The thing I do not like is how the scroll wheel moves the slice when the cursor is NOT over one of the slice views.</td>
</tr>
<tr>
<td>It also feels that the controls are a bit interleaved, maybe it would feel better having the volume-related controls (coordinates, colormap, zoom) grouped together (eg on the left of the slice views) and well apart from the other plot-related controls.</td>
</tr>
<tr>
<td>Nice.</td>
</tr>
<tr>
<td>Would it be useful to see the correlation map overlaid on the anatomical image?</td>
</tr>
<tr>
<td>Good.</td>
</tr>
<tr>
<td>Very nice. I would only like buttons and drop down menus to be positioned next to the images to which they belong (see page 13).</td>
</tr>
<tr>
<td>First I did not understand what was meant by the [] parenthesis but it was quite easy to discover.</td>
</tr>
<tr>
<td>I like the layout. Big focus on what’s important (the images / plots). Not overly crowded with other UI elements.</td>
</tr>
<tr>
<td>Overall, nice. The +/- buttons for marker position and selected volume seem a little superfluous.</td>
</tr>
<tr>
<td>Good, maybe the coordinate numbers should not be in pink though.</td>
</tr>
</tbody>
</table>
Part 2: “Image Comparison”

8. In this UI module, you only see one view at a time (Coronal, Sagittal or Axial). What do you think about this? Is it clear how to change the view?

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>It’s clear how to change the view and I think it’s a good idea because we want to compare the image.</td>
</tr>
<tr>
<td>I think it is very clear and straightforward.</td>
</tr>
<tr>
<td>I think this is good. It’s clear how to change the view.</td>
</tr>
<tr>
<td>Maybe it would be hard to include all the views at the same time. It’s clear how to switch views.</td>
</tr>
<tr>
<td>I guess it would be nice to see all three views at the same time but it is likely that the screen would be overcrowded. I think one view at a time works well.</td>
</tr>
<tr>
<td>It’s good, allows the user to compare Undeformed, Deformed, ..., without the UI being too crowded.</td>
</tr>
<tr>
<td>I had no problems changing the view.</td>
</tr>
<tr>
<td>Yes, clear.</td>
</tr>
<tr>
<td>Yes</td>
</tr>
</tbody>
</table>

9. What do you think about the setting to switch the left-most image between undeformed image and "difference image"?

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>I don’t have any comment on this part.</td>
</tr>
<tr>
<td>I think it is ok, but I would rather alternate between Jacobian and difference, since it may be useful to see undeformed / deformed / reference / diff all together.</td>
</tr>
<tr>
<td>Good.</td>
</tr>
<tr>
<td>OK, but you forgot to put the [ ] around the D in the menu. Maybe you can move things around so you won’t need a drop-down menu for only 2 alternatives.</td>
</tr>
<tr>
<td>Pressing U for Undeformed doesn’t seem to work all the time.</td>
</tr>
<tr>
<td>I like it. But is it possible to find colormaps with different nuances of green. The light green nuances are a bit difficult to look at.</td>
</tr>
<tr>
<td>It’s good to be able to see both, a small thing though is that the grouping (undeformed and difference) makes it a bit unclear what the difference is.</td>
</tr>
<tr>
<td>I think it would make more sense to have the &quot;difference volume&quot; option for the Jacobian determinant image, ie the rightmost image. Both the Jacobian and difference image relate to the registration quality.</td>
</tr>
<tr>
<td>Good</td>
</tr>
</tbody>
</table>
10. Is it clear how to change the display range of the Jacobian image? What do you think about this functionality?

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>It’s clear how to change the display.</td>
</tr>
<tr>
<td>I think it has some problems, I will put the details in question 20 since it is a bit long answer.</td>
</tr>
<tr>
<td>Yes, this is good. In some instances it might be useful to also be able to change the display range for the anatomical images.</td>
</tr>
<tr>
<td>Maybe put min and max closer together? I don’t know much about what the Jacobian is for, so I can’t answer concerning the functionality.</td>
</tr>
<tr>
<td>My knowledge of the Jacobian is quite limited but wouldn’t it be useful to be able to display a range min &gt; 0?</td>
</tr>
<tr>
<td>Yes, it was clear, would like an “find limits”-button here as well. Grouping min and max together would also be good, now they’re a bit far apart.</td>
</tr>
<tr>
<td>The controls could be spread closer together. Also, would be nice to have an “auto” button, to display the full range.</td>
</tr>
<tr>
<td>It might be helpful to see which value range is actually present in the image.</td>
</tr>
</tbody>
</table>

11. What do you think about the layout?

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good but still I cannot make the application to fullscreen.</td>
</tr>
<tr>
<td>I think it is pretty neat, same considerations as in answer 7.</td>
</tr>
<tr>
<td>Nice layout. Subject data overlaid on reference subject data would be good. With a transparency option. To assess the registration.</td>
</tr>
<tr>
<td>Mostly good.</td>
</tr>
<tr>
<td>It is intuitive and nice.</td>
</tr>
<tr>
<td>The layout is good. See question 7.</td>
</tr>
<tr>
<td>Overall nice. I think the slider for scrolling might be replaced by a text field instead. Either you are just browsing, in which case the scroll wheel is fine, or you want to see a specific slice, in which case you may want to enter that numerically.</td>
</tr>
<tr>
<td>Good</td>
</tr>
</tbody>
</table>
General questions

12. What do you think about the keyboard shortcuts? (In [L]oad, [L] means that the “L” key is a shortcut for using this button.)

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>I don’t get that idea, until I read this question why you give bracket at “L”.</td>
</tr>
<tr>
<td>Very nice, it looks very standard and intuitive.</td>
</tr>
<tr>
<td>Useful.</td>
</tr>
<tr>
<td>Good.</td>
</tr>
<tr>
<td>Yes, once I got it, I think it’s good.</td>
</tr>
<tr>
<td>Didn’t use them during my task. But I would probably use them if I started using the program a lot.</td>
</tr>
<tr>
<td>To avoid confusion, you should probably not map the same key to different functions in different modules ([L]imits vs [L]oad).</td>
</tr>
<tr>
<td>Nice and clear.</td>
</tr>
<tr>
<td>Good</td>
</tr>
</tbody>
</table>

13. There are menus for changing colormaps and marker color. What do you think about the colormaps and marker colors?

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>It will give different detail with different colors.</td>
</tr>
<tr>
<td>I think it is very good to provide a choice of colormap. But I think it would be useful to have an additional colormap tailored to show correlations more intuitively, eg white = zero, and two diverging colors toward +-1. -1: blue &lt;- 0: white -&gt; +1: red. (Same for the Jacobian, which is centered on 1 instead.) (same also for difference images)</td>
</tr>
<tr>
<td>Inverted gray might be useful for some applications?</td>
</tr>
<tr>
<td>Good, but if I press M the color is switched to magenta, and if I press C nothing happens. Maybe remove these?</td>
</tr>
<tr>
<td>As I said before, the light green is very bright and difficult to look at, especially if it is shown as background, is it possible to choose a different green nuance. Otherwise, the maps are good.</td>
</tr>
<tr>
<td>They’re good. Really like that it is easy to change.</td>
</tr>
<tr>
<td>OK. Might consider other colormaps as well (perceptually uniform).</td>
</tr>
<tr>
<td>Good choice. Magenta seems rather unusual, but might be necessary in order to set it apart from the colormaps.</td>
</tr>
</tbody>
</table>
14. What do you think about the use of drop-down menus?

<table>
<thead>
<tr>
<th>It's good because it helps to change some parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I think they are OK, they are used where it is most reasonable to expect them.</td>
</tr>
<tr>
<td>Good.</td>
</tr>
<tr>
<td>It's good when there are many alternatives. Not so good when there are only 2 alternatives.</td>
</tr>
<tr>
<td>Useful.</td>
</tr>
<tr>
<td>I like them.</td>
</tr>
<tr>
<td>Fine.</td>
</tr>
<tr>
<td>Good</td>
</tr>
</tbody>
</table>

15. You can use the mousewheel to scroll through the volumes. Is this function working as desired?

<table>
<thead>
<tr>
<th>It's working well.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes. Only the scroll is a bit slow, it takes too much to go through the volume.</td>
</tr>
<tr>
<td>Yes!</td>
</tr>
<tr>
<td>It's pretty slow. Maybe add a shift + mouse wheel option to make it faster.</td>
</tr>
<tr>
<td>Yes. It would be nice to have the marker positions somewhat closer to the images so that you can easily monitor how they change while you scroll and reposition the marker. Maybe on top of or beside each image (see page 14).</td>
</tr>
<tr>
<td>Yes. A bit slow sometimes but not enough to ruin the workflow.</td>
</tr>
<tr>
<td>Yes. It is nice that the active window is marked.</td>
</tr>
<tr>
<td>Sometimes the mouse wheel affects other fields than intended. This might be impossible to change in Matlab though.</td>
</tr>
</tbody>
</table>

16. How is the response time of the applications? Did you experience any “lag”?  

<table>
<thead>
<tr>
<th>The response is good and there isn’t lag.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moving through the slices is a bit slow, and it takes a while to load a file from disc.</td>
</tr>
<tr>
<td>There is a lag time, but it’s quite short.</td>
</tr>
<tr>
<td>Good, no ‘lag’.</td>
</tr>
<tr>
<td>Yes, there is a slight lag.</td>
</tr>
<tr>
<td>Some milliseconds delay here and there but nothing major. Usually when working with the image views.</td>
</tr>
<tr>
<td>No problem.</td>
</tr>
<tr>
<td>Loading images takes a while, as does navigating through a volume. This might be impossible to fix in Matlab as well though.</td>
</tr>
</tbody>
</table>
17. Is there any data or functionality that these UI modules are lacking, that you would like them to have?

| **A data probe that shows the value under the cursor, when moving the pointer over any of the slice views, would be a very useful addition. Zoom with ctrl + mouse wheel would be very handy.** |
| **To add other imaging volumes. But perhaps this is out of scope for your project?** |
| **Se tidigare svar. See previous answers.** |
| **+ Copy the position from “Correlation Explorer” to ”Image Comparison”. You could do this automatically when using ”Load Saved ID”.** |
| **It would be nice to be able to change the position of the marker by the keyboard. It is easier than using the mouse.** |
| **It would be nice to be able to change the contrast of images in module 2. To click inside a volume to select a voxel rather than dragging the marker around.** |
| **Putting volumes on top of each other for comparison, eg deformed on reference.** |
| **Not that I can think of.** |
| **I can't think of anything.** |

18. What data or functionality in these UI modules seems unnecessary or redundant?

| **Maybe having a numeric index in the “select image volume” (Correlation Explorer) could be redundant, I guess the POEM ID could be used instead.** |
| **Reset position. At least in its current format. Could be useful if the user can set his/her own default position.** |
| **+ och – för ”selected image volume”.** |
| **Nothing I can come to think of.** |
| **Manual scroll sliders should not be necessary as all mouses have a scroll today. Maybe trim down the marker position, indication on where the marker is is good, but you probably don’t need the +/- or even the manual input.** |
| **+/- buttons, as mentioned before.** |

19. Did you find any bugs or strange behavior in the UI? Please describe it in as much detail as possible.

| **No, I do not think I encountered any bug.** |
| **No. Well done!** |
| **For a while I didn't get any image in ”Correlation Explorer” when I selected ”Image Volume”. But later I could not make the problem happen again.** |
| **No.** |
| **Had some issues with the cursor releasing the marker while dragging. Only happened once or twice during this session however.** |
| **No.** |
| **Image Comparison -> POEM ID sometimes seems to change randomly when pressing ”[L]oad saved ID”?** |
20. Any additional comments?

<table>
<thead>
<tr>
<th>I think the program is very good.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I think the range control for the Jacobian is a bit limited. In theory, the Jacobian should be centered on the value of 1 (no value changed, with values in (0, 1) denoting compression, and values in (1, +\infty) denoting expansion. Hence, it would be useful being able to window the Jacobian colormap on an interval around 1 (eg [0.7, 1.3]), but with the current controls it is not easy to do so, since the MIN is bounded \leq 0 (btw, values of the Jacobian \leq 0 are not physically meaningful and come from registration errors)). Also, it would be very valuable to be able to select different colormaps in the Image Comparison, one for the volumes and one for the Jacobian (which is visualized best with a diverging CM as mentioned in the reply to question 13).</td>
</tr>
<tr>
<td>In module 1 the “limit”-panel was a bit confusing until I realized I could open Jacobian images. You could maybe hide this until a Jacobian image is opened.</td>
</tr>
</tbody>
</table>
6 Implementation

Here the implementation of the UI is presented. The back-end of the most important features and the most challenging parts of the implementation process are discussed.

The implementation consists of a number of source code files that are collected in a GitHub repository. There is an m-file and a fig-file for every UI module (part 1 and part 2) along with some support scripts. The fig-files (generated by GUIDE) are binary and not readable. This code will be published on GitHub by the author after the completion of the project.

The data visualized by the UI is imaging data (MRI volumes in the \texttt{vtk} file format) and non-imaging data (numbers contained in tables). These originate from a dataset (POEM) studied within the Imiomics research project, and they are kept separate from the repository and will not be made public. The reasons for this are medical data privacy as well as the large file size of the image volumes.

More information about the source code can be found in the Appendix.

6.1 Processing image data

This section will explain the basics of digital image coordinate systems and how the 2D images in the UI are obtained from 3D data volumes.

6.1.1 The image volume coordinate system

This project has led to a deeper understanding of the coordinate systems of digital images, which can be helpful for anyone displaying images in MATLAB or similar programs.

In basic mathematics, the most common graphs look like this:

![Coordinate system](image)

Origo is in the lower left. The y-axis increases upwards, and the x-axis to the right. Coordinates are described in the form \((x, y)\).
In MATLAB, and many other forms of digital image handling, images are handled like matrices. Origo is in the upper left corner of an image and the coordinates are described as (row, column), like this:

![Coordinate System](image.png)

This can potentially cause confusion when handling digital images. It is important to note that the coordinate axes are discrete, not continuous, thus all image coordinates are integers. Furthermore, the coordinates start at 1, not 0, as 1 denotes the first row or first column etc.

In the image volumes of the POEM dataset, origo (which has coordinates (1,1,1)) is located on the patient’s right side and above the patient’s head, as shown in figure 19.

![Coordinate System](image.png)

*Fig 19: The coordinate system of the 3D data volumes, in relation to the patient.*

### 6.1.2 Creating 2D images from 3D volumes

One of the aims of the project, as stated in 2.4, was to “implement suitable data processing methods” – this means handling the image volumes. Based on existing applications, like Slicer, the choice was made to visualize the 3D image data as 2D slices aligned with the anatomical planes. This subsection will explain how the image processing was implemented.
There are many ways to show the anatomical planes of a human body, depending on the location and orientation of the point of view (the “camera”). For example, a coronal whole-body image can have the patient’s head pointing in any direction and the point of view can be either in front of or behind the body. The orientations of the images in the UI, as seen in the screenshots in previous chapters, were chosen to be intuitive for the user and to utilize the screen space as efficiently as possible.

The coronal and sagittal images are rectangles with the longer side roughly 4 times the size of the shorter side. They were placed vertically, so the patient appears to be standing. The axial image is almost square-shaped and requires less space, and it shows the patient lying down. These design choices mean that the UI must map the data from 3D volumes to 2D images in a specific way to achieve the desired point of view.

The following pseudocode shows the principle of how the mapping is implemented. First, a 2D “slice” is created from an image volume using the function squeeze. Then, imshow is used to generate the image. In two cases (the coronal and axial images) the slice needs to be transposed to obtain the chosen orientation (otherwise the coronal image would be horizontal).

```matlab
CoronalSlice = squeeze( Vol(:,,:,zval) );
CoronalImage = imshow( CoronalSlice' );

SagittalSlice = squeeze( Vol(xval,:,,:) );
SagittalImage = imshow( SagittalSlice );

AxialSlice = squeeze( Vol(:,yval,:) );
AxialImage = imshow( AxialSlice' );
```

A different dataset may have an entirely different coordinate system (origo is in another place in relation to the patient) and will require other operations to achieve the same viewing angles.
6.2 Common structures
Several UI functions and data structures are implemented in the same way (or in very similar ways) in both modules, and the most important of these are described in this section.

6.2.1 The current point
The “current point” is an important concept in the UI. This is a position in 3D-space (a voxel) described by the coordinates \((x, y, z)\). This is represented by three numbers in the handles structure:

\[
\begin{align*}
\text{handles.xval} \\
\text{handles.yval} \\
\text{handles.zval}
\end{align*}
\]

Most functions in the UI uses \(x\text{val}\), \(y\text{val}\) and \(z\text{val}\) in some way – either reading their values or changing them.

When the UI is opened, these coordinates are set to a starting value in the middle of the image volume. For the volumes in the POEM study, this middle point is \((128, 126, 128)\). The “Reset position” button simply resets the coordinates to these values.

The lower boundary for voxel coordinates is 1 and the upper boundary is the volume size in the dimension in question. Some safety features have been implemented to prevent users from giving input outside these boundaries, for example in the +/- buttons.

6.2.2 Markers
For the markers, an existing MATLAB object was used, the \texttt{impoint}[10]. In the code, they have been called “dragpoints” because they can be moved by click-and-drag. They are placed “on top of” an image.

An “impoint” object is created like this:

\[
\text{hpoint} = \text{impoint}(\text{hparent}, x, y);
\]

where \texttt{hpoint} is a handle to the newly created impoint, \texttt{hparent} is a handle to the parent axes (containing an image), and \texttt{x} and \texttt{y} are the coordinates on the axes where it will be placed.

The position of an already existing impoint can be updated using:

\[
\text{setPosition(hpoint, x, y)};
\]
The impoints have a callback function (which is set upon their creation) that is automatically called whenever the impoint changes position. This happens both when the impoint is moved by a user and when its position is updated by other functions. One important functionality in the UI is that when the user moves marker A, markers B and C need to also move accordingly. This was implemented by letting the callback function of A update the positions of B and C. To prevent an “infinite loop” where the callbacks of B and C activate and try to update the position of A, the following safeguard was built into all the callback functions.

A callback is only allowed to execute in its entirety when the impoint was moved by the user, and not when the position was updated by another function. This was implemented by adding a flag (handles.allowDPCB) to the handles structure that is set to 0 whenever an unnecessary callback will be triggered by the code inside a function. The first thing every callback function does is to check the flag value. If it is 1, the function executes as normal, but if the flag value is 0 nothing happens. For example, inside callback function A there is a block of code that updates the positions of markers B and C. Before that block is executed, the flag is set to 0, and after the block, the flag value is returned to 1 (otherwise any subsequent user input would not work).

The markers are updated whenever the current point is changed, and whenever the user moves a marker, the current point is updated.

### 6.2.3 Drop-down menus

Drop-down menus are called “pop-up menus” in the GUIDE interface, but here they will be referred to as drop-down menus which is a more accurate description of their function.

A drop-down menu is easily created by placing the object on the UI and then “filling” the menu. This is done, perhaps rather unintuitively, by editing the String property of the drop-down menu object to simply form a list with one menu option per row.

For example, the String belonging to the menu for changing the marker color looks like this:

```
Black
White
Magenta
```

In the callback function for the menu, we can use the get method to get the value of the currently selected option. This value is 1 for the first menu item, 2 for the second item and so on. As an example, the callback function for the “Marker color” drop-down menu looks like this:
function popupmenuMarkerColor_Callback(hObject, eventdata, handles)
% /…/
value = get(hObject, 'Value');
switch value
    case 1
        setColor(handles.dragpoint1, 'black');
        /…/
    case 2
        setColor(handles.dragpoint1, 'white');
        /…/
    case 3
        setColor(handles.dragpoint1, 'magenta');
        /…/
end
guidata(gcf, handles);

All the drop-down menus used in the UI work in a similar way, using switch-cases based on the menu value.

6.2.4 Colormaps

One use of the drop-down menus is to change the colormaps. To switch the colormap of an image that is already being displayed, use the colormap method like this:

    colormap(handles.axes1, jet);

The colormaps jet, gray, parula and hot are built-in colormaps in MATLAB. The colormap “BlueWhiteRed” was custom-built for this project. If used on an image with range [-1, 1], like a correlation map, it will start at blue (-1), then there is a linear transition to white (0) and then another linear transition to red (1). The implementation of BlueWhiteRed is the following:

    % Custom colormap BlueWhiteRed
    blue = [0 0 1];
    white = [1 1 1];
    red = [1 0 0];
    blueinc = [1 1 0];
    redinc = [0 1 1];
    BWRmap = [ blue
                blue + 0.1* blueinc
                /…/
                blue + 0.9* blueinc
                white
                red + 0.9* redinc
                /…/
                red + 0.1* redinc
                red];
The colormap is represented by a matrix with values in \([0, 1]\) which has 2 columns and any number of rows (the more rows, the smoother the transitions in the colormap can be). Any custom colormap can be created using the same pattern.

### 6.2.5 Zoom in and Zoom out

The *Zoom in* function essentially works by “cutting” smaller slices of the image volume, centered on the current point. There is a zoom number defined as a number between 0 and 3. Zoom number 0 is the normal view, and 3 is the greatest level of “zoomed in”. The Zoom in button raises the zoom number by 1, and the *Zoom out* button lowers it (until it’s back to 0).

The related values are defined like this in the opening function:

```matlab
handles.zoomNr = 0;
handles.radius = 20;
handles.xrange = 1:xmax;
handles.yrange = 1:ymax;
handles.zrange = 1:zmax;
```

The callback functions for the *Zoom in* and *Zoom out* buttons change \(\text{zoomNr}\) and reduce or increase the ranges of \(x\), \(y\) and \(z\). The radius affects how much the range changes with each step.

The functions for updating the images uses the \(\text{xrange, yrange}\) and \(\text{zrange}\) when creating slices of the image volumes to display.

### 6.2.6 Scroll through volume

When looking at an image in the coronal view, the user is shown a 2D slice from the volume that is parallel to the \(x\)-axis and \(y\)-axis with \(z\) at a fixed value. When “scrolling” through or traversing the volume, by using the slider or the mouse wheel, the view shifts to different slices along the \(z\)-axis.

In a coronal slice, the marker can be used to change the current point (voxel) in the \(x\)- and \(y\)-directions. The scroll function is needed to move in the 3\(^{rd}\) dimension. For sagittal and axial slices it works the same way, except the scrolling is done along the \(x\)-axis or the \(y\)-axis.

The scrolling is implemented slightly differently in the two UI modules, but in both there is at least one slider present. The slider can be used as an alternative to the mouse wheel. The user can click and drag the slider left or right. The slider callback function is executed once the user has finished the dragging motion and releases the slider. In the callback function, the following line of code
value = get(hObject, 'Value');

returns a value between 0 and 1. The value is 0 when the slider is in the left-most position, and 1 when it is in the right-most position. This value is then mapped to an integer between 1 and, for example, \( z_{max} \). The current point is updated with the new \( z \)-value, and the image is then updated.

In the Correlation Explorer, there is a view number in the handles structure that denotes which image is the currently selected one – this controls in which direction scrolling is done. The selected image changes when the user clicks on one of the three images, and a text (“Scroll through volume [MouseWheel]”) is shown above the image to indicate the current selection.

The implementation of Image Comparison also uses a view number but as all images have the same orientation, there is no function for selecting any of them. Instead the view number (and thus the direction for scrolling) is controlled by the selected option in the “View” panel.

### 6.2.7 Jacobian display range or limit values

Both UI modules have the following components for controlling the display range of Jacobian determinant images:

- Sliders for two values (upper limit and lower limit)
- Number boxes for two values (upper limit and lower limit)
- Button “Find limits”
- Button “Set to (0, 2)"

All of these change the display range. For example, the “Set to (0, 2)” button has a callback function that does the following:

```matlab
set(handles.axes3, 'CLim', [0 2]);
```

It also updates the sliders and number boxes to reflect this change. Similarly, all the other components are tied to each other to update when any one of them is changed.

The functions for all these components contain rules that keep the upper and lower values within a predetermined range. The range of the upper limit is \([1, 5]\) and the range for the lower limit is \([-3, 0.99]\). The reason why the lower limit cannot exceed 0.99 is that it must never be the same as the upper limit. If the user were to set both limits to 1, then updating the image display range would cause an error in MATLAB. Hence, during the implementation a boundary of 0.99 was set for the lower limit. It is close to 1 but not close enough to cause a problem.

The “Find limits” button (keyboard shortcut F) finds the highest and lowest values in the image slice, adjusts them if needed to be within the allowed range, and then sets
the limits to these values. This is used to display the Jacobian determinant image with “as much quality as possible”, as these images have a much more varied range than the other images.

In the Correlation Explorer, any changes to the display range limits also affect the scatter plot, in that the y-axis is altered when displaying Jacobian determinant values. When using “Find limits”, small margins are added to the y-axis max and min values so that the outlier data points will fall inside the plot and not be hidden by the border.

6.3 Part 1: Correlation Explorer

The part 1 module displays one image volume at a time (either a correlation map or a regular MRI volume of an individual patient) but it requires that data from more than 100 image volumes are available for the scatter plot. To solve this, a “storage class” was created and the data was packaged for easier access. This is presented in further detail in 6.3.1. The following subsections present the other features unique to the part 1 module.

6.3.1 Data structures: The “storage class”

One of the challenges posed in this project was the size of the files involved. Each of the ~300 vtk-files are in the 100 MB size range. When these files are kept in the MATLAB workspace they take up a rather large amount of memory. Additionally, the script that imports vtk-files into MATLAB takes some time to execute for every image volume. When the UI was built, a choice had to be made between speed and memory usage.

It could be possible to only read data directly from the vtk-files when it is needed. However, for the scatter plot it is necessary to read a voxel in ~150 different data volumes every time the marker is moved. With the current method of reading vtk-files, that would make the UI slow. It is possible this would work better with another method or in a different environment than MATLAB. To ensure the speed of the UI, the choice was made to hold the data volumes in the MATLAB workspace memory while the UI is running (however, there is a way to reduce the memory usage, which will be presented shortly).

There is some non-imaging data that needs to be stored together with the image volumes: POEM ID numbers, DXA fat values and a 0 or 1 denoting a male or female subject. It is also important to have the fat fraction and Jacobian volumes of the same subject ID associated with each other. For this reason, a “storage class” was created to store all the data needed. It is defined as follows:
classdef imiFatVol
    %IMIFATVOL Wrapper class for Imiomics fat volumes and data

    properties
        % numbers
        poemid % int
        female % logical
        dxaval % float
        % 3D volumes
        DefVol
        JacDet
    end

    methods
        ...

An “imiFatVol” object stores all the data related to a certain subject from the POEM study.

Another issue encountered during the project was which numeric type to use for the volumes in MATLAB. If they are stored as double, the numbers have very high precision but also take a lot of memory space (64 bits per voxel, and every volume has 256*252*256 voxels). For the purpose of this UI a lower precision is sufficient. Therefore, the memory usage could be greatly reduced by converting the image volumes in the storage class to the int8 format, which has the much smaller numeric range [-127, 128].

In the image data, the fat fraction volumes should ideally have values limited to [0, 1] but there are some negative values due to calculation errors. The reasoning behind the choice of int8 instead of an even smaller “unsigned int” type, was that the UI should preserve these negative values instead of discarding them, so that it might be possible for a user to analyze them.

When loading volumes into MATLAB they are originally created as doubles. Then they are “packaged” for storage through the following steps: First, the volume values are multiplied by 100 (a so-called “scaling factor”), so the range is changed to (roughly) [0, 100]. Then, the volumes are converted to int8 – this way, more of the range of int8 is used.

Later, when the UI reads the numbers from the volume they are “unpackaged” by turning them back into doubles and then scaling them down (by dividing by 100) to retrieve the true values.

The Jacobian determinant volumes were a bit harder to handle since their values vary greatly. The assumption was made that most of the values are within the range [-3, 5] (in theory, there should be no negative Jacobian determinant values but there are negative values in the image volumes due to calculation errors). The Jacobian
determinant volumes were packaged in the same way as the fat fraction volumes but using a scaling factor of 20 instead of 100.

The script `create_data.m` must be run before part 1 can be used on a new dataset. It loads the image volumes from the provided vtk-files, packages the volumes and creates an array of ~300 `imiFatVol` objects from the volumes and tables with DXA values and other non-imaging information. Running this script takes ~10 minutes, and it saves the resulting array as a mat-file on disc. This file, called `uidata.mat`*, is ~2 GB in size, and it is loaded into the MATLAB workspace when the UI is opened. For this reason, opening the part 1 module takes about 2 minutes.

The correlation maps are also included in `uidata.mat`, but they are not packaged, as there is only 8 of them. It is worth noting that both `create_data.m` and the part 1 module use a good amount of RAM memory, but it works well in the current environment (a desktop computer with 32 GB of RAM).

For part 2 an entirely different approach is used, as only 5 data volumes need to be stored in the MATLAB workspace at any time. Here, the UI loads the needed volumes directly from the vtk-files, and there is no need to package and unpack them. When the user switches the POEM ID, 5 new volumes are loaded (which takes only a few seconds) and the old ones are discarded.

*Both the image volumes and the tables with non-imaging data were only stored locally during this project and will not be shared along with the source code (both for data privacy reasons and filesize reasons). The same is true for the generated file `uidata.mat`.

### 6.3.2 The correlation map menu and parameter buttons

The correlation map drop-down menu allows the user to choose the displayed correlation map by its filename. It can be cumbersome to handle these long filenames, so the parameter buttons were added to make it easier to both select a correlation map and to quickly see which one is selected.

There are 8 correlation maps which are described by 3 parameters: Male or female, P-value or R-value and fat fraction or Jacobian determinant. Because of this, it was suitable to use groups of 2 radio buttons for each parameter. MATLAB GUIDE has a feature called “button groups” – when there are at least 2 radio buttons in a button group, they are connected in such a way that no more than one button can be selected at any time.
In the same panel as the parameter buttons, the R-value and P-value of the current point (voxel correlation values) are also shown. These values are taken from the two correlation maps appropriate for the currently selected parameters.

6.3.3 The “Scatter plot of data points” plot
As was described in 6.3.1, all the image volumes (except for the correlation maps) are stored in a large array of objects of the custom-made class imiFatVol. Each such object holds one fat fraction volume and one Jacobian determinant volume.

The scatter plot is updated whenever the current point changes, or when the data type is switched between fat fraction and Jacobian or between male and female subjects. Every time the plot updates, the aforementioned array is traversed, and for every volume of the current type, the value of the voxel at the “current point” coordinates is read. Two smaller data arrays are also held in the handles structure – one for female and one for male values, rowMale and rowFemale. When the imiFatVol array is traversed, the values found are stored at the appropriate place in the appropriate “row”-array. There is a lookup table that maps the index in the imiFatVol array to the index in the male or female arrays, and also to POEM ID numbers. Thus, the values in these arrays are changed every time the marker is moved.

There are also data arrays for the DXA values of the male and female subjects, but these arrays are always the same. The scatter plot is created by mapping the DXA values to the fat fraction or Jacobian determinant values.

The green line in the plot is generated using polyfit and polyval. Polyfit uses a least-squares method to find the linear function that best approximates the data, and polyval is used to draw the line in the plot.

A blue square marks the data point of the currently selected subject (as shown in figure 20). The lookup table mentioned above is used to find the index in the male or female row-array that is connected to the current POEM ID. The blue square is generated in the same plot function call as the data points, on top of the appropriate point.
6.3.4 The “Data cursor” replacement
An important piece of functionality in part 1 is for the user to be able to click on data points in the scatter plot to select them. The currently selected point is marked by a blue square as seen in figure 20. Changing the selected data point updates the current POEM ID number.

![Scatter plot of data points](image)

*Fig 20: The scatter plot in part 1, where the data points can be clicked by the user to be selected. The selected point is marked by a blue square.*

In GUIDE, there are several pre-defined Toolbar items that can be added to a UI using the Toolbar editor, like save, zoom in/out, pan, and rotate. One of them is the “data cursor” tool.
The data cursor tool can be used to, for example, select any data point in a graph. It creates a small pop-up box that displays the x- and y-values in the point, and it can also be modified to show other values, like in this case the index in the row-array.

The first implementation attempt utilized the data cursor, however, it did not function as desired in this UI. It turned out to be difficult to modify the tool as needed, since trying to call other functions from its callback function caused problems.

Instead, a replacement for the data cursor was built in the following way:

A **button down function** was added for the axes (called axesDat in the code) that holds the scatter plot. This way, a function call is executed when the user clicks on the plot. Within this function, it is easy to retrieve the x- and y-coordinates (within the plot) of the point that was clicked by using the **CurrentPoint** property:

```matlab
point = get(handles.axesDat, 'CurrentPoint');
```

After that, the function needs to find the data point which is closest to the clicked point. This calculation included Pythagoras’ theorem \((x^2 + y^2 = d^2)\), where \(d\) is the distance. A large “weight factor” was added to the y-values to compensate for the fact that the x-axis has values of a much greater magnitude. Once the closest data point has been identified, the selection marker (the blue square) is moved, and all the appropriate variables – most importantly the POEM ID number – are updated.
### 6.4 Part 2: Image Comparison

In this module, only one subject is studied at a time. Five images are compared, which means only five image volumes need to be loaded into memory at any time. This is not much, so the volumes can be kept at their full size (with no need for type conversion). The folders with vtk-files are searched and the appropriate files are identified, based on the POEM ID that is present in the file name, and loaded into MATLAB as needed.

In this module, the choice was made to only show one view at a time (coronal, sagittal or axial) in order to have room to display five images simultaneously. The user can switch between the different views using the radio buttons in the “View” panel (which is implemented using the button group feature).

#### 6.4.1 The “Load saved ID” button

When the user presses the “Load saved ID” button, the latest POEM ID that was selected in part 1 will show up in the POEM ID number box and the relevant images will be loaded. This is possible because whenever the POEM ID is changed in part 1, the number is saved to a small file (IDselected.mat) in the current directory. Part 2 simply reads this file to retrieve the selected POEM ID. This function will not work as intended if the user changes the current directory in MATLAB between using the modules.
7 Discussion

The UI that has been built during this project represents one possible way to solve the visualization problem mentioned in the beginning of this report. The testing process verified that the UI fulfills the users’ needs and is “good enough”. As is always the case, more work could be done to improve the functionality of the implementation - some improvements suggested by the user testers are listed in section 8.1.

An alternative implementation could have been built in Slicer, since Slicer has an interface for coding in Python. In that case, a lot of Slicer’s already existing functionality could have been used. However, the author had no prior experience with Slicer while being very familiar with MATLAB. Using GUIDE meant learning about a new aspect of the MATLAB toolbox while still having time to build a working implementation.

For any project larger and more complex than this one (especially if more people are involved), a more detailed written requirements specification would be extremely useful. One of the greatest challenges was formulating the detailed requirements based on oral feedback, and to surmise which parts would be too difficult or time-consuming to implement.

Lastly, to finish the project in a reasonable amount of time it was necessary to limit the scope of the UI to only handle a small subset of the existing data. A UI that is able to handle all the data related to Imiomics would be possible but more complicated to build.

7.1 Advantages and disadvantages of MATLAB GUIDE

MATLAB GUIDE has been suitable for the particular development context of this project but is has some quirks that users should be aware of. Here its major advantages and disadvantages will be listed.

**Advantages**

- The GUIDE window makes changing the size and position of images and graphs very simple.
- It is relatively fast to learn and you get a lot of functionality “for free”.
- Almost anything you can use in MATLAB – functions, objects, data structures – you can also use in a GUIDE UI.
Disadvantages

- GUIDE auto-generates a new m-file when you save a fig-file with a new name. To avoid losing your code, refrain from changing the name of your figure.
- GUIDE runs on MATLAB which is not the fastest or most memory-efficient environment.
- It seems impossible to change the standard appearances of graphics objects like buttons and sliders.
- The callback functions give some ease of use, but also some functionality restrictions. Example: When using a slider, GUIDE will only call the callback function once after the slider movement has finished. Sometimes, like when the user is scrolling through an image, it would make more sense to evoke a function continuously during the slider movement.

In conclusion, GUIDE is suitable for making prototypes of graphic user interfaces when the data accompanying the interface is already handled in MATLAB or is easily imported there.

7.2 Limitations of this work

Because of the complexity of the data involved in Imiomics, only a small subset of the data was used in this project (only the fat fraction of MRI images, and only one non-imaging parameter, the DXA fat value). To accommodate all image data, and all non-imaging parameters, a more complex UI would be needed.

The user test was essential in evaluating the implementation, thus it is important to be aware of how the testing procedure could affect the validity of the test results. Here the known limitations of the testing procedure will be listed.

Limitations of the user test

- Only 8 participants.
- The majority of questions were qualitative rather than quantitative.
- The participants had at least some basic knowledge of image volumes and Imiomics.
- Questions were chosen to cover most of the important functionality, while attempting to keep the test as short as possible.
- The answers to the few quantitative questions (like the testers’ previous experience with similar software) have only been summarized and not correlated to other feedback, as the sample size is small.
• Only the UI modules themselves were tested, not the process of preparing the data. Before each test, the author started up MATLAB GUIDE and launched the UI modules.

• More could be done to accommodate users with different ability levels. For example, a user with color-blindness could hopefully find at least one of the provided colormaps useful, but this aspect has not been tested. In its current state, the UI requires the user to be able to use the mouse with some precision. For any software that is to be released to the public, inclusion issues should be taken into consideration during development and testing.

After the user test, the improvements suggested by the users were either implemented or added to the list presented in Future work. In the cases where several people made similar suggestions, the suggestions were combined into one and given a higher priority.

All improvements suggested in the user feedback are presented in this report, either listed under “Implemented improvements” in chapter 5 or as “Suggested improvements” in section 8.1. The decision on which improvements to implement before the end of the project was made by weighing the priority based on user feedback against the time required for implementation. Several small but critical improvements were made, leaving some larger ones for Future work.

7.3 Conclusion

As the delivery of this project, a UI has been developed. The UI was tested and overall deemed useful by the testers, and it was subsequently improved based on their suggestions. It can be used as a prototype for user interfaces of future software related to Imiomics.

Since a working UI has been built, there is at least one possible solution to the visualization problem stated in the beginning of this report. The UI represents one viable visualization which fulfills the needs of the users as well as possible based on the time and resources spent on constructing it.

The project has also been a learning experience, and the knowledge gained has been collected in this report. Some chapters can be used as a beginner's introduction to MATLAB GUIDE.
8 Future work

The Imiomics pipeline has recently passed the proof-of-concept stage and is still being developed and improved upon. In the future it may be distributed for wider use as a standalone software package or hosted on some web platform. At that time, the UI presented in this report could possibly be used as a prototype for part of the UI of said software or web application. The visualization and functionality used here could be replicated or reworked as needed.

8.1 Suggested improvements

Here are listed, in no particular order, the improvements suggested in the user feedback that have not yet been implemented.

- Make it possible to zoom in and out in the scatter plot. (Data points that are close together can be hard to see.)
- Make it easier to select data points by clicking close to them instead of having to click exactly on top of them. (This is hard to implement because of a technical limitation in GUIDE.)
- Make a probe near the cursor that shows the value when the cursor is held over a data point in the scatter plot.
- The function of scrolling through a volume using the mouse wheel should only be active when the cursor is over an image, so it is not activated by mistake.
- Make it possible to scroll faster, for example using shift + mouse wheel.
- Make the UI windows scalable and/or full screen. (Currently the UI scalability is turned off, as it is difficult to maintain the layout when scaling. The objects move around in unpredictable ways if the window size is changed.)
- The sliders and the +/- buttons are possibly redundant and could be removed.
- Make it possible to change the display range of all images, not just the Jacobian determinants.
- Make a function for showing the deformed image overlaid on the reference subject, with a transparency option, to assess the registration.
- In addition to the POEM ID, also save and copy the marker position from part 1 to part 2.
- Implement borders that prevent the markers from being dragged outside the image (which can cause the (x, y, z)-values to go below 1 or above the max allowed value).
References


3D Slicer is accessible at https://www.slicer.org/ (available on 2018-05-31).


Article is accessible at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5471941/. ENIGMA-Viewer is accessible at: http://enigma-viewer.org/HBM-16-0075.html (both available 2018-06-05)


The Nora medical imaging platform is accessible at http://www.nora-imaging.com/ (available 2018-06-05)


Appendix: Code information

The source code related to this thesis project will be available in a GitHub repository created by the author (github.com/AnnaRipley/UIforImiomics) after the completion of the project.

The repository includes the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Readme.txt</td>
<td>Contains instructions</td>
</tr>
<tr>
<td>create_data.m</td>
<td>Run once for every new dataset, to create data structures.</td>
</tr>
<tr>
<td>dell_script.m</td>
<td>Is called by dell.m</td>
</tr>
<tr>
<td>dell.m*</td>
<td>Is called by dell.fig (when running the UI part 1)</td>
</tr>
<tr>
<td>dell.fig</td>
<td>Opened by GUIDE</td>
</tr>
<tr>
<td>del2_script.m</td>
<td>Is called by del2.m</td>
</tr>
<tr>
<td>del2.m*</td>
<td>Is called by del2.fig (when running the UI part 2)</td>
</tr>
<tr>
<td>del2.fig</td>
<td>Opened by GUIDE</td>
</tr>
<tr>
<td>imiFatVol.m</td>
<td>Storage class used in part 1</td>
</tr>
<tr>
<td>vtk_read_header.m</td>
<td>Volume loader script</td>
</tr>
<tr>
<td>vtk_read_volume.m</td>
<td>Volume loader script</td>
</tr>
</tbody>
</table>

The **scripts** `vtk_read_header.m` and `vtk_read_volume.m` were made by a member of the research group and are used to convert vtk-volumes into MATLAB data volumes.

The other m-files are written by the author, apart from the auto-generated skeleton code (created by GUIDE) in the m-files marked with (*) The fig-files are binary files that are generated by GUIDE and contain all the graphics objects and their properties.

Note that the MRI volumes from the POEM data set (as well as the tables of non-imaging data) will not be shared, as they are medical research data, and the volumes are very large files (~100 MB per file). The image volumes were only stored locally during this project.
However, the UI code could potentially be modified to handle a similar set of image volumes and non-imaging data. See instructions in the Readme file for starting the UI and “loading” data.

The MATLAB version used during the project was R2018a.