

# Multicontrast interaction platform

---

Arvid Rudling



UPPSALA  
UNIVERSITET

**Teknisk- naturvetenskaplig fakultet  
UTH-enheten**

Besöksadress:  
Ångströmlaboratoriet  
Lägerhyddsvägen 1  
Hus 4, Plan 0

Postadress:  
Box 536  
751 21 Uppsala

Telefon:  
018 – 471 30 03

Telefax:  
018 – 471 30 00

Hemsida:  
<http://www.teknat.uu.se/student>

## Abstract

### Multicontrast interaction platform

---

*Arvid Rudling*

The use of magnetic resonance (MR) images involves navigation in multiple dimensions, which may comprise not only volumetric images but multiple representations of the same tissues with varying contrast parameters - multispectral. Inspection and processing of this information puts special demands on human-computer interaction; this master thesis is a study in the design and development of an open source platform for processing volume image data. Images are processed using basic data structures implemented in the platform, or through the common Insight segmentation and registration ToolKit (ITK). The project includes an application for inspecting bispectral MR images using a 2D histogram - this scheme was adapted into a tool for the common task of image segmentation. Segmentation performance was evaluated against automated test data produced in an earlier study and manually produced segmentations. The complete workflow for interacting with image data was evaluated in user tests. The 2D histogram used for segmentation was found comprehensible by novice as well as experienced users. Interaction mapping between image and histogram was found to have more impact in understanding the histogram than its theoretical background. Segmentation results were limited in raw form, but approached those from automatic clustering when basic post-processing was applied. The minimum-effort API design for defining user controls demonstrated good usability in testing.

Handledare: Joel Kullberg  
Ämnesgranskare: Hans Frimmel  
Examinator: Elisabet Andréddóttir  
ISSN: 1650-8319, UPTec STS 07 019  
Sponsor: AstraZeneca



UPPSALA  
UNIVERSITET

Teknisk- naturvetenskaplig fakultet  
UTH-enheten

Besöksadress:  
Ångströmlaboratoriet  
Lägerhyddsvägen 1  
Hus 4, Plan 0

Postadress:  
Box 536  
751 21 Uppsala

Telefon:  
018 – 471 30 03

Telefax:  
018 – 471 30 00

Hemsida:  
<http://www.teknat.uu.se/student>

## Abstract

### Interaktionsplattform för multispektrala bilder

## Multicontrast interaction platform

---

*Arvid Rudling*

Användning av magnetresonansbilder (MR) innefattar orientering i flera dimensioner, som kan utgöras av såväl volymbilder som flera avbildningar av samma vävnad med olika kontrastparametrar – multispektrala bilder. Undersökning och behandling av multispektrala volymer ställer speciella krav på människa-datorinteraktion. Det här examensarbetet är en studie i design och utveckling av en plattform för behandling av volymbilder, med öppen källkod. Bilder kan behandlas antingen med robusta datastrukturer som ingår i plattformen, eller med bildbehandlingsbiblioteket ITK. Examensarbetet innehåller även en tillämpning för att undersöka bispektrala MR-bilder med hjälp av tvådimensionella histogram – denna metod utvecklades till ett verktyg för den återkommande uppgiften att segmentera bilder. Precisionen vid segmentering jämfördes med data från en tidigare undersökning av automatisk segmentering, och med manuellt segmenterade bilder. Plattformens arbetsflöde för bildbehandling undersöktes i användartest. 2D-histogrammet för segmentering var hanterbart för både datorovana och erfarna användare. Interaktionen mellan histogram och bild visade sig ha större betydelse än den bakomliggande teorin för att nå förståelse hos användarna. Segmenteringsresultatet blev i obehandlad form ordinärt, men med enklare efterbehandling uppnåddes samma nivå som den automatiska segmenteringen. Användartestet visade att plattformen ger reglage med hög användbarhet genom sitt minimalistiska API.

Handledare: Joel Kullberg  
Ämnesgranskare: Hans Frimmel  
Examinator: Elisabet Andrésdóttir  
ISSN: 1650-8319, UPTec STS 07 019  
Sponsor: AstraZeneca

# Table of contents

<b>1. Platinum: a medical image interaction platform</b>	<b>3</b>
1.1. Introduction	3
1.2. Goals	6
1.3. The name	8
1.4. Implementation	8
<b>2. Multispectral segmentation</b>	<b>18</b>
2.1. Introduction	18
2.2. Implementation	20
2.3. Quantitative test	21
2.4. User test	22
2.5. Conclusions	26
<b>3. Appendices</b>	<b>29</b>
3.1. Quantitative test results	29
3.2. User test results	32
3.3. Class reference	34
3.4. Class diagram	36
3.5. References	37

# Expressions & acronyms

## **API**

*Application Programming Interface*, a standardized set of programming language instructions for using a software library, the “vocabulary” of the library.

## **Axial/coronal/sagittal**

Medical terms for the bodily directions bottom-up, back-front and left-right, respectively.

## **Clustering/classification**

Clustering is a data mining technique for partitioning similar objects into distinct groups using some distance measure. The underlying statistical procedure that is performed in clustering is known as *classification*.

## **CMake**

A software development tool that enables one definition list to produce projects for various computing platforms and development environments.

## **Copyleft**

A licensing concept aimed to counteract negative effects of copyright (e.g. inefficient information usage) by allowing usage, modification, as well as redistribution provided that the copyleft licensing is retained on any derivatives. This is intended over time to create a “viral” growth of the canon of copylefted material.

## **FLTK**

*Fast Light ToolKit*. A software library providing the graphical user interface in Platinum, enabling the software to have a uniform look and feel across multiple platforms.

## **FOSS**

*Free and Open Source Software*. Software which can be used, copied, studied, modified and redistributed with little or no restriction. This implies that the software is also without cost and it is one of the most stringent definitions of “free software”.

## **ITK**

*Insight segmentation and registration ToolKit*. An open source software library providing registration (alignment of multiple images), segmentation and image file operations.

## **Likert scale**

A questionnaire measurement scale where respondents specify their level of agreement to a statement.

**MPR**

*Multi-Planar Reformatting*, the technique of using slice-wise (tomographic) data to construct images along other axes, typically sagittal or coronal.

**Segmentation**

The process of partitioning a medical image (usually a volume) into separate parts: organs, tissue types, pathologies.

**Tomography**

The process of imaging a solid body in sections, or slices.

**UI**

*User interface*. The means by which a user interacts with a computer.

**VTK**

*Visualization ToolKit*. A comprehensive software library for scientific visualization, including tessellation, volume rendering and combining 2D and 3D images. There is a VTK file format (suffix *.vtk*) that stores multidimensional images and various kinds of shapes.

**XML**

*Extensible Markup Language*. A general-purpose markup language reminiscent of HTML, providing a standardized and extensible way to record structured data

# I. Platinum: a medical image interaction platform

## I.1. Introduction

### Medical imaging

Medical imaging is one of the primary tools in medical diagnosis and research. Images are used to assess damage from accidents or disease, prepare for and evaluate surgery and other treatment, and to study body structure and function for research purposes. Of the non-invasive sources of internal body images, the common include:

- (X-ray) Computed Tomography (CT)
- Magnetic resonance imaging (MRI)
- Single photon emission tomography (SPECT)
- Positron emission tomography (PET)
- Ultrasound (US)

Among these MRI stands out for providing excellent and customizable soft tissue contrast coupled with high resolution full body-reach, while considered harmless since it does not rely on ionizing radiation. Its main drawbacks are the relatively slow speed of image acquisition and physical hurdles caused by the bulky scanner and its large magnetic fringe field. The imaging modalities listed above generate 3-dimensional images – *volumetric images*. This data cannot be viewed directly; the volume has to be somehow rendered into a flat 2-dimensional image or presented interactively on a computer monitor. In many cases, particularly for research, more information can be extracted through post-processing and measurements on the volume data.

### Master thesis project

The imaging platform described in this thesis project is being developed for the *Image-and function center* of Uppsala university hospital. The department is the hospital's main clinical imaging hub and hosts numerous researchers pursuing various imaging-related projects, including digital image processing.

In this environment, image processing algorithm development tends to be a multi-stage cycle, where several steps are required to evaluate each change in the developing software. The steps include coding, compiling, execution – which may comprise multiple steps including specifying arguments – and evaluation by using a separate image viewer. The imaging platform is intended to aid these research projects in reducing the development cycle time and effort, allowing scientists to focus on their research:

- A framework is provided for using the ITK and VTK medical image analysis libraries.
- A durable user interface allows for experimentation with parameters and processing order for the evaluation of new algorithms without recompilation.
- An integrated viewer facilitates direct evaluation of processing results, both cutting time and allowing more frequent and detailed review of algorithm output.

Though created at the department's MRI branch the platform is intended to eventually be used with all imaging modalities and to promote synergistic usage among them.

This master thesis is an applied study in user-centered software design. The development and testing of the software has been performed with support of the MRI branch of the Uppsala university hospital, whose staff are among the potential users of the software. These connections allowed the project to be designed with the requirements of both individual users and the general medical user group in mind. The design process takes into account user interface, software architecture, implementation and the process of technology adoption (including licensing) of the final product. A wide perspective like this is suitable for my MS program, Science and technology studies (STS).

### Similar projects

Since many medical image processing applications use a handful of rendering and interaction modes (particularly axial slices, arbitrary MPR slices and volume rendering), they benefit substantially from interaction frameworks. Algorithms may be specific to a particular project, but the requirements on a basic user interface remain the same. Consequently there are a number of frameworks for similar purposes available:

- *Medical Imaging Interaction Toolkit* (public open source project at Sourceforge.org)  
The MITK has a purpose similar to this platform; it is however more general-purpose (no preset workspace with interaction widgets). Neither does it provide any ready-made interface to a UI toolkit, as Platinum does with FLTK. Developing Platinum using MITK was considered but would introduce a large dependency only being utilized as a special case; the generality available through MITK is not needed in the intended applications for this platform.
- *Julius/JuliusLight* (Caesar research center)  
The Julius framework and the Julius Model Viewer provides a framework and ready-made user interface much like Platinum. It is distributed as free software (not open source), under the condition of non-commercial usage.
- *MeVisLab* (MeVis research GmbH)  
User interface framework with IDE (Integrated Development Environment) for designing image processing applications. A commercial product with cost-free version for non-commercial usage.
- *VolSuite* (Ohio supercomputer center)  
Promising open interaction platform under development. Built with FLTK and C++, but unlike Platinum there is no ITK or VTK integration. No inherent multi-contrast visualization functionality, but a capable user-configurable OpenGL volume rendering pipeline. Rendering, I/O and processing algorithms are added as modules. The software is very flexible - including user tracing and session management facilities - but with limited support for volumetric image I/O at the time of writing.
- Several vendor-specific research platforms, such as *PRIDE* (Philips Research Integrated Development Environment). Despite being open sourced, its corporate connections disqualifies it from being FOSS: usage requires a collaboration agreement with the vendor in question.



The demands of the platform call for a deployment-ready independent and interoperable system. It still has to be full-featured for its purpose: *to facilitate the development of specific application projects in medical image analysis*. As described further in the *Goals* section, the user interface should not allow more flexibility (and consequential cognitive load) than necessary for the expected applications.

## Licensing

Platinum was designated from the start as an open source project. The major advantage of open source licensing is the prospect of widespread adoption. If adoption is successful substantial “free” improvements can be expected since open source code invites contributions by its users to a higher degree than other common practices like user surveys or bug tracking. Open source licensing is becoming increasingly popular, occasionally a prerequisite, in academic environments. Freely available code and distributed development effort makes the development process a software counterpart of the scientific peer review mechanism, and even more disclosed at that. Being first and foremost as a research tool this makes open source a suitable distribution model for the intended audience. The disadvantages of open source licensing includes ruling out the possibility of commercialization – though not necessarily monetization, as there are numerous business models utilizing open source software. There are a number of free software licenses available, most are either:

- **based on the GNU General Public License (GPL)**

The GPL (and its limited library counterpart LGPL) is a common and popular license used in about 70% of current open source software projects, as indicated by software distribution site Freshmeat<sup>1</sup>. It uses the concept of *copyleft* to facilitate a multiplication effect where any derivatives (which any person are allowed to create) also have to be GPL-licensed. This condition, however, only applies when *distributing* the derivative work; alterations and derivatives can be made in private regardless. To comply with the GPL, a licensee has to make available the source code along with the executable program.

The *Lesser GNU General Public License* (LGPL) is available as an option for library code. It has the same conditions for the library itself, but unlike the GPL it allows non-GPL software linking with (using) the library. This creates an interface where proprietary software can combine with and create momentum for GPL projects, these projects will nonetheless remain free software in themselves.<sup>2</sup>

- **based on the Berkeley Software Distribution (BSD) license**

The BSD is a so-called *permissive license*, making it closer (in terms of author rights) to releasing software into the public domain. This stance may also be described as *copycenter*, indicating its moderate approach to free software; the licensing requirements only amount to acknowledging the original authors.<sup>3</sup>

- **a license tailored for a specific (typically corporate) organization or FOSS project**

e.g. Apple Public Source License, European Union Public License, Mozilla Public License, Nokia Open source license, Python license

Since the Platinum project was intended to be and remain open source, and provide value by being a tool for its maintainers rather than sold as a product, copyleft licensing

would not limit the intended use. A more permissive license might in fact reduce the amount of contributions in a later stage, as potential contributors may be tempted by the possibility of instead turning the derivative work into a product, whether or not that is a viable prospect.

With the above considerations in mind, after discussion with representatives of both academic and corporate potential users, the Platinum software was decided to be deployed as a library and licensed through LGPL. This will allow corporate users to develop proprietary applications using Platinum, while any contributions to the platform itself are kept in the GPL sphere. Support libraries are also released under various open source licenses:

- FLTK (LGPL)
- ITK (BSD-like)
- VTK (BSD-like)

## 1.2. Goals

### **Target groups**

The image interaction platform is designed to be used by two characteristic user groups:

- Image analysis engineers
- Radiologists or other medically trained users

Common features of these two groups include:

- Professional users, who will likely spend much time using the platform and learn special features
- Skilled users with an academic background, with experience and training in handling abstract concepts

For **engineers**:

- Engineers using the platform will mainly be involved with software development

For **physicians/medical researchers**:

- Physicians have a busy schedule that puts a limit to the current use of tomographic data; any time saved on routine or maintenance tasks is of high value
- While highly educated, this group has less computer science knowledge. Conversely, they are experienced professionals and generally uncomfortable in the position of a novice.<sup>4</sup>

The demands of the target groups suggest a number of prioritizations: The user interface must be optimized for fast and uncomplicated routine use. It needs to be clear and well laid out, but not necessarily pretty or inviting.

In fact, given the seriousness of the medical field it may even be an advantage to have the interface look austere in order to impart the right expectations to users. Advanced functions are relevant and will probably be utilized, but must not complicate the interface or they will interfere with basic use.<sup>5</sup>

Since a non-technical interface is necessary, the API has to make this as undemanding as possible so that the programmer is not tempted with taking shortcuts that sacrifice the usability of the application. Though aimed at professional users, usability is an important issue. There are demands and guidelines for professional user interfaces in terms of ergonomics, information presentation and control which have to be met.

## **Functionality**

The feature goals are the following:

- **Visualization:** the ability to inspect images prior to and past processing
- **Data interchange:** one or more ways to import from data sources (such as an MR scanner) and to export for storage or further processing
- **Interactivity:** the specifics of an operation should be user-configurable just as easy as when viewing, i.e. without hard-coding or use of configuration files
- **Follow-through:** unlike typical research software, being both a user front end and an API, this software needs stability and completeness within the implemented feature set to attain sufficient usability. This requires more time for each implemented feature than without such demands
- **Extensibility:** with the multi-purpose scope of the platform coupled with the limited time frame of this master thesis project, all the possibly needed or useful features cannot be implemented or even foreseen. To guarantee future usability the code is organized in a modular form. This has the additional advantage of improving software quality, but like the above point it also increases development time.

## **Interoperability**

The platform is designed from an open software model, where

- Code and required libraries are open source-licensed (LGPL) and can be built on a variety of computing platforms using the CMake system
- The graphical user interface toolkit (currently FLTK) has few and well defined connections with the functional code – allowing the user interface toolkit to be replaced with a different one (including a proprietary toolkit) when the need arises
- Internal data structures are standard C++ types, chosen so that a mature algorithm can be easily moved to another environment or turned into a special-purpose program of its own.
- Common medical image formats can be read using the ITK library

## **Limitations**

Platinum is designed to have the user interface toolkit loosely coupled. This has the advantage of making the switch to another toolkit easier, should it someday be needed. Using such abstractions is a common way to make multi-platform software that employs each of the supported platforms' native toolkit. This project does not, however, maintain a complete abstraction to the toolkit and substantial rework will be needed to

adapt another user interface API. The FLTK platform-independent user interface toolkit is intertwined with the platform code to varying degree, most in interface-specific sections.

FLTK is completely self-reliant (in terms of the end result, though it takes advantage of native drawing routines available on various platforms) and maintains the same look and feel across platforms. Besides compiler idiosyncrasies this makes adding support for another computing platform a trivial task. The disadvantage of a generic user interface toolkit is that it makes the software a less good “OS citizen”, unfamiliar in comparison with the system's native toolkit and unable to take advantage of each platform's unique strengths. One example of this is the hurdles of the FLTK open/save dialog boxes, discovered in the user test described later.

### 1.3. The name

The name Platinum was chosen for

- its scientific connotation and general positive association (as in “platinum record”)
- the effect as a catalyst (for application development, in this case)
- its phonetic association with “platform”
- not previously being used for any notable software project – though not registered, from a trademark point of view it is an *arbitrary* name (an existing word used out of context).

### 1.4. Implementation

#### **Previous work**

At the beginning of this thesis project, much work had already been contributed to the interaction platform. A prototype of the current multi-pane interface was implemented with a fixed-viewpoint MPR renderer. An ITK-based reader for the VTK file format was also functional. A three- (now four-) part architecture was designated for communication between the platform's different functionalities (see *Architecture*, p. 9). Most of the data structures employed were prototyped, though not implemented.

#### **Interface**

A well-designed core visualization interface and feedback presentation assures that the usability goals are sufficiently addressed in applications derived from the platform. The platform's workspace is shown in figure 1, consisting of three parts,:

1. A viewport area with a matrix of images rendered from volume data sets
2. A list of loaded data, including data produced within the program
3. A control area, containing a list of operations and controls for their associated parameters

Viewports are the visible areas onscreen in which presentation of data takes place. In practice it is useful to have multiple views with shared parameters, e.g. one as close-up

and another as overview. The view in each viewport is defined by the following objects (that can be shared between viewports):

- `renderer` (base class `renderer_base`) Determines (and executes) rendering in a specific manner, e.g. multiplanar reconstruction, volume rendering or variations of these displaying more than three dimensions.
- `rendercombination` determines the visibility, color and blend modes of multiple superimposed volumes.
- `rendergeometry` sets view “look at” position, orientation and zoom level.

By dividing render parameters this way, the interface may be extended to have multiple views with blend modes or point of view synchronized simply by using references to the same parameter object.

Images are rendered off-screen and delivered to the widget toolkit for displaying. A number of wrapper methods exist so that refresh can be requested when loading another data set, panning, modifying data etc.

## Architecture

The platform is composed of four “manager” objects that collect and process data based on the model-view-controller (MVC) pattern described by Reenskaug.<sup>6</sup> This architecture had been selected prior to project onset – the MVC pattern was introduced subsequently to guide development, as well as to impart familiarity to developers who may get involved at a later stage.

- **Datamanager:** manages the creation, deletion and retrieval of volume data sets – *model*.

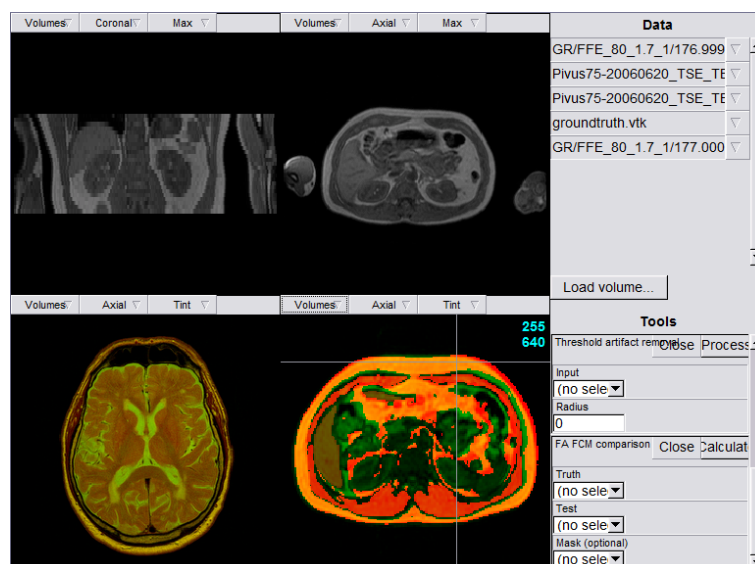


Figure 1 – main UI layout

- **Viewmanager:** manages refresh and selection of viewports in which rendered images of volume data are displayed - *controller*, where each `viewport` object that is “managed” comprises a *view*.
- **Rendermanager:** connects renderer objects with `viewports` and image data.
- **User IO manager:** lists and relays queries/commands between the user and applications implemented in the platform – *editor*.

The `viewport` is split into one toolkit-specific part that receives interactive events (mouse clicks etc.) and the actual view part which receives abstracted action codes. Similarly, the `userIOmanager` has toolkit-specific implementation of the individual controls while the communication with applications is done through an interface independent of user input which allows commands to be invoked with or without user input as specified in the MVC pattern.

References to the setup components (`images`, `renderers`, `viewports` and `userIO` blocks) are expressed internally as numerical IDs independent of memory addressing, or the environment in which the software is built and run. This creates a robust architecture and also leaves open the possibility of storing a session – all active objects - for re-opening or distribution using the same IDs. The concept of unique IDs provides an easy-to-grasp abstraction of this data structure-independent approach, one which is shared throughout the various parts of the program.

Brief descriptions of the full class tree are available in appendix 3.3.

## Image storage

Volumetric images are stored as C++ arrays of a templated voxel type. This design allows custom voxel data types with negligible performance loss, as template instantiation is done at compile time.

Since volume objects (`image` class) vary in size, the array needs to be dynamically allocated. Because of C++ limitations it is necessary to do storage as a one-dimensional array with lines and slices laid out sequentially with x, y, z in decreasing rate of growth. Volume data may also be replicated into the storage format used by the ITK library (necessary for doing image processing with ITK) and back to the default format.

## Processing

The `image` object forms the basis of image processing in Platinum. There are functions available for setting and retrieving voxel values, or processing algorithms may be implemented in `image` (or a subclass). Use of the ITK library for image processing is not required; typically simple algorithms are better custom-written to operate directly on the storage arrays, as ITK has some performance limitations following its flexible and generalized design.

## Applications

Currently Platinum facilitates image processing applications in the role of a library, providing a user interface for image I/O and display. Applications are C++ programs linked to the library, which define appropriate `userIO` objects and their callback functions which do the actual processing. This separates the particular usages from the general

platform, avoiding computational and cognitive overhead of functions for unrelated uses within the same program.

A different, possible future solution instead defining image processing algorithms and their associated controls in a plug-in system and run Platinum as one program that loads plug-ins of choice. This would have the advantage of assembling different algorithms into a package without recompilation. The disadvantages amount to the more complicated definition of plug-ins consisting of multiple files and a dynamically linked library for the code, which have to be grasped and appreciated by application programmers.

## Coordinates

The local coordinate system of each renderer – whose `rendercombination` may include several volumes - was chosen with the following points in mind:

- Simultaneously displayed volumes have different data sizes
- Yet, volumes rendered superimposed typically have the same geometry
- Though described as arrays of voxels, volume data represents a physical object
- In practice, rendering is done to inspect a particular area, rather than fitting data in an onscreen box

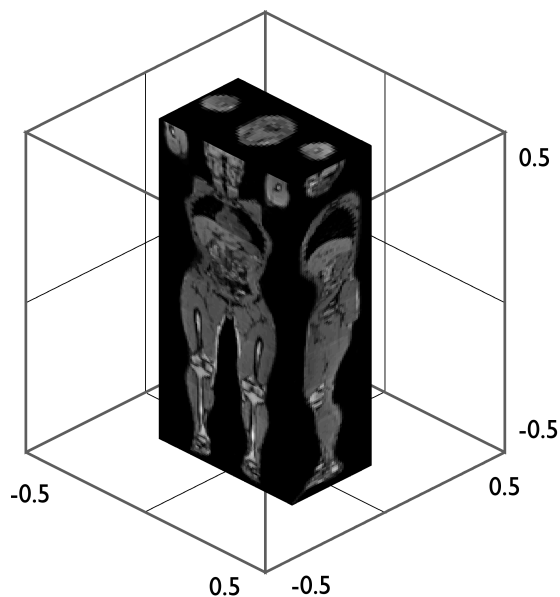


Figure 2 – local coordinate system

A system based in voxel units will no longer make sense when volumes of different resolutions are combined. A non-proportional (as in proportionality to physical dimensions) system will result in unnecessary work for the application programmer in converting locations to proper coordinates. Accurate dimensions on objects are fundamental for medical and research purposes, and selecting a suitable coordinate system as basis will reduce the risks of misinterpretation of data.

The chosen solution is a unit coordinate system proportional to physical units but scaled to each volume so that it maintains proportions with the longest edge

scaled to 1.0. The origin is the geometric center of the volume, making the edge locations -0.5 to 0.5 at the volume's longest edge(s) – see figure 2. This design makes it easy to add another positioning layer where volumes are positioned and oriented in a global coordinate system.

## View navigation

View navigation is defined by a look-at point and an orientation matrix in a concept common to the 3D graphics field. Scale (zoom) is represented by a scalar value where 1.0 will precisely fit the volume within the display area (when viewed in an orthogonal

direction). Zooming in the viewport maintains the look at point, which may also be set by the application.

Mouse controls were selected in an attempt to conform with established practices for users of image analysis software on the site, namely *ImageJ*, *Impax* and the proprietary control interface for the installed Philips MR scanner. Since the control mappings differ largely among these, some mappings had to be chosen arbitrarily. The result is shown in figure 9.

## User input/output

To control the processing taking place, the programmer needs a way of requesting and displaying information to the user. The chosen approach is a list of input tasks (figure 3), located below the list of open volumes. The following typical data types/interaction modes have been implemented:

- Binary value
- Integer and floating-point scalar value - input is made with a slider control
- Integer value – with keyboard entry and incremental control
- Selection of a particular volume (for acting as processing mask, etc.)
- Results - one-way display of information to the user

*Additional data types which may be needed in later use:*

- 3D position - specified by a mouse click in the relevant viewport
- Series of points (polygon) defining a region of interest - including save/load capabilities
- Scalar interval
- Progress bar – may be combined with timing functionality for assessing the efficiency of operations

The programmer defines one or multiple operations using one of the data types above, and corresponding tasks are stacked in a list in Platinum's main window. The workflow (depicted in figure 4) goes as follows:

1. Operation widget(s) are added in code
2. The user loads and browses images

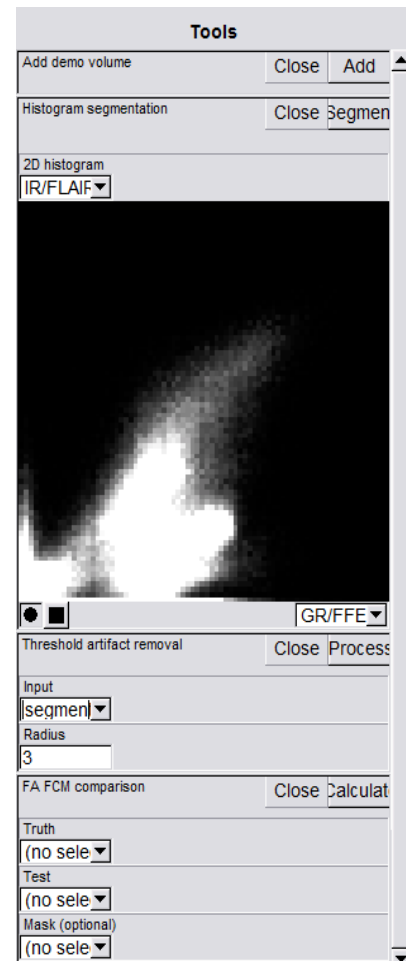


Figure 3 – userIO list



3. *Optional*: altering controls engages a callback supplying real-time feedback for the value of one or more volumes
4. The user engages an operation (with the OK button)
5. Processing callback is run

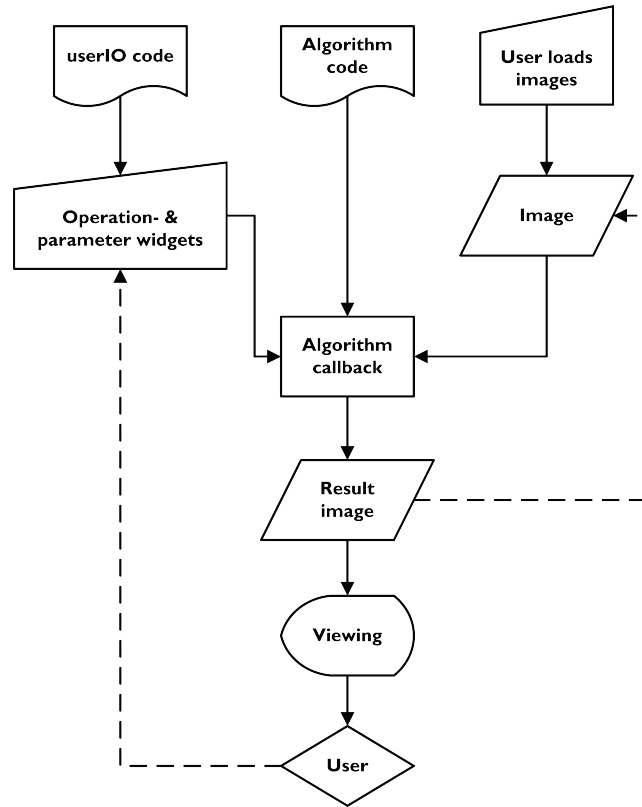


Figure 4 – userIO workflow

Parameters are marked as belonging to a particular task, for each task parameters are grouped in a box with shared “close” and action buttons. User tests showed that a boilerplate “OK” title on the action button was confusing to the users as to what the result would be from pressing the button; the API was altered by removing the default button title, forcing the programmer to give the button a more descriptive name. Grouping of parameters enables the user to adjust and re-select parameters before eventually running the operation. This avoids a major modality – a generally bad design choice<sup>7</sup> – present in simple command-line interfaces where parameters are given in a question-response format one after another. Input task widgets are created/removed and assigned unique IDs by a manager object in the same manner as volumes and renderers are handled.

### Superimposed slice rendering

During medical image processing it is useful to render multiple volumes together (e.g. one raw and one processed image). The platform provides this mode of interaction in the following way:

- Which volumes to render is selected in a menu associated with the `viewport`
- Blending mode can be selected for the view (`rendercombination`)

The supported rendering modes are:

- Max: the pixel with the highest intensity is used
- Min: the pixel with the lowest intensity is used
- Average: the average pixel value is used
- Tint: consecutive layers are assigned separate color channels (i.e. first red, second green) thus new colors are created where the images intersect

The *min* mode presents a special challenge: to produce proper results, minimum rendering starts with a full-intensity (white) background. The background is left unchanged outside the perimeter of the images, which creates a distracting bright white surrounding. Since image blending is done in the final RGB buffer, and the initial background color is of importance, there is no trivial way of assigning a color to the surroundings of the image. This was solved by using only the red channel for storing original intensity and setting the value for all three channels in rendered pixels. This way unchanged pixels can be identified post-rendering (from not being monochrome) and assigned a new background color of choice.

Navy blue was selected for background color, since it is known to have low visual impact, yet can be distinguished from dark parts of the rendered image.

## Optimization

The trivial choice for pixel-by-pixel 2D rendering of a structure is evaluating a function at each pixel in the destination image, a function which defines the transformation from source to destination. With the large volumetric images in current (and upcoming) medical applications and the objective of real-time performance some optimizations to the rendering algorithm were necessary.

The pixel-by-pixel technique described above was altered

to maintain *sequential locality of reference*: the processor can be expected to retrieve into the cache a chunk of memory following the address accessed. If data contained in this chunk is accessed next it will be available faster through the cache. This is the case with the trivial mapping of x (fastest-growing) axis to x onscreen, and y (second fastest-growing) to onscreen y.

To make this optimization available in other rendering directions and other scales than 1:1, the rendering direction is determined by memory order rather than screen pixel order (figure 6) in the following way:

1. memory voxel position is advanced and onscreen position accordingly by a vector which maps memory to screen coordinates. The slice to be rendered is traversed in voxel space, memory order, i.e. first x, then y, then z. Position in voxel space is stored as integer pixels in screen space (the minimum distance for each iteration)
2. the onscreen distance to next voxel is calculated, if it is greater than one pixel the spanning pixels are filled with the same value in a local loop

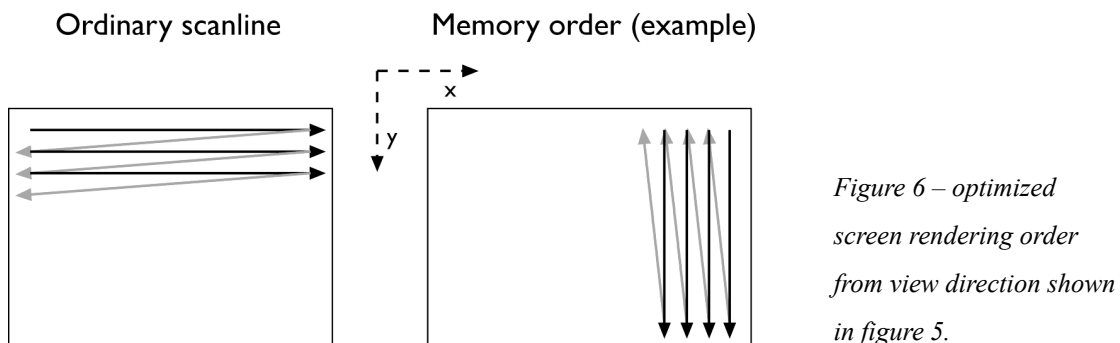


Figure 6 – optimized screen rendering order from view direction shown in figure 5.

Using this technique data loaded into the processor cache in chunks will get maximum utilization. For slices in the x direction – traversing only y and z directions - the access pattern may fall outside the cache scope, ultimately depending on size of cache and volume data.

With the rendering loop implemented as a general case for all twelve (in practice three, as the other are mirrored variations of the axial, coronal, and sagittal projections) orthogonal directions there are a number of calculations for the position and bounds of voxels in memory and screen space. Position (“look-at”) and direction are defined as 3D vectors (figure 5), and provide necessary positioning for a future arbitrary-direction reconstruction renderer as well. To further improve performance, intermediate values depending on the slower-growing axes are stored and recalculated only in the respective outer loops.

### **Transfer function**

When rendered, volumes are put through a linear transfer function. The transfer function implemented as part of this project is not user configurable; it is selected from each volume's maximum and minimum values. Possible improvements include basic intensity/contrast controls, and a polynomial transfer function.

### **File format**

Loading of images involves using data generated by other software, and saving generated images for use with other programs. There are several volumetric image formats common in medical image processing: DICOM, Analyze, VTK and a number of formats specific to scanner manufacturers. The format used varies between sites; the DICOM standard is common in medical applications. While then a natural choice for main input format, DICOM is very comprehensive with a lot of overhead and the hassle of handling a collection of image files when copying or opening.

For repeated loading and saving, a format requiring less system resources would be suitable. One such is VTK. It is self-contained (as opposed to DICOM which consists of redundant headers for each file containing a slice), supports data compression and is implemented through the already-used ITK library. The VTK file format is thereby a suitable choice for image output. Input and output of images from the platform is a straightforward process of using available libraries to convert between the respective files and the platform's internal representation (the `image` class).

### **Image loading**

The system can open single-file formats (such as VTK) as well as multi-file series like the DICOM standard. Once loaded, both result in the same type of volume and therefore the two formats should be treated similarly from a usage perspective. The DICOM format stores series membership within each file (as opposed to identifying series by one common file in the directory where the file resides, or by the directory itself). Therefore, several series may be stored in the same directory and a DICOM series cannot be unambiguously chosen by selecting a particular directory.

By treating any DICOM file as a specimen for its series, both single-file images (like VTK) and DICOM series can be selected using the same file chooser window. In a mul-

tiple selection, any DICOM file will load its associated series, duplicate files from the same series will be ignored and single-file images will also load as expected.

### **Tools**

Platinum is written in C++ using templates to handle various image voxel types. The user interface is implemented in the FLTK open source widget toolkit. The platform's architecture integrates usage of the ITK medical image processing library.

Development was done alternatingly in Microsoft Visual Studio .Net 2003 and Apple XCode. Cross-platform build files are produced using the CMake tool.

## 2. Multispectral segmentation

### 2.1. Introduction

MRI provides a harmless and information-rich source of body images. In MR imaging a radio frequency (RF) signal is emitted by atomic nuclei in the scanned subject, placed in a strong magnetic field and administered a carefully tuned input signal. The image is generated by Fourier transformation of the emitted signal. Several parameters in this process can be altered to produce images which accentuate certain features, e.g. fat content.

To be really useful, the images have to be partitioned into the tissue types of interest. This process is called *segmentation*, and is for most part a computational imitation of the neurological process that takes place when an observer studies an image and identifies its parts. Once segmentation is done, the information can be used to calculate the volume of organs, to highlight a tissue or organ of interest, as density information for radiotherapy or a 3D volume rendering.

Basic segmentation takes place when just looking at an image, and manual segmentation is simply the process of sketching down the tissue/organ boundaries observed. Just like machine vision, automatic segmentation is more complicated than it may sound. In theory each tissue type would be identified simply by its MR signal intensity in the image. In practice however, there are variations in magnetic field added to variations and noise in the RF signal. While these disturbances can be compensated for by human vision, they make automatic segmentation nontrivial - counter to CT where the contrast of each tissue type remains homogeneous throughout the image.

To manage these deficiencies one may use combinations of

- Multiple data<sup>8</sup>
- Pre-measured inhomogeneity profile estimate<sup>8</sup>
- Inhomogeneity model fitted to image data<sup>9</sup>
- Hints from spatial distribution<sup>8 10</sup>
- Transformed-domain correction<sup>8</sup>

### 2D histogram selection

This section describes a tool for graphically selecting areas of a certain intensity combination out of two images. The purpose of this project was to assess the performance of manual clustering using the same input as automatic techniques, and to explore the comprehension and viability of a 2D histogram representation as a tool for MR image analysis - with other possible applications than segmentation. Additionally, it provided a practical application for testing various aspects of the Platinum interface, that lent itself to testing better than the platform itself.

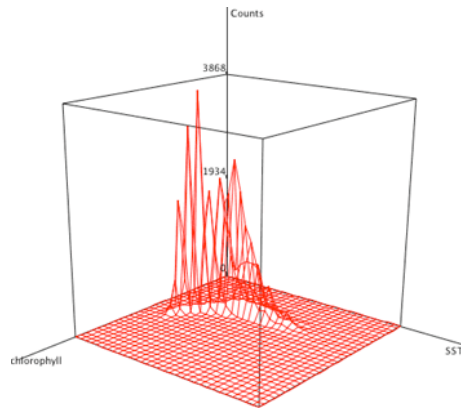


Figure 7 – example of 2D histogram as 3D surface. ([http://earthobservatory.nasa.gov/Laboratory/ICE/ice\\_user\\_guide.html](http://earthobservatory.nasa.gov/Laboratory/ICE/ice_user_guide.html))

The method described in this section is a pure multiple-data approach. As described above medical MRI relies on a number of parameters which produce varying contrast between tissue types. The combined information in two images with varying contrast contain complementary information to outperform current single-image techniques in inhomogeneity correction<sup>11</sup>. The intensity of a tissue type varies non-linearly among images acquired with diverse contrast parameters (hereafter *multispectral* images). Accordingly, there is an N-dimensional feature space (where N is the number of images) that increases separation of tissue types.

There have been numerous studies (<sup>12 13 14 16</sup>) using multispectral images for improving segmentation, several by applying clustering in the feature space to classify various tissue types. In this project a 2D histogram is presented to the user for recognizing and selecting feature space clusters corresponding to tissue types of interest. This technique could be implemented for higher-dimension feature spaces as well. With the highest dimension for an onscreen linear graph being a 3-dimensional perspective projection, comparative studies of 2- and 3-dimensional graphs show that while 3D graphs are preferred by users in a significant way they yield less performance than 2D graphs as interface.<sup>15</sup> Hence the 2D histogram was selected for this study.

While current data mining techniques for clustering multidimensional data are efficient and produce adequate results, the clustering abilities of human perception are powerful and interesting for comparison. With MR imaging being a developing field with many open ends and possibilities, there is also merit in exploring multispectral data without a predetermined goal.

A similar histogram interface is used in Mayo Clinic's *Analyze* image analysis software (and possibly others). In that software the selection of training areas for an automatic Bayesian classification algorithm is displayed in a “scattergram” - a 2D histogram displaying the occurrence of distinct values instead of frequencies.

## 2.2. Implementation

The 2D histogram used is an intensity-projection of the more common topographical 2D histogram consisting of a 2-dimensional map of peaks and valleys displayed in 3D perspective (figure 7). Bucket size is dynamically selectable so that the user can alter perspective from rough frequency estimate to fine separation (but with imprecise frequency resolution). Typical histograms have a very high dynamic range and require more sophisticated scaling than mapping the highest bucket value to maximum screen intensity. A simple heuristic was used; linearly scaling max intensity to the mean bucket value. The histogram is produced from the entire volume as opposed to the visible slice – having advantages and disadvantages discussed in *improvements* (p. 26).

In slices of volumes the user may select a rectangular area whose bispectral voxel values will be highlighted in the histogram. This enables the user to identify tissue types from a sample area; the highlight also gives a clue to the amount and distribution of noise. In a corresponding way to the histogram highlighting, the volumetric image can be segmented by selecting either a rectangle (regular 2D thresholds) or a circle (target value pair with allowed distance). The circle is actually an ellipse, corresponding to a user-selectable scaling of the two intensity values from the respective images from which distance is calculated. The elliptic option was included from an assumption that tissue feature clusters have a normal distribution and therefore more accurately delineated by an elliptic region.

Upon selecting an area of the histogram using either of these two methods, the indicated range in the image is highlighted (figure 8).

This highlighting updates interactively during selection which makes it easy to fine-tune the selected range. The histogram selection can also be resized using the mouse wheel to expand or narrow the selection window. In both image and histogram display the highlight is drawn using translucent colors so that the original image remains visible during selection.



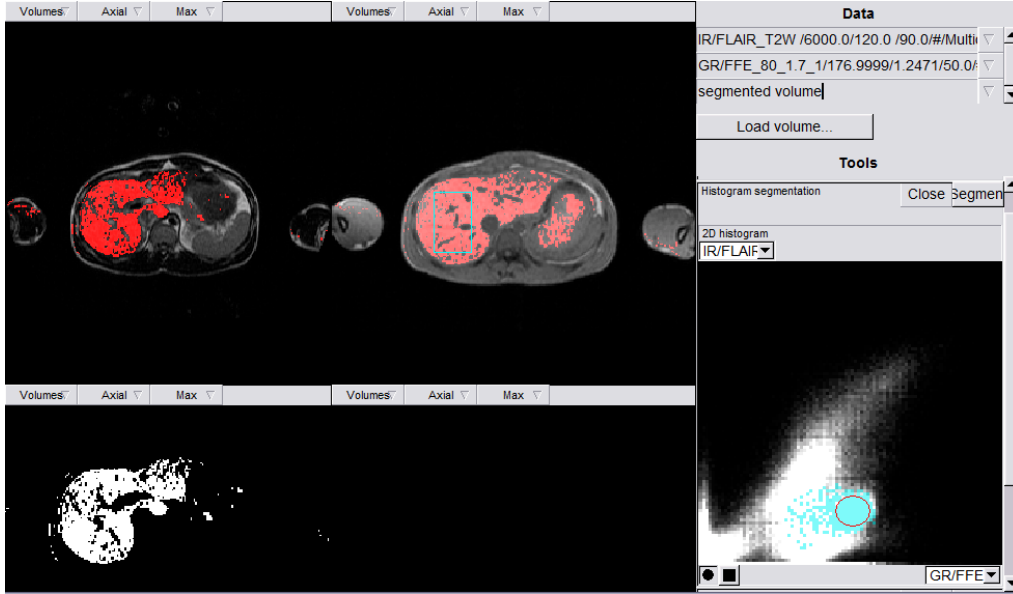


Figure 8 – 2D thresholding of liver tissue

### Platform integration

The 2D histogram is presented to the user as a `userIO` parameter, part of a `userIO` object which in the typical case (and during testing) has the action of doing a segmentation using the selected threshold. The frame containing the histogram image itself was implemented as a FLTK widget corresponding to a `histogram2D` object that calculates the histogram and produces images. The `histogram_2D` class is derived from a base class that facilitates future variations, including 3D- or regular 1D histograms.

The viewport selection rectangle for highlight in the histogram was inserted in the viewport event handler for action code `CB_ACTION_DRAG_PASS` (a general drag event not associated with panning/zooming). When more features are added, this functionality would be more suitably located in the `userIOmanager` where the priority of several possible event handlers can be properly assigned. To keep the event handling code clean in the meantime, the selection functionality is kept in the viewport's associated `FLTK2Dregionofinterest` object.

### 2.3. Quantitative test

To get a quantitative measure of 2D histogram selection performance, a quantitative test was performed where a user experienced with manual segmentation used the software to segment the liver from abdominal data. Tankyevych's<sup>16</sup> algorithm test data set was used: abdominal volumes from 10 subjects with FLAIR and flip 50 gradient echo breath hold sequences for each subject. The FLAIR and flip 50 sequences were chosen among 5 available sets as these were shown in the Tankyevych study to have the best separation for features of liver tissue among available pairs. Additionally, the dataset contained manually segmented ground truth volumes and binary masks cutting out arms, MR motion artifacts and noise. This data was used to assess segmentation precision.

The test user was asked to select the liver volume out of each test image, and save the result. No feedback of precision was given during the test. The total elapsed time per-

forming the test was 47 minutes for ten images, or about 5 minutes for each image. The time was measured only for reference; the test subject was instructed to work with each image until they were happy with the result, without time pressure. Considering that this user had only brief previous experience with the histogram segmentation software, it will likely decrease with more practiced users.

## Results

The results were measured 1. unaltered, 2. with Tankyevych's<sup>16</sup> mask and 3. using a basic algorithm to remove the numerous but minor unwanted elements resulting from noise. In the algorithm a mask is created through the following steps:

1. erosion in an 18-neighborhood (radius 1.0 voxels)
2. keep largest connected volume
3. two iterations of 18-neighborhood dilation

Finally the mask is applied to the original image. This treatment is fully automated and could be integrated into the 2D thresholding to create a one-step segmentation procedure. The effect can be described as removing elements with dimensions of less than 3 voxels in one direction, while preserving full shape detail of any object that passes this condition. Eroding once and dilating twice ensures that the mask encloses fine details at the surface of the target volume. The "keep largest" volume step limits the result to one connected object, as is desired in the case of liver segmentation.

Test results were run through an algorithm to produce comparison diagrams that indicate the amount and distribution of false positives and -negatives, and produce a measure of segmentation precision. The measure used was the quotient between true positives and the combined volume of segmentation and ground truth, as proposed by Udupa et. al.<sup>17</sup>

The untreated 2D thresholding segmentation performance was an average 57.8% - The best result achieved by Tankyevych with Spatial FCM automated liver segmentation was 86.3 % using triple-spectral images: SPAIR, FFE flip 50 and FLAIR. In this metric the abdominal sections were pre-processed using the data set's torso mask. Using this same mask on the manual data obtained in this study increased score to an average 64 %. However, using the artefact removal algorithm described above raised segmentation precision to 82,7 %.

Complete results are available in appendix 3.1.

## 2.4. User test

To evaluate the user interface design decisions, a qualitative user test was performed. As the platform will not be used by itself, but rather in supporting specific applications, an application needs to be part of the test. A combined test was devised assessing the interface and usability of both Platinum and the 2D histogram application. Test methodology was adapted from Lewis & Rieman<sup>7</sup>.

In a first step, an experienced user (medical student in final semester, familiar with manual image segmentation) did a pilot test. This preliminary test exposed several minor points of clarity and usability, resulting in a new color scheme and a more informa-

tional method of automatic naming for DICOM series. The use of multiple mouse buttons, as is common in professional applications, was confirmed as far beyond self-explanatory and an instruction diagram was made to alleviate this. (see figure 9)

With these issues straightened out (but some more intricate points remaining - see the [revisions/extensions](#) section for the platform on p. 27), a qualitative user study was performed by six users representing the various combinations of medical and image processing knowledge.

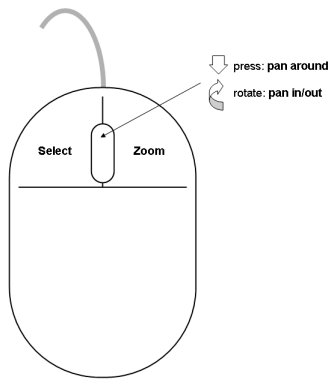


Figure 9 - mouse button mappings

Users were given a first try of the software without any instruction, then a brief explanation of concepts and purpose of the program, including the mouse button chart (figure 9). They were then asked to do segmentation of a liver in an abdominal image (one of the images used in the later quantitative test), and finally to answer 14 questions regarding the impressions of the software on a Likert scale. Full questionnaire answers are presented in appendix 3.2.

The original intention was to initiate the test with open exploration of the software without instruction. This part was abandoned as the unguided test turned out too difficult to carry out under the circumstances, particularly with experienced users: these users had difficulties

grasping the concept of aimless testing, and waited idly for instructions rather than going ahead and experiment with the interface. Contributing to this might be the perceived awkwardness of being asked to do something without instruction and being observed at the same time, but also that the software is intended for a professional audience and not designed to be self-explanatory.

During the test, the subjects were encouraged to ask questions and comment following the *think aloud* method as described by Lewis<sup>18</sup>. Notes were taken on the testers' comments and observed behavior. With limited time available with the test subjects, think aloud provides a way for test subjects to provide feedback in sufficient amounts for drawing conclusions. It also creates a more participatory atmosphere than passively observing or recording participants' behavior.

## Test observations

Participating users suggested the following features:

- The most success in explaining the function of the histogram was achieved by a hands-on demonstration of the selection and segmentation functions. Nonetheless, several users suggested axes for indicating what is shown in the histogram. A way of displaying the histogram value (bucket content) at any point was also suggested. However, these suggestions may be even more an indication that the histogram is nonobvious, than axes being the optimal solution. In any case, tasks (here: feature-space selection) must determine which features get implemented<sup>5</sup>.
- Various mouse cursor shapes can be used to give better feedback on the effect of mouse clicks in various locations

- The ability to close all open volumes and reset state in one action to start afresh
- Present histogram for the currently visible slice instead of the entire volume, to exclude tissue in regions not of interest

One test user familiar with both the medical and computational aspects of MR image processing was able to identify various tissues (liver, spleen, fat) just by looking at the histogram shape. While this user was very skilled with MR technology he had no previous experience of 2D histograms. Thus, with necessary background knowledge and practical experience with the histograms of a vast body of images users might acquire the skills to operate directly on the histograms and take advantage of the extra information present in these for further understanding the scanned image data.

The feature for highlighting corresponding values of an area of the image turned out to be more useful to test subjects than determining the extents of a particular tissue type using the histogram figure. The lack of descriptive axes may have played a part in this occurrence, but once the relationship between image and histogram was established by highlighting this was no longer a concern. In fact, the highlighting seemed to be the major means of comprehending the histogram/segmentation to such an extent that the histogram image itself only served as map for remembering previous areas of highlights.

The 2D histogram is clearly an unfamiliar way of interaction with images. The use of one spatial image mapped to another in the nonlinear fashion that is image-to-histogram transformation, is at best challenging to comprehend. The relationship was most easily explained with the aid of highlighting image samples in the histogram to show their corresponding domain. This explanatory effect could probably improve further with faster software, making this interaction more responsive.

By the potential for confusion apparent from user tests, it appears a sound decision to stick with a 2D representation of the feature space. Since the actual frequency in histogram buckets is not as important as a rough hint of areas of concentrated frequency – clusters – the topographical kind probably would not contribute significantly to the clustering task. However, a genuine 3D histogram (using 3 multi-contrast images) might improve segmentation performance. To be comprehensible, such a histogram requires careful and attentive design of presentation and the navigation and selection tools.

While the 2D intensity histogram appears to be a viable display mode, two interrelated questions have emerged:

- Appropriate bucket size is important for an informative display. Big size (few buckets) gives a good view of the density of different areas but will obscure small yet distinct clusters. Small size gives a detailed picture of the tissue areas' boundaries but is easily clouded by variations from noise. In the described implementation the image sample highlighting is done for every bucket to which at least one selected voxel belongs. This means that the highlighting becomes much more dense with larger bucket sizes, which is desired in most cases.
- Scaling of histogram values to screen intensity is a non-trivial problem. Since the histogram image appears to be more of a rough guide than precise figures for analysis, an interesting and feature-rich image is more desirable than an image less so but with a more defined relationship to the data.

- The load & save file dialogs supplied by FLTK caused a varying degree of confusion among most of the test subjects, possible reasons include:
- Unfamiliarity: the most commonly used dialog for these purposes is the one supplied by the operating system used in case of Mac or Windows users. Save file names are defined using the full path, when altered it changes the displayed directory in a consistent manner, but unlike the save dialogs familiar to users of other platforms.
- POSIX-centric design: FLTK uses an internal file representation modeled from the POSIX standard and relies on this representation more than what is necessary, creating confusion in users unfamiliar with UNIX file system conventions. I.e., to move upwards in the directory structure, there is no “up” or similar button, instead the above directory is presented as “..” at the top of the current file listing.

The FLTK 1.1.7 API does not allow the file name in a save dialog to be changed by software, otherwise a suitable default value would be the user definable image name

One way to alleviate all the above issues would be to use the file selection dialog native to the current platform.

Selection of DICOM images by selecting a single image in a series (thereby loading the entire series) appeared intuitive to users experienced with medical image processing and the concept of image series, selecting folders or all the image files in a series was suggested by less experienced users.

The multi-pane mouse-button driven interaction design draws from elements of pre-existing software, particularly of the kind used to control MR scanners. Prior knowledge of this type of user interface appeared to instill some familiarity in test users with experience in this field. Yet, user interfaces of this type are far from standardized and familiarity did not cover the particular mouse-action and data-view mapping.

The use of multiple mouse buttons (typically 3) turned out to be nonobvious to most test users, experienced or not. This is commonly used in professional interactive applications, where the two auxiliary mouse buttons (middle and right) available on common personal computers have different functions than the convention of a right click bringing up a contextual menu. All three buttons may also vary in function in different parts of the interface, a practice reserved for professional user interfaces where the user can be expected to find out from documentation and then memorizing the meaning of the buttons. While there is no straightforward way to label mouse buttons on a per-application basis, computer games (a software category having both casual and power users) have used onscreen images of the mouse with labels to continually inform the user of mouse button effect.

The “action” button for each `userIO` at testing time was labeled with a default “OK” instead of indicating the particular action, which was only suggested by the name of the `userIO` group displayed next to the button. The purpose of the button was unclear to several of the test users. This was corrected by altering the API in a way that programmers must supply a label text in order to enable the action button.

Test response from the less computer-experienced users suggested that the user interface is lacking in feedback. When unaware of the internal architecture (data-renderer-

viewport), the available controls do not seem to do much right after launching the program. The connection between a viewport and the displayed image(s) is only shown in a drop-down menu; making this information more accessible is a non-straightforward design issue but should benefit both novice and experienced users.

## Improvements

To alleviate the bucket sizing problem, highlighting could be done at a lower bucket size than histogram display. The highlighting may also be carried out with varying opacity depending on the number of voxels consisting the highlight of each bucket.

Four viewports in a grid is a reasonable general purpose configuration, but ultimately the user needs the option to add or remove a column/row of viewports to suit the needs of a task.

The current method of choosing datasets for each viewport from a menu should be complemented by a way of quickly displaying one particular image in a viewport, e.g. dragging-and-dropping approach.

## 2.5. Conclusions

The multicontrast histogram approach, as exemplified by this 2D implementation, has two main opportunities for clinical use that may be further explored: segmentation, and diagnosis or other image assessment from the histogram itself. As there are equally successful automatic methods of segmentation through clustering, the most potential lies in using the histogram to evaluate the composition of the scanned subject. The shape and location of the various clusters correspond to the amount and composition of tissue, including tumors and any abnormal accumulations of fat or water. The multispectral histogram acts as a hash or “fingerprint” of the scanned area in that deviations from the expected composition are visible in pure feature space without the spatial component - that is, without knowing where in the body to look. Manual examination of the histogram means that the flexibility of human perception is available for alleviating the noise and distortion present in MR images.

An examination of various software for the field of medical tomography imaging (image viewing programs and frameworks including those listed in *Similar projects* on page 4) shows numerous variations on the multi-pane layout. While this is a recurring pattern, navigation - less trivial than the mapping of multiple 2D slices to a 2D screen - is performed with indefinitely varying combinations of on-screen controls and mouse/keyboard actions. Following this, it was expected to find experienced users having little advantage of their experience when trying out the navigation controls of Platinum.

An interaction platform will have one or more layers of complexity added onto its native interface from any particular application. The platform's interface by itself has to be exemplary in terms of usability to optimize the whole experience, and to set example for developers making the specific applications. The user test suggests that the file and image navigation controls have room for improvement in this regard. Icons (as opposed, or in conjunction with, text labels) and drag-and-drop functionality have not been employed in the current interface and may improve the intuitiveness, particularly users without image processing experience.

## **Development issues**

The multi-platform approach means that the same code is reused and runs on Linux, Mac OS X and Windows systems with only small adaptations involving compilation and runtime initialization of the various libraries. In practice, however, large programming projects involving multiple contributors will contain bugs or mere glitches which may show up only on certain combinations of operating system and computer architecture. The CMake build system uses one single project specification which can produce actual build files in a number of formats, alleviating some of the potential difficulties. CMake has the same interpretation of scripts on different platforms, this is not the case at the compiler level however. In particular the GCC (GNU Compiler Collection) compiler used with Linux and Mac OS X is fairly compliant with standardized C++, while Microsoft's compiler used on Windows has more features including nonstandard extensions (and also supports some standards not implemented in GCC) meaning that programmers will have to respect strict coding rules – code may be unusable in this type of project even though it compiles and runs on one particular system. The alternative is to do frequent recompilation on supported platforms, which needless to say is time-consuming, and will only assure that the code works on the particular test systems. Numerous problems of this type were encountered during the development of this project - suggesting that cross-platform development is, and will probably remain an extra effort compared to centering on one computing platform.

## **Revisions/extensions**

Currently, image display is possible only using C arrays. Data produced by ITK image manipulation functions must be converted prior to displaying. The internal storage of volume data may be implemented with full unbiased support for C arrays and ITK, or simply changed to using just ITK data as there are iterators available for high-performance access to image voxels. This way image display may be implemented with VTK as well, without the need to go back and forth between the formats for processing and display.

Storing the user interface setup – or “workspace” – can be a time-saving feature that also makes it more convenient to introduce new users to the platform by providing them with ready-made setups for doing a particular task.

While not an integral part of Platinum, the 2D histogram tool might be suitable extended in the following ways:

- **Contrast adjustment**

A 2D histogram is an information-rich image which unfortunately has a wide dynamic range with details both in peaks and in the low-intensity areas around them. Either automatic or manual adjustment, or some combination, would improve histogram perception and the ability to draw conclusions about the source images directly from the histogram.

- **Noise reduction**

Noise is an expected component of MR images, reducing clarity and contrast. The definition of noise in this case is unrelated intensity combinations that overlap with the tissue of interest in feature space, but not in image space. Another distin-

guishing property of noise is that it varies randomly across the image, with high spatial frequency (corresponding to a wavelength of about one voxel). This characteristic can be exploited by weighting each voxel's contribution to the histogram proportionally to the image intensity gradient. Thus the impact of image noise in the histogram may be reduced.

- **Multiple ROIs**

The current functionality allows segmenting multiple intensity combinations only sequentially - by selecting and extracting one tissue type after another. Multiple selections within the histogram would make the process more convenient, and provide a way of handling overlaps by, for instance, using Gaussian weighting.

- **Save/load classifiers**

The Gaussian classifier regions described above may be saved/loaded and reused on different image pairs (having the same contrast, or same intensity ratios in the case of MRI).

User interface setup is internal to the platform and there is little possibility, or need, to incorporate it into an existing standard. Still, to facilitate development and maintain human-readability of the setup file, it might be stored as XML (extensible markup language). This allows other software to read the files easily, should the need arise. It also provides a familiar syntax for editing by hand, and even allows the use of existing XML editors. Since XML is tag-based (as opposed to binary or position-based) it facilitates extending the data format in upcoming versions while maintaining compatibility with earlier files.

The relationship between the setup components (`images`, `renderers`, `viewports`, `rendergeometries` and `rendercombinations`) are expressed internally as numerical IDs independent of the environment. This allows the connections, in addition to the necessary instance variables (the individual objects), to be easily represented as XML text. In this solution each object has a method for manifesting itself as XML, and a constructor that creates a new object from an XML object argument.

## **Acknowledgements**

I would like to thank my supervisors Joel Kullberg and Petter Ranefall for their open-minded guidance and support throughout this master thesis. Shouts to Mårten Öhman for doing in-depth tests and providing valuable points of view. I deeply appreciate the friendly atmosphere and creative input of the staff at the MRI branch of Uppsala university hospital.



### 3. Appendices

#### 3.1. Quantitative test results

*Volume expressed in number of voxels*

Dataset	<b>Raw</b>				Truth volume	% TP	% FP	Udupa precision	False balance
	True positives	False positives	False negatives	Combined volume					
AAQ	31376	5223	5163	41762	36539	85,87%	14,29%	75,13%	0,00
AQ	45692	37206	4841	87739	50533	90,42%	73,63%	52,08%	0,64
BE	33401	7521	5303	46225	38704	86,30%	19,43%	72,26%	0,06
EQ	43933	19333	3163	66429	47096	93,28%	41,05%	66,14%	0,34
EV	40713	24990	6418	72121	47131	86,38%	53,02%	56,45%	0,39
FA	34573	12474	5937	52984	40510	85,34%	30,79%	65,25%	0,16
FON	33201	33081	2029	68311	35230	94,24%	93,90%	48,60%	0,88
HF	43852	45638	2120	91610	45972	95,39%	99,27%	47,87%	0,95
HF2	37248	35881	2628	75757	39876	93,41%	89,98%	49,17%	0,83
JL	42637	48917	3792	95346	46429	91,83%	105,36%	44,72%	0,97
								<b>57,77%</b>	<b>0,52</b>

<b>Masked</b>									
Dataset	True positives	False positives	False negatives	Combined volume	Reference volume	TP	FP	Udupa precision	False balance
AAQ	31057	4116	4915	40088	35972	86,34%	11,44%	77,47%	-0,02
AQ	45688	20903	4761	71352	50449	90,56%	41,43%	64,03%	0,32
BE	33401	3644	5290	42335	38691	86,33%	9,42%	78,90%	-0,04
EQ	43933	16288	3163	63384	47096	93,28%	34,58%	69,31%	0,28
EV	40713	13929	6418	61060	47131	86,38%	29,55%	66,68%	0,16
FA	34573	10802	5937	51312	40510	85,34%	26,67%	67,38%	0,12
FON	33201	22183	2028	57412	35229	94,24%	62,97%	57,83%	0,57
HF	43852	31072	2120	77044	45972	95,39%	67,59%	56,92%	0,63
HF2	37248	31902	2620	71770	39868	93,43%	80,02%	51,90%	0,73
JL	42637	34891	3789	81317	46426	91,84%	75,15%	52,43%	0,67
								<b>64%</b>	<b>0,30</b>

<b>Processed</b>									
Dataset	True positives	False positives	False negatives	Combined volume	Truth volume	% TP	% FP	Udapa precision	False balance
AAQ	30605	371	5934	36910	36539	83,76%	1,02%	82,92%	-0,15
AQ	44823	2896	5710	53429	50533	88,70%	5,73%	83,89%	-0,06
BE	31856	515	6848	39219	38704	82,31%	1,33%	81,23%	-0,16
EQ	43043	3774	4053	50870	47096	91,39%	8,01%	84,61%	-0,01
EV	39237	1403	7894	48534	47131	83,25%	2,98%	80,84%	-0,14
FA	31168	1282	9342	41792	40510	76,94%	3,16%	74,58%	-0,20
FON	32922	1266	2308	36496	35230	93,45%	3,59%	90,21%	-0,03
HF	43622	4641	2350	50613	45972	94,89%	10,10%	86,19%	0,05
HF2	36822	6867	3054	46743	39876	92,34%	17,22%	78,78%	0,10
JL	41287	2448	5142	48877	46429	88,93%	5,27%	84,47%	-0,06
								<b>82,77%</b>	<b>-0,06</b>

### 3.2. User test results

Subject	LJ	AE	TH	FON	JA	RE	Average
Profession	MR scientist	Psychology student	MD	Electrical engineering MS	CompSci student, Photoshop wizard	Bioscientist	
<i>1-5 scale, disagree - agree</i>							
Usage is clear	4	1	3	5	1	1	2,50
Facilitates image segmentation	3	4	5		4	4	4,00
Responsive	3	5	5	4	2	2	3,50
Intuitive	4	3	4		2	3	3,20
Enjoyable	4	2	3	5	3	4	3,50
Forgiving of errors	5		4	5	3	1	3,60
Pretty	2	3	4	4	2	3	3,00
Looks professional	3	4	4	4	4	4	3,83
Want to use in work	5		3,5	4	3	5	4,10

## User test results (continued)

Profession	MR sci- entist	Psychology student	MD	Electrical engineering MS	CompSci student, Photos- hop wi- zard	Bioscien- tist
------------	-------------------	-----------------------	----	---------------------------------	---	-------------------

*-(2) – 2 scale; negative: too little, positive: too much, 0 is adequate*

Number of com- ponents	-1	0	0	1	-1	-1	-0,33
Ability to custo- mize	-1	0	0	1	-1	-1	-0,83
Ways to ac- complish task	-1		0	2	-2	0	-0,20
Similarity with others				0		-1	

*1-10 scale*

Final score	7	7	6	9	5	7	6,83
-------------	---	---	---	---	---	---	------

### 3.3. Class reference

Classes in *italics* are planned (some prototyped), but not implemented.

#### **View & rendering**

##### **Viewport**

Abstraction of widget containing controls and image pane (implemented in `FLTKviewport`) for each viewpoint of data.

##### **FLTKviewport**

FLTK implementation of the central pane of each viewport that displays the rendered image and can be clicked/dragged in.

##### **Viewmanager**

Viewmanager lists the available viewports and allows access through viewport ID and various ways of requesting updates.

##### **Renderer\_base**

Renderers produce a 2D viewport pixmap out of images and point sets. The base class has virtual functions for coordinate storage conversion, common to different renderer types.

##### **RendererMPR**

MPR renderer subclass which also renders thumbnails via a static function. Like other possible renderers it also converts between local and global coordinate systems.

##### **Rendercombination**

Stores list of volumes and blend mode(s) for renderers.

##### **Rendergeometry**

Rendergeometry stores rendering coordinates (look at, orientation and zoom) for renderers.

##### **Rendermanager**

Keeps a list of renderer objects and handles communication between viewport and renderer.

#### **Data handling**

##### **Image\_base**

Untemplated base class for image storage, including volumes, contains 3D position and orientation information.

##### **Image3D**

Volumetric image class templated over voxel data type.

##### **Point\_base**

The untemplated base class for point collections.

##### **Datamanager**

The datamanager maintains a list of work data loaded or created at runtime. It has functions for loading and saving data (that invoke functions in the relevant classes).

## **User interaction**

### **Datawidget**

The datawidget is the GUI representation of a dataset (image/point). It provides controls for saving, deleting and shows the thumbnail.

### **UserIO**

Class defining an operation on data presented to the user in a list.

### **FLTKuserIOparameter\_base**

Base class for numerous parameter widgets populating userIO blocks. Handles value retrieval, updates and creation/deletion.

### **UserIOmanager**

Maintains a list of userIO blocks, and relays all operations on blocks and their parameters. The interface to this class is independent of the FL toolkit.

## **Helper classes**

### **Color**

Color storage class.

### **Colormap**

Interpolated color map class.

### **Cursor3D**

3D cursor that could be implemented as a point3D.

### **Histogram\_base**

Base class for processing of histograms in 1-3D (separated from user interface).

### **Histogram\_2D**

Displays and does area highlights of a 2D histogram.

### **Thresholdparvalue**

N-dimensional threshold values with optional shape variations. Includes a function for creating a new volume through thresholding.

## **2D histogram segmentation specials**

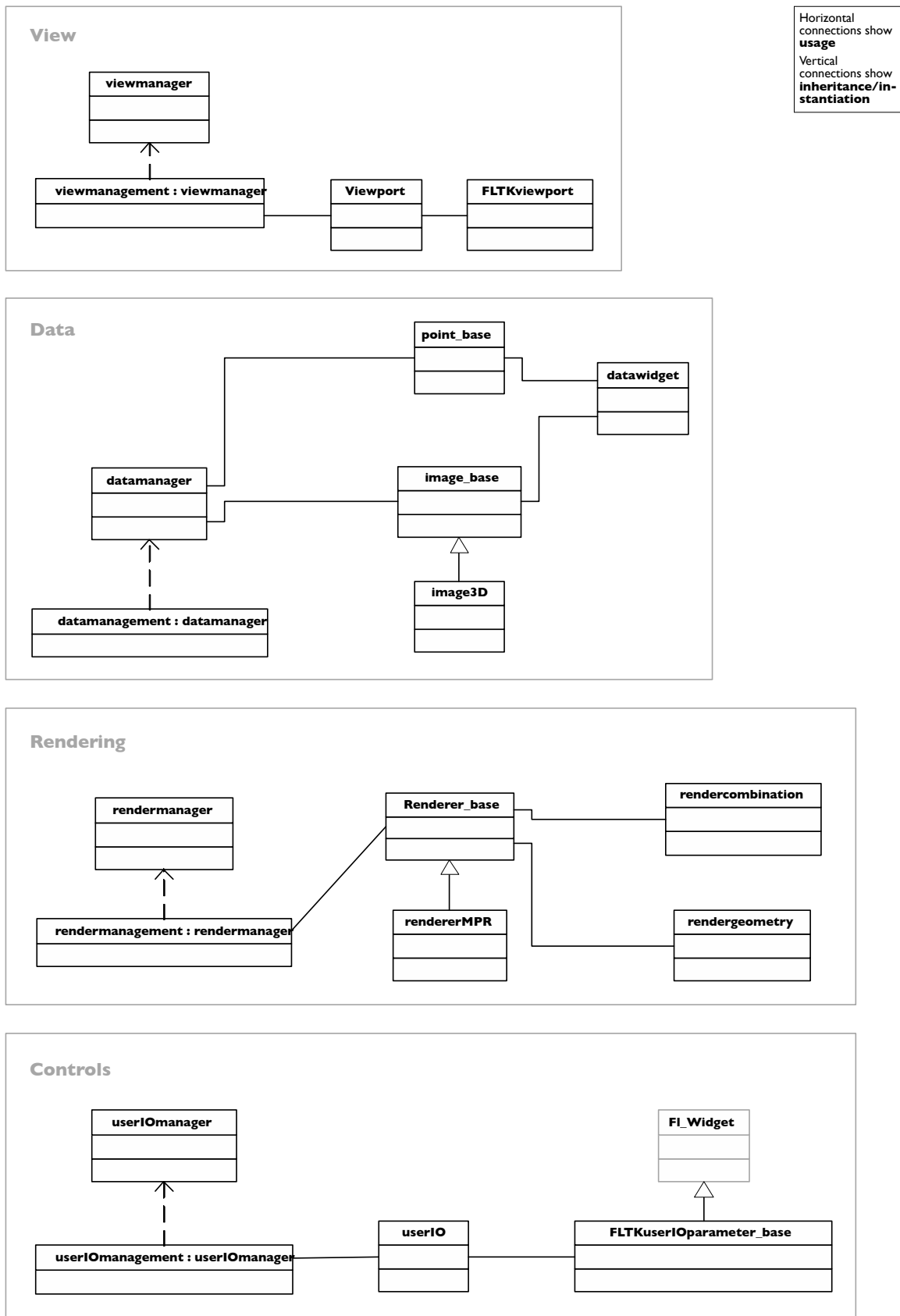
### **Threshold\_overlay**

Overlay to viewports that displays a segmentation preview for the 2D histogram segmentation project.

### **FLTK2Dregionofinterest**

Overlay to viewports that displays region of interest information, including a rectangular selection and threshold\_overlay segmentation previews.

### 3.4. Class diagram





### 3.5. References

- <sup>1</sup> Freshmeat (2007). “Freshmeat license breakdown”, <http://freshmeat.net/stats/#license> (2007-01-26)
- <sup>2</sup> GNU foundation (2007). “GNU Licenses”, <http://www.gnu.org/licenses/licenses.html> (2007-01-25)
- <sup>3</sup> (contributors of) Wikipedia (2007). “BSD licenses”, [http://en.wikipedia.org/wiki/BSD\\_licenses](http://en.wikipedia.org/wiki/BSD_licenses) (2007-01-26)
- <sup>4</sup> Gosbee, J. and Ritchie, E. 1997. Human-computer interaction and medical software development. *interactions* 4, 4 (Jul. 1997), 14
- <sup>5</sup> Sellers, M. 1994. Designing for demanding users. *Interactions* 1, 3 (Jul. 1994), 54-64
- <sup>6</sup> Reenskaug, T. (1979). “Models-views-controllers”, <http://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf>, Xerox
- <sup>7</sup> Lewis, C & Rieman, J (1994). “Task-Centered User Interface Design: A Practical Introduction”, <http://www.hcibib.org/tcuid/>
- <sup>8</sup> Belaroussi, B, Milles, J, Carme, S, Zhu, Y.M, Benoit-Cattin, H (2006). “Intensity non-uniformity correction in MRI: Existing methods and their validation”, *Medical Image Analysis* 10 (2006) 234–246
- <sup>9</sup> Styner, M, Brechbühler, C, Székely, G and Gerig, G (2000). “Parametric Estimate of Intensity Inhomogeneities Applied to MRI”, *IEEE Transactions on medical imaging*, vol. 19, no. 3, March 2000
- <sup>10</sup> Chuang, K-S, Tzeng H-L, Chena S, Wua, J, and Chen, T-J (2006). “Fuzzy c-means clustering with spatial information for image segmentation”, *Computerized Medical Imaging and Graphics* Volume 30, Issue 1 , January 2006, Pages 9-15
- <sup>11</sup> Vovk, U, Pernuš, F and Likar B. (2006). “Intensity inhomogeneity correction of multispectral MR images” *Neuroimage* 2006 Aug 1;32(1), 56
- <sup>12</sup> Andersen, A, Zhang, Z, Avison, M. J and Gash, D. M (2000). “Automated segmentation of multispectral brain MR images”, *Journal of Neuroscience Methods* 122 (2002) 13-23
- <sup>13</sup> Boudraaa, A-O, Dehak, S.M.R, Zhub, Y-M, Pachaib, C, Baob, Y-G and Grimaud, J (2000). “Automated segmentation of multiple sclerosis lesions in multispectral MR imaging using fuzzy clustering”, *Computers in Biology and Medicine* 30 (2000) 23±40

- <sup>14</sup> Zhang, Y, Goldszal, A, Butman, J and Choyke, P (2001). “Improving Image Contrast Using Principal Component Analysis for Subsequent Image Segmentation”, *Journal of Computer Assisted Tomography* 25(5):817–822
- <sup>15</sup> Cockburn A, McKenzie B (2001). “3D or not 3D?: evaluating the effect of the third dimension in a document management system”, *Conference on Human Factors in Computing Systems, Proceedings of the SIGCHI conference on Human factors in computing systems*
- <sup>16</sup> Tankyevych, O (2006)  
“Automated Abdominal Tissue Segmentation of Multi-Contrast Magnetic Resonance Images”, master thesis, Uppsala University
- <sup>17</sup> Udupa, J. K, LeBlanc, V. R, Zhuge, Y, Imielinska, C, Schmidt, H, Currie, L. M, Hirsch, B. E and Woodburn, J (2005). “A framework for evaluating image segmentation algorithms”, *Computerized Medical Imaging and Graphics* 30 (2006) 75–87
- <sup>18</sup> Lewis, C (1982). “Using the thinking-aloud method in cognitive interface design”, *IBM Research Report RC 9265*, Yorktown Heights, NY, 1982