



UNIVERSITÄT  
KOBLENZ · LANDAU

Fachbereich 4: Informatik

# **Visualizing anatomical labels on brain images: developing a multi-functional display method**

## **Bachelorarbeit**

zur Erlangung des Grades Bachelor of Science (B.Sc.)  
im Studiengang Computervisualistik

vorgelegt von  
**Marius Meyer**

Erstgutachter: J.-Prof. Dr. Kai Lawonn  
(Institut für Computervisualistik, AG Medizinische Visualisierung)

Zweitgutachter: Prof. Dr. Rolf A. Heckemann  
(Department of Radiation Physics, University of Gothenburg)

Koblenz, im März 2019



## Erklärung


Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ja    Nein

Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.

Altenhof, 03.03.2019

(Ort, Datum)



(Unterschrift)

# Contents

<b>1</b>	<b>Abstract</b>	<b>1</b>
<b>2</b>	<b>Background</b>	<b>1</b>
<b>3</b>	<b>Problem</b>	<b>2</b>
3.1	Challenges of anatomical segmentation methods . . . . .	2
3.2	State of the art . . . . .	2
3.3	Related work . . . . .	4
<b>4</b>	<b>Material</b>	<b>5</b>
4.1	Images and segmentations . . . . .	5
4.2	Software . . . . .	5
<b>5</b>	<b>Development</b>	<b>5</b>
5.1	First theoretical steps . . . . .	5
5.2	Programming of the double contour . . . . .	6
5.3	Programming the showing of multiple contours . . . . .	8
<b>6</b>	<b>Result</b>	<b>10</b>
6.1	Interface . . . . .	10
6.2	How a potential work flow could look like . . . . .	10
<b>7</b>	<b>Testing and test result</b>	<b>11</b>
<b>8</b>	<b>Discussion</b>	<b>11</b>
8.1	Double Contour . . . . .	11
8.2	Why the display mode is useful . . . . .	11
8.3	Personal lessons and experiences . . . . .	12
8.4	Limitations of the development results . . . . .	12
8.5	How the program can be further developed . . . . .	13
<b>9</b>	<b>Conclusion</b>	<b>13</b>
<b>10</b>	<b>Notes</b>	<b>13</b>

## 1 Abstract

This work describes a novel software tool for visualizing anatomical segmentations of medical images. It was developed as part of a bachelor's thesis project, with a view to supporting research into automatic anatomical brain image segmentation. The tool builds on a widely-used visualization approach for 3D image volumes, where sections in orthogonal directions are rendered on screen as 2D images. It implements novel display modes that solve common problems with conventional viewer programs. In particular, it features a double-contour display mode to aid the user's spatial orientation in the image, as well as modes for comparing two competing segmentation labels pertaining to one and the same anatomical region. The tool was developed as an extension to an existing open-source software suite for medical image processing. The visualization modes are, however, suitable for implementation in the context of other viewer programs that follow a similar rendering approach. The modified code can be found here: [soundray.org/mm-segmentation-visualization.tar.gz](http://soundray.org/mm-segmentation-visualization.tar.gz).

## 2 Background

In recent years the use of IT in the medical sector made substantial advances. One of these advanced technologies evolved around the computational processing of medical image data generated, for example, by magnetic resonance (MR) imaging or X-ray computed tomography (CT).

The medical image data generated is a spatial map of an object in two or three dimensions where each image element (pixel or voxel) carries a value of a physical measurement, such as magnetic resonance or radioopacity.

The analyzing of this data is called medical image analysis and is generally used for researching and understanding the function, structure and diseases of living beings, especially humans.

Due to the availability of experts to interpret images visually being limited and not matching the increasing amount of image information generated, there is a drive in interest for automatic methods that perform or support image interpretation.

This interest gets addressed by the growing numbers of automated methods, which enable the computer to read and analyze the data of a picture by itself.

Those methods are often adapted from several fields for example the one of image and pattern recognition.

A potential method could be i.e. the automatic recognition of a certain organ or tissue visible in a certain set of image data.

Due to the challenging precision these programs have to deliver, their results must be evaluated properly.

Those programs' output usually are a label images. A label image accompanies a regular image, in this case a medical image. The label corresponds spatially with the image to the extend that voxel values represent anatomical categories instead of physical measurements.

A possibility to evaluate these labels of a program, is visualizing them and comparing them to other automatic and also manual segmentations.

### 3 Problem

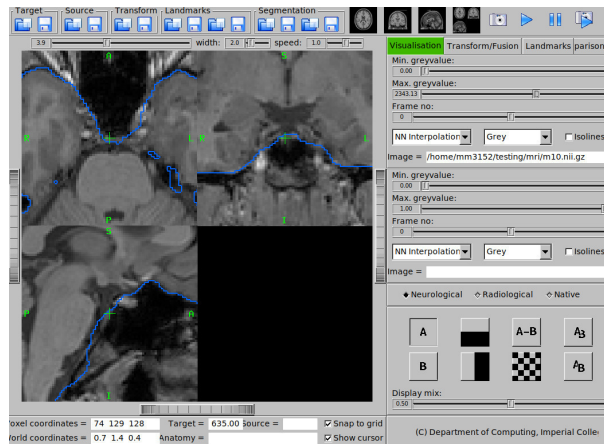
#### 3.1 Challenges of anatomical segmentation methods

Automatic anatomical segmentation is difficult to implement because it is an ill-posed problem: even experts can disagree and there is no objective truth. The developers of automatic anatomical often have visualize the results to properly evaluate them. This is usually done by somehow projecting a single segmentation label in the context of the segmented image, for example a MR image. Evaluation often requires a comparison of a pair of labels. These can be one label done by an expert and one generated by an algorithm or two labels from two competing segmentation algorithms. Solutions for comparative visualization are usually unsatisfying. (see e.g. Figure 1)

#### 3.2 State of the art

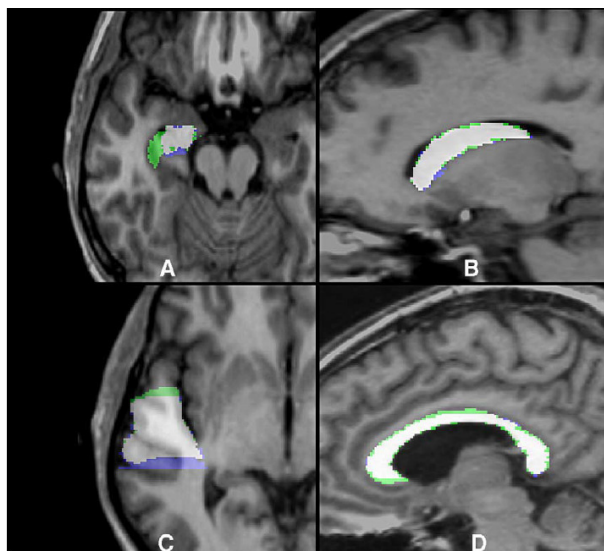
Standard methods of programs that offer label displaying are not geared to meet these requirements. The two most common displaying modes each resolve and cause a problem respectively.

Contour overlays depict the label boundary using a narrow line (cf. Figure 1),so the image remains nearly unaltered. The disadvantage of this method is that one can easily lose anatomical orientation. Therefore it is hard to figure out what is inside and what is outside of the segmentation label, especially on higher zoom levels.



**Figure 1:** The image shows three panels showing 2D sections in orthogonal orientations centering on a common location. The contour of a single label is shown. There is no orienting information for determining what is inside and what is outside of the label, therefore making it hard to figure it out for the user. (Marius Meyer, 2019)

In contrast, Area overlays (cf. Figure 2) usually provide good anatomical orientation, but often hamper the view on parts of the image information underneath the overlay. Most programs do not support a direct comparison of two competing segmentations or labels that regard the same anatomical region. Thus for the comparison it is often required to open two instances of the program. Each instance displays one of the segmentations, comparing them side-by-side.

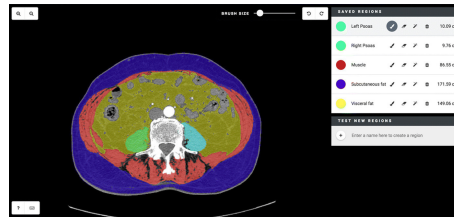


**Figure 2:** Example for the problem of the area overlay obscuring image information (Heckemann, 2006):

This image shows four 2D sections from different regions in 3D MR brain images with label comparisons. In each panel, one of the labels shown is a generated label, the other is a reference drawn by an expert. The labels are shown as semitransparent area overlays of the MR image. The used colors mean the following: regions where the labels intersect are shown in grey, regions included in the automatic label that the expert did not include are shown in green, regions included in the expert label that the automatic label did not include are shown in blue.

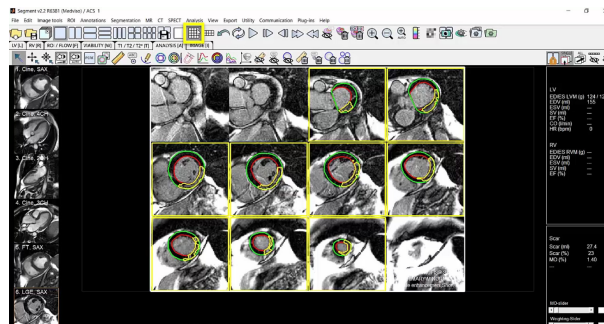
### 3.3 Related work

One related work I found regarding the field of segmentation visualization was the Coreslicer (Mullie et al., 2019) (cf. Figure 3), a free and open-source web-based interface aiming to facilitate measurement of analytic morphomics by clinicians. It also enables muscle and fat measurements. While this open source software uses DICOM images that were taken by CT, it also focuses around the the visualization of segmentations.



**Figure 3:** Coreslicer Interface, displaying the area overlay method. The image shows an sagittal view of the human spine where regions are marked with colored areas, that obscure parts of the image information.( Louis Mullie, 2019)

Another program that also focuses mainly on visualization is segment (Heiberg et al., 2010)(cf. Figure 4). Segment offers analysis tools for MRI, CT and more. It can be utilized for Radiology and Cardiology applications. This program uses the contour overlay for its displaying methods.



**Figure 4:** The Segment Interface is displaying the contour overlay method in 12 different panels. Each one is showing a different image, while always the same region of interest marked. (Heiberg et al., 2010)



## 4 Material

### 4.1 Images and segmentations

The 3D MR images used for this project work were retrieved from the Hammers Atlas Database (<http://brain-development.org/brain-atlases/adult-brain-atlases/>), a publicly available collection of MR brain images with corresponding volumes of anatomical segmentation labels drawn by experts. Two types of brain labels were used in the project. The first type of labels was created merging expert labels into a single binary label outlining the brain. The second type of labels was automatically generated by a segmentation method called pinfram, which labels the brain on MR images of the human head. (Heckemann et al., 2015)

### 4.2 Software

In my search for a new better way to show labels for comparison, there were several options that I had to choose from: implementing a viewer from scratch, using a visualization library or extending an existing viewer software. Due to limited time for the project I chose to extend the rview package of IRTK. It is an open-source project and stands for image registration tool kit. IRTK provides medical image registration including rigid, affine and non-rigid registration. The original IRTK software was used for a refactoring by BioMedia, implementing a new modular registration framework and the tools for processing cortical surface meshes to create MIRTk. In my thesis work I used the viewer package of IRTK which has been converted into a MIRTk compatible version by Dr. John Cupitt. The reason I chose MIRTk for the project was that it is programmed in C++ and an open source project. It was also the most used platform of the people I knew that were working in this area of science. Dr. Andreas Schuh, the maintainer of the program, was very easy to reach for questions and his answers furthered my suggestion to extend this viewer-program. Rview provides standard label display methods. The MRITk rview module was extended by a novel label display mode. The interface of the rview module is programmed by using FLTK. This extension was developed within Ubuntu Linux with an integrated development environment (Eclipse 2018-09).

## 5 Development

### 5.1 First theoretical steps

The first work that was done on this bachelor thesis project was an exploration to get a better idea of what purposes my program should fulfill.

For this purpose I started with a test task of the viewer function of MIRTk to compare two MRI image labels, one generated by an old version of pinfram and one generated by the current version of pinfram. The comparison was planned to include scrolling through the slices of the 3D-image and, furthermore, the evaluation which of the

generated labels is the better one. To avoid bias, the method of a certain label was not named and each of the two folders included images of both methods of pinfram. During the test task it became clear where the problems within the current version of the program for doing this job could be found: It takes a long time and is error-prone. It was sometimes hard for me to figure out whether a circled region was inside or outside of the label. Therefore, the goal of the end results was: there was the need to introduce a clearer outlining technique for better region orientation around the labels borders. Additionally, a feature was needed to distinguish the differences of the label images more easily within a shorter duration of time. After that I put effort into how it would be possible to mathematically process both images into a resulting image that shows exactly which parts of the two labels are common and which are just included in one of these. After several approaches with distinguishing multiple cases or a formula like  $(f * g) + ((f - g) - (g - f))$  there was a thought of a much simpler method.  $f + 2 * g$  Therefore, a pixel with the value 0 is found in none of the labels, 1 is only in label f, 2 is only in image g and 3 is in both labels. This thought was very useful for later steps in programming. Another point to address was how the displaying of the in-/out-contrast should look like or how the display could be generally improved. One option was stippling or hatching, to make it clear which regions are marked without covering most of it. Usually, it is used to create 3D-effects, but in this case it would instead be useful for the aforementioned reasons. Another option was marching ants, which would have also been useful for the aforementioned reasons. A third option was the combination of both methods, what would be a marching version of either hatching or stippling. Finally, the decision was made on a double contour, that means an adjacent border around the original contour.

## 5.2 Programming of the double contour

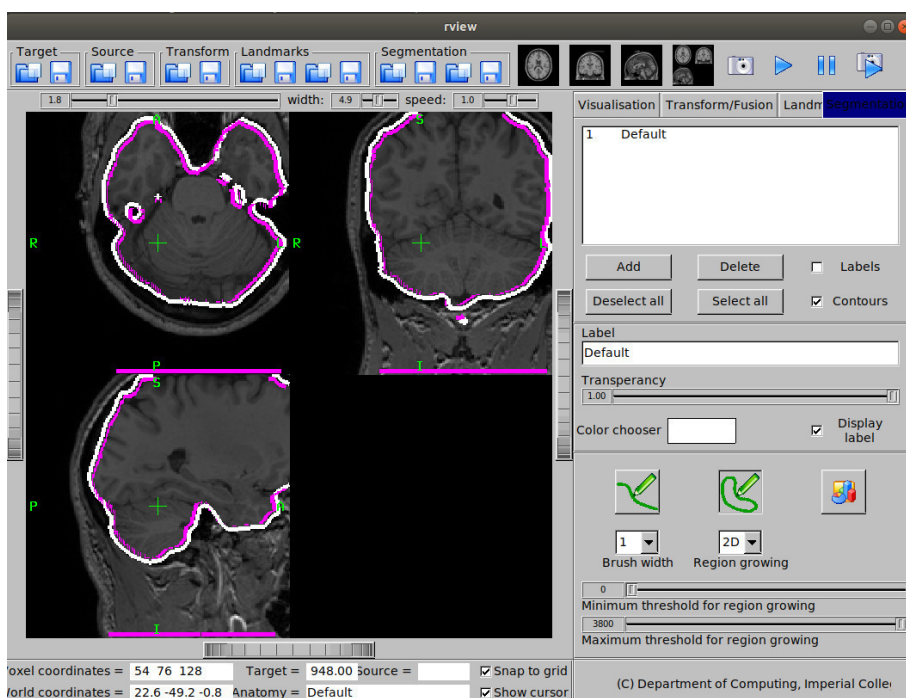
The programming part of the thesis was started off by editing the segmentation mode of the viewer. The programming of the double contour was the quintessential part of this thesis, therefore, the first goal was to create the second contour around the original one to make it clearer to the user which parts are inside and which are outside the label. Therefore, an understanding of the way the existing algorithm works was needed, especially the drawing part of it.

The algorithm works with a nested for-loop over all pixels in the selected area. Within the loop the algorithm reacts to non-black pixels via an if-clause. If a pixel has indeed a value over 0, it is checked whether there is a black pixel in the four straight line neighbor pixels. If this is the case, the algorithm draws a line via `GL_LINE`, an Open-GL primitive, between this and the aforementioned found pixel with value over 1. To check the results for other drawing modes like `GL_triangles`, those were put into the code, but neither of those were helpful. Next the first approach for the drawing of the double contour was implemented. This approach included the registration of each vertex of the first contour.

After that the algorithm would run to the vertices and move them one pixel into the

outside region of the picture. Which direction this was, was determined by the quarter of the picture the vertex was in. This did not work out for multiple reasons. First, the execution and therefore the drawing was slow. Second, the moving of all vertices in one area in one certain direction did not work out due to the regular occurrence of concave edges. Additionally, the new contour was not a constant line, but uncountable small line pieces that were barely attached. Focusing on the last problem a method was implemented that always remembered the second vertices of the two that were used to draw a line between and made it the first vertex for the next line. By doing so, all vertices would be attached. After implementing this, the contour was too thick and the line ran into spaces it was not supposed to. Therefore, another approach was needed.

I considered, but not did completely implement an algorithm that would run through the whole image that included the first contour already. But then it would be nearly the same algorithm and, therefore, it would be inefficient. Due to that the next idea was to implement it in the original loop. With this approach, after a border was found, another step in the target direction was required, followed by the drawing of a second line there. At this stage of the project the color of this line was fixed. What was also needed was the implementation of an edge detection to prevent the program from crashing. The crash was caused by a null pointer exception, due to the program trying to read a value from a pixel that was not part of the image. Following that a problem occurred that caused some of the double contour pixel to drop into the bottom line of the display.

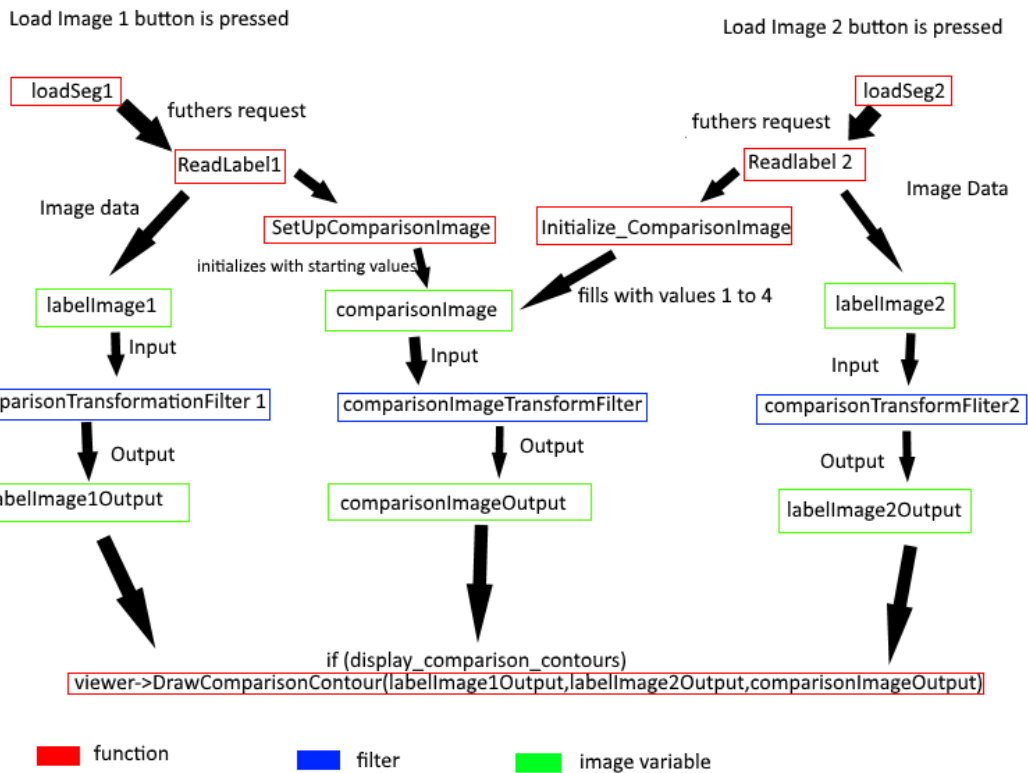


**Figure 5:** Programming issue during the work on the double contour ( Marius Meyer, 2018)

What was also needed was an offset for the line drawing that was influenced by the selected width, otherwise an increasing width would lead to the second contour to be drawn over most parts of the first one. An error that was more challenging than expected was the one that occurred when putting the focus on the upper edge of the label borders. After various failed attempts to find the issue for whose search s-trace was used, it was decided to finally renew the conditions of edge treatment from scratch. After that the second contour worked like it was supposed to and the program did not crash anymore.

### 5.3 Programming the showing of multiple contours

After finishing the programming of the double contour, the next task was the programming of an extension of the original program that would improve the comparison of images by allowing the user to load two segmentation labels and use multiple options to compare them. At first there was a need for a new slot in the user interface that would provide interactive elements to customize the display for the user's needs. Therefore, there was also a need to design a class for the comparison. The new interface had to include two buttons to load the label images that would be compared. Additionally, there was a "Generate image"-button that would create a contrast map out of both images. Below that was the space for visualization options for the image: selecting visibility, the first contour's color and the second contour's color. The drawable contours were implemented as common contour (the parts, where the label boundaries matched), segmentation 1 (showing the outline of label 1) segmentation 2 (showing the outline of label 2),  $1 \setminus 2$  (the areas, that were included in label 1, but not in 2) and  $2 \setminus 1$  (the areas, that were included in label 2, but not in 1). Later the contour for intersection was added. Last, there had to be a button for switching on and off the displaying of the second contour. Due to the fact that FLTK was used for the entire original interface, the label-comparison part was also built on FLTK. Initially the algorithms were called overinclusion, underinclusion and intersection. The algorithms ran over both labels and decided for each pixel if it was 0 in both labels, greater than 0 in one or even in both. Overinclusion was the case  $1 \& 0$ , underinclusion  $0 \& 1$  and intersection  $1 \& 1$ . Those cases resulted in the values 0 to 4 in the respective order that were put into the comparison image. The display of the generated image needed to be on every station of the pipeline. There were several possible approaches regarding the integration of multiple contour colors. One possibility was the usage of Segmenttable, a pre-existing color selection class for the segmentation part of the program. There were several issues with the communication between the new functions and the ones already existing in Segmenttable. Because of those issues the class Comparisontable was designed. The comparison map needed an interpolator and a transform, before being usable. The map was used for the contours  $1 \setminus 2$ ,  $2 \setminus 1$  and intersection. The other contours were generated out of the two label images themselves.

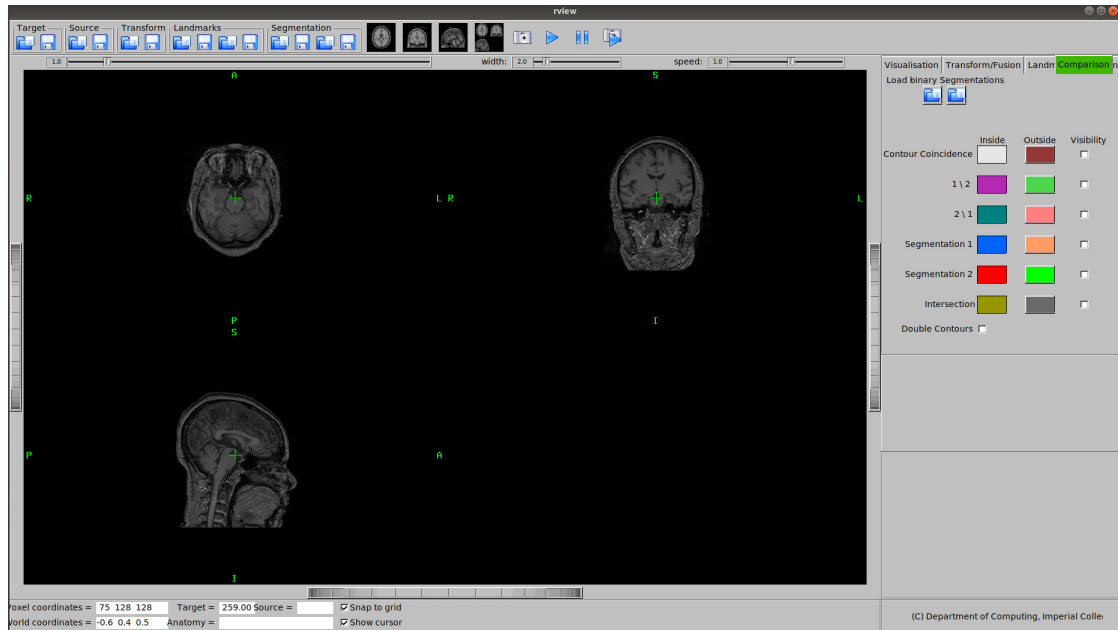


**Figure 6:** Overview of the model from image to drawing: Button pressing, loading images, generating the comparison image and filling it with values regarding image similarities of 1 & 2. Then all images are filtered and possibly drawn (Marius Meyer, 2019)

## 6 Result

### 6.1 Interface

The interface that was part of the project is integrated into the interface of the original version of rview as an additional tab that sits between the tabs for landmarks and segmentation.



**Figure 7:** Initial appearance after opening the comparison tab( Marius Meyer, 2019)

The tab contains the loading option for the two images that are supposed to be compared. Below them are the color selections for the first and second contour of the respective viewing options with a visibility check box behind it. On the bottom of this area there is also a check box for setting the second contour on and off.

### 6.2 How a potential work flow could look like

After starting the program and opening the "Comparison" tab, the button the "Open Source Image" is used to load the MR image into the viewer-display. Next two label images that are to be compared are loaded into the system by selecting the "Comparison"-Tab by using the buttons right underneath the tab selection. The comparison image is generated as soon as the second image is loaded. Now it is up to the user if they just want to look at the segmentations or actually compare them. In case they just want to view them individually, it is possible to take a very close look at the label contour of one label by selecting "Segmentation" "1" or "2" and, for better orientation, turning the double contour on. Zooming is done by moving the zoom

slider to the right. In case that the contour colors differ to little, they can be adjusted using the color changer. For the case that the user wants to compare two images simultaneously, it is possible to select both segmentations, or use one of the other viewing modes. To check similarities between the segmentations, it makes sense to use "Common Contour" to see where the label boundaries match, or to use "Intersection" to view which areas are included in both labels . To get an overview of the areas where the segmentations differ, the "2\1" and "1\2" checkboxes should be used.

## 7 Testing and test result

While the program was developed iteratively, a testing stage followed one developing step and before going forward to the next step. Any errors were eliminated before proceeding to this next step. Near the end of the development cycle, the program was tested in a scenario similar to real-world applications to ensure that the design enables a smooth workflow.

## 8 Discussion

This project delivered a display tool for segmentation labels that solves several problems of conventional display modes. The core elements of the success of the project are the double contour display and a selection of several comparison modes for pairs of binary label images.

### 8.1 Double Contour

The implementation of the double contour solves both problems that area overlays and contour overlays have. The double contour display avoids the problem of hiding information of obscuring parts of the image underneath. The problem of lack of orientation, especially at higher zoom levels, is solved by the second contour that indicates which side of the contour is the inside and which is the outside.

### 8.2 Why the display mode is useful

This program extension of MIRTk Viewer can be used in multiple modes for comparison of two binary images of a certain data type. It is designed for label images, but without major concerns it can also be used for other binary images. The visualization options for the comparison offer four different modes to highlight the difference: contour coincidence, 1\2, 2\1, intersection and the direct comparison of the two displayed contour lines. A decisive advantage is the providing of the double contour feature, which helps the user to keep a sense of orientation, without having information in the underlying gray-level image covered.

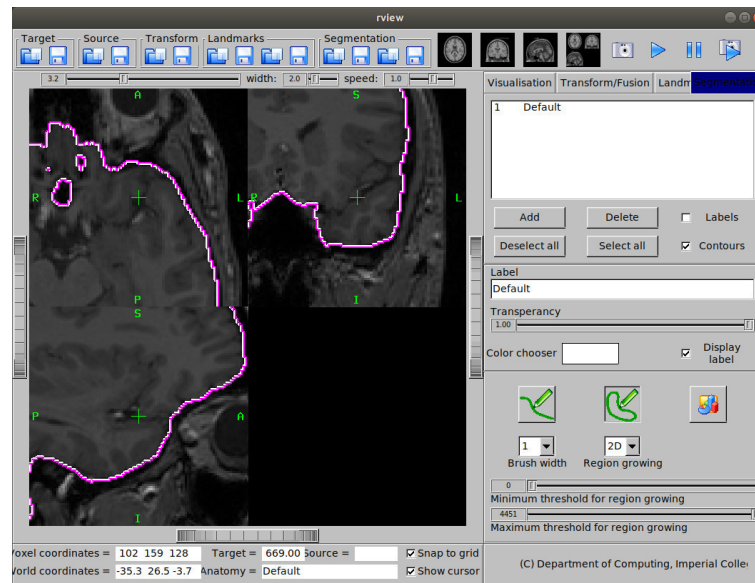


Figure 8: Double Contour display ( Marius Meyer, 2018)

### 8.3 Personal lessons and experiences

I personally gained a lot of programming experience and some medical knowledge from this project. Due to the fact that at university I had written smaller programs only before, but had never dealt with a project with the size of MIRTk, I learned that one first has to understand the program before one can really implement a new part into it. For myself the best way to find out how a program works is at first to make a small overview about classes and how they are connected. This approach is good for the start, but following one subsequently has to start to work with the program in order to understand how it works. Starting with some small changes and seeing how the program reacts is the next step. Following that it is usually possible to change or to expand the code for one's own purposes.

### 8.4 Limitations of the development results

Fundamentally the program is limited by the same things as rview. That means it supports the same data types and only them. The interface is somewhat outdated and its resolution can cause viewing problems on some monitors. It is only possible to use the new feature within rview. Additionally, it is not possible to compare more than two labels at the same time. Furthermore, the drawing is often not that fast. Regarding the testing it should be considered to do tests with independent people, who do not know the program yet or at least who do not know the extension. That should reduce the bias caused by the testing done by myself.



## 8.5 How the program can be further developed

The current state of the program poses several challenges for a developer who wants to improve it. Firstly as in nearly every program, the speed could be optimized. The delayed drawing is caused by the many drawing options and adjustments that have to be checked. By using specialized libraries it would be possible to speed it up.

Secondly, the drawing of the second contour creates one-pixel-big overlaps at some points between the first and the second contour. Most users probably would not notice this and it does not take anything away from the usefulnesses of the program, but it should be fixed to optimize the program.

Finally rview could be overhauled regarding its graphics and interface entirely, but just for optical and not for functional reasons.

## 9 Conclusion

This thesis project resulted in a program extension for the medical image viewer rview that visualizes binary segmentation labels while avoiding common problems that overlays usually have. It provides multiple modes and options for comparing two binary label images within a common image space.

## 10 Notes

Mathematical notation:

set (capitals):  $\mathcal{V}$

vectors (lower case, bold):  $\mathbf{v}$

scalars (lower case):  $s$

matrices (capitals, bold):  $\mathbf{M}$

## List of Figures

- 1 The image shows three panels showing 2D sections in orthogonal orientations centering on a common location. The contour of a single label is shown. There is no orienting information for determining what is inside and what is outside of the label, therefore making it hard to figure it out for the user. ( Marius Meyer, 2019) . . . . . 2
- 2 Example for the problem of the area overlay obscuring image information (Heckemann, 2006): . . . . . 3
- 3 Coreslicer Interface, displaying the area overlay method. The image shows an sagittal view of the human spine where regions are marked with colored areas, that obscure parts of the image information.( Louis Mullie, 2019) . . . . . 4

4	The Segment Interface is displaying the contour overlay method in 12 different panels. Each one is showing a different image, while always the same region of interest marked. (Heiberg et al., 2010) . . . . .	4
5	Programming issue during the work on the double contour ( Marius Meyer, 2018) . . . . .	7
6	Overview of the model from image to drawing: Button pressing, loading images, generating the comparison image and filling it with values regarding image similarities of 1 & 2. Then all images are filtered and possibly drawn ( Marius Meyer, 2019) . . . . .	9
7	Initial appearance after opening the comparison tab( Marius Meyer, 2019)	10
8	Double Contour display ( Marius Meyer, 2018) . . . . .	12

## References

[1] E. Heiberg, J. Sjögren, M. Ugander, M. Carlsson, H. Engblom, and H. Arheden, Design and Validation of Segment – a Freely Available Software for Cardiovascular Image Analysis, *BMC Medical Imaging*, 10:1, 2010. Image from website :medviso.com

[2] Louis Mullie, Jonathan Afilalo, CoreSlicer: a web toolkit for analytic morphomics, *BMC Medical Imaging* 19:15 2019.

[3] PinCram: Heckemann RA, Ledig C, Gray KR, Aljabar P, Rueckert D, Hajnal JV, Hammers A. Brain Extraction Using Label Propagation and Group Agreement: PinCram. *PLoS One*. 2015 Jul 10;10(7):e0129211. doi: 10.1371/journal.pone.0129211. eCollection 2015.

[4] IRTK: J. A. Schnabel, D. Rueckert, M. Quist, J. M. Blackall, A. D. Castellano Smith, T. Hartkens, G. P. Penney, W. A. Hall, H. Liu, C. L. Truwit, F. A. Gerritsen, D. L. G. Hill, and D. J. Hawkes. A generic framework for non-rigid registration based on non-uniform multi-level free-form deformations. In *Fourth Int. Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI '01)*, pages 573-581, Utrecht, NL, October 2001

[5] MIRTk: A. Schuh, A. Makropoulos, E. C. Robinson, L. Cordero-Grande, E. Hughes, J. Hutter, A. N. Price, M. Murgasova, R. Pedro A. G. Teixeira, N. Tusor, J. K. Steinweg, S. Victor, M. A. Rutherford, J. V. Hajnal, A. D. Edwards, D. Rueckert. *bioRxiv*, 2018.

[6] Hammers Atlas: Hammers A, Allom R, Koepp MJ, Free S., Myers R, Lemieux L, Mitchell TN, Brooks DJ, Duncan JS, Aug.

2003. Three-dimensional maximum probability atlas of the human brain, with particular reference to the temporal lobe. *Hum. Brain Mapp.* 19 (4), 224-247.  
<http://dx.doi.org/10.1002/hbm.10123> Gousias IS, Rueckert D, Heckemann RA, et al., 2008, Automatic segmentation of brain MRIs of 2-year-olds into 83 regions of interest, *Neuroimage*, Vol:40, ISSN:1053-8119, Pages:672-684.  
<http://dx.doi.org/10.1016/j.neuroimage.2007.11.034>

- [7] Image from figure 1: Heckemann, R. A., Hajnal, J. V., Aljabar, P., Rueckert, D., Hammers, A., October 2006. Automatic anatomical brain MRI segmentation combining label propagation and decision fusion. *NeuroImage* 33 (1), 115-126.  
<http://dx.doi.org/10.1016/j.neuroimage.2006.05.061>