Diplomarbeit

Tissue-differentiating segmentation for a regional anaesthetic simulation

vorgelegt von
Cand.-Inform. Christian Teich
(Mat.-Nr.: 225720)

RWTH AACHEN UNIVERSITÄTSKLINIKUM
INSTITUT FÜR MEDIZINISCHE INFORMATIK
Abteilung Medizinische Bildverarbeitung

Aachen, 12.12.2007

Volume 3, Band 3
ISSN 1860-8906
ISBN 978-3-9811848-8-4
Ich, Christian Teich, versichere an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner Prüfungskommission vorgelegt.
## Contents

1 **Introduction** ................................................. 9  
   1.1 Motivation ................................................. 9  
   1.2 The RASim-Project .................................... 10  
   1.3 Goal formulation ........................................... 10  
   1.4 Structure of the thesis .................................... 12  

2 **State of the Art** ........................................... 13  
   2.1 Medical Image Segmentation .................................. 13  
   2.2 Tissue Classes ............................................. 16  
      2.2.1 Bones Segmentation .................................... 16  
      2.2.2 Soft Tissue Segmentation ................................ 16  
      2.2.3 Vessel Extraction and Segmentation ...................... 17  
      2.2.4 Nerve Extraction and Visualization ...................... 18  

3 **Material and Methods** .................................... 21  
   3.1 Data acquisition ........................................... 21  
      3.1.1 MRI background ......................................... 21  
      3.1.2 Sequences and results .................................. 23  
   3.2 File Formats .................................................. 27  
      3.2.1 DICOM file format ....................................... 27  
      3.2.2 Object file format .................................... 28  
   3.3 Software Tools ............................................... 29  
      3.3.1 Insight Segmentation and Registration Toolkit (ITK) ........ 29  
      3.3.2 Visualization Toolkit (VTK) ............................ 30  
      3.3.3 Medical Imaging Interaction Toolkit (MITK) ............ 31  
      3.3.4 Qt ....................................................... 31  
      3.3.5 CMake .................................................... 34  
   3.4 Algorithms ................................................... 35  
      3.4.1 Fuzzy Connectedness ................................... 35  
      3.4.2 Fuzzy C–Means Clustering ................................ 36  
      3.4.3 Deformable Models ...................................... 38  

4 **Implementation** ........................................... 41  
   4.1 Basic Concepts ............................................... 43  
   4.2 Framework ................................................... 45  
      4.2.1 Basic Data Structure ................................... 47  
   4.3 Scripting and Processing .................................... 49  
      4.3.1 Creating Scripts ....................................... 49  
      4.3.2 Exporting Segmentation Scripts ......................... 49  
      4.3.3 Importing Segmentation Scripts ......................... 50  
   4.4 Labeling and Data Export .................................... 51  
   4.5 Graphical User Interface .................................... 53  
      4.5.1 RASimSeg Graphical User Interface ...................... 53  
      4.5.2 Viewer Controls ......................................... 55  
      4.5.3 Processing Controls ..................................... 56  
      4.5.4 Labeling Controls ........................................ 57  
   4.6 Plugins ....................................................... 58
5 Results

5.1 Vessel .......................................................... 68
5.2 Soft Tissue ..................................................... 70
5.3 Muscles .......................................................... 71
5.4 Bones ........................................................... 71
5.5 Proposed Segmentation Script ................................. 74

6 Conclusion ......................................................... 76

7 Future Work ....................................................... 76

8 References ........................................................ 78
List of Figures

1. Femoral nerve block ................................................. 9
2. General processing pipeline for image segmentation ............... 14
3. Bone segmentation with active contours .......................... 16
4. Soft tissue segmentation with fuzzy connectedness ............... 17
5. MR angiography visualized via MIP ................................ 17
6. Nerve visualization ................................................... 19
7. Diffusion model ...................................................... 19
8. Nuclei spin, precession and magnetic moment ..................... 21
9. Impact of using additional coils .................................... 23
10. Nerve in MR imaging ............................................... 25
11. MR imaging artifacts ............................................... 26
12. Chosen MRI sequences ............................................. 26
13. MITK viewer/data tree concept .................................... 32
14. Qt signal/slot concept ............................................... 34
15. Level Set Concept and Evolution .................................. 40
16. RASimSeg Activity diagram ....................................... 43
17. RASimSeg application logic ....................................... 46
18. Relations between the RASimSeg application and used toolkits 46
19. Basic data structure .................................................. 47
20. Example segmentation script ....................................... 51
21. Label assignment .................................................... 52
22. Layout of the main application .................................... 53
23. Screenshot of the main application ................................ 54
24. GUI viewer tab ....................................................... 55
25. GUI processing tab .................................................. 56
26. GUI labeling tab ..................................................... 57
27. Thresholding plugin .................................................. 58
28. Simple region growing plugin .................................... 59
29. Mathematical morphologic erosion plugin ......................... 60
30. Fuzzy connectedness plugin ...................................... 61
31. Fuzzy c–means clustering plugin ................................ 62
32. Effect of thresholding the membership function images ........ 64
33. Geometric active contours plugin ................................ 65
34. Initial level set model .............................................. 66
35. Feature image for a gradient magnitude based level set segmentation 66
36. Feature image creation for a Canny–edge level set segmentation 67
37. Sample slices taken from the MR angiography image .......... 68
38. Sample slices taken from the MR morphology image ............ 69
39. Region growing and dependence on the selection of seed points 69
40. Segmentation results with fuzzy connectedness ................ 70
41. Labels produced by fuzzy c–means clustering .................... 70
42. Isosurface of the soft tissue segmentation ....................... 71
43. MR morphology bones ............................................. 72
44. MR sequence of the femur-joint .................................. 72
45. Results of level set bone segmentation .......................... 73
46. Problems of level set bone segmentation ........................ 73
47. Models created with the proposed script ......................... 74
Diagram of the proposed segmentation script
**Abbreviation**

CT ................. computed tomography  
DICOM ............. digital imaging and communication in medicine  
FC ................. fuzzy c-means  
FC ................... fuzzy connectedness  
FOV ................ field of view  
ITK ................ insight toolkit  
MIP ................ maximum intensity projection  
MITK ............... medical interaction toolkit  
MR ................... magnetic resonance  
MRI ................ magnetic resonance imaging  
RA ................... regional anesthetics  
RASim ............. regional anesthetic simulator  
SNR ................ signal-to-noise ratio  
TOF ............... time of flight  
XML ............... extensible markup language
1 Introduction

1.1 Motivation

Medical image analysis is an important impetus in the area of image processing, registration and segmentation research. With the technological advance in medical imaging modalities and techniques, it is possible to gather more and more heterogeneous data, with increasing resolution and accuracy. This often huge amount of data have to be prepared and abstracted for a suitable visualization and further usage in medical applications. Beside the steadily increasing acceptance and integration of computer aided diagnosis and surgery planning in the clinic routine, there is also a growing need for training with virtual simulators. The “RASim” project (Regional Anesthetic Simulator) [1] has been initialized in 2006 to develop a virtual simulation for performing local anesthetics on individual virtual patient models.

Regional anesthetics (RA) is a common used technique for deafen particular body regions, without narcotizing the patient all-over. General anesthesia is always afflicted with risks due to the loss of consciousness and respectively with the associated deficiency of the pharyngeal and respiratory reflexes. In regional anesthesia only these nerves are narcotized, which innervate the specific region. In that way, signals transmitted by the efferent and afferent nerve fibers are blocked, leading to paralysis and analgesia in the desired region. Efferent nerves lead signals from the central nervous system to the body region and afferent nerves the opposite way around. Typical scenarios for RA are operations on the upper extremities (e.g. brachial plexus block), lower extremities (femoral nerve block) and abdominal operations or cesarean sections (spinal block).

![Figure 1: Femoral nerve block: left) outer leading structures with 1) inguinal ligament 2) arteria femoralis 3) spina iliaca ant. sup. 4) puncture site; right) puncture with stimulation-needle](image)

To determine the position for puncture, the anesthetist uses so-called outer leading structures or anatomical landmarks for a coarse orientation (Fig. 1). After that, an electric stimulation–needle is used to locate the nerve cord within the tissue. An electric impulse triggers a contraction of associated muscles or a paresthesia stimulus. The strength is directly correlated with the distance between needle and the desired nerve. With decreasing distance the potential of the electric impulse is being lowered. If a minimum distance has been found, a local anesthetics is injected through the needle [2]. To gain experience in regional anesthetics techniques, there is still no satisfying alternative to practicing in real situations. Beside the possibility to damage the nerve and adjacent structures like veins or even arteries with the needle itself, a physician can


run the risk of an irreversible lesion while closing in on the nerve and trigger an electric impulse with a too large potential.

The RASim project was established to simulate such procedures and give the user a visual and haptic feedback of his actions. The simulation can be used by clinicians for training and preparation for individual patients.

1.2 The RASim-Project

The RASim project [1] started in 2006 and is funded through the START program (Aachen University Hospital) and also partially sponsored by a grant from the German Research Foundation (DFG). It is a cooperation of three institutes at the University Aachen. The Clinic of Anesthesia takes care of the medical aspects in the project, the Department of Medical Informatics supervises data acquisition, processing and model creation. The simulator itself is developed by the Virtual Reality Group at the Center for Computing and Communication of the RWTH Aachen.

The goal of the project is a comprehensive virtual environment that allows to perform different block techniques on virtual patient models. A user can train all steps of a regional anesthesia procedure as described above and will get visual and haptic feedback of his actions. Beside the puncture procedure itself, the haptic response also covers the feel for the anatomic landmarks. Another feature is that the simulation is not restricted to a single dataset, i.e. a static model of one patient — the most important drawback of other approaches to this topic. So the underlying virtual patient database can cover a wide range of anatomical variances and even patient individual models [3]. The virtual model required for simulation has to be an anatomically plausible one. This covers representations of bones, musculature, blood vessels, skin, soft tissue and of course the nerve cords.

1.3 Goal formulation

This diploma thesis covers the aspects of the dataset creation for the virtual database in the RASim–Project. Here the focus is not set on the creation of a highly accurate and medic plausible model, like it is required by tools for diagnosing or operation planning. Instead the handling and execution of a semi–automatic segmentation and classification of the different tissues out of a given volume dataset is considered. The relevant tissue classes for the simulation are:

- bones
- muscles
- soft tissue (skin, subcutaneous fat and connective tissue)
- blood vessels
- nerve cords

This thesis with its goal to develop a tissue differentiating segmentation of a medical volume dataset that meets the requirements of the simulation can be divided into the following four subtasks:
1) **Data Acquisition** The medical imaging modality for the project is restricted to magnetic resonance imaging (MRI), so imaging sequences have to be found (i.e. appropriate values for parameters and weights involved in the MRI process) that provide enough information for a suitable segmentation of the desired tissues.

2) **Framework** To perform and control processing and segmentation of the acquired data, a framework with a graphical user interface (GUI) has to be developed. It is designed to guide a user through the segmentation process by mapping the workflow of a segmentation task, keeping an unskilled user in mind. The framework allows to open and manage multiple input image data in various formats (most important the DICOM file format – the common interface to medical imaging modalities). An integrated viewer provides an intuitive visualization of all inputs, intermediate and final results. In order to create a segmentation, the user can run subsequent processing steps. Thereby all parameters and inputs are given over the GUI’s provided by the algorithms. After a segmentation session a user can combine the results of single steps to a final result dataset and export it in various formats. The key idea is that he can also store the order of the subsequent processing steps including their parameters to a XML file. If there is need for a new segmentation of a corresponding dataset, an approved script can simply be loaded in the application and run on the dataset after providing still required parameters (e.g. user-defined seed points). The user also has the possibility to modify and adjust parameters of each step to optimize the result. After all a flexible and extensible framework will be provided that allows an user to perform the model creation for arbitrary datasets, representing a body region, where a blockade–technique will be applied.

3) **Algorithms** Usually a specific acquisition technique is used for each tissue class to enhance and optimize the contrast for segmentation and visualization. Therefore suitable filters and segmentation algorithms have to be found, adapted to the given task and implemented for usage in the framework. A preselection has to be made with respect to a minimize user interaction and a preferable automated initialization. The implementations of the algorithms itself will be loaded as shared libraries into the application via a plugin mechanism. Therefore a common interface will be provided that allows to implement new algorithms and further extend the application.

4) **Demonstration** At the current state of the project, only the femoralis blockade is considered. So the framework is explicitly demonstrated by creating a script that performs a segmentation of a dataset representing the region of the pelvis. The created model will be used as a first prototype in the RASim–project.
1.4 Structure of the thesis

The following section gives a short description of the contents and structure of this thesis. In Chapter 2 a definition and an introduction to image segmentation for medical data is given, including an overview about existing segmentation approaches. The remainder of the chapter deals with related work and the state of the art concerning the segmentation of the single tissues listed above with respect to the underlying MR imaging modality. The first part of Chapter 3 deals with the aspects of the data acquisition for the project, including a short introduction to the physical background of MR imaging, the difficulties of the data acquisition and the results of the different tested settings and sequences. The second part introduces the libraries and tools that are used for the realization of this thesis. The chapter ends with the presentation of the integrated and implemented algorithms and their the mathematical background.

The basic concepts, mechanisms and data structures of the actual implementation are figured out in Chapter 4. Further the application and its GUI components are introduced and the provided algorithms, their GUI's and some aspects concerning the implementation are described. A processing script for performing a segmentation of the pelvis is introduced in Chapter 5 and preliminary results are presented. In Chapter 6 a short conclusion followed by an outlook on possible future work is given.
2 State of the Art

In the field of digital image processing, the image segmentation is a central part and therefore a contemporary and diversified issue. Segmentation denotes the process of partitioning the image domain into disjoint, spatially connected and semantically relevant regions. In the medical domain, it denotes the delineation and isolation of different diagnostically or therapeutically relevant image regions [4]. That can be e.g. an organ, a functional part of an organ, a specific tissue or a pathological structure like a tumor. Image segmentation is often used as a preprocessing step for more abstract tasks in image analysis. These tasks can range from visualization, object recognition to image classification or compression. In medical imaging applications, adequate segmentations are needed to extract quantitative data in order to support the spectrum of biomedical investigations and clinical activities from diagnosis, to radiotherapy, to surgery [5].

In this chapter, a short overview about common segmentation techniques is given. After that, current work is presented with respect to specific tissues and the used MR modality.

2.1 Medical Image Segmentation

The segmentation of structures from medical images is a challenging task, due to the sheer size of the datasets and the complexity and variability of the anatomic shapes of interest. Even structures easily detected by a human observer are difficult to specify for a computer. A huge amount of approaches and variations have been proposed and implemented in recent years. In spite of the effort and the advances addressing this problem, there is still no method that yields acceptable results for every medical image. Therefore the bigger part of publications in the field of medical image segmentation deals with extending and combining existing approaches, optimizing parameters and adapting them to specific problems. The proposed methods for performing segmentations vary widely depending on various factors:

**Application:** What objects should be segmented (i.e. is the location, the size or the shape of these structures known) and is this a priori knowledge available for integration in the algorithm? How are the specifications for accuracy and error tolerance of the resulting segmentation? What demands on the runtime have to be met? E.g. there might be applications that require even real-time processing, so high level approaches with complex mathematical models are not feasible.

**Data:** First, the used imaging modality (e.g. computer tomography (CT), MRT, Ultrasound or X-ray radiography) and in the process utilized parameters are to be considered. Each modality has its own characteristics, like the distribution of intensities for different tissues, which can be essential for an algorithm. Second, the dimensionality and the resolution of the data are important for selection of an approach. Beside the fact that not every algorithm can be applied in each dimension, both dimension and resolution determine the size of the dataset and with it the runtime of the algorithm. A small resolution (respectively a small sample rate) can also cause partial volume effects. This effect occurs when a single discrete pixel or voxel at a border of different tissues represent a mixture of these. Thus, the intensity distinction between the tissue classes is blurred and complicates the
detection of boundaries. Last but not least the overall quality of the image data can have significant impact on the performance of segmentation algorithms. Here the signal–to–noise ratio (SNR), describing the power ratio between a signal (i.e. the meaningful information) and the background noise is critical, because various approaches differ largely in stability depending on the noise. Other aspects to consider are inhomogeneities inside the image and the presence of artifacts caused by motion (e.g. pulsation artifacts) or other susceptibilities of the underlying imaging modality.

**User:** To what extent can the segmentation be automated? Is there a user, who can be involved in the segmentation process? A user can bring in additional knowledge for initialization (e.g. specifying a region of interest or selecting seed points) and controlling a segmentation algorithm. Another point is the possibility of user interaction during a run of an algorithm, in terms of inspecting and correcting intermediate results.

**System:** What possibilities and capacities are given on the target hardware system? Is it possible to run an algorithm parallelized or respectively multithreaded? Is it possible to run the algorithm in an adequate time? Also the space complexity of an algorithm is still a criterion. Beside the fact that the input data itself can be very large, there are algorithms that produce a large computational overhead during runtime.

In general, a segmentation task consists of the course of actions depicted in Figure 2. Once an input dataset is loaded, there is often performed a preprocessing step to enhance the image, including noise reduction or correcting inhomogeneities by appropriate filters techniques. Next step is the selection of the algorithm and the determination of involved parameters. In some cases one need to make some additional pre–calculations in order to run the selected algorithm. That can be a resizing of an image or the calculation of gradient images, as well as estimations of initial parameters. After a successful run, the output of the algorithm can be postprocessed for a further optimization of the result. Some examples are: closing holes, smoothing boundaries, connected component analyses or reducing the complexity of a geometric representation. After all the result can be visualized or exported.

![Figure 2: General processing pipeline for image segmentation](image)

```plaintext
Figure 2: General processing pipeline for image segmentation
```
Because of the enormous spectrum of image segmentation approaches, there exist no complete classification scheme that captures all aspects. Some of these segmentation schemes are listed in [6]. On the basis of the underlying semantics that are used in segmentation approaches, the following classification can be made [7, 8]:

1. **Pixel–oriented approach** Here the criterion for homogeneity is only based on the intensity value of the current pixel. Therefore, these approaches are also characterized as local approaches. The assumption is that single structures in the image have different intensity distributions and correspond to different maximums in the histogram. Therefore threshold values (e.g. the local minimums in the histograms) are used to differentiate the objects. This approach is typically used for bimodal histograms i.e. separating an object from the background by applying a global, local or dynamic thresholding algorithm [7]. Results from pixel oriented approaches completely lack of global connectivities, but they are simple and in most cases also fast to compute.

2. **Region–oriented approach** Here the criterion for a pixel belonging to a certain region is defined by a similarity or distance measure. A distance measure represents the degree of similarity between two regions (a single pixel is also treated as a region). An overview about different distance measures is given in [9]. Typical algorithms in this class are region growing, region merging or splitting techniques. The basic approach of a region growing is to start from a seed region (typically one or more pixels) that is considered to lie within the object to be segmented. For all neighbor pixel of the region the distance is evaluated to determine if they should also be considered as part of the object. If so, they are added to the current region and the process continues iteratively until no more pixel can be added.

3. **Edge–oriented approach** The criterion for this class of algorithms is based on local gradients in the image. Typically, the gradient is computed from the intensities in the image, but can also be computed from other values (e.g. from a variance image, representing the local variance of intensities of each pixel). The basic idea of these approaches is that the gradient inside an object has low and at boundary regions large values. Edges detection and tracing algorithms belong to this class, as well as the large field of deformable model approaches.

4. **Model based approach** Additional knowledge about the geometric form of the object to be detected is integrated in the segmentation process. Algorithms of this class try to fit a shape into the image domain. The shape itself can vary from simple geometric approximations like cylinders to highly complex geometric representations, so-called shape prior models.

Beside the approaches described above, there also exist techniques based on stochastical or statistical models, namely classification, cluster and atlas–based techniques. Other algorithms adapt neuronal network techniques. Hybrid segmentation strategies are combinations of different algorithms to overcome the drawbacks of single approaches and are also widely used. Overviews about segmentation techniques can be found in [10, 11, 6].
2.2 Tissue Classes

2.2.1 Bones Segmentation

Figure 3: Bone segmentation with active contour: *left*) original MR image; *middle*) weighting function for active contour; *right*) intermediate and final result of the active contour segmentation, the initial contours are shown in black [12]

The usual used imaging modality for bone structures is X-ray radiography. If three dimensional data must be acquired, CT imaging is used. Bone structures have a high absorption rate for X-rays, so this produce high intensity values and a good contrast to the surrounding tissue. Although numerous MRI segmentation techniques are described in the literature, there is little research dedicated to the problem of segmenting bone in MRI.

A method for segmenting the human skull is presented in [13]. A hybrid algorithm combining thresholding and mathematical morphological operations in multiple steps is proposed. For segmentation of bone and cartilage of the knee joint, Fripp [14] presents a automated atlas-based framework that uses an underlying point distribution model. Other solutions utilize deformable models (active contours) and address the problem of the weak boundary between bone and adjacent tissue. In [12], texture information (here: the local variance of the intensity) is incorporated to the formulation of the external forces of the deformable model (Fig. reffig:segbones). Heric [15] enhances the edges by applying a wavelet transform at different scales and detecting points that lie on the boundary. In [16], the segmentation is performed on the single 2D slices of the volume with a fast marching algorithm. After that, the 3D segment is composed and smoothed with a min/max flow algorithm.

2.2.2 Soft Tissue Segmentation

Ideally suited for visualization and segmentation of skin and soft tissue are CT and $T_1$-weighted MR-sequences [18]. For a long time the segmentation of soft tissue had no medical application, and therefore was mainly ignored in research. In the last years, segmentations of soft tissue are more and more used to determine the ratio between hypodermic and intra abdominal fat. This ratio can give much more accurate information about the risk to come down with diseases type 2 or cardiovascular diseases, as the well-known Body Mass or Quetelet Index. Wittsack et al. [19] present a method that first performs an intensity correction on the MR image. After that a global threshold is determined by a histogram analysis and applied to the image to separate fat tissue from background. In [17] a hybrid segmentation approach is proposed. A connected partition
is extracted with a fuzzy connectedness algorithm that starts from a user selected seed point inside the soft tissue (Fig.4). Then, a homogeneity criterion is derived from this partition and used in a subsequent Voronoi diagram classification. An automated and unsupervised method is presented in [20] and realized with a fuzzy c-means algorithm that classifies the image in background, soft tissue and remaining tissues.

2.2.3 Vessel Extraction and Segmentation

Medical imaging techniques that are used to visualize the inner opening of blood filled structures, including arteries, veins and heart chambers are denoted as angiography. They are designed to maximize the intensity of these structures and suppress signals from all the other tissues. The modality used frequently for angiography is digital subtraction or bimodal X-ray angiography (DSA). Here two images are acquired: first a normal X-ray image, second an X-ray image after injection of a contrast agent and subtracted. The resulting image shows only the vascular tree. Computed tomography angiography (CTA) is the most accurate way to generate 3D vascular images, but also requires injection of contrast enhancers. Lately magnetic resonance angiography (MRA) has become a commonly used acquisition process. MRA enables to obtain images of flowing blood without radiation or contrast agent injection. Two different non-invasive MRA techniques have been developed: time of flight MRA (TOF) and phase-contrast MRA (PC). However, contrast enhanced (ce) MRA is widely used, because it produces the most accurate image quality. A survey about existing techniques is given in [21].

Although angiography images can be visualized directly (Fig.5), the segmentation of vessel structures are widely used. The segmentation of the vascular network is not only essential for planning and performing surgery procedures, but also for diagnosing and localizing vascular pathologies such as aneurysms or stenoses. Blood vessels are also often used
as landmarks for image registration. Because of this important role there exist a vast amount of publications addressing segmentation and extraction of vessel. The challenge for segmentation algorithms is the large complexity of the vascular tree and its variability in form and size. The increasing density and the decreasing diameter along the vascular tree push the partial volume effect and lead to inhomogeneities inside the structure.

An overview about filtering techniques to enhance angiography images is given in [21]. Basic thresholding techniques [22] are used for segmentation, as well as region growing techniques [23, 24, 25, 26]. Many segmentation techniques reduce the complexity of the 3D vessel shape to their 1D centerlines. These techniques vary widely in the way they detect the centerline and estimate the diameter of the vessel [27]. Model–based approaches try to fit a geometric or parametric model in order to find the vessel structure. In [28] a soft shape prior in form of a sphere in addition to image statistics is proposed to deform an active contour. In [29] elliptical cylinders are used as a model to detect tubular structures. Statistical approaches integrate anatomical knowledge in the segmentation process. In [25, 30] atlas–based approaches are presented to segment the cranial vessel tree. Feng and Horace [31] utilize a point distribution model to assist the segmentation. Much effort is also invested into optimizing geometric models for vessel segmentation. The main problem is the design of the external force pulling a contour to the vessel boundary. While iteratively propagating the front of the model along the vessel structure, the contour tends to swap over the boundary or to leak through weak points on the border. Deschamps [32] addresses this problem by freezing the contour when it reaches the boundary. The incorporation of weighted local variances in the external force term is proposed in [33]. In [34], the forces are modeled according to the physical forces of cohesion and adhesion occurring in thin tubular structures (capillary action). Yin and Ladak [35] present a semi-automated framework, that allows a user to approximate the vessel structure with geometric primitives (generalized cylinders and spheres) for a more accurate initialization of the geometric model.

Kirbas and Quek [36] give a comprehensive and comparative survey about existing vessel segmentation approaches. Felkel, Wegenkittl and Kanitsar [37] go into methods applied to CT angiography. Techniques for MR angiography are reviewed by Suri et al. [27].

2.2.4 Nerve Extraction and Visualization

For the visualization and extraction of nerve fibers, the Diffusion Tensor MR imaging technique is used. This technique allows the measurement of the molecular diffusion rate in tissues in vivo and therefore the molecular organization in tissues. Since 1999, it has been successfully used to analyze the spatial arrangement of fiber bundles in the central nervous system and since 2004 also of peripheral nerves [38]. In DTI, radiologists use specific radio-frequency and magnetic field-gradient pulses to track the movement of water molecules in tissues. Water molecules are dipoles and cannot easily pass biological barriers (e.g. the hydrophobic membrane of a cell). In general, axons are additionally enveloped in myelin sheaths (actually these are also just the cell membranes of the so-called Schwann-cells). This causes the molecules to diffuse along the length of axons. For each voxel of a volume, a diffusion tensor $D$ is determined:
Because of the symmetry of the matrix in (1), the diffusion tensor is completely determined by six values \( D_{ij} \) that represent the diffusion coefficient in direction \( \vec{ij} \). These values are measured by collecting diffusion-weighted MR images along these gradient directions. Once the diffusion tensor has been determined, the eigenvalues \( \lambda_i \) and eigenvectors \( \epsilon_i \), \((i = 1, 2, 3)\) corresponding to the ellipsoid model in Fig. 7 can be computed by diagonalizing \( D \):

\[
D = E^T \begin{pmatrix}
\lambda_1 & 0 & 0 \\
0 & \lambda_2 & 0 \\
0 & 0 & \lambda_3
\end{pmatrix} E, \text{ with } E = (\epsilon_1, \epsilon_2, \epsilon_3)
\]  

(2)

The eigenvector \( \epsilon_i \) for the largest eigenvalue \( \lambda_i \) is the main diffusion axis. The measurement can also be done for more directions (e.g. 12, 42) resulting in more accurate determination of the fiber orientation. The derived vector field can be visualized directly e.g. by color coding [39]. Other visualizations use techniques for flow field visualization (line integral convolution, streamlines/-tubes oder particle tracing) [40, 41].(Fig.6)

In [42, 43] the detection and visualization of peripheral nerves with MR imaging is investigated. Here high resolution MR tomography in combination with specific coils
to enhance the signal is used for the data acquisition. Because of the low number of this hardware in the clinical routine, these are still theoretical studies.
3 Material and Methods

3.1 Data acquisition

To collect suitable/feasible data for the project i.e. the segmentation, there were some restrictions for the imaging modality to consider. The imaging process should get along without additional stress and be non-invasive for a patient. That forbids the use of computed tomography (CT) imaging and the injection of contrast agents; the acquisition time itself should be as short as possible. Therefore, the contrast of MRI was improved by appropriately setting parameters and time constants involved in the relaxation processes of the tissue nuclei.

3.1.1 MRI background

Nuclear magnetic resonance

The physical base of MR imaging is the phenomenon of nuclear magnetic resonance (NMR) discovered in 1946 by Bloch and Purcell. Resonance is the absorption of energy from a source at a specific frequency (resonant frequency). In MRI the energy comes from a high radio frequency and the object resonating is the nuclei of atoms in a static magnetic field $B_L$. If energy is absorbed by a nucleus, it changes to a higher energy or excited state. The excited state is highly unstable, so the nucleus seeks to return to its original or ground state, while releasing the absorbed energy.

Spin and precession

Like an electron, a proton also possesses a spin around its axis. The spinning charge of the proton produces a magnetic field or a magnetic dipole moment. The magnetic moments of all protons and neutrons of a nucleus superposes to the so called net magnetic moment of the nucleus. Note that this model is incorrect, because subatomic particles have no spatial extension and the neutrons also have a magnetic moment, although they are uncharged. If the nucleus has the same number of protons and neutrons, the resulting moment equals zero. As described below, MRI requires atoms with a magnetic moment, this means the sum of neutrons and protons has to be odd. In MRI the nuclei are typically protons, i.e.
the nuclei of a hydrogen atom (\(^1\)H), but also \(^{13}\)C, \(^{14}\)N, \(^{17}\)O, \(^{19}\)F, \(^{23}\)Na, \(^{31}\)P and \(^{39}\)K. The external magnetic field \(B_L\) (pointing into \(z\)-direction) aligns the spin axes of the nuclei to the direction of the field. The orientation of this alignment can be parallel or anti-parallel, depending on the current state of the nucleus (ground or excited). In the equilibrium state of a system, the part of nuclei in the unstable excited state are negligibly small. While in the static field, the magnetic moments causes the spin axes themselves to begin a precession around the magnetic field direction. This can be best compared to a spinning gyroscope. The frequency of this precession is termed the Larmor frequency:

\[
\omega_L = \left(\frac{\gamma}{2\pi}\right)B_0
\]  

The Larmor frequency is linearly dependent on the strength of the external magnetic field \(B_L\) at the nucleus. The factor \(\gamma\) is a fundamental nuclear constant which has a specific value for every nucleus, called the magnetogyric ratio.

The assembling of the static field itself provides no measurable information about the nuclei (in terms of inducing a current, while changing the field gradient). Statistically the bulk magnetization \(M\) (i.e. the sum of the individual moments) has the same direction as the external field, because the vectors pointing into the actual direction of a moment are uniformly distributed around the cone-shaped shell defined by the precession. If a second high frequency (HF) field \(B_T\) is assembled orthogonally to \(B_L\) with a frequency equal to the Larmor frequency \(\omega_L\), the protons are forced to spin in phase and the individual nuclear magnetic moments reinforce each other to produce a strong bulk magnetic moment. The NMR phenomenon causes some of the nuclei to raise to a higher state, which come along with an anti-parallel orientation. The proportion of the excited nuclei depends on the amplitude and the pulse duration. A saturation is reached, if the quantity of nuclei in the ground state resp. in parallel orientation equals the quantity of nuclei in the excited state resp. in anti-parallel orientation. Then the longitudinal component of \(M = M_z\) is zero, because the longitudinal components of parallel and anti-parallel oriented individual magnetic moments cancel each other. In this state, the bulk magnetization is maximal and points into transversal direction \((M = M_{xy})\) of the field \(B_T\).

Relaxation process

If the transversal field \(B_T\) is switched off, you can distinguish between two relaxation processes. One relaxation process involves the return of the nuclei to the parallel orientation in the field, observable in the rapid drop of \(M_{xy}\). The time needed to emit the absorbed energy by the so-called spin–lattice and to restore the equilibrium is denoted as longitudinal or spin-lattice relaxation time constant \(T_1\). The second relaxation process has a much longer duration and refers to the loss of synchrony of precession, respectively the phase among the nuclei, observable in the restoration of the longitudinal component \(M_z\). The transverse or spin–lattice relaxation time constant \(T_2\) characterizes this process. More about the physical background and the image reconstruction can be found in [44, 45].

MR–Signal and parameters

The detected signal in MR imaging depends on the chemical environment, but is mainly correlated with the density of hydrogen atoms, respectively water molecules in a tissue. By adjusting the HF to a specific Larmor frequency \(\omega_L\), one can excite certain types
of nuclei and measure the concentration of these types of atoms in the tissue. The signal’s amplitude also depends on the impulse duration and HF amplitude, because both determine the proportion of nuclei that will raised to a higher state. The time to echo (TE) denotes the time between HF impulse resp. a series of HF impulses and the readout of the signal. Time to repeat (TR) is the time between two subsequent impulses. Another parameter is the so-called flip angle (FA). It determines the angle by which the bulk magnetization vector is rotated away from the direction of the static magnetic field during the application of an HF pulse. By finding a good trade-off between a small flip angle and adjusted TR, the acquisition time for sequence can be reduced. Some interesting dependencies can also be derived from the signal-to-noise ratio, given by the following term:

\[
\frac{S}{N} \sim B_L \times V \times \sqrt{\frac{t}{D^5}} \quad (4)
\]

\(B_L\) is the static field, \(D\) the diameter of MR tomograph or the distance between emitter and receiver, \(t\) the measurement period and \(V\) the size resp. the volume of the measured location. The volume \(V\) is determined by the image matrix (resolution in the transversal plane) the slice thickness in longitudinal direction. This means the higher the resolution of an MR image, the more noise is present. This can be compensated by using longer measurement times or using a stronger magnetic field. Using an additional coil, the image quality can be significantly increased by reducing the diameter \(D\) (see Fig.9). These (surface) coils are attached to the patient and serve as receiver. Note the inbuilt coils in the tomograph can transmit and receive signals.

![Figure 9: Impact of using additional coils, reducing the signal drop off: left) \(T_1\)-weighted sequence without coil; right) \(T_1\)-weighted sequence with coil](image)

**3.1.2 Sequences and results**

In clinical practice a large number of different MR imaging techniques (sequences) is used. The different sequences vary widely in setting of the used parameters. An detailed description of commonly used sequences and settings are given in [46, 47]. To optimize the base for the segmentation, several image series acquired with different sequences have been evaluated.
### Table 1: Sequences and used settings

<table>
<thead>
<tr>
<th>Series</th>
<th>Sequence (Protocol)</th>
<th>Region</th>
<th>Resolution [pixel]</th>
<th># Slices</th>
<th>Thickness [mm]</th>
<th>Space between slices [mm]</th>
<th>Spacing [mm]</th>
<th>TR [ms]</th>
<th>TE [ms]</th>
<th>FA [°]</th>
<th>AV</th>
<th>Coil</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SE (*tfl3d1)</td>
<td>knee</td>
<td>448 x 448</td>
<td>39</td>
<td>2.0</td>
<td>0.2</td>
<td>0.29 x 0.29</td>
<td>6960</td>
<td>82</td>
<td>150</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>SE (*tfl3d1)</td>
<td>knee</td>
<td>448 x 448</td>
<td>39</td>
<td>2.0</td>
<td>2.2</td>
<td>0.29 x 0.29</td>
<td>6960</td>
<td>82</td>
<td>150</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>SE (*tfl3d1)</td>
<td>spine</td>
<td>448 x 448</td>
<td>39</td>
<td>2.0</td>
<td>2.2</td>
<td>0.29 x 0.29</td>
<td>8000</td>
<td>77</td>
<td>120</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>GR/TOF (*fl2d1r.ts)</td>
<td>pelvis</td>
<td>384 x 208</td>
<td>128</td>
<td>3.0</td>
<td>-1.0</td>
<td>1.04 x 1.04</td>
<td>20</td>
<td>4.8</td>
<td>60</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>GR (*tfl2d1)</td>
<td>pelvis</td>
<td>384 x 208</td>
<td>128</td>
<td>3.0</td>
<td>3.0</td>
<td>1.04 x 1.04</td>
<td>30</td>
<td>3.6</td>
<td>40</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>6</td>
<td>GR (*tfl3d1)</td>
<td>pelvis</td>
<td>384 x 208</td>
<td>128</td>
<td>2.0</td>
<td>-2.0</td>
<td>1.04 x 1.04</td>
<td>2250</td>
<td>3.4</td>
<td>9</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>SE (*tse2d121)</td>
<td>pelvis</td>
<td>512 x 512</td>
<td>128</td>
<td>3.0</td>
<td>0.0</td>
<td>0.86 x 0.86</td>
<td>7.9</td>
<td>3.0</td>
<td>80</td>
<td>6</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>GR (B-FFE ax)</td>
<td>pelvis</td>
<td>512 x 512</td>
<td>128</td>
<td>6.0</td>
<td>-3.0</td>
<td>0.78 x 0.78</td>
<td>9.4</td>
<td>4.2</td>
<td>30</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>9</td>
<td>GR (F12 3D sa CLEAR)</td>
<td>pelvis</td>
<td>512 x 512</td>
<td>100</td>
<td>-3.0</td>
<td>1.04</td>
<td>0.88 x 0.88</td>
<td>4.8</td>
<td>2.4</td>
<td>80</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>10</td>
<td>GR (F12 ax CLEAR)</td>
<td>pelvis</td>
<td>512 x 512</td>
<td>90</td>
<td>3.0</td>
<td>1.04</td>
<td>0.88 x 0.88</td>
<td>218.1</td>
<td>4.6</td>
<td>80</td>
<td>3</td>
<td>yes</td>
</tr>
<tr>
<td>11</td>
<td>GR (REC T1W/FFE/SENSE)</td>
<td>pelvis</td>
<td>256 x 256</td>
<td>90</td>
<td>3.0</td>
<td>3.3</td>
<td>1.46 x 1.46</td>
<td>10000</td>
<td>100</td>
<td>90</td>
<td>2</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Table 2: Subjective comparative evaluation of image features for the segmentation of the tissues, through visible inspection of the sequences (*+) with respect to the vessel signal

<table>
<thead>
<tr>
<th>Series</th>
<th>Purpose</th>
<th>Vessel I/G</th>
<th>Bone I/G</th>
<th>Soft tissue I/G</th>
<th>Signal-to-noise ratio</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>nerve enhancement</td>
<td>–/-/+</td>
<td>–/+</td>
<td>–/-</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>nerve enhancement</td>
<td>–/-/+</td>
<td>–/+</td>
<td>–/-</td>
<td>++</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>nerve enhancement</td>
<td>–/-/+</td>
<td>–/+</td>
<td>–/-</td>
<td>++</td>
<td>signal drop</td>
</tr>
<tr>
<td>4</td>
<td>angiography</td>
<td>++/+</td>
<td>–/-/+</td>
<td>–/-</td>
<td>–/-</td>
<td>++ *</td>
</tr>
<tr>
<td>5</td>
<td>morphology incl. vessels</td>
<td>++/+</td>
<td>–/-/+</td>
<td>–/-</td>
<td>–/-</td>
<td>–/-</td>
</tr>
<tr>
<td>6</td>
<td>morphology</td>
<td>–/-/+</td>
<td>–/-/+</td>
<td>–/+</td>
<td>–</td>
<td>motion, overlay</td>
</tr>
<tr>
<td>7</td>
<td>morphology</td>
<td>–/-/+</td>
<td>–/-/+</td>
<td>–/+</td>
<td>–</td>
<td>overlay</td>
</tr>
<tr>
<td>8</td>
<td>morphology incl. vessels</td>
<td>++/+</td>
<td>–/-/+</td>
<td>–/+</td>
<td>++</td>
<td>motion</td>
</tr>
<tr>
<td>9</td>
<td>morphology incl. vessels</td>
<td>++/+</td>
<td>–/-/+</td>
<td>–/+</td>
<td>++</td>
<td>overlay</td>
</tr>
<tr>
<td>10</td>
<td>morphology incl. vessels</td>
<td>++/+</td>
<td>–/-/+</td>
<td>–/+</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>morphology</td>
<td>–/-/+</td>
<td>–/+</td>
<td>++/-</td>
<td>+</td>
<td>overlay</td>
</tr>
<tr>
<td>12</td>
<td>morphology</td>
<td>–/-/+</td>
<td>–/-</td>
<td>++/-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>
Table 1 shows exemplarily the large variation in parameter settings and resulting images. In mostly all sequences, a distance is let between single slices, but it is also possible to overlap slices (see sequences 4, 6, 9). In fast protocols, a too small space would cause corruptions of the signal in the current plane, because the relaxation process of the subsequent step is still not finished. Repetition time (TR), echo time (TE) and flip angle (FA) are described above, the average (AV) refers to the number of samples taken for the result. Using a sequence with averaging will increase the acquisition time, but also smooth the result. One can distinguish between two basic sequences: spin echo (SE) and gradient echo sequences (GR). SE with a short TE responds to the longitudinal relaxation process, therefore denoted as $T_1$–weighted sequences, while sequences with longer TE responds to the transversal de–phasing process, denoted as $T_2$–weighted, respectively. The GR echo sequences use a second HF pulse that is introduced shortly after the excitation pulse in anti–parallel direction ($180^\circ$). The GR echo then responds to the gradient of the read–out signal after the first and after the second HF pulse.

First evaluations have been done in cooperation with the Research Center Jülich. The sequences 1–7 are acquired with a Siemens MAGNETOM Trio, A Tim System 3T. All sequences uses the nucleus $^1$H and the according Lamour frequency of 123.26272 Hz for a magnetic field $B_L = 3\, T$. At the beginning several MRI sequences have been tested to enhance the nerve fibers itself for segmentation, but with less convincing results. Although they can be detected on several single slides, a tracking of them is difficult even for by an experienced clinician (Fig.10). Due to their small structure and the properties to share the same intensity band like the surrounding connective tissue, in which the nerves are embedded, another approach has to be found. The nerve cords were not considered during the segmentation process; instead they are additional modeled along some user–specified control points.

First results to achieve a good sequence for differentiating the tissues have been not very promising and were corrupted by noise and non–acceptable artifacts, due to the necessary trade–off in the choice of parameters and the unavailability of a body coil, (fig.11).

The sequences 8–12 are acquired at the Department of Diagnostic Radiology (RWTH Aachen) using a Achieva 1.5T MR tomograph from Philips Medical Systems. Additionally, a SENSE Torso coil is applied. The image nucleus is $^1$H, the resulting Lamour frequency for the magnetic field $B_L = 1.5\, T$ is 63.897566 Hz. The data shows a much better

---

**Figure 10:** Nerve in MR image; enlarged pictures taken from subsequent slices (in braces) in sequence 2: left) nerve noticeable in axial direction (18); middle) continues in transversal (19); right) and disappears (20)
signal-to-noise ratio and less artifacts, due to the used coil.

The goal to adapt a single sequence to gain sufficient information for a classification of the desired tissues has been discarded. After all, a combination of the following angiography and a morphology sequences has been most promising: a time of flight (TOF) MR-angiography without a contrast agent is used to extract the vascular system and a T1-weighted morphology sequence provides the information for the other tissues classes (Fig. 12). Both sequences are applied in succession and cover the same volume, so they are a-priori co-registered and no additional registration of both datasets is needed. The overall time for data acquisition is currently about 20 minutes.

Figure 12: chosen MRI sequences: left) MR angiography referring to settings of sequence 4; right) $T_1$-weighted sequence with parameter set to 11
3.2 File Formats

3.2.1 DICOM file format

The introduction of the first CT scanners in the early 1970’s has pushed the technological advance in digital medical imaging techniques and the integration of computer systems into the clinical routine, but also raised the problem of data incompatibility for interchange between the different hardware devices and software applications. To address this problem, the American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA) formed a joint committee in 1983 to develop a standard with the following goals [48] in mind:

- Promote communication of digital image information, regardless of device manufacturer.
- Facilitate the development and expansion of picture archiving and communication systems (PACS) that can also interface with other systems of hospital information
- Allow the creation of diagnostic information databases that can be interrogated by a wide variety of devices distributed geographically.

Earlier versions of the standard were referred to as ACR–NEMA standards (version 1.0 and 2.0). The current version is the Digital Imaging and Communications in Medicine (DICOM) Version 3.0 and was developed in collaboration with other standardization organizations such as CEN TC251 (Europe), JIRA (Japan) and reviews by other organizations including IEEE, HL7 and ANSI (USA). The members of the DICOM Standards Committee consist of almost all vendors of medical devices and some national societies. The DICOM standard is structured as a multi–part document consisting of 16 related, but independent parts. For a complete documentation see [48, 49]. It defines a comprehensive set of standards for handling, storing and transmitting image and associated data.

Part 10 of the standard is the Media Storage and File Format for Data Interchange. Here, a file format for images is defined. Image files which are compliant with Part 10 of the DICOM standard are generally referred to as DICOM format files. Like other formats, it contains a header and the image data itself. The header consists of standardized entries defined in the data dictionary (part 6), as well as empty fields for free usage. In the standard entries, the information to address the image data (e.g. pixel type, resolution, compression method etc.) is given. Furthermore, various information about patient, modality and so on can be stored. The image data can contain contain multiples frames, allowing storage of volumes or animations. The pixel elements can be simple data types (up to 64 bit) or complex types for color images with multiple channels and vector data.

Although DICOM has established itself as standard format, it has some drawbacks. The actual version 3.0 is nearly 15 years old and supports e.g. no 4D data, which is often applied in newer imaging techniques. In the research field, the DICOM format often seems to be too complicated in terms of handling, reading and writing files and series.
3.2.2 Object file format

While the DICOM format is utilized to exchange medical image data, the Object file format (OBJ) is used to represent the geometry of the model in the simulation. The object file format is an open format developed by Wavefront Technologies. In current version 3.0, the format supports both polygonal objects and free–form objects. Polygonal geometry uses points, lines and faces to define objects while free–form geometry uses curves and surfaces. The OBJ file format represents a 3D geometry by listing first all elements (vertices, texture coordinates, vertex normals) and then the faces using reference numbers of of specific elements. Elements are indexed by their occurrence and can be freely intermixed in the file, but typically they are grouped to their types. Object files are in the ASCII format and commonly structured as followed:

```
# this is a comment
# here begins the vertex list
v 0.111 0.222 0.333
v ...
# here begins the texture coordinate list
vt 0.112 0.223
vt ...
# here begins the normal vector list
vn 0.122 0.233 0.344
vn ...
# here begins the faces list
f 1//1 2//2 3//3
# if no texture is used for the face
f 1/1 2/2 3/3
# faces can be grouped by enclosing them using following tags:
# for a object/polygon group use
# materials groups are enclosed by
usemtl [material name]
```

Elements are listed starting with a specific tag, followed by their coordinates. Texture coordinates can have one parameter for 1D textures, two for 2D and three coordinates for 3D textures. The faces are constructed out of triples representing the vertex data. Each triplet is composed of reference numbers to a geometric vertex, texture vertex and vertex normal. A reference number i refers to the ith occurrence of the according element in the file. Additionally it is possible to group faces to objects or material groups sharing a common property. Material properties (e.g. colors) are defined in an optional external
material file (.mtl). Because of this simplicity, the OBJ file format is widely supported by 3D graphics application vendors and can be used to transfer geometric data between various applications. The full specification can be found at [50].

3.3 Software Tools

3.3.1 Insight Segmentation and Registration Toolkit (ITK)

The Insight Toolkit (ITK) is a platform–independent open source software toolkit for the registration and segmentation of (medical) image data [51]. ITK is completely implemented in the C++ programming language, but also supports automatic wrapping for interpreted programming languages such as Tcl, Java and Python. The term wrapping refers to the process of generating interfaces between the source and the target programming language. The Insight Software Consortium was founded in 1999 under a contract by the US National Library of Medicine of the National Institutes of Health including academic and industrial partners. ITK provides data representations in a general form for images (arbitrary dimension) and (unstructured) meshes and further processing algorithms, including I/O routines and various segmentation and registration algorithms. Visualization of data or graphical user interface support are not in the scope of the ITK. Some of the basic concepts [52] important for the implementation in this work are:

C++ generic programming ITK is implemented using generic programming principles. This means the actual implementation of a function or algorithm is abstracted from the data type it uses. ITK uses generic types or templates for the algorithm implementation and the class interfaces themselves. The big advantage of this design is that algorithms can be used with different data types (e.g. images with different basic pixel types and arbitrary dimensions) and that the design of a new algorithm is simplified. But it also has some disadvantages: first it makes the code difficult to understand, so a firm grounding in templated programming in general, and the Standard Template Library (STL) in particular is assumed. Furthermore the usage of generic types in the interface has one negative consequence: you have to explicitly specify the data types at compile time. E.g. there is no possibility to allocate an image object first, and then dynamically set its type. In ITK the image type (using STL container to store the pixel data) is templated, so you have to define and initialize it with a fixed type.

Memory management The memory model in ITK depends on so–called smart pointers that retains a reference count to objects it refers to. Smart pointers can be allocated on the stack like normal pointers, and when their scope is exited, smart pointers disappear and decrement their reference count to the object that they point to. If the reference count is zero, i.e. the object is not needed anymore, the object itself is deleted and the memory is deallocated. So there is no need to keep track of deleting objects before a reference to it is lost. This is comparable to the garbage collection in the programing language Java.

Data pipeline In ITK all process objects can be stitched together (via the SetInput() and GetOutput() methods) forming a so–called data pipeline. These process objects can be simply divided into:
sources, objects that produce data (e.g. reader)

• filter, objects that require input data and process it to produce new data

• mapper, objects that take data and produce an output to a file

Calling the Update() method on any object in the data pipeline will recursively call
the method of all preceding objects. If any process object changes its internal state,
due to changes of its data the GenerateData() method is called to update the output
and forward the changes through the data pipeline.

Error handling Critical portions of the ITK code are set under exception inspection using
the C++ try/catch statements.

Event Handling Event handling is realized by the Subject/Observer respectively Com-
mand/Observer design pattern. Here, an object (observer) registers at another
object (subject) to receive a specific event. If the subject invokes the event all
subscribed observers get a notification i.e. a specified function is called (callback
function) for each of them.

ITK also provide classes to utilize multi–threading independent of the underlying system.
It can be used to execute a single method (e.g. a filter) on multiple threads, or to specify a
method per thread. Furthermore, the streaming of large data is supported by subdividing
the image and processing only requested image regions. Detailed information can be found
at [51] and in the documentation [52].

3.3.2 Visualization Toolkit (VTK)

The Visualization Toolkit (VTK) is an open source software system for 3D computer
graphics, image processing and visualization [53]. VTK follows a strictly object–
oriented design (C++), but like ITK it supports wrapping to interpreted programming
languages. VTK offers a variety of data representations for a wide field of applications,
including unorganized point sets, polygonal data, images, volumes, and structured and
unstructured grids. For these types, VTK provides I/O routines (readers/importers and
writers/exporters) to exchange data with other applications or toolkits. Furthermore, the
toolkit offers a lot of algorithms to process and manipulate this data, even though the focus
is set on filters processing geometric data. Finally, the software is a visualization system
providing a high–level abstraction of the underlying OpenGL API (e.g. an interactive
renderer). A wide variety of visualization algorithms are supported, including scalar,
vector, tensor, texture, and volumetric methods. Moreover, it is easily possible to merge
2D imaging and 3D graphics/data in visualization.

Although VTK is a separate toolkit, the development of VTK was strongly connected to
the ITK. Therefore, some of the basic concepts are identical. The data pipeline concept
is also implemented in VTK, and it is possible to connect ITK and VTK pipelines. The
main difference is that VTK abstained from the generic programming concept and that it
has no built–in memory management, so the user must take care of memory leaks. VTK
provides no GUI support, but it offers components (render widgets) that can be integrated
in GUI libraries such as Qt, Fast Light Toolkit (FLTK) or the Microsoft Foundation Class
Library (MFC).
3.3.3 Medical Imaging Interaction Toolkit (MITK)

The Medical Imaging Interaction Toolkit (MITK) [54] is developed at the German Cancer Research Center (DKFZ) and extends and combines the Insight Toolkit (ITK) and the Visualization Toolkit (VTK). While ITK offers a comprehensive library of algorithms for image processing, segmentation and registration tasks, but no support for interaction and visualization of image data, VTK is designed to visualize a wide range of scientific data in many different ways. Last provides GUI-support in form of render widgets and low-level support for interaction and manipulating data such as panning, rotating, moving and scaling. For medical image processing and visualization, it is straightforward to use a combination of both toolkits. One big drawback of this solution is the effort to keep data and the view(s) on the data consistent. On the one hand, changes in the data to be visualized have to be propagated to the view(s) itself. On the other hand, if multiple different views on the same data are used, changes in one view may have impact to the other displays, so they also have to be updated. E.g. in case of the common used cross-sectional view on a volume dataset: changing the position in one of the three orthogonal planes will change the currently shown sliced image in the other views. MITK overcomes this gap by managing different data (input, intermediate and final processing results or geometric objects) in one so-called data tree, which defines the contents of the views comparable to a scene-graph and also can be used to represent logical dependencies (Fig. 13). The data tree nodes themselves contain

- the data (i.e. images, points or surfaces)
- a list of mappers,
- the geometry of the data object,
- and a list of properties.

Mappers are used to control the creation of rendering primitives that interface with the graphics library (e.g. OpenGL, VTK), which actually draws the data. The geometry of the data object is the bounding box, the origin and the currently used transformation matrix. The extensible property list contains at least the name of the node as well as basic entries that controls the visualization e.g. the visibility, color or opacity of the data in the viewer. Single nodes can be addressed by traversing the tree or a subtree with various tree iterators, such as preorder, inorder or child iterators provided by the MITK and ITK, respectively. Although it is designed to be independent from an underlying GUI library, MITK offers a collection of basic Qt GUI elements that deal with this data structure, such as single node selectors and data tree views. Currently MITK supports the Qt library up to version 3.3. Furthermore MITK offers extended interactions like defining points and polygons in the image domain and an undo/redo concept while interacting with data objects [55].

3.3.4 Qt

Qt is often simply referred to as GUI library, but it is a comprehensive toolkit for creating complex graphical applications, as well as the underlying infrastructure. This includes process handling, threading, networking and database access and much more. Furthermore, Qt offers additional development tools for GUI design or internationalization.
The Qt Designer

As mentioned above, Qt provides some helpful tools and one of them is the Qt designer. Qt designer facilitates the development of graphical user interfaces and can be best described as a construction kit for GUIs. It allows the creation of widgets and dialogs, placing of single basic GUI elements via drag and drop, defining layouts and setting properties and behavior of components via a comfortable user interface. Qt designer also provides a preview of constructed components.

The result of a Qt designer session is a user interface (.ui) file. It is a well-formed XML-file that holds all information about the constructed GUI class. Before it can be used, the user interface compiler (uic) had to be run that translates the XML-file into C++ code. Typically a class is inherited from this base class and implements the functionality of the user interface. This design pattern allows a separation of functionality and appearance of a GUI element. Anyway the Qt designer is well suited to create “static” user interfaces, but the dynamic changing (i.e. during runtime) of basic layouts or the exchange of GUI element is not supported. So it is inevitable to write a bulk of additional code to implement.

Figure 13: (left) data tree with different node types; (right) four consistent views of the 6 data objects in the tree (image, segmentation, isosurface and the 3 planes) requires the coordination of 19 scene–graph objects in four scene–graphs (5 objects in each 2D view and 4 objects in the 3D widget) of applications. Qt it is available for nearly every platform, including various Unix derivatives, Microsoft Windows or Apple MacOS. It is developed by Trolltech (Oslo, Norway), a commercial software developer. Nevertheless, even the actual version (4.3) it is freely available for non-commercial applications. Beside the preeminent documentation of the library, these are the reasons Qt is considered as standard for cross-platform development. The Qt GUI library provides a various amount of classes, reaching from a simple button to file opening dialogs and the well-known error message boxes. The design of the elements respectively whole applications can be dynamically adapted to a target system, denoted as the Look&Feel concept. In the following part some basic concept will be explained, because the implementation makes heavy use of it.
dynamic GUIs. The best way to do so is realizing static parts with the Qt designer (it really saves some time), and writing dynamic components in the subclass.

The Signal/Slot Concept

The signal-slot concept realizes event handling between Qt classes, respectively derived classes. In all classes derived from the minimal base class QObject (so all Qt classes) this concept can be used. The Qt classes provide a lot of predefined slots and signals for interacting, but new types can be defined. A class can emit a signal due to a user interaction, a system or internal state change. A slot (i.e. a function) connected to the signal is called, whenever the signal is emitted. It is also possible to call the slot function with a single parameter.

In the class that attempts to trigger an action, you just declare the signal in the interface (header), using the keyword signals. In the implementation part you can emit the signal whenever it is supposed to, by simply using the statement emit wakeUp(i).

```cpp
#include <QObject>

class SenderClass : public QObject
{
    Q_OBJECT

public:
    ...

signals:
    void wakeUp(int);
    ...
};
```

In a class that should react to a signal/event (i.e. execute some code), a function is declared using the keyword slots and implemented as usual.

```cpp
#include <QObject>

class ReceiverClass : public QObject
{
    Q_OBJECT

public:
    ...

public slots:
    void makeCoffee(int cups);
    ...
};
```

In both classes the macro Q_OBJECT has to appear at the top of their declaration. Note that slots and signal always have the return value void. Signal and slots can also be defined as private or protected, following the standard C++ rules for the scope. Now you can establish a connection between two instances of the classes, as follows:

```cpp
connect( m_sender, SIGNAL(wakeUp(int)), m_receiver, SLOT(makeCoffee(int)) );
```

A signal is emitted for every connection established, if a connection is simply duplicated, two signals will be emitted. An arbitrary number of connections can be established to a slot or a signal (see fig.14). Connections are valid as long as both objects exist or they are manually broken using QObject::disconnect().
Of course, the source code of the above classes will not compile, since it is not conform to the C++ standard. Before so, Qt’s meta–object compiler (moc) has to be run. It parses the class files removing the signals, slots, and emit keywords and generates C++ code that initializes the meta–object. The meta–object contains the names of all the signal and slot members, as well as pointers to these functions. It is comparable to the Subject/Observer design pattern used in ITK, but here also type–safeness of a parameter is guaranteed.

3.3.5 CMake

Cross platform make (CMake) is a tool to manage the compilation process in a platform–independent way. It was actually developed to control the build process in the ITK project. It also is an open source project implemented in C++, sources and binaries are freely available at [57]. Other important projects currently using CMake are VTK, MITK, and KDE 4. CMake can be used to control the software compilation process using simple platform– and compiler–independent configuration files that are placed in the source tree. CMake generates appropriate makefiles, project or workspace files for different development resp. compiler environments out of the information in the configuration files. So it is easy to switch to a different platform or compiler environment e.g. from KDevelop (Linux) to Visual Studio (Microsoft Windows). CMake features auto–detection of external libraries and packages and supports setting additional build parameters via a simple user interface. CMake can compile source code, create static and dynamic libraries, generate wrappers, and build executables in arbitrary combinations. You can also execute user commands and user defined macros in each build step. CMake features in–place as well as out–of–place builds, and can therefore support multiple builds from a single source tree.
3.4 Algorithms

As described in 2.1 the choice of an algorithm and the resulting performance depends on various factors. The criteria for the pre-selection of the algorithms currently implemented resp. integrated in framework are framed as followed:

**Application:** The RA simulation requires no highly accurate medical model, but an abstract plausible one. Target tissues of the segmentation are vessel, muscles, soft tissue and bones. The segmentation process itself is not time-critical.

**Data:** The used image modality is MR. The data consists of two co-registered volumes, namely a MR angiography and a $T_1$-weighted MR image.

**User:** The user of the algorithm is supposed to have anatomical knowledge, therefore he can provide at least desired seed-points in anatomical structures. A semi-automated approach is therefore recommended. However, the goal is to minimize user interaction.

**System:** The target system is supposed to have enough capacity, so space complexity is not a limiting factor.

The vascular tree is extracted from the 3D MRA data by applying a fuzzy-connectedness algorithm on user-provided seed points. For the segmentation of the soft tissue, an approach based on [20] is chosen. A fuzzy c-mean clustering algorithm with a predefined number of three classes to discriminate the $T_1$-weighted dataset into background, soft tissue and a class containing the remaining tissue. For the segmentation of the bones, a geodesic/active contour approach has been selected.

3.4.1 Fuzzy Connectedness

Medical images are inherently “fuzzy”, because the imaging process maps a composition of signals to single pixel values. A signal can be referred to different tissues types, noise, acquisition modality specific effects or partial volume effects. The fuzzy connectedness algorithm introduced by Udupa and Samarasekera in 1996 [58] takes this into account and defines global fuzziness. To each adjacent pixel pair a fuzzy affinity is assigned. A fuzzy connectedness relation or fuzzy affinity map is defined to represent the global connectedness in the image domain. The actual object segmentation is then achieved by defining a group of pixels that show a certain degree of global hanging togetherness.

A fuzzy affinity $k$ is any reflexive and symmetric relation in an $n$-dimensional image domain $D$:

\[
k = \{((c, d), \mu_k(c, d)) | (c, d) \in D\}
\]

\[
\mu_k : D \times D \rightarrow [0, 1]
\]

\[
\mu_k(c, c) = 1, \forall c \in D
\]

\[
\mu_k(c, d) = \mu_k(d, c), \forall (c, d) \in D
\]

Generally $\mu_k$ can be written as:

\[
\mu_k(c, d) = h(\mu_a(c, d), \mu_i(c, d), \mu_g(c, d), c, d) \forall (c, d) \in D,
\]

\[1\]
\( \mu_a(c, d) \) is the degree of coordinate space adjacency, \( \mu_i(c, d) \) represents the degree intensity space adjacency and \( \mu_g(c, d) \) the degree of gradient space adjacency between two pixels \( c \) and \( d \). The fuzzy \( k \)-connectedness \( K \) is a fuzzy relationship in \( D \), where each path between any pixel \( c, d \) has a strength which is the smallest affinity along the path. The fuzzy affinity \( \mu_k(c, d) \) between \( c \) and \( d \) is the strength of the strongest of all paths.

A fuzzy object \( O_\theta \) of strength \( \theta \) containing a point \( c \) is a set of pixels such that:

\[
\begin{align*}
    d \in O_\theta & \iff \mu_k(c, d) \geq \theta, \text{ and} \\
    d \notin O_\theta & \iff \mu_k(c, d) < \theta
\end{align*}
\]

In [58], the following definition for fuzzy affinity was proposed:

\[
\begin{align*}
    \mu_k(c, d) &= \mu_a(c, d)[w_i h_i(f(c), f(d)) + w_g h_g(f(c), f(d))], \\
    \mu_k(c, c) &= 1
\end{align*}
\]

\( f(c) \) and \( f(d) \) are the intensities of the pixels \( c \) and \( d \). The fuzzy affinity \( \mu_k(c, d) \) is thereby the weighted sum of an intensity feature \( h_i(f(c), f(d)) \) and a gradient feature \( h_g(f(c), f(d)) \), where weights \( w_i + w_i = 1 \). The adjacency function \( \mu_a(c, d) \) is assumed to be the hard adjacency relation, that means it is zero if \( c \) and \( d \) are not 1-neighbors:

\[
\mu_a = \begin{cases}
    1, & \text{if } \|c - d\|_2 \leq 1 \\
    0, & \text{otherwise},
\end{cases}
\]

where \( c_i \) are the pixel coordinates for \( 0 \leq i \leq n \). The functions \( h_i \) are \( h_g \) given as Gaussian functions of the mean \( \frac{1}{2}(f(c) + f(d)) \) and gradient \( |f(c) - f(d)| \):

\[
\begin{align*}
    h_i(f(c), f(d)) &= e^{-\frac{1}{2} \left( \frac{(f(c) + f(d) - m_i)}{s_i} \right)^2} \\
    h_g(f(c), f(d)) &= e^{-\frac{1}{2} \left( \frac{|f(c) - f(d)| - m_g}{s_g} \right)^2}
\end{align*}
\]

The parameter \( m_i \) is the mean intensity and \( s_i \) the standard deviation of the intensity in a sample region; \( m_g \) and \( s_g \) are the mean and standard deviation of the gradient of a sample region.

### 3.4.2 Fuzzy C–Means Clustering

The fuzzy c–means clustering (FCM) algorithm was introduced by Bezdek [59] in the 1970s. The FCM allows fuzzy segmentation based on the fuzzy set theory and generalizes the k–means clustering algorithm. The term clustering refers to the process of partitioning a data set into subsets (clusters). The fuzzy c–means algorithm is a robust unsupervised method. That is, it works without the use of a specialized set of data (training data) for determining parameters of an algorithm. Furthermore, it is relatively independent from the initialization of the cluster centers. Fuzzy clustering is a clustering method that minimizes an objective function, which is used to measure the compactness of the clusters with respect to the given features. The minimization is done by iteratively updating of the membership and cluster centers. In contrast to the k–means method that forces a
hard classification (i.e. pixels can be only member of one class), FCM evaluates a fuzzy membership function for each pixel. It determines for each pixel the degree of membership to each class, which denotes the closeness of a pixel to the cluster centers.

The fuzzy c–means objective function has the following format [59]:

$$J_m(U, v) = \sum_{k=1}^{N} \sum_{i=1}^{C} (u_{ik})^m d_{ik}^2$$  \hspace{1cm} (1)

and

$$d_{ik}^2 = \|y_k - v_i\|_A^2 = (y_k - v_i)^T A(y_k - v_i)$$ \hspace{1cm} (2)

where

- $U$ fuzzy c-partition of $y$
- $v = \{v_1, v_2, ..., v_C\}$ vector of cluster centers
- $y = \{y_1, y_2, ..., y_N\} \subset \mathbb{R}^n$ the data set
- $C$ the number of clusters $2 \leq C \leq N$
- $u$ membership or partition $(C \times N)$ matrix
- $m$ fuzziness coefficient, determining the fuzziness of the membership, $1 \leq m \leq \infty$, if set to 1 c–means equals k–means clustering
- $\|\cdot\|_A$ A–norm on $\mathbb{R}^n$, and $A$ is positiv–definite $(n \times n)$ weight matrix, if $A = I$ (identity matrix) then it refers to the Euclidean norm

The fuzzy c–means algorithm locally minimizes $J_m$ iteratively updating the cluster center with

$$v_i = \frac{\sum_{k=1}^{N} (u_{ik})^m y_k}{\sum_{k=1}^{N} (u_{ik})^m}, \text{ where } 1 \leq i \leq c$$ \hspace{1cm} (3)

and the membership or partition matrix with

$$u_{ik} = \frac{1}{\sum_{j=1}^{C} \left(\frac{d_{ik}}{d_{jk}}\right)^{\frac{1}{m-1}}}, \text{ where } 1 \leq k \leq N \text{ and } 1 \leq i \leq c$$ \hspace{1cm} (4)

To perform a hard segmentation, you can just assign pixels to the class with the highest membership.
3.4.3 Deformable Models

Deformable models are one of the most common approaches to medical image segmentation. Therefore it is well investigated and dozens of different implementations are available. The basic idea is to evolve a closed curve in a 2D image domain or a surface in 3D, respectively to approach the boundary of the object to be segmented. The curve or surface is also referred to as interface. The evolution of the interface depends on internal forces of the interface itself and external forces derived from the image information. The evolution can be described as energy–minimizing problem as well as dynamic force problem. The advantages of deformable models are the immanent closed contour, scalable accuracy and the possibility to include a prior modeled as additional forces. Disadvantages are the necessary need for an initialization and that the model have to reach an equilibrium of forces to stop.

Deformable models were first introduced in 1987 by Kass, Witkin and Terzopoulos as so–called snakes [60]. The curve in the image domain $D$ is represented in a parametric description

$$X(s, t) = (X(s, t), Y(s, t)), \quad (1)$$

where $s \in [0, 1]$ is arc length and $t \in \mathbb{R}^+$. The formulation as force equation is then given with [61]

$$-\gamma \frac{\partial X}{\partial t} = F_{\text{int}}(X) + F_{\text{ext}}(X) \quad (2)$$

where $\gamma$ is the damping factor, an arbitrary positive constant. The internal force $F_{\text{int}}$ is composed of two terms

$$F_{\text{int}}(X) = F_{\text{elast}}(X) + F_{\text{rigid}}(X) = \frac{\partial}{\partial s} \left( \alpha \frac{\partial X}{\partial s} \right) - \frac{\partial^2}{\partial s^2} \left( \beta \frac{\partial^2 X}{\partial s^2} \right) \quad (3)$$

where the constant coefficients $\alpha$ and $\beta$ can be used to control the strength of the contour’s elasticity and rigidity, respectively. The external or potential force $F_{\text{ext}}$ is typically given as the edge potential

$$F_{\text{ext}}(X) = -\nabla P(X) \text{ and } P(x, y) = -w |\nabla [G_{\sigma}(x, y) * I(x, y)]|^2 \quad (4)$$

with a positive weighting parameter $w$, $G_{\sigma}(x, y)$ is the 2D Gaussian function with standard variation $\sigma$, $\nabla$ is the gradient operator and $*$ is the convolution operator. $F_{\text{ext}}$ pulls the contour towards the desired object boundaries, while the internal force $F_{\text{int}}$ discourages stretching and bending of the curve. The description in the energy minimizing formulation and an overview about external forces is given in [61].

The problem of this snake model is the initialization. If the initial model is too far away from the boundary, the evolution will stick to local minimums. Therefore Cohen [62] proposed the balloon model that adds an additional pressure force to the classic snake model. The pressure force inflates or contracts the contour, so it can overcome local
minimums. Regardless of that, using parametric models is difficult due to the fact that topological changes (i.e. splitting and merging of curves) have to be explicitly handled.

In 1993 geometric deformable models were introduced [63, 64] which overcome the problem of the topological changes and can handle arbitrarily complex shapes. They base on the curve evolution theory and the level set method. In the curve evolution theory the deformation of an arbitrary parametric curve given in Eq. (1) is only expressed in geometric measures such as the unit normal and spatial derivatives of the curve:

\[ \frac{\partial X}{\partial t} = V(\kappa)N, \] (5)

where \( V(\kappa) \) is called the speed function, depending on the curvature \( \kappa \) and \( N \) is the inward unit normal. The movement of the curve in tangential direction is omitted, because it only affects the curve’s internal parametrization (i.e. the positions of control points) but does not change the form of the curve itself. The evolving curve \( X \) is implicitly embedded as the zero level set of an higher dimensional hypersurface \( \Phi \):

\[ \Phi[X(s,t),t] = 0 \] (6)

The level set function \( \Phi \) is defined in the image domain \( \Omega \) and constructed as follows:

\[
\Phi(x, y, t) = \begin{cases} 
0 & \text{if } (x, y) \in X, \\
< 0 & \text{if } (x, y) \in X_{in}, \\
> 0 & \text{if } (x, y) \in X_{out}
\end{cases} \] (7)

\( X_{in} \) denotes the area inside the curve and \( X_{out} \) the area outside (see Fig. 15). Now instead of the evolution of the interface, the hypersurface itself is deformed. Differentiating Eq. (6) with respect to \( t \) using the chain rule leads to

\[ \frac{\partial \Phi}{\partial t} + \nabla \Phi \frac{\partial X}{\partial t} = 0, \] (8)

where \( \nabla \Phi \) is the gradient of \( \Phi \). Using Eq. (5) and the inward unit normal given by \( N = -\nabla \Phi / |\nabla \Phi| \), the above evolution equation can be simply written as

\[ \frac{\partial \Phi}{\partial t} = V(\kappa) |\nabla \Phi| \] (9)

where the curvature \( \kappa \) at the zero level set is given by

\[ \kappa = \nabla(-N) = \frac{\nabla \Phi}{|\nabla \Phi|} = \frac{\Phi_{xx} \Phi_y^2 - 2\Phi_x \Phi_y \Phi_{xy} + \Phi_{yy} \Phi_x^2}{(\Phi_x^2 + \Phi_y^2)^{3/2}} \] (10)

Like the modeling of the forces in the parametric model, the critical point is the design of the speed function. It has to get smaller with decreasing distance to edge indicators and zero if the boundary has been reached. Some typical formulations are described in
A generic formulation of the speed term that can be used to implement almost any parametric deformable model is given with

\[
\frac{\partial \Phi}{\partial t} = V(\kappa) |\nabla \Phi| = V_{\text{curv}} \kappa |\nabla \Phi| + V_{\text{press}} |\nabla \Phi| - V_{\text{ext}} \nabla \Phi
\]  

(11)

where \( V_{\text{curv}} \), \( V_{\text{press}} \) and \( V_{\text{ext}} \) are functions defined in the image domain. \( V_{\text{curv}} \) is called the spatial modifier term (typically set to a constant) and corresponds to the internal force in the active contour model. \( V_{\text{press}} \) is the propagation or expansion term and corresponds to the pressure force in the balloon model. \( V_{\text{ext}} \) is the advection term that snaps the surface to the boundaries, corresponding to the external force in the parametric model.

**Figure 15:** left) Concept of zero set in a level set; right) Evolution of the level set surface
4 Implementation

The aim of this thesis is to provide a compact and intuitive framework for creating segmentations that fit the requirements of the RASim project. Further the framework is demonstrated by segmenting the acquired datasets and creating a first reference model. The requirements for the segmentation are outlined in the following points:

(1) the resulting segmentation should be differentiating. That is, all desired tissue classes should be extracted out of a provided dataset. The following tissues are defined important for the simulation:

1. **bones**, which are needed for the skeleton model (animation) of the virtual patient and also as anatomical landmarks for the detection and location of the puncture site;

2. **musculature**; which is recommended for the simulation of contractions, while stimulating with an electric impulse. In some body regions a clinician also has to penetrate the musculature to address the specific nerve;

3. **blood vessels**, which are also needed to detect the puncture site (here a clinician feel for the pulsation inside a vessel). There will be made no distinction between the arterial and the venous system;

4. **soft tissue** including skin, subcutaneous fat and connective tissue is needed on the one hand simply for the representation of the skin of the model, and on the other hand to determine the resistance, while actually performing the puncture. Furthermore, the nerve and vessels are typically embedded in a surrounding protective coat of soft tissue.

As a consequence of 3.1.2 the nerve fibers are not considered in the segmentation process. The classification of the voxels in the image not only determines the visual model, but also the functionality and resistance at specific locations. The proposed medical imaging modality is MR imaging and the data consists of two images acquired with different parameters (see 3.1.2). One dataset is a MR (TOF) angiography that will be used to segment the vessel structure, the other is a $T_1$–weighted MR image representing the morphology of the other tissues. Both datasets will be acquired one after another in a single acquisition session without relocating the patient. So both images cover the same volume or field of view (FOV) and are thereby a priori co-registered.

(2) The resulting segmentation should be at least provide an abstract model. Although this is preferable, the RASim project does not need a highly accurate medical model that is needed in critical medical applications e.g. for computer assisted diagnose or computer guided surgery planning. Instead the segmentation should emphasize the substantial structures.
The segmentation process should be automated as far as possible. Because manual or even semi-automated segmentation can be a very time consuming task, the user interaction should be minimized during the segmentation process. Furthermore, the segmentation approach should be adaptable to individual patients (i.e. datasets of the same body region) as well as adaptable to different body parts depending on the regional anesthetic technique.

The mentioned requirements yield in the task to find adequate methods and algorithms, adapt them to the problem and provide them for usage in the segmentation process. As the result of the discussion in 2.1, it can be assumed that there is no single algorithm that discriminates all tissues and thereby fulfills the goals. Therefore, a hybrid and combined approach is proposed for creating a segmentation of the desired tissues in one segmentation session. The consequence of using multiple algorithms is to find a suitable order of the algorithms to achieve the desired segmentation. Further the problem arises, how to control and visualize the attaching of the single algorithms in an intuitive manner. Therefore a framework has to be developed that integrates an interactive viewer on the loaded and processed data and the functionality to build up various segmentation schemes. Last refers to build up a processing sequence that is adapted to the current task and the underlying dataset.

The basic idea of this thesis is to create initial segmentations (so-called segmentation scripts) for a specific task running different algorithms step-by-step in a suitable order. These scripts can be exported and later loaded again by a user, when there is need for a segmentation of new datasets representing the same body region. The following procedural method to create the initial segmentation script is introduced (Fig. 16):

1. user starts a new segmentation session
2. loading of the data set(s) in the RASimSeg application
3. running single algorithm steps by
   a) selecting an algorithm
   b) setting the input and the required parameters
   c) executing the algorithm
   d) inspecting the result with the possibilities to
      • correct the result by setting more appropriated parameters and rerun the step,
      • discard the result of the segmentation step or
      • proceed with a following segmentation step
4. if the result of the overall segmentation is satisfying, the user can
   a) combine the results of single segmentation steps (so-called labels) to a single result dataset corresponding to a partitioning in the tissues classes
   b) can export the result in various output formats
   c) can store the segmentation script, describing the sequence and the input parameters of the steps
Thereby single algorithms are typically applied directed to segment a specific sort of tissue. The order of addressing the tissues can be ignored, unless the user want to utilize a particular result as input for a subsequent segmentation step. A stored segmentation script that has been approved for a specific task, can be imported and applied to another corresponding dataset. Before running a loaded script the user has to provide those parameters that could not be exported like user–specified seed points. He further has the possibility to change and adjust parameters of each step to optimize the segmentation result for the current dataset.

The developed framework offers a comfortable graphical user interface that intuitively maps the workflow depicted in the section above.

4.1 Basic Concepts

The design and implementation of applications is a big challenge — not only from the technical point of view. Many applications are well–designed and offer comprehensive functionalities, but also lack easy accessibility, usability and representation (in particular applications that come from the academic and scientific domain). The far end is specialized software applications that are developed and adapted for a specific task or a target user group. They mostly provide a good usability in their context, but in exchange lack of flexibility and extensibility. It is even harder to find a adequate compromise from the developers point of view. Someone involved in the design and developing process knows about the operating modes of the application and the background of the used algorithms. But when the application is really deployed, an unexperienced user has to be assumed.

In [65] a criterion scheme for the implementation and integration in particular for image processing applications in the medical routine is introduced. It can be used as guideline for the software developing and evaluation process. In following section, the basic concepts of the implementation are presented with respect to the following criteria:
• flexibility, adaptivity and stability of the application
• control by the user
• data, functioning, presentation and context integration
• validation

**Flexibility** can be referred to as the ability to adapt the software to changing basic conditions. A certain degree of flexibility is obtained by the design decision to implement the application platform–independently. Although it is currently developed on a Linux system, due to the usage of platform–independent toolkits and the CMake environment the framework has been already successfully ported to Windows. Further the implementation and the GUI classes are separated as far as possible to support an adaption to a new GUI library (e.g. a new version of the Qt library), if required in the future.

**Adaptivity** identifies in how far an algorithm can be adapted to specific tasks or to concrete image materials. The framework is geared to be adaptive for different tasks (i.e. performing segmentations for different body regions in the RASim project). This is achieved by using combined algorithms that can be arbitrarily arranged to perform a segmentation. Also it is possible to extend the application by additional algorithms with the implemented plugin support. This is, new algorithms can be implemented as independent dynamic libraries and are loaded into the application, during the runtime. The currently realized algorithms are pre–selected under the condition to fit the task, the image data and modality respectively.

**Stability** means the reproducibility of the result by an algorithm. Stability can not be obtained insofar a user is currently still involved in the segmentation process and therefore the results also depends on his input. Since no stochastic algorithms are involved, the concept of the segmentation scripts can be used to reduce parameter variance to a minimum and maximize the stability of the approach.

**Control by the user** is afforded by the integrated viewer. The viewer allows the visualization of all data in a consistent way. Different data can be visualized at the same time. So it is possible to show the original image overlaid by a translucent image containing a segmentation result, as well as the mixed representations of images, surfaces and points. The viewer consists of three 2D widgets – one widget respectively one plane for each body axes and a 3D widget showing a cross-sectional view on the volume. The cross-sections are defined by the planes of the 2D widget and their actual position in the volume. In each 2D widget the positions of the other planes are visualized by lines that allows an intuitive navigation through the volume. Further panning and zooming is possible in all views and the 3D widget additionally allows to rotate the view around the shown data. In each step of processing the results can be visualized and inspected by the user, give him the possibility to correct the result by adapting parameters or even completely discard it.

**Presentation integrity** is given by various factors. The application offers a intuitive interface that lean against the common standard interface design used in medical imaging. The Qt library supports the dynamic adaption of interface styles through the Look&Feel concept. That is, changing the appearance (design, form and color scheme) of GUI elements according to that used on the current system. Further all GUI elements used for performing the basic tasks are located in a control area on the left side of the
application, separated in semantic units. Additional interfaces are only provided if it is more comfortable (e.g. file opening/saving dialogs and special input masks) or absolutely necessary (e.g. various warnings).

**Context integrity** is reached through various concepts. To avoid repeating steps the introduced scripts are used. A once created processing flow that has shown to fulfill a desired task can be saved and reused later for an application in the same context. On the the GUI level, the single parts of the work flow (viewing, segmenting, labeling) as well as inner elements in the layout form clearly distinguishable separated units. Single GUI elements are utilized to restrict the user’s freedom of action insofar that he can only perform valid actions. Buttons and other elements are enabled or disabled according to the current state. Much effort has been made to avoid that a user has to switch between the current tab and the viewer controls again and again. When a user selects a data item in another part then in the viewer controls e.g. in the data input selection fields (comboboxes) of an algorithms or in the labeling GUI, it can be assumed that the user also wants to see what he has selected. All required parameter for performing the segmentation task, are exclusively set up in the GUI. That includes interactively specifying seed points in the viewer area (if needed in the current context) or setting values in the designated fields. Here various validators are used to restrict the values to a context sensitive range or type, respectively. All these little optimizations seem naturally, but implementing it cost much time to realize and make up the bigger part in the code of an application supporting a GUI. Furthermore, parameters have to be checked in respect to the underlying image before actually running an algorithm. For a simple example, you can not perform an isosurface extraction algorithm (marching cubes) on a 2D image. There are more complex situations and due to the template concept of the ITK images these parameter checks are not easy to perform and often requires to implement decision trees.

**Data integrity** is achieved by simply holding no additional representations of the input data. The application is designed to be the interface between the medical image data and the virtual patient data base of the RASim project. The input refers to an MRT volume dataset in the DICOM file format. The interface to the RASim database is a geometric representation of the segmented tissues in the obj file format, respectively a colored image dataset also in the DICOM format.

**Function integrity** means the integration of the application and its functionality within a system. The RASimSeg application is designed to be a stand–alone application.

**Validation** of the application can not be performed because it requires an appropriate database and further exceeds the time frame of this thesis.

### 4.2 Framework

For the implementation of the framework, the Medical Imaging Interaction Toolkit (MITK) has been chosen, because the concept for visualization and managing of the data is well–suited for the framework that has to be developed. MITK comes along with a basic application that can be used to integrate so–called functionalities that are loaded in this application. The class of the application has been taken out and serves as base of the RASimSeg main application. The concept of the functionalities has been maintained
to keep the application flexible in terms of adding additional functionalities in the future. The application logic is depicted in Fig. 17. The framework utilizes the built-in input support of the basic class to perform the loading of the DICOM data set(s) into the application. The communication between different parts (i.e. the functionalities) of the application is exclusively done over the common data tree. That is, data is changed or added to the tree and the change is announced by invoking signals on the node or the data tree itself. The actual functionality of the RASimSeg application is implemented in BasicFunc. It consists of three parts: the viewer controls, the processing controls and the labeling controls. Basically all of them can be regarded as self-contained functionalities.

The MITK consists of a GUI-independent part MITK and the Qt related QMITK. QMITK classes are used and adapted to allow changing the settings of the viewer widgets and for the selection of input nodes in the GUI's of the algorithms. The MITK classes provide the general logic, including the data tree, tree nodes, tree iterators and so on. Beside the used elements of QMITK, the GUI is realized with the Qt library. Furthermore the general event handling is done using the Signal/Slot concept of Qt. VTK is only used for handling the export of surfaces data to the object format as part of the interface to the RASim project. ITK classes are utilized for the algorithms as far as possible and performing the export in various image formats. Figure 18 shows the relations between the toolkits and the application.
4.2.1 Basic Data Structure

As depicted in Fig. 17 the basic data structure of the application is a tree structure that holds all data loaded in and generated by single modules. The MITK data tree concept is used and adapted to represent the work flow in a consistent way and to traverse the tree in a more convenient manner. The following hierarchy is introduced (Fig. 19).

Figure 19: Introduced data tree hierarchy

The data tree consists of empty hierarchy nodes and nodes holding different kinds of data. QmitkMainApplication was modified to put all data loaded by a user in the according subtree **Loaded Data** and name them with a consecutive number and a time stamp. The subtree **Final Labels** contains nodes representing the currently defined final labels, each consisting of a unique name, an isovalue and a color. The combined results of the overall segmentation (i.e. the current session) is put in the **Result** branch. This subtree will be hidden until a result is actually created. The hidden subtree **widgetPlanes** holds the geometry of the three planes (sagital, axial and frontal plane) used in the viewer (see 4.5.1). Now for each processing step, a subtree will be created and all produced outputs...
of the current algorithm will be located here.

For realizing the creation and loading of segmentation scripts, the dummy node concept is introduced. Dummy nodes are empty nodes that contain no data, but determine the type of data they will contain. Each algorithm creates a dummy node for each output according to its default parameters, when initialized and add them to the data tree. If the user changes settings that affects the kind and number of outputs, it will be dynamically updated. This allows to specify the input of subsequent algorithm steps, although no data has actually been created. If the algorithm is executed and has produced the according data, it is assigned to the already existing dummy nodes. Although a user will normally create a script step–by–step while executing the current algorithm and inspecting the result, this will fail when loading scripts. Dummy nodes are determined by adding a boolean property to the node property list. Because it is still unknown what data will be hold by a dummy node, the type has also to be specified. The RASimSeg application internally differentiates between five types of data tree nodes:

1. **Hierarchy nodes** in the data tree
2. **Image nodes** holding various types of images
3. **Label nodes** representing binary images
4. **Point set nodes** containing one or more points
5. **Surface nodes** representing geometric data

Furthermore, some other entries are added to the data nodes to realizes the aimed functionalities. All additional entries in the property list used in the application are:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Type</th>
<th>Description</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>rasimseg_dummy</td>
<td>bool</td>
<td>node is a dummy or not</td>
<td>true</td>
</tr>
<tr>
<td>rasimseg_nodetype</td>
<td>integer</td>
<td>set to value according to above enumeration</td>
<td>-1</td>
</tr>
<tr>
<td>rasimseg_step</td>
<td>integer</td>
<td>step number, in which the node is created</td>
<td>-1</td>
</tr>
<tr>
<td>rasimseg_createdby</td>
<td>string</td>
<td>unique identifier of the algorithm</td>
<td>MainMenu</td>
</tr>
<tr>
<td>rasimseg_assigned</td>
<td>integer</td>
<td>only used by label nodes, determines the assignment to a final label</td>
<td>-1</td>
</tr>
</tbody>
</table>

To set the right values you can use simply use the provided helper functions placed in `BasicHelp.h`.

**Listing 2:** Setting properties for a label node

```cpp
static void setLabelNodeProperties(mitk::DataTreeNode::Pointer &node, std::string name, std::string uid, int step)
{
    if (step > 0) {
        std::stringstream temp;
        temp << "(Step " << step << ") " << name;
        node->SetProperty("name", new mitk::StringProperty(temp.str()));
    } else node->SetProperty("rasimseg_dummy", new mitk::BoolProperty(true));
    node->SetProperty("rasimseg_nodetype", new mitk::IntProperty(2));
    node->SetProperty("rasimseg_step", new mitk::IntProperty(step));
}
```
As you can see first the standard MITK properties are set, after that the RASimSeg specific properties. Note in line 19 the color for visualization of this node is set according to the next color of a nice global rainbow color scheme. For all other node types similar functions are available. If the output is successfully created, the dummy node flag is simply set to false.

4.3 Scripting and Processing

4.3.1 Creating Scripts

If a user chooses a new processing step the GUI of the algorithm is shown and the interface will set all inputs to the last image node that meets the input type by default. That is, if an algorithm e.g. asks for a label image (binary image) the input will be assigned to the last node that matches the label type, evaluated on the setting of the node property rasimseg.type. If no appropriate node can be found the input is not assigned. A user can only proceed, either with setting up the next step or executing the current step, when all inputs are specified. Typically just one input is needed. If the check fails, the user has to load an according dataset. So at the moment of the creation of a segmentation script the validity is guaranteed at each step. Now it could be possible to go back to a previous step and change the input to a node that will be created later in the processing pipeline. To prevent a user from constructing such invalid scripts, some filter functions/classes that works on the data tree were implemented and can be used in the QMITK GUI elements. They can be simply initialized with the current step number, when creating the algorithm GUI and will only filter these nodes that match a specific node type and a smaller step number (property rasimseg.step). So in each step it can be ensured that a user can only choose those inputs, that already have been created in the processing pipeline.

4.3.2 Exporting Segmentation Scripts

Scripts are stored in the XML format. Implementing native writing and reading of XML files is an exhausting effort. Therefore MITK’s capability to store and load complete data trees is utilized. The class mitk::PropertyList (member of each data node) can write its own elements and their values as XML stream. Furthermore, the class can be used to store arbitrary basic value types:

```cpp
mitk::PropertyList::Pointer settings = mitk::PropertyList::New();
settings->SetProperty("string prop", new mitk::StringProperty("test") );
settings->SetProperty("int prop", new mitk::IntProperty(3) );
settings->SetProperty("bool prop", new mitk::BoolProperty(true) );
```
When a user selects to store a script, an export dialog will show up. The script can be exported in whole or in part. This allows more flexibility through dividing and exporting a processing script in single parts e.g. a part that perform some preprocessing and a part that actually does the segmentation. It is also possible to store the final label descriptions and their assigned labels (see 4.4) within the script. Of course the last makes only sense for those labels that are actually created in the exported steps. Further the user can provide a name, a description and an additional comment for the script. A description is recommended, so that another user knows what additional parameters he has to set in order to run the script in the indented way.

When exporting the script a simple XML tree is constructed with the MITK/VTK XML writer. At first the script info is written (also via a property list), then each step is requested for a property list that contains its current settings (Fig. 20). To encode the data inputs (i.e. data nodes in the tree), a function is provided that takes a node as parameter and returns a string in the following format:

```
[nodeType; nodeOffset = algorithmStep - inputNodeStep; nodePosition].
```

That is, the input node will be simply encoded with its type (rasimseg.type), an offset computed from the current step and the step the node was created in (rasimseg.step) and finally a number that specifies its position in its subtree according to the node type. This encoding of inputs is sufficient for all integrated algorithms, but since each algorithm itself take care of writing its settings into the property list and reading them out of it, appropriated input encodings and decodings can be used. So it is easily possible to encode things like take all outputs from a certain algorithm or all outputs of a certain step as input. After all the assigned labels are encoded in a similar way and the final label descriptions are exported by simply writing out the existing subtree in the data tree.

### 4.3.3 Importing Segmentation Scripts

The import of a script is done by invoking the `mitk::DataTree::Load( treeIterator, filename )` method, where `treeIterator` is an iterator on an empty data tree and `filename` is the selected XML file. The method constructs the dual data tree out of the XML tree. The resulting data tree consists of nodes, only containing a property list and can be parsed using common tree access methods. The first node holds the script info that is read out and presented to the user in the import dialog. For each node representing a processing step an instance of the algorithm is created (determined by the unique identifiers) and initialized with the property list. As mentioned above the algorithm reads out its parameters from this list and decodes the input nodes. Note that the correctness of the decoded inputs is only guaranteed for output–input relations within the saved script. This is due to the fact that it can not be ensured that the user has loaded the datasets in the same order or that the number of processing steps applied before loading the script does not differ from the recommended one used in the script creation. This is, for the example given in Fig. 20 skipping the smoothing step or...
Figure 20: Example script and corresponding exported XML script tree

running an additional preprocessing step, both causing the encoded offsets to the script inputs to be incorrect. The simplest way to overcome this problem is just letting the user specify all input nodes, that refers to steps outside the script. Nevertheless it is attempted to automatically set up all inputs. The implemented decoding function called with a string [nodeType; nodeOffset; nodePosition], will return a node according the following scheme:

1. find a node that match conditions nodeType, nodeOffset and nodePosition,
2. else return first node that match conditions nodeType and nodePosition,
3. else return first node that match conditions nodeType
4. else return empty node

For the most common cases this will be sufficient to successful set up the correct input nodes, but even so the user has to check the critical steps.

4.4 Labeling and Data Export

Label nodes are either intermediate results of the segmentation process or represent a tissue or a part of a tissue, respectively. The values of single voxels in the binary image determine their membership to the tissue. The user or a loaded script defines a mapping of the available labels to the predefined final labels i.e. the desired tissues see Fig. 21. Thereby only those label nodes have to be assigned that really represent a tissue. Note that only the order of the final label descriptions itself is important. The resulting class for a voxel always depends on the last assigned label that is non-zero on this position. So the final label descriptions should represent a hierarchical order or non-overlapping
tissues. The combined label respectively the combined color image are initialized with the values specified by the last final label that does not have any assigned nodes (typically the background label).

![Available Label Nodes and Final Label Definitions](image)

**Figure 21:** Label assignment, defining an 1:n–mapping of created binary images and the resulting tissue classes

Before the result dataset is created, some tests are forced on the assigned labels. Clearly all of them must be non–dummy nodes and actually contain image data. Second the dimensions respectively the resolution in each dimension of all images must be equal. If the conditions are satisfied, the labels are combined forming one dataset with the segmented tissues represented by the sum of those voxels sharing the predefined isovalue or color.

**Listing 3:** Pseudocode for combining the labels

```plaintext
for each voxel
    for last_assigned_label to first_assigned_label
        if voxel position = 1
            -> set according isovalue in combined image
            break this loop
    end for
end for
```

After successfully creating the result images, they can be exported in various formats specified by the user. RASimSeg supports the following formats:

- **png** colored image series
- **dcm** DICOM series containing the colored image
- **raw** write a RAW volume file (uses the isovalues)
- **obj** surface file for each tissue

The first three cases are implemented utilizing the image writer classes of the ITK. Exporting obj files is a little bit more complicated: a `mitk::LabeledImageToSurfaceFilter` is applied to the result dataset to created surfaces for each isovalue (except for background). The `vtkOBJExporter` asks for a `vtkRenderWindow`, so a short rendering pipeline had to be constructed. Finally an XML file is written that contains all label descriptions and basic image informations, including dimension, resolution and spacing (note that the png file format can not store pixel spacing informations). If the raw output was created,
additional informations to access the raw data is provided too. This is, the pixel type and the endianness (the byte order).

4.5 Graphical User Interface

4.5.1 RASimSeg Graphical User Interface

The graphical user interfaces of the RASimSeg application is based on the QmitkMainTemplate of the MITK and is only slightly changed (e.g. the menu structure). The layout follows the common standard design (Fig. 22) that is widely used in medical applications. It is composed of two areas – a view and a control area. The view area contains the already mentioned 2D cross-sectional views and the 3D widget. The controls are placed on the left side by default. If preferred, a user can also switch it to the left side via the options of the application (there was no workaround to avoid a restart of the application). The QmitkMainTemplate provides some useful features, including a memory indicator on the right bottom corner, that shows the current status of main memory, a toolbar that holds some GUI elements to control the behavior and the layout of the viewer widgets and a statusbar showing information about the pixel under the mouse cursor. That includes the position in the bounding volume, the pixel coordinates in the image space and its value. Further an additional windowing level controller is added to the view area depending on the current shown data. It can be used to define and adjust an interval in the extended gray scale range of medical images that will be mapped to the 256 levels of gray, which can be typically displayed by common indicators.

![Figure 22: Layout of the RASimSeg application](image-url)
Figure 23: RASimSeg application

The actual functionality of the application is located in the control area. As depicted in Figure 17 it can be considered as three separated functionalities, so each has its own interface placed in a shared tabulator widget.
4.5.2 Viewer Controls

The main component of this GUI is a `QmitkDataTreeListView` widget. This QMITK GUI element allows a representation of the data tree. It is placed in a Qt `scrollarea` to avoid ugly stretching of the tab widget itself, while the data tree grows and `QmitkDataTreeListView` exceeds the current size. Besides providing an overview of the existing respectively created data nodes to the user, it also allows to change some properties of the data nodes affecting their visualization. That includes toggling the visibility and changing the overlay color and the opacity. For a better usability, especially for larger data trees, some filter functions are provided that can be used to show only specific types of data nodes in the list view. Three further sliders for a more accurate way to position the planes are embedded, although it can be done interactively in the orthogonal views.

![Viewer Controls Diagram]

**Figure 24:** Viewer controls
4.5.3 Processing Controls

The GUI of the processing controls splits up into three parts: the upper step controller, a middle area showing either the algorithm selection widget or the GUI of an algorithm and the lower button area, that contains buttons for all available actions. The step controller can be used to browse the processing pipeline. It always shows the step number and the name of the algorithm. If the user wants to add a new step or append a script he simply step to the end of the processing pipeline (see Fig. 25). In the middle part the algorithm selection widget will show up, that allows to select an algorithm from a Qt `qlistview` holding all loaded plugins. Further a short description of the marked algorithm is provided below. The lower button area allows to perform the following actions:

- **Save Script** opens the export script dialog and is only enabled if at least one algorithm has been set up.
- **Load Script** shows the import script dialog.
- **Run All** will cause the execution of all processing steps which have not been run or have been changed.
- **Select** initializes the currently marked algorithm and shows its GUI.

If the processing tab shows an algorithm there are only two available actions in the button area. A **Cancel** button can be used to discard the step and its results (deletes the subtree in the shared data tree). It is only enabled for the last step in the processing script, because deleting arbitrary steps in a script is currently not supported. The **Run** button executes the active algorithm and respectively depending preceding steps.

![Processing Controls Diagram](image)

**Figure 25:** Processing controls
4.5.4 Labeling Controls

Fig. 26 shows the GUI that is used to perform the final labeling act. It contains two Qt qListView, the above shows all available unassigned label images, the bottom one displays the defined final labels and their currently assigned nodes. The user can select one or multiple label nodes and simply assign it to a final label, as well as removing again an assigned label. At the bottom there are buttons for the following actions:

- **New** creates a new final label description. An input dialog is presented, where the user can select an isovalue and a color and further enter a name, a description and a comment for the label.

- **Edit** allows the user to change the values of a label (see New).

- **Delete** removes a final label from the list.

- **Load** imports a final label list from an extern XML file.

- **Export** opens the export dialog and performs the actual combination of the result dataset and its exporting in the specified output formats (see 4.4).

![Figure 26: Labeling controls](image-url)
4.6 Plugins

In the following section, the most important algorithms that have been integrated and implemented for usage in the segmentation process are introduced. The GUI for each algorithm is presented and the available parameters described. Further some issues concerning the implementation are addressed.

4.6.1 Thresholding

Figure 27: left) Thresholding GUI; right) example of thresholding an image with an upper and a lower threshold

The thresholding algorithm can be used to threshold arbitrary input images in three different modes. An upper threshold can be used to select pixel with lower values, a lower threshold to select pixel with higher values and both can be used to select those pixel that have values between the given thresholds. The threshold values itself can be provided as absolute values or relative to the highest pixel value in the image. For the actual processing the `itk::BinaryThresholdImageFilter` is utilized.
4.6.2 Simple Region Growing

Simple region growing creates connected regions from initial seed points specified by the user. The algorithm allows to select an existing point set or to create a new one. A `mitk::PointSetInteractor` is established on the current selected node, so the user can change, add or delete points directly in the viewer widgets. Note that MITK allows only one active `mitk::PointSetInteractor`, so it must be ensured they are disconnected/reconnected each time the user change the point set or switch to another processing step. The upper and lower thresholds that are used in the region growing (i.e. only adjacent pixels that lie within the range are added to the region(s)) are determined from the values of the seed points. The range can be adjusted by adding offsets to both thresholds (min/max offset). The resulting values have to be checked against the numeric traits of the pixel type in the input image, otherwise if they are exceeded the algorithm produces unintended results. Internal an `itk::ConnectedThresholdImageFilter` is set up and produces the output.

**Figure 28:** left) Simple region growing GUI; right) region growing performed from a user-defined seed point
4.6.3 Mathematical Morphological Operators

Mathematic morphological operators are typically used as post-processing step in order to smooth a binary image, to close holes or to eliminate outliers. The operators are applied pixel wise and only depend on a so-called structure element. The structure element defines an arbitrary mask for the neighborhood of a pixel. Generally a symmetric convex structuring element (i.e. a discretized ball) is used. There are two basic operations: the erosion deletes pixels whose neighborhood does not match the mask and the dilation adds those pixels defined by the mask, when the pixel itself is non-zero. It is also possible to use mathematic morphological operators on grayscale images, causing the erosion to extend regions with small values and the dilation to extent regions with higher values.

Both operations are realized as plugins and can be used on binary label images as well as on arbitrary grayscale images. For setting up the structuring element an `itk::BinaryBallStructuringElement` with the given radius is constructed and passed to a `itk::BinaryErodeImageFilter` or `itk::GrayscaleErodeImageFilter`, `itk::BinaryDilateImageFilter` or `itk::GrayscaleDilateImageFilter`, respectively.

The choice of the right filter (binary or grayscale) depends on the data node provided as input. The number of iterations determines how many times the filter is applied. With both basic operators, the well-known closing and opening operations can be realized by simply running first a dilation followed by an erosion step or an erosion followed by a dilation step.
4.6.4 Fuzzy Connectedness

The fuzzy connectedness algorithm is described in section 3.4.1. Just like in the region growing plugin the user can specify the initial seed–point directly in the viewer widgets. Again, a `mitk::PointSetInteractor` is connected to the current selected node, but in this case it is limited to a single point (i.e. the user can specify at most one point). If an already existing pointset is selected and contains more then one point the algorithm will take the first one. The algorithm is realized using the `itk::SimpleFuzzyConnectednessScalarImageFilter`. The ITK implementation uses a native version of the fuzzy affinity equation that only involves the intensity features of pixels (Eq. 4). The initialization expects parameters for the mean intensity \( m_i \) and the standard deviation of the intensity \( s_i \) in a sample region of the object (Eq. 6). Both parameters are estimated by an `itk::ConfidenceConnectednessImageFilter`. This filter performs a region growing from the same initial seed–point using the following inclusion criterion [52]:

\[
I(X) \in [m - f \sigma, m + f \sigma],
\]

where \( I \) is the image, \( X \) is the position of the particular neighbor pixel considered for inclusion in the region. The variables \( m \) and \( s \) are the mean and standard deviation of the region intensities and the factor \( f \) is a constant set to 2.5. The filter is iterated \( n = 5 \) times and in each iteration the mean and standard deviation is recomputed from the actual grown region. The output of the filter is a rough segmentation of the object of interest.

Figure 30: left) Fuzzy connectedness GUI; right) outputs of the fuzzy connectedness algorithm
anatomical structure and an estimation of the mean and variance of gray values in this structure. The values are multiplied with user defined factors and then passed to the `itk::SimpleFuzzyConnectednessScalarImageFilter` in order to compute the global affinity map. The affinity map can be directly thresholded with a given value (i.e. all pixels with a higher affinity belong to the object) and/or saved in the data tree.

### 4.6.5 Fuzzy C-Means Clustering

![Fuzzy c–means clustering GUI](image)

The fuzzy c-means algorithm is described in section 3.4.2. The implementation of the algorithm additionally accepts a mask image that allows to perform a clustering restricted to masked voxels (i.e. those voxels which are non–zero in the mask image). A mask can be used either to define an arbitrary region of interest or in the intended way to allow an hierarchical sub–clustering. Last can be simply done by setting a label image representing a class as input mask in a subsequent fuzzy c-mean clustering step. The implemented algorithm operates directly on ITK images and makes exclusive use of fast `itk::ImageRegionIterator` to access pixel positions and values. The following listing shows the intensely simplified pseudocode for the algorithm:

% Listing 4.6.5

The fuzzy c-means is algorithm is described in section 3.4.2. The implementation of the algorithm additionally accepts a mask image that allows to perform a clustering restricted to masked voxels (i.e. those voxels which are non–zero in the mask image). A mask can be used either to define an arbitrary region of interest or in the intended way to allow an hierarchical sub–clustering. Last can be simply done by setting a label image representing a class as input mask in a subsequent fuzzy c-mean clustering step. The implemented algorithm operates directly on ITK images and makes exclusive use of fast `itk::ImageRegionIterator` to access pixel positions and values. The following listing shows the intensely simplified pseudocode for the algorithm:
Listing 4: Pseudocode for the fuzzy c-means algorithm

```plaintext
initRandom(membershipMatrix)
do // big loop
copy(membershipMatrix, membershipMatrix_last_iter)
updateClusterCenter(clusterCenter, membershipMatrix)
for each voxel
  if (mask(voxel) == 0) continue loop
  else updateMembership(clusterCenter)
  ++iteration
end for
error = difference(membershipMatrix, membershipMatrix_last_iter)
while (error > maxError) or (iteration > maxIteration)
```

The `membershipMatrix` stores the membership function for each class at every voxel location. The values of the membership functions can vary from 0 to 1, with the constraint that at any voxel location the sum of the membership functions of all the classes must be the unity. The `membershipMatrix` is represented by a vector of ITK images with basic pixel type `<double>` (one image per class) and initialized with random values between zero and one. In each iteration of the main loop the cluster center are updated (line 5) according to the equation (3). Then the membership values for each voxel are recomputed and normalized with respect to the new cluster center (Eq. 4). Note, that there is need for a case differentiation to avoid a division by zero. This will occur if a value of a voxel is equal to a cluster center, which will lead to a membership value of one for this particular class and zero for the other classes. The function `difference(membershipMatrix, membershipMatrix_last_iter)` in line 11 computes the Euclidean distance between the membership matrix of the last iteration and the one updated in the current iteration and is a decreasing function that converge to zero. The main loop is iterated until a specified maximum number of iterations is reached or the difference drops below an user provided maximum allowed error.

The algorithm produces two types of output images depending on the users choice. The hard segmentation will create a binary image for each class, thereby each voxel is assigned to the class belonging to the highest value of the membership function. Performing a hard segmentation equals the output of a k-means clustering algorithm and therefore undermines the concept of fuzziness segmentation. It is often recommended to store the result of the soft segmentation (i.e. the membership images) itself for further processing. By applying a subsequent thresholding on the membership images, only those voxels can be selected that show a certain degree of membership to this class. A threshold closer to one will lower the uncertainty, while a threshold closer to zero will raise the uncertainty of voxels belonging to the class (Fig. 32).

Note, that the order of the classes itself is not defined and can vary from run to run, due to the random initialization of membership images. However, for addressing a particular output in further steps of a segmentation script, a fixed order is essential. So the class labels and membership images are sorted with respect to the values of their cluster centers from the lowest to the highest one, before storing them in the data tree.
4.6.6 Geometric Models

The concept of geometric models and the level set theory are described in section 3.4.3. The method is realized using the level set framework of the ITK. The common implementation of ITK level set filter make use of the generic equation (see Eq. 11)

\[
\frac{\partial \Phi}{\partial t} = \alpha V_{\text{curv}} \kappa |\nabla \Phi| + \beta V_{\text{press}} |\nabla \Phi| - \gamma V_{\text{ext}} \nabla \Phi,
\]

where \( V_{\text{curv}} \) is the spatial modifier term for the curvature \( \kappa \), \( V_{\text{press}} \) the expansion term and \( V_{\text{ext}} \) the advection term. The parameters \( \alpha \), \( \beta \) and \( \gamma \) are scalar constants that relatively weight the influence of the single terms while the interface evolves. If the an user wants to refine an already good approximation of an anatomical structure, \( \beta \) can be set to zero causing the surface to snap to the nearby boundary. On the other hand, if a segmentation should be run from initial seed points the influence of the expansion term has to be adjusted to higher values. An appropriate choice of this weights is a process of trial & error. The plugin requires two images as input: the first one defines the initial model \( \Phi(X, t = 0) \), the second image is the feature image that encodes the speed function.

The user can select an existing point set or a label image for as initial image. The initial models/images are constructed as follows:

- **Point Set** The point set is passed to an `itk::FastMarchingImageFilter`. The fast marching filter is then executed with a constant speed and stops when it reaches a user-defined minimal distance. The output is a valid level set function, where the interior of the slightly grown region around each seed point has negative values, and the exterior has positive values.

- **Label Image** For the construction of a valid level set function, where the isosurface of the region(s) in the binary image represents the zero level set an `itk::DanielssonDistanceMapImageFilter` is utilized. The output of this filter is an image with the values of the signed distance to the isosurface (Fig. 34).

Two approaches for the level set segmentation have been realized, which differ in the underlying feature image they use. The feature image provided to the geodesic active contours segmentation bases on the magnitude of the Gaussian gradient image. The Canny–edge level set segmentation uses a distance map to edges detected by an Canny–edge filter.
Figure 33: (left) Geometric models GUI; (right) segmentation results using different weights for the propagation term, pushing the front of the contour more or less in the convex region.

Geodesic Active Contours

This plugin uses the `itk::GeodesicActiveContourLevelSetImageFilter`. The feature image is obtained from the first input image by applying an `itk::GradientMagnitudeRecursiveGaussianImageFilter` with an user provided standard deviation $\sigma$ of the Gaussian to produce the gradient image. The output is then passed to a sigmoid filter. The sigmoid filter performs a linear transformation, that maps the intensity range to a target range given by an interval $[\min, \max]$ and thereby raises a specific intensity range defined by a width $\alpha$ and its center $\beta$ according the following equation

$$f(z) = (\max - \min) \ast \left(1 + e^{-\frac{z - \beta}{\alpha}}\right)^{-1} + \min$$  \hspace{1cm} (2)

The target range is set to $[0, 1]$ and the parameter $\alpha$ and $\beta$ are user provided. The parameter $\alpha$ should be negative to perform an inverse mapping, so the resulting feature image has lower values at high gradient magnitudes and higher values in homogeneous regions (Fig. 35).

The initial level set model for the filter is constructed from a user selected label image or a point set as described above. Both images are passed to the `itk::GeodesicActiveContourLevelSetImageFilter`. The user can adjust the weights of the advection, the curvature and the expansion term and further specify a maximum number of iteration and a maximum allowed error as stopping criteria. The segmentation is simply obtained by thresholding the output of the level set.
**Figure 34:** Initial level set model: *left*) binary label image; *right*) scaled distance image: brighter values corresponds to positive distances, darker ones to negative distances. The contour representing the zero level set is shown in red.

**Figure 35:** *left*) original input image; *middle*) Gaussian gradient image; *right*) Sigmoid filtered segmentation filter that corresponds to the level set function $\Phi(X, t = i)$, where $i$ is the final iteration step. This is, all voxel that have a value equal or smaller than zero.

**Canny–Edge Level Set**

This plugin uses the `itk::CannySegmentationLevelSetImageFilter`. It uses speed term that minimizes the distance to the Canny edges in an image. The feature image is computed from the first input image by applying an `itk::DanielssonDistanceMapImageFilter` to the image obtained from an `itk::CannyEdgeDetectionImageFilter` (see Fig. 36). The user has to specify the canny variance and a upper threshold for the canny edge detection.

The target range is set to $[0, 1]$ and the parameter $\alpha$ and $\beta$ are user provided. The parameter $\alpha$ should be negative to perform an inverse mapping, so the resulting feature image has lower values at high gradient magnitudes and higher values in homogeneous regions.

The level set filter itself uses the same parameters as the `itk::GeodesicActiveContourLevelSetImageFilter` above and the segmentation is created in the same way.
4.6.7 Other

- **Gaussian Smoothing** The plugin utilizes the `itk::DiscreteGaussianImageFilter` to compute the convolution of the input image with a Gaussian kernel, given a user provided variance and a maximum size of the Gaussian kernel.

- **Anisotropic Gaussian Smoothing** This plugin uses an `itk::GradientAnisotropicDiffusionImageFilter` to perform an edge-preserving smoothing an an input image. The parameters are the step size (common values are 0.125 for 2D and 0.0625 for 3D images), the conductance that controls the strength of the diffusion and the number of iterations.

- **Rescaling** This plugin is intended to get a visual impression of algorithm outputs that can be visualized in a appropriate manner, e.g. membership image or distance images. That is, images that have pixel types like `float` or `double`. The plugin uses an `itk::RescaleIntensityImageFilter` to map the intensity range of the input image to an user provided interval (standard is [0, 255]).

- **Create Isosurface** This plugin can be used to create a surface representation of voxels sharing a common isovalue in arbitrary images. Internal a `mitk::ManualSegmentationToSurfaceFilter` is utilized that performs a marching cube algorithm. Beside the isovalue (for binary images just select a value unequal zero), the user can choose to perform a Gaussian smoothing and to perform a reduction of the resulting polygon mesh (the value provided is given in percentage of the complete mesh).

- **Create Point Set** This simple plugin can be used to create arbitrary point set for usage as input in other plugins (e.g. level set segmentation).

- **Convert Point Set** This plugin converts an point set to a binary image. The user has to select an existing (label) image to specify the geometry of the resulting image.

---

**Figure 36:** *(left) original input image; middle) Canny–edge image; right) distance image*
5 Results

After all the application is used to achieve a differentiating segmentation of the tissues described in section 4. The goal is to develop a segmentation script that minimizes on the one hand the knowledge and experience needed to reproduce the result and on the other hand the user interactions themselves. Last refers to the quantity of parameters and inputs a user has to supply interactively in order to successful run the script. Therefore, the segmentation process should be preferably initialized automatically and the parameters should be chosen as generally as possible. The pre-selection of the individual segmentation algorithms has already taken these aspects into account. The challenge of this task is the width variety of anatomical structures and their characteristics according to the used MR modality. The proposed segmentation script has been exclusively developed and tested on the currently available datasets, consisting of the MR angiography (TOF) and the $T_1$–weighted morphology selected in chapter [?]. The datasets do not have the desired quality and also are unregistered. The angiography dataset consists of 128 slices with a resolution of 384 x 209 pixel, the morphology has 70 slices with a resolution of 480 x 480.

![Sample slices taken from the MR angiography image](image1)

**Figure 37**: Sample slices taken from the MR angiography image: a) slice 1; b) 29, c) 68 and d) 125

5.1 Vessel

For the segmentation of the vessel tree(s) in the MR angiography dataset, two algorithm are proposed which produce comparable results with appropriate parameter settings. The region growing and the fuzzy connectedness algorithm. Both produce connected regions, but require at least one user–provided seed point for each region or vessel tree. Before applying any of these algorithms, the input image is preprocessed using the gradient anisotropic Gaussian smoothing filter.
Results

Figure 38: Sample slices taken from the MR morphology image: a) slice 1; b) 26; c) 46 and d) 70

Region Growing  The advantage of this approach is its simplicity and the execution speed. It allows the user to adjust the parameters which determine the intensity range used for the inclusion decision of the region growing nearly in real–time. Here only the critical lower threshold respectively the user defined added offset has to be considered, because the upper threshold can be automatically set to the maximum value by adding a large offset. The disadvantage of the approach is its dependence on the selection of the seed points (see Fig. 39). Their minimum intensity value determine the lower threshold. A selection of seed points in darker regions or even the selection of an outlier inside the vessel structures will lower the threshold and could lead to additional inclusions of voxels which not belong to the vessel. On the other side, the selection of seed points in brighter regions will raise the lower threshold and can lead to omitting parts of the vessel tree. The right image in Fig. 40 shows a segmentation where the seed points are selected in slice 1 that causes the segmentation of the left vessel tree to stop in slice 26 that shows a weak signal (see image a) and b) in Fig. 37).

Figure 39: Results from region growing using same parameters but different seed points: left) intended result; middle) seed points with lower intensity values; right) seed points with higher intensity values

Fuzzy Connectedness  This algorithm is more stable to the selection of the initial seed points, but it is much more time consuming. Furthermore the algorithm has to be executed
for each region to be segmented. This is, for each seed point the global affinity map is computed for the whole image domain and then thresholded to produce the segmentation result (Fig. 40). To process the dataset the algorithm runs for approx. 14 min on a Intel Dual Core 1.80Hz machine with 2 GB RAM. Note, the algorithm do not run multithreaded, so actual only one cpu core is used.

Figure 40: Results using fuzzy connectedness: left) initial image with seed point; middle) sagital snapshot of the rescaled fuzzy affinity map; right) resulting surface of the left vessel tree

The results of both algorithm can be additional smoothed using a mathematical morphological closing operation. The drawback of such a smoothing step is that spatial disconnections of nearby vessel structures can get lost. Therefore a post-processing on the final polygon mesh is recommended.

5.2 Soft Tissue

The segmentation of soft tissue is achieved using the fuzzy c–means algorithm on the $T_1$–weighted MR image. The algorithm is automatically initialized with a fixed number of three classes and requires no further user interaction. The algorithm runs about 15 min to drop below an error of 0.001. The labels are directly produced by performing a hard segmentation on the membership images. The output is ordered with respect to the cluster center and represent the following classes/labels: background, other tissue and soft tissue (Fig. 41). The resulting label for the soft tissue will not only cover the subcutaneous but also abdominal fat and connective tissue.

Figure 41: left) three classes: black = background label; red = remaining tissue; white = soft tissue; right) original images and overlaid by the soft tissue label
Again the result can be additional smoothed using mathematical morphological closing and/or opening operations. In Figure 42 the resulting isosurface of the smoothed segmentation is visualized. The right image shows that the segmentation produces a “hole” in the right rear section. This is due to a regional inhomogeneity in the MR dataset that can be seen in Fig. 38 b).

![Figure 42: Isosurface of the soft tissue segmentation: left) front; right) back with “hole”](image)

### 5.3 Muscles

The segmentation of muscle tissue is implicitly done when clustering the MR morphology image for differentiating the soft tissue. The resulting second class can be simply denoted as muscle (Fig. 41). Note, that both soft tissue label and muscle label will contain the other structures like blood vessels and parts of the bones, but because of the order in the final label assignment these voxels will be correctly assigned to their labels.

At this state the final result dataset will contain body regions that are labeled as background (e.g. the bladder will be classified as background, due to its low intensity). If a disjunct segmentation in the tissues is wanted and the muscle class is defined to collect all voxels that do not belong to another defined tissue class, a region growing algorithm can be applied to fill the internal region of the soft tissue label. This step does not need an additional initialization, because the user already has specified seed points for the vessel extraction and they surely lie within the desired region. For the current dataset this will fail, because of the unclosed boundary (Fig. 42).

### 5.4 Bones

The segmentation of bone structures in MR images is a difficult task, especially in the region of the pelvis with the complex femoral joint. One major problem is, that the definition of “bone” tissue is quite ambiguous. There are cortical bone, trabecular bone and marrow cavities. Each structure has its own characteristics (e.g. intensity distribution), which make it difficult to define some common segmentation criteria. Further the gradient to the surrounding tissue can widely vary because the cartilage interface and the epiphysis where the calcification is lower have a significant different
intensity gradient on boundaries compared to other regions of the bone.

Figure 43 shows the variety of form and composition as well as the different intensity distributions in single regions. The mean and variance of the intensity in a (3D) sample region of the pelvis (left image) are about 1182 and 1253 respectively. In a sample region in the middle part of the femur (right image) the measurement gives values about 618 and 512 respectively.

Figure 43: left, middle) parts of the pelvis; right) middle part of the femur with broad cortical (black)

The sequence shown in Figure 44 visualizes the complexity of the femur–joint.

Figure 44: Femur–joint (from superior to inferior)

Both intensity and gradient features are used to perform a suitable segmentation. The aspired aim to archive a rough prior model for a level set segmentation with a hierarchical fuzzy c–means clustering approach has been not promising. Thereby the initial region ought to be differentiated by subsequently clustering the second output class of the clustering algorithm that has been already performed for the soft tissue segmentation. But due to the width range of intensity values in the bone tissue, it is unequally distributed over all three classes.

Therefore the initial model is provided by a region growing approach or user–defined seed points, showing the problems of the level set segmentation itself. It is a time–consuming process to find appropriate values for the level set parameters as well as for the creation of the feature images. Figure 45 shows that the segmentation works well for some bone regions.

The major problem is to move the contour in inhomogeneous regions towards a desired boundary. If the weight of the propagation term is too small, the contour can easily
Figure 45: Level set segmentation starting from a seed point: *left*) original image with seed point; *middle*) computed feature image; *right*) result of the level set segmentation.

become stuck on image features close to their initialization. If the influence is too large, the contour can pass these inhomogeneous regions but can also easily be pushed through weak boundaries. The middle image in Fig. 46 shows the problem. Here a value has been chosen to successfully segment the bone, but the contour has already leaked through the weak boundary at the superior femoral cartilage interface and now is growing into neighboring regions from above. The right image shows a segmented femur with an incomplete femoral head.

Figure 46: *left*) initial region; *middle*) contour leaked through a weak boundary; *right*) femur with incomplete femur head.
5.5 Proposed Segmentation Script

The diagram 48 depicts the preliminary processing script that is proposed for the segmentation of the desired tissues. The region growing algorithm is preferred to the fuzzy connectedness for segmenting the vascular system. Because of its excellent runtime, an user can adjust the single necessary parameter just–in–time to optimize the segmentation result. The soft and muscle tissue will be discriminated by a fuzzy c-mean clustering algorithm that is fully automatically initialized and supplies conclusive results. The critical point stays the segmentation of the bones. At the moment there is no alternative to the described initialization of the algorithm (i.e. the initial model will be constructed from a grown region or from user–defined seed points). Appropriate default values for the parameter used in the creation of the feature images (Canny–edge or Gaussian magnitude based) and for the weights of the level set method have to be determined experimentally. In Fig. 47 models created with the proposed processing procedure are presented.

![Models created with the proposed script](image)

**Figure 47:** Models created with the proposed script
User loads datasets into application

User specifies seed points for each vessel tree and if needed adjust the lower threshold

User specifies seed points to grow an initial region for the bone level set segmentation

User adjusts/changes weights

Figure 48: Preliminary proposed segmentation script
6 Conclusion

The main idea behind the developed framework is that an arbitrary order of imaging processing steps that has been successfully set up for a specific task can be stored with all user-independent input parameters. This stored script can be loaded again and the processing pipeline will be reconstructed. For a successful rerun of the script a user just have to provide the remaining inputs (e.g. seed-points). The framework is capable of handling multiple input data, not restricted to a specific imaging modality. A comfortable graphical user interface is provided to guide an unskilled user through the segmentation process, forcing an intuitive work flow. The GUI window is separated in two areas, the control area and a view area. The viewer uses the common view for volume data consisting of three orthogonal 2D–views and a 3D–widget for volume and polygonal data rendering. The control area is used for setting the content and properties of the viewer. Once an user has loaded the input data, he can choose either to load an existing processing script, providing the necessary parameters and run it on the data or run single image processing steps in order to create a new script. All input parameters can be entered and changed in a robust fashion over the GUI's provided by each algorithm. Various algorithms have been integrated and can be arbitrarily attached to perform specific segmentation tasks. At the moment the application does not provide support for a registration of two images, so all datasets intended to be used for a combined segmentation are assumed to be co-registered.

The framework is designed to be extendable in two ways: on the one hand, completely new modules can be integrated via the functionality support that is inherited from the MITK base application. On the other hand, a dynamically plugin support has been implemented that allows to developed arbitrary algorithms and load them as dynamic library into the application during runtime. The common interface of all algorithms is designed to be as generally as possible.

First segmentation results created with the application are much promising and demonstrate the usability of developed framework.

7 Future Work

With the availability of appropriate dataset an evaluation of the methods and their results has to be done, i.e. comparing the semi-automated segmentation to a manually labeled one. Further the concept of the transferability and interchangeability of predefined segmentation scripts itself has to be evaluated. This is, in how far are the results reproducible on other datasets.

The experience gained while collecting feasible data for the project have shown that it is difficult to acquire MR datasets that are co-registered a priori (i.e. they cover exactly the same volume). Therefore algorithms for a registration of image data should be integrated and could further expand the scope of application.
Much work has to be done for optimizing the results of the segmentation of bone structures. The adjustment of the single weights used to control the terms in the level set equation, is difficult due to the fact that the movement of the curve is hidden from the user until a stopping criterion has been reached. Here a way to observe and control the evolution of the level set interface has to be found and integrated (e.g. a step wise execution with visual feedback).
8 References


[38] Meek MF, Stenekes MW, Hoogduin HM, Nicolai JPA. In vivo three-dimensional reconstruction of human median nerves by diffusion tensor imaging. Experimental Neurology. 2006;198(2):479–82.


[65] Lehman TM. Digitale Bildverarbeitung für Routineanwendungen. DUV; 2005.