


Harvard Medical School
Surgical Planning Laboratory

University of Heidelberg
University of Heilbronn

Segmentation of the Cerebrospinal Fluid from MRI Images for the Treatment of Disc Herniations

A thesis prepared
by

Martin Löpprich

Department of Medical Informatics
in partial fulfillment of the requirements for the degree of

Diplom-Informatiker der Medizin

at the

University of Heidelberg, Germany

November 2010

Advisors

Ron Kikinis, M.D.
Surgical Planning Laboratory
Harvard Medical School
Boston, Massachusetts
United States of America

Prof. Dr.-Ing. Hartmut Dickhaus
Department of Medical Informatics
University of Heidelberg
Heidelberg, Baden-Württemberg
Germany

Abstract

About 80 percent of people are affected at some point in their lives by lower back pain, which is one of the most common neurological diseases and reasons for long-term disability in the United States. The symptoms are primarily caused by overly heavy lifting and/or overstretching of the back, leading to a rupture and an outward bulge of an intervertebral disc, which puts pressure on and pinches the nerve fibers of the spine. The most common form is a lumbar disc herniation between the fourth and fifth lumbar vertebra and between the fifth lumbar vertebra and the sacrum. In recent years the diagnosis of lower back pain has improved, mainly due to enhanced imaging techniques and imaging quality, but the surgical therapy remains hazardous. Reasons for this include low visibility when accessing the lumbar area and the high risk of causing permanent damage when touching the nerve fibers. A new approach for increasing patient safety is the segmentation and visualization of the cerebrospinal fluid in the lower lumbar region of the vertebral column. For this purpose a new fully-automatic and a semi-automatic approach were developed for separating the cerebrospinal fluid from its surroundings on T2-weighted MRI scans of the lumbar vertebra. While the fully-automatic algorithm is realized by a model-based searching method and a volume-based segmentation, the semi-automatic algorithm requires a seed point and performs the segmentation on individual axial planes through a combination of a region-based segmentation algorithm and a thresholding filter. Both algorithms have been applied to four T2-weighted MRI datasets and are compared with a gold-standard segmentation. The segmentation overlap with the gold-standard was 78.7 percent for the fully-automatic algorithm and 93.1 percent for the semi-automatic algorithm. In the pathological region the fully-automatic algorithm obtained a similarity of 56.6 percent, compared to 87.8 percent for the semi-automatic algorithm.

Acknowledgements

First and foremost I would like to thank my advisor Prof. Dr. Hartmut Dickhaus (University of Heidelberg, Germany) for his full support throughout my study. He introduced me to the Surgical Planning Laboratory and offered me the possibility to write my diploma thesis abroad. Without his confidence and continuous help this thesis would not have been possible.

Furthermore I would like to express my deepest gratitude to Prof. Dr. Ron Kikinis (Harvard Medical School, Boston, USA) for the opportunity to participate in a project at the Surgical Planning Laboratory, and his honest advice and encouragement, which went beyond my research and thesis. It was a great honor for me to work in his laboratory.

A special thanks to Dr. Sylvain Jaume (Massachusetts Institute of Technology, Cambridge, USA) for sharing his knowledge, guidance and algorithmic ideas upon which I was able to build this thesis. I really appreciate the time he spent supervising and providing me with valuable background information.

In addition, a great many thanks to Dr. Steve Pieper (Harvard Medical School, Boston, USA), who provided me with many suggestions on how to proceed with the software development, and to Dr. Roland Metzner (University of Heidelberg, Germany), who supported me with the clinical background of my research. I am indebted to both of them for their help.

I gratefully acknowledge the Baden-Württemberg Landesstiftung foundation for their Baden-Württemberg scholarship and financial support.

Finally, I would like to emphasize that I am incredibly grateful for the unconditional support of my parents. Without their confidence and encouragement my stay in the United States would not have been possible.

Contents

List of Abbreviations

CAD	Computer-Aided Diagnosis
CSF	Cerebrospinal Fluid
CMake	Cross-platform Make
CT	Computed Tomography
EMG	Electromyography
GUI	Graphical User Interface
ITK	Insight Toolkit
LPS	Left, Posterior, Superior
MRI	Magnetic Resonance Imaging
MRT	Magnetic Resonance Tomography
NCS	Nerve Conduction Studies
NMR	Nuclear Magnetic Resonance
NMRI	Nuclear Magnetic Resonance Imaging
PDw	Proton Density-weighted
RAS	Right, Anterior, Superior
RF	Radiofrequency
ROI	Region of Interest
SE	Spin Echo
T1w	T1-weighted
T2w	T2-weighted
TE	Echo Time
TR	Repetition Time
VTK	Visualization Toolkit
XML	Extensible Markup Language

List of Tables

List of Figures

1. Introduction

Lower back pain is after headache the second most common neurological disease [1] and the fifth most common reason for all physician visits in the United States [2]. Around 60 to 80 percent of adults develop back pain symptoms during their lifetime, which are primarily caused by degenerated intervertebral discs [3]. The occurrence of lower back pain is about five percent of the population [4] and changes with age: below the age of 50, the observed incidence is 30 percent, while 85 percent of adults above 50 years exhibit it [5]. Approximately one quarter of U.S. adults reported having lower back pain lasting at least one whole day in the past few months [2], and seven percent reported at least one episode of severe acute lower back pain within a one-year period [6]. Lower back pain is also very costly: as published by the Archives of Internal Medicine in February 2010, the direct costs of treating lower back pain in the United States are over $50 billion a year [7]. In addition, indirect costs related to the days lost from work are about two percent of the U.S. workforce [8].

As Chapter 2 "Clinical Background" shows, lower back pain can be caused by a variety of conditions, of which the most likely is overly heavy lifting and/or overstretching the back that leads to a rupture and an outward bulge of the intervertebral disc [9]. However, it is not the disc itself that hurts, but rather the pressure of the disc and the resulting pinching of the nerve fibers in the spinal cord [1, 10]. A bulged or herniated intervertebral disc can occur in any disc of the vertebral column, but the most common form is a lumbar disc herniation between the fourth and fifth lumbar vertebra and between the fifth lumbar vertebra and the sacrum [11]. This is largely because of the fact that the pressure load of the body is greatest in the lumbar region and that the spinal cord in this area consists only of a bunch of nerves, called cauda equina, which are not as solid and well protected as in the cervical and thoracic region [1, 12, 13].

The diagnosis of lower back pain has improved during recent years, mainly due to enhanced imaging techniques and imaging quality, and it can now be determined with certainty whether the cause of lower back pain is a bulged intervertebral disc or not [9]. Often referred to as the gold-standard modality of diagnosing lower back pain is magnetic resonance imaging (MRI), which has been shown to be the most sensitive and specific imaging technique for herniated discs [12, 14, 15]. Compared to other diagnostic methods such as X-ray imaging, CT or Myelography, MRI is routinely used and most preferred since it is not based on ionizing radiation and can be applied safely and noninvasively to screen high risk patients above the age of 50 [14]. Even though the imaging quality has improved, the surgical therapy to treat a disc herniation is still hazardous and has to be performed in approximately 20 to 30 percent of patients, whose condition and symptoms did not improve during a conservative nonsurgical therapy [1, 16]. The challenges of the surgical removal of the disc material are the low visibility while accessing the lumbar area and the high risk of causing permanent damage when touching the nerve fibers [17].

A possibility of increasing patient safety during the surgery is to separate the individual structures on the MRI images and to display them as a three-dimensional model in a Computer-Aided Diagnosis (CAD) software application. This model can afterwards be used for a better planning of the surgical intervention but also to assist the clinician during the delicate operation, for example by visualizing the three-dimensional model in addition to the current position of the surgical instruments.

Corresponding to the state of the art, several approaches exist to perform a computer-assisted surgery of lumbar disc herniations. While one method published in 2006, for example, provides an automated segmentation of the whole spine on MRI datasets using a model-based searching algorithm to locate each vertebra and disc [18], a similar algorithm was applied in 2009 on CT scans [19]. However, the currently available methods and published papers are mainly focused on the automatic detection and separation of vertebrae and soft tissues [18, 19, 20, 21], but do not discuss the nerve fibers, which are most sensitive during a surgical intervention. This is in particular due to the fact that common MRI techniques such as T1- or T2-weighted images, which are described in detail in Chapter 2 "Magnetic Resonance Imaging", are still not able to display the nerve fibers within the vertebral column adequately and clearly visibly, mainly because of the small size and low contrast between the nerve fibers and the surrounding cerebrospinal fluid (CSF) [15, 22].

A new approach to determine at least the location of the herniation and of the pinched nerve fibers is to segment the fluid within the vertebral column, since it is also affected by the deformed disc and can serve as an appropriate visualization of the pathological region. Although the method of segmenting the CSF is not entirely new, it has never been applied to the lumbar region of the spine. Instead, it was always detected together with the different tissues of the brain and mainly performed on T1-weighted images [23, 24, 25]. Because this weighting is well-suited for visualizing the anatomical structures of the brain and body, a segmentation of the CSF, gray and white matter is principally useful for long-term studies, such as the measurement of alcohol's influence on the brain [26]. Consequently, a detection of the CSF has so far not often been performed on T2-weighted datasets, which are more sensitive for pathology.

In total, the segmentation of the CSF in the lumbar region of the vertebral column is motivated by two aspects: first to apply an already existing registration method to T2-weighted MRI datasets and second to provide a novel visualization of a herniated intervertebral disc, which can be used to assist surgical interventions.

The approaches to segment the CSF on T2-weighted MRI images range from manual and semi-automatic with user interaction to fully-automatic. While a manual segmentation, as shown in the figure below, is not a realistic option because, firstly, a high level of knowledge of the anatomic structures is required, and secondly because it is a very time-consuming and complex task to follow the structures on three-dimensional images, semi- and fully-automatic algorithms are required to perform the segmentation and visualization. Furthermore, an automated segmentation allows reproducibility and reliability of the result and is independent of the experience of the clinician.

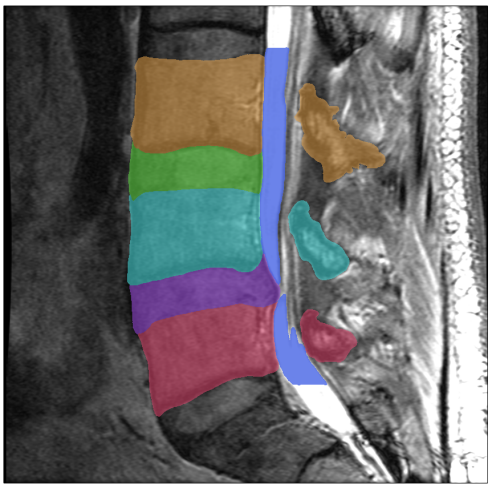


Figure 1.1.: Manual segmentation of the lumbar spine

The resulting model of the manual segmentation is illustrated by the following figure, which can be rotated, flipped, cropped and superimposed by the surgical instruments to derive a maximum benefit for the surgical intervention.

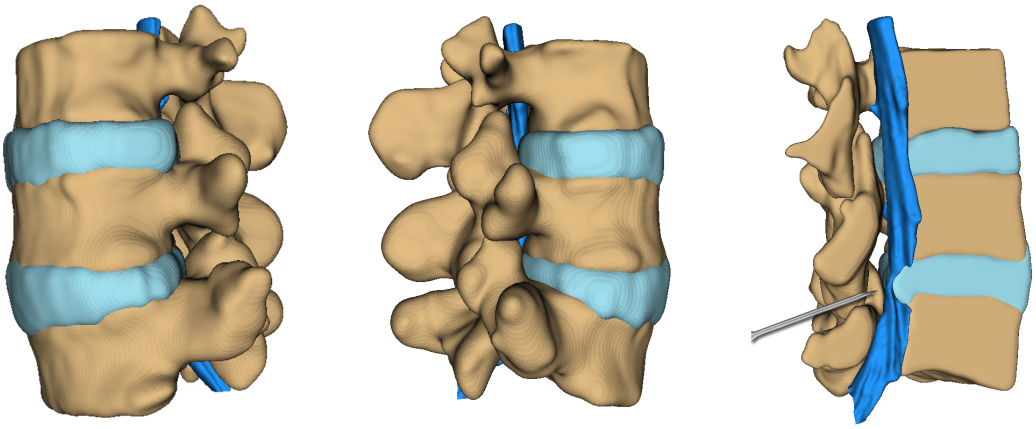


Figure 1.2.: Three-dimensional model of the manual segmentation of the lumbar spine

The advantages of such a model and of the underlying segmentation are numerous; the manual creation by a clinician is unrealistic and so automatic algorithms are worth pursuing.

An overview of the main principles in designing a simple segmentation module is provided in Chapter 3 “Materials and Methods”, and Chapter 4 presents both a fully-automatic and a semi-automatic approach to performing the segmentation of the CSF in the lower lumbar region of the vertebral column on T2-weighted MRI datasets. While the fully-automatic algorithm takes advantage of the high contrast between the CSF and the spine and realizes the registration with a model-based searching method, the semi-automatic algorithm benefits from a user interaction to locate the CSF. The implemented segmentation methods differ fundamentally from each another and refer either to a volume or to individual axial planes.
Besides a detailed description of the appropriate algorithmic steps, Chapter 4 also provides several illustrations of the segmentations and three-dimensional models that result from the application of both algorithms to four MRI datasets.

In the last chapter, the two algorithms and their outcomes are contrasted and discussed in terms of their completeness and accuracy. Furthermore, a gold-standard segmentation of one dataset is available, which is used to obtain a better assessment of the resulting segmentations and to determine a similarity coefficient between the gold-standard segmentation and the outcomes of the automatic algorithms.

2. Background and Theory

2.1. Clinical Background

The human back is the surface opposite to the chest, extending from the bottom of the neck and shoulders to the loin area. The central feature of the human back is the vertebral column, which is composed of 24 segmental bones, the vertebras, as well as the sacrum and coccyx. The vertebrae move against each other and thereby allow movement forward, backward, left, right and around the longitudinal axis. This range of movement is supported by the discs, which also stabilize the vertebral column by several ligaments [27, 28].

A representation of the vertebral column is shown in the illustration below.

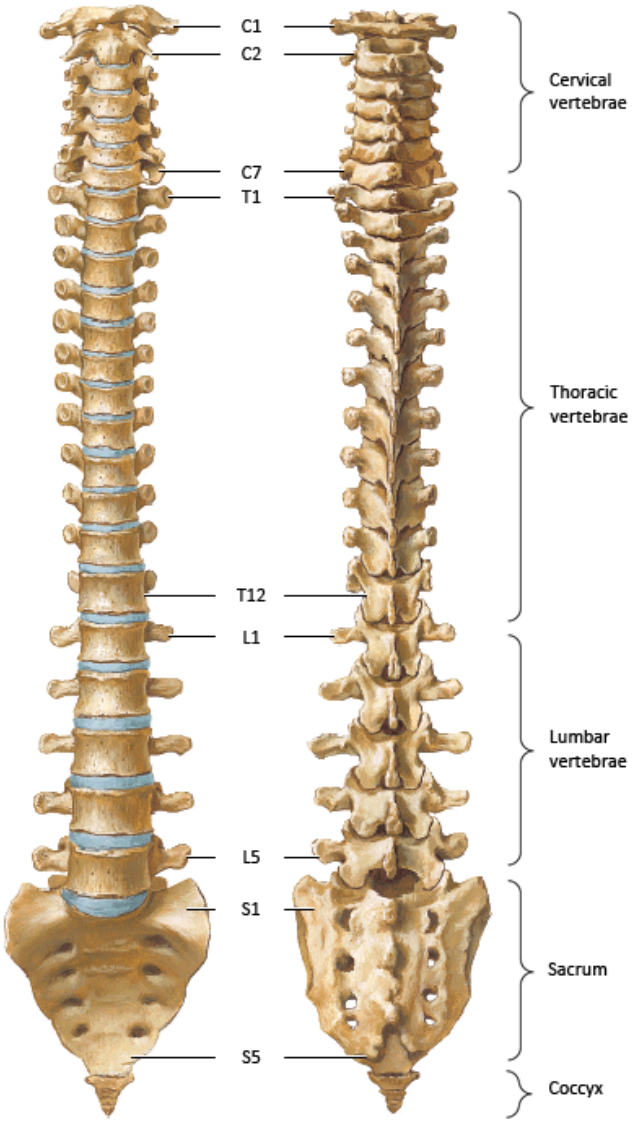


Figure 2.1.: Anterior and posterior view of the vertebral column [29]

In detail, the vertebral column consists of the following regions [10]:

- The cervical vertebrae with 7 cervical bones (numbered top-to-bottom from C1 to C7)
- The thoracic vertebrae with 12 thoracic bones (Th1 to Th12), that are articulated with the ribs
- The lumbar vertebrae with 5 lumbar bones (L1 to L5)
- The sacrum with 5 sacral bones (S1 to S5) that are fused into one compact bone
- The coccyx with usually 4 fused coccygeal bones

The structure of the vertebrae column is from the third cervical (C3) to the fifth lumbar vertebrae (L5) consistent and differs only in size and shape. The two essential parts are an anterior segment, the vertebral body, which transmits most of the body weight on to the lower limbs, and a posterior part, the vertebral arch, which encloses the vertebral foramen. The vertebral arch is formed by a pair of pedicles as well as a pair of laminae, and supports seven processes: four articular, two transverse, and one spinous [27, 30].

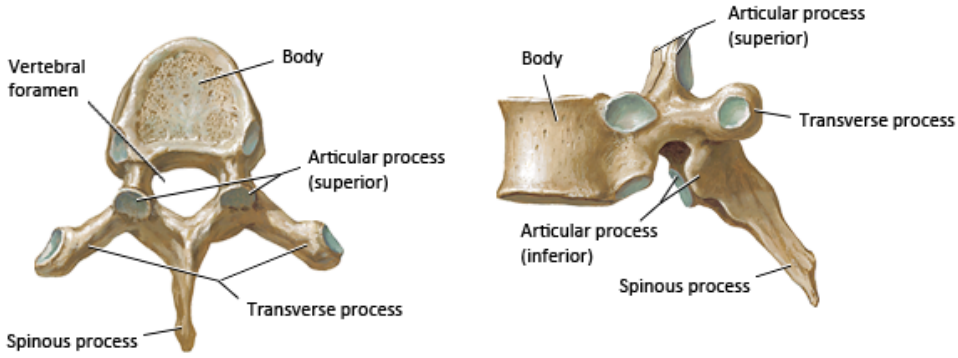


Figure 2.2.: Superior and lateral view of the T6 vertebra [29]

Between adjacent vertebrae in the cervical, thoracic and lumbar spine, and between L5 and the sacrum, are intervertebral discs [10]. Each disc has a thickness of about 5 mm and forms a cartilaginous joint to allow slight movement of the vertebrae [30]. A disc consists of an inner gelatinous core, the nucleus pulposus which compensates pressure and absorbs shocks, and an outer ring of fibrocartilage, named annulus fibrosus. Furthermore, the nucleus pulposus bulges laterally when bending or stretching and thereby distributes the load in an optimal way [28].

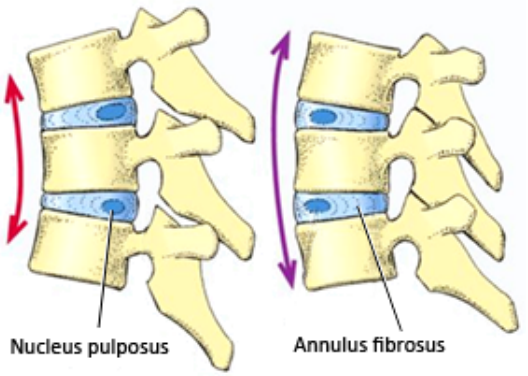


Figure 2.3.: Flexion (bending) and extension (stretching) of the vertebral column [30]

The vertebral column surrounds and protects the relatively shorter spinal cord, which serves as an "information highway" between the brain and the peripheral nervous system [28]. It is a tubular bundle of nerve cells and fibers wrapped together about as thick as a finger that extend down to the space between the first and second lumbar vertebrae. The spinal cord functions primarily in carrying information from the brain to the rest of the body and vice versa (through ascending and descending tracts made up of white matter), but serves also as center for coordinating reflexes (through the gray matter) [13, 31]. In the upper part of the vertebral column, spinal nerves exit directly from the spinal cord, whereas in the lower part nerves pass further down the vertebral column before exiting. This is motivated by the fact that the vertebral column grows more quickly during fetal development than the spinal cord and has therefore an average length of only 45 cm in males, compared to 71 cm of the vertebral column [10, 32]. The part below the conus medullaris, the terminal end of the spinal cord which is at the level of the second lumbar vertebra, consists only of a collection of nerves, called cauda equina [30].

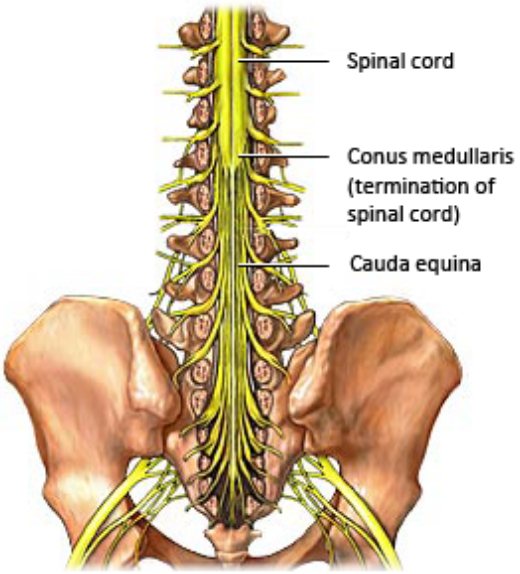


Figure 2.4.: Posterior view of the lumbar area [33]

The spinal cord is covered by three layers of tissue, named dura mater, arachnoid mater and pia mater. These three spinal meninges are continuous with those in the brainstem and cerebral hemispheres [10]. The dura mater is the outermost layer and forms a tough protective coating. Between the dura mater and the surrounding bone of the vertebrae is the epidural space, which is filled with fat and blood vessels to protect and supply the spinal cord [31]. The space between the middle layer, the arachnoid mater, and the innermost layer, the pia mater, which is tightly associated with the surface of the spinal cord, is called subarachnoid space and made up of cerebrospinal fluid (CSF) [13]. In contrast to the spinal cord and due to the fact that the spinal cord ends higher, the cauda equina is only covered by the dura and arachnoid mater [31]. Nevertheless, the nerve bundles of the cauda equina in the lower lumbar region are also located within the CSF of the subarachnoid space.

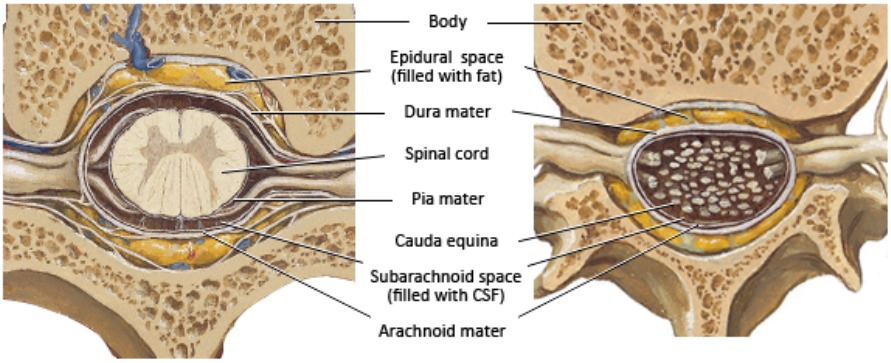


Figure 2.5.: Cross-section through a thoracic and a lumbar vertebra [29]

The CSF, also known as liquor cerebrospinalis, is produced in the brain (more precisely in the choroid plexus) by ependymal cells [10]. It is a clear bodily fluid that is similar in composition to blood plasma, from which it is formed [31]. Once produced, CSF circulates freely through the ventricular system around the brain and enters the subarachnoid space via apertures in the walls of the ventricles, from where it flows to the spinal cord and to the cauda equina [28]. In adults, the total CSF volume is about 150 ml. However, since large amounts are drained into the blood through arachnoid granulations and are cleaned by the choroid plexus, about 500 ml of CSF is produced daily [31]. Besides the purpose of protecting the brain and spinal cord from blows and traumata, the CSF also serves for chemical regulation [28] and reduction of the brain weight [31].

CSF is usually obtained for diagnostic purposes by a puncture of the lumbar spine in cases of suspected inflammation of the nervous system such as meningitis, encephalitis or myelitis [10]. To do so, a needle penetrates the dural sac to enter the subarachnoid space and to collect a sample of CSF [29]. Subsequently, the extracted CSF can be analyzed for bacteria, blood impurities, pus and clots [10].

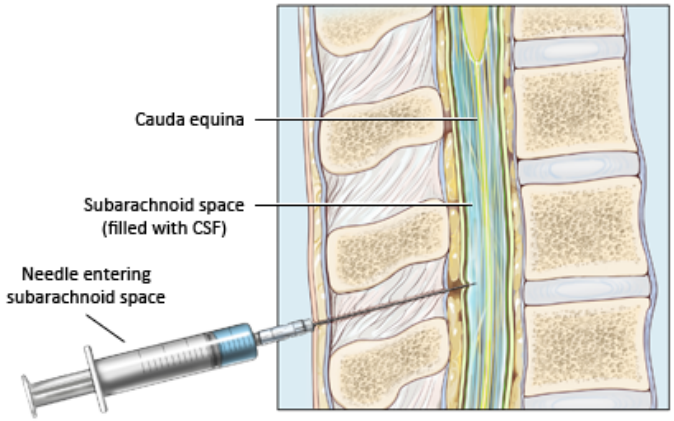


Figure 2.6.: Lumbar puncture to collect a sample of CSF [34]

In total, the delicate nerve fibers of the spinal cord and cauda equina are surrounded by three protective barriers (a bony vertebra, an adipose tissue in the epidural space and liquor in the subarachnoid space), which all act as a cushion to protect the nerves against damage from outside. Although the nerve fibers are wrapped very well, intervertebral discs offer only minimal protection against external influences and are mainly responsible for pinched nerves [1]. This occurs when the inner gelatinous nucleus pulposus bulges through a tear in the annulus fibrosus, putting pressure on and pinching the nerve fibers [10].

According to [9], [11] and [30], the main causes of bulged or herniated intervertebral discs are:

- Traumatic injuries such as too heavy lifting or overstretching the back
- Abnormal biomechanical stress such as lifting while bent at the waist, rather than lifting with the legs while the back is straight
- Aging, because the annulus fibrosus loses fluid, flexibility and the ability to contain the nucleus pulposus in its center
- Genetic components that lead to a gene mutation of the proteins which are involved in the regulation of the extracellular matrix

A disc herniation can occur in any disc of the spine, but approximately 90 percent of bulging discs occur in the lower back, 8 percent in the cervical area and only 2 percent in the thoracic region [9, 11]. The most common form of a lumbar disc herniation is between the fourth and fifth lumbar vertebra and between the fifth lumbar vertebra and the sacrum, since the pressure load on the discs is greatest in this area [12].

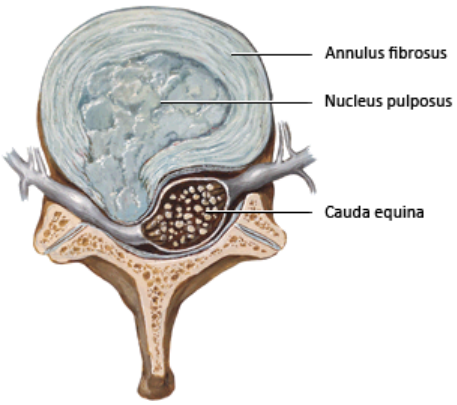


Figure 2.7.: Lumbar disc herniation [29]

As Figure 2.7 illustrates, usually one side of the nerve roots is compressed by the herniation and clinical symptoms therefore affect only one side of the body. However, if the prolapse is large enough and presses on the spinal cord or on the cauda equina in the lumbar region, affection of both sides of the body may occur, often with serious consequences. Symptoms usually depend on the level of herniation and can range from severe pain to paresthesia and paralysis [30].

The following table shows the most common clinical features of a herniated lumbar disc dependent on the level of herniation [12, 29].

Herniation level	**Location of pain**	**Location of numbness**	**Motor deficit**
L4 - L5	Sacroiliac joint Hip Lateral thigh Lateral leg	Lateral leg First three toes	Difficulty walking on heels
L5 - S1	Sacroiliac joint Hip Posterolateral thigh Posterolateral leg Heel	Back of calf Lateral heel Lateral foot	Difficulty walking on toes

Table 2.1.: Symptoms of a L4-L5 and a L5-S1 disc herniation

The finding and diagnosis of a disc herniation is made by a radiologist based on the physical examination, medical history and symptoms. In addition, a variety of diagnostic methods are available to visualize the herniation and to confirm the cause of the lower back pain, such as X-ray imaging, CT and MRI scans, Myelography or electrodiagnostic procedures [1, 9, 11]:

- X-ray imaging is a relatively inexpensive method and often the first imaging technique used by a general practitioner to locate the cause and site of the symptoms. The bony structure of the vertebra column and any vertebral misalignments or fractures are clearly visible. Although traditional plain X-rays are limited in their ability to image soft tissues such as discs, muscles and ligaments, they are still used to confirm or exclude other possibilities like tumors, fractures or infections. A final confirmation however cannot be provided with X-rays alone.
- Computed Tomography (CT) uses X-rays, which are passed through the body at various angles and are detected by computerized scanners to produce two-dimensional slices of the spine. The images show both the bony structure as well as the soft tissues and are therefore very useful for confirming which disc is damaged.
- Magnetic Resonance Imaging (MRI) produces three-dimensional images of body structures using a magnetic field and radiofrequency waves to provide a detailed view (even better than CT scans) of the soft tissues of the spine. It can detect the spinal cord, nerve roots and surrounding areas, as well as nerve compression, enlargement, degeneration and tumors.
- Myelography involves an injection of a special contrast dye into the subarachnoid space to enhance the visibility of the CSF. Afterwards, an X-ray fluoroscope or a CT scanning records the images and allows to recognize the spinal cord and the nerve compression caused by herniated discs or fractures. Because this procedure involves the injection of foreign substances, Myelography is usually suggested for patients who are considering lumbar surgery or whose pain has not responded to conventional treatments.
- Electrodiagnostic procedures such as Electromyography (EMG) and Nerve Conduction Studies (NCS) are not imaging techniques in the classical meaning. These tests are used to measure the muscle response to electrical stimulation (EMG) or the electrical impulse along nerve roots and peripheral nerves (NCS). Thereby nerve damage or muscle weakness can be detected.

The gold-standard modality and most preferred method for diagnosing a disc herniation is MRI, which has a higher sensitivity and is more specific than X-ray imaging and CT scans and is in contrast to Myelography noninvasive [12].

Most lower back pain can be treated with conservative nonsurgical treatments, which are usually attempted first. Some studies indicate that symptoms of lower back pain have been reduced in approximately 70 to 80 percent of cases without any surgical intervention [9, 16, 35]. These treatments involve medication, ice and heat compresses, bed rest and physical therapies to reduce the pain. If the condition does not improve with conservative treatments or the patient has a significant neurological deficit, a surgery of the herniated lumbar disc, called discectomy, is the most common procedure [16].

A discectomy, also known as open discectomy, is the surgical removal of the nucleus pulposus which presses on a nerve root. During this procedure, the surgeon first makes an incision down the middle of the lower back and dissects the spinal muscles to expose the vertebra. Afterwards, a laminectomy is usually performed to remove a small portion of the vertebral bone (called lamina) and to permit access to the compressed nerve root. This nerve root is moved aside and the injured disc can be examined. Next, the surgeon cuts a small opening in the annulus fibrosus and injects a substance through a tube to liquefy the nucleus pulposus. Forceps are placed inside the opening to remove the nucleus pulposus and to clean out the material within

the disc. Finally, the muscles and soft tissues are put back in place, and the skin can be stitched together [1, 9].

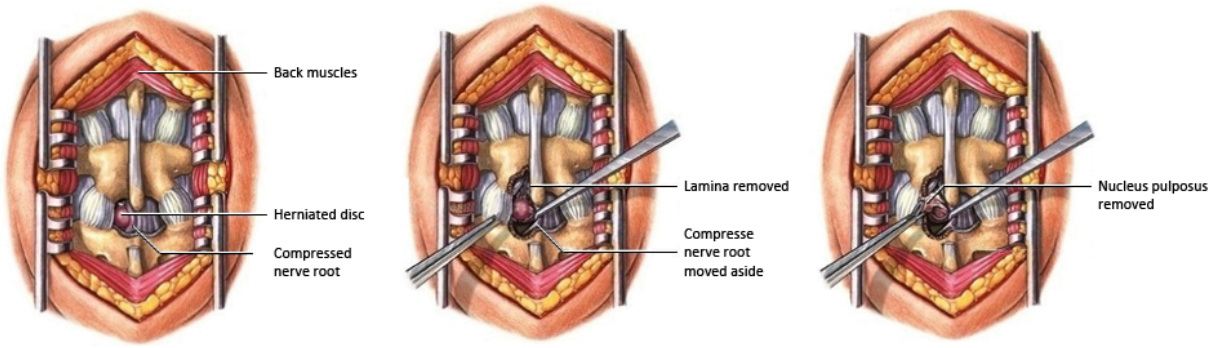


Figure 2.8.: Lumbar open discectomy [36]

A less invasive technique to treat a herniated disc is a microdiscectomy, which causes less muscle injury than a traditional discectomy due to a smaller incision in the back. By using an endoscope or a microscope only a tiny hole in the vertebra, instead of removing the lamina, is required [9, 37].

Even if advances in surgery have produced effective alternatives to traditional discectomy procedures, the risk of complications is about 10 to 15 percent [17]. These complications include infections, bleeding and nerve damage, which can occur from bumping or cutting nerve tissue [38]. An injury to the spinal nerves can lead to an increase in symptoms and cause a severe muscle weakening or a complete loss of sensation in the areas supplied by the nerve. Since lower back surgery is hazardous, it should be performed only in patients whose condition does not improve with conservative treatments or who suffer from neurologic diseases [1].

2.2. Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI), also known as Nuclear Magnetic Resonance Imaging (NMRI) or Magnetic Resonance Tomography (MRT), is a noninvasive imaging technique used primarily in medical settings to visualize detailed structures of the inside of the human body. The MRI scanning equipment creates a powerful magnetic field around the body to align the nuclear magnetization of water molecules. Afterwards, radiofrequency (RF) fields are briefly turned on to alter the alignment of the nuclear magnetization. When the radio waves are switched off again, the magnetization returns back to its previous alignment, which can be detected by a scanner to construct an image of the body [39, 40, 41].

In contrast to other imaging and diagnostic methods, such as CT or X-ray, MRI uses no ionizing radiation and provides a greater contrast between the different soft tissues of the body [42]. Therefore, it is a common procedure in neurology and orthopedics to evaluate for example the lumbar region for bone degeneration, injuries or diseases in muscles, discs, ligaments, nerves and blood vessels [1].

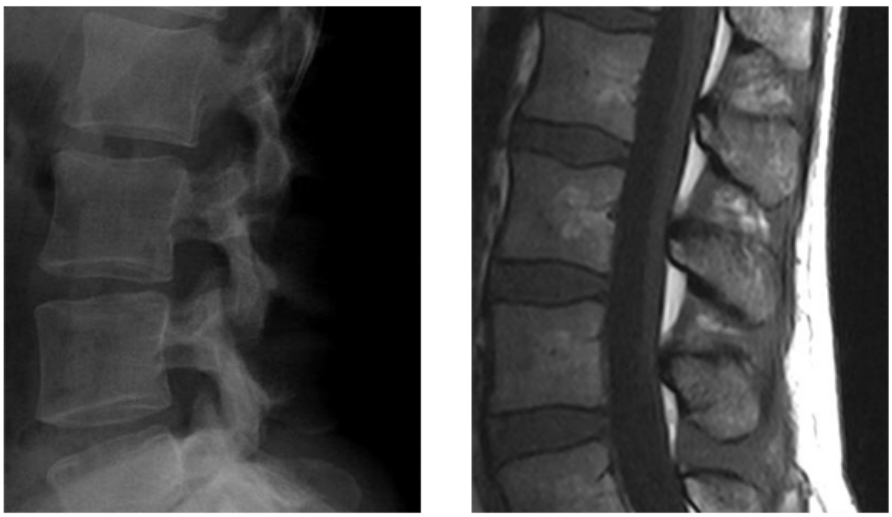


Figure 2.9.: Comparison between lumbar X-ray imaging and MRI [43]

MRI is based on the principles of Nuclear Magnetic Resonance (NMR), discovered independently by Felix Bloch and Edward Purcell in 1946 [40]. Between the period from 1950 to 1970, NMR was primarily developed and used as an analytical tool for chemical and physical molecular analysis [42]. Due to the negative connotations associated with the word nuclear in the 1970s, the imaging technique was renamed MRI (rather than NMRI) and first demonstrated on small tube samples by Paul Lauterbur in 1973 [44]. About four years later, in 1977, the first MRI studies performed on human beings were published [45, 46].

MRI uses the signal from the nuclei of the hydrogen atom ^1H for image generation [40]. A hydrogen atom consists of a single proton which is orbited by a single electron. Since the proton has a positive charge and the electron a negative charge, the hydrogen atom as a whole is electrically neutral. The importance of hydrogen atoms for the MRI technique is also justified by the fact that the human body is mostly composed of fat and water containing many hydrogen atoms and making up approximately 63 percent of all atoms in the human body [42].

Atoms with an uneven number of protons or neutrons, such as hydrogen atoms, also have the property that the proton rotates or spins about its axis [41]. As a rotating mass with an electrical charge, the proton has a magnetic moment B and creates a magnetic field parallel to the axis of rotation [47].

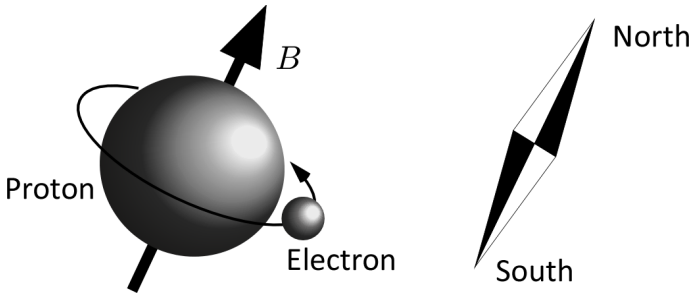


Figure 2.10.: A hydrogen atom and its direction of the magnetic field

In a volume of tissue containing a collection of hydrogen atoms, the rotation axes and the magnetic moments B are oriented randomly in all directions. Therefore, the magnetic fields created by all protons of the hydrogen atoms are mutually compensated, which results in a net magnetization $\vec{M}_0$ of zero [48]. However, if the tissue is exposed inside an external magnetic field $\vec{B}_0$, the situation changes and the individual protons align to the direction of $\vec{B}_0$ [47].

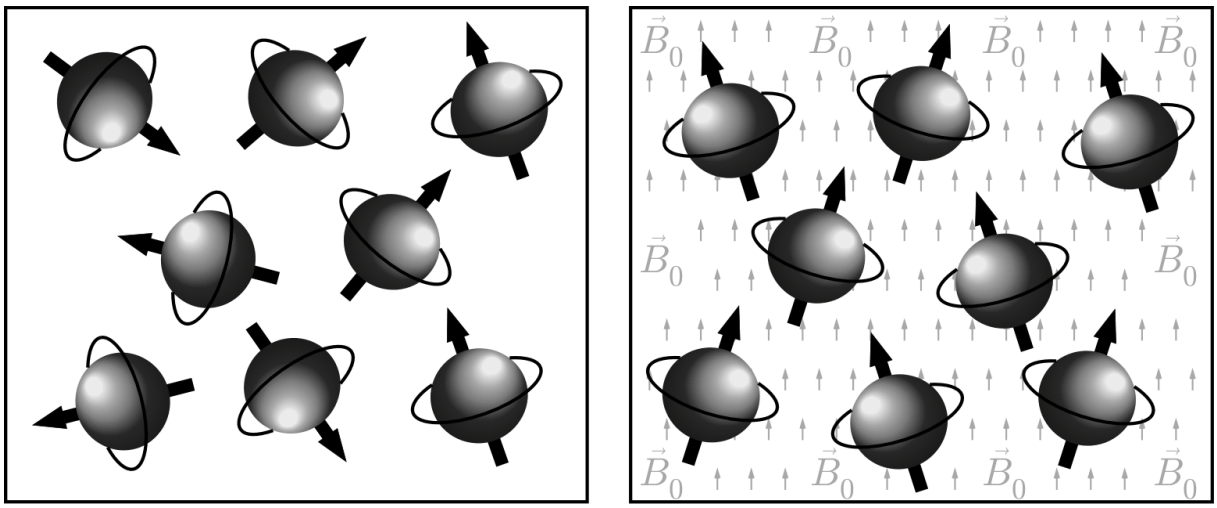


Figure 2.11.: Tissue of hydrogen atoms outside and inside an external magnetic field

Figure 2.11 shows that the alignment of the protons is tilted slightly away from the direction of the magnetic field, because of the proton spinning [48]. This property is called "precession" or the Larmor frequency and occurs because of the interaction of the magnetic field with the spinning charge of the protons. The Larmor frequency is proportional to the strength of the magnetic field $\vec{B}_0$ and is given by the Larmor equation [41]:

$$\omega_0 = \gamma \cdot \vec{B}_0$$

where ω_0 is the Larmor frequency in MHz, γ a gyromagnetic ratio specific for each nucleus and $\vec{B}_0$ the strength of the magnetic field in tesla.

Even if the spin axes of the protons in the presence of an external magnetic field are not parallel to the magnetic field $\vec{B}_0$, all protons rotate with the same frequency around the same precession axis, which means that all individual magnetic moments add together and build up a net magnetization $\vec{M}_0$ [47]. However, in the direction perpendicular to $\vec{B}_0$, the spin axes are still randomly distributed just as they were outside the magnetic field, so that there is still no net magnetization perpendicular to $\vec{B}_0$. The whole net magnetization $\vec{M}_0$ is therefore oriented parallel to the magnetic field [42].

By convention, $\vec{B}_0$ and the axis of precession is defined to be oriented in z direction of a Cartesian coordinate system [47]. The projection of the net magnetization $\vec{M}_0$ on the z axis is called longitudinal magnetization $\vec{M}_z$. As described previously, inside a magnetic field there is [49]:

$$\vec{M}_z = \vec{M}_0$$

The projection of $\vec{M}_0$ in the xy plate is known as transverse magnetization $\vec{M}_{xy}$, which is in the presence of $\vec{B}_0$ [41]:

$$\vec{M}_{xy} = 0$$

Overall, the magnetic field $\vec{B}_0$ causes an aligned stable state of all protons and a net magnetization $\vec{M}_0$ parallel to the magnetic field $\vec{B}_0$ (z axis).

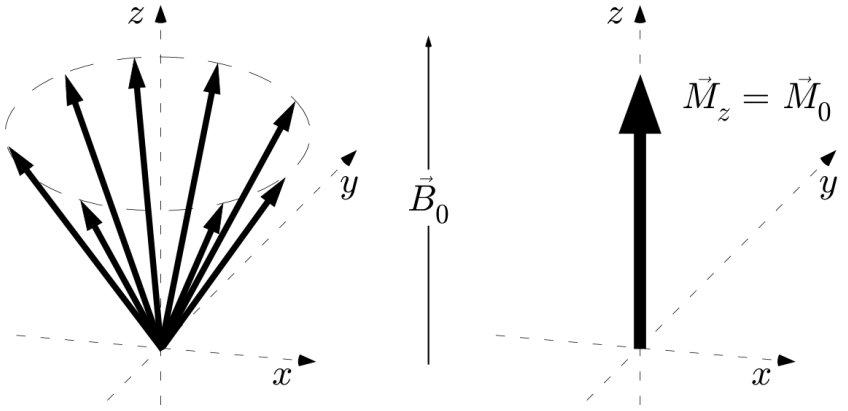


Figure 2.12.: Magnetization inside an external magnetic field

By applying a radiofrequency (RF) pulse into such a stable system, energy is absorbed by the protons, which are deflected from their steady position [48]. This is only possible if the RF pulse has the same frequency as the Larmor frequency, called resonance condition [39]:

$$\omega_{RF} = \omega_0$$

The process of energy absorption results in the net magnetization $\vec{M}_0$ being more and more tipped away from the z axis towards the xy plate [42]. If the RF pulse is strong enough and applied long enough, the magnetization can be flipped over by exactly 90° and rotates afterwards entirely in the xy plate [47]. As a result, the transverse magnetization $\vec{M}_{xy}$ increases. When the RF pulse is turned off, the protons and the net magnetization slowly return to their previous state and the transverse magnetization decays again. The return to the stable state present before excitation is caused by two independent processes, called spin-lattice interaction and spin-spin interaction [41].

The spin-lattice interaction describes the return of the hydrogen nuclei to their ground state by dissipating the additional energy, applied by the RF pulse, to their surroundings [47]. The time required for this recovery is known as T1 relaxation time and the whole process as T1 or longitudinal relaxation [48]. If a 90° pulse is applied, $\vec{M}_0$ rotates to the xy plate and no longitudinal magnetization will be present following the pulse. As time goes on, the longitudinal magnetization $\vec{M}_z$ returns to $\vec{M}_0$ as protons release their energy [41].

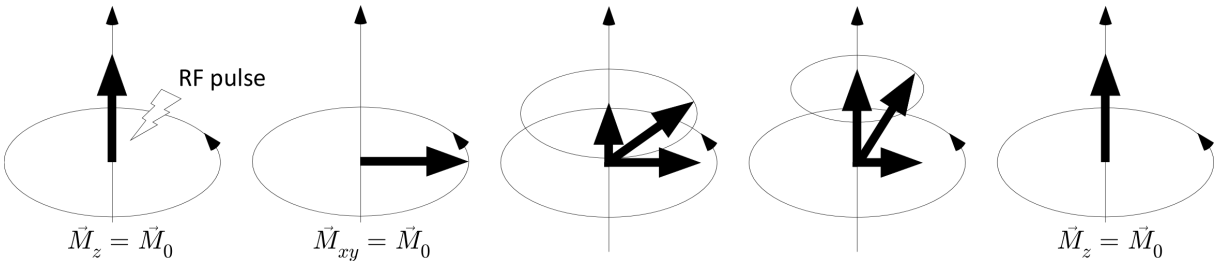


Figure 2.13.: T1 relaxation process

The recovery of the longitudinal magnetization magnitude M_z to 63 percent of its equilibrium value M_0 typically follows an exponential growth process [39]:

$$M_z(t) = M_0 \cdot \left(1 - e^{-\frac{t}{T1}}\right)$$

where T1 is the time constant describing the rate of growth.

The figure below shows that following a 90° RF pulse no longitudinal magnetization is present and at $t = T_1$ the magnetization $\vec{M}_z$ has returned to 63 percent of its value prior to the pulse.

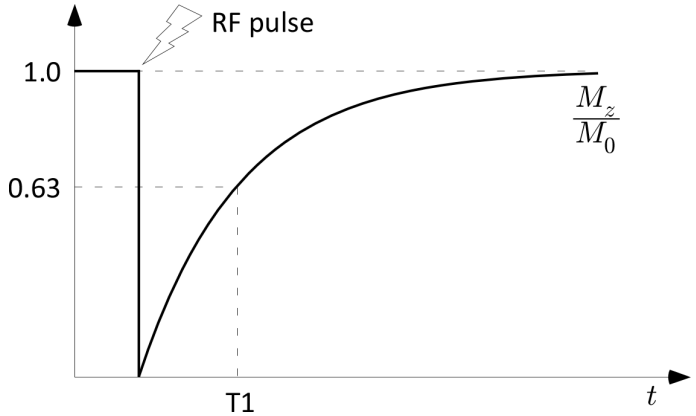


Figure 2.14.: T1 relaxation curve

Besides the releasing of the energy to their surroundings, protons also exchange energy with each other, described by the T2 or transverse relaxation [49]. At the end of a 90° RF pulse all protons have absorbed energy and are oriented in the transverse plane, which results in a transverse magnetization $\vec{M}_{xy}$ of the same value as $\vec{M}_0$. Since all protons rotate synchronized around the precession axis at the same frequency ω_0, energy can be released by one and be absorbed by its neighbor [39]. Therefore, the exchanged energy remains longer in the system and causes a loss of phase coherence between the proton spins. The gradual "dephasing" of the spins results in a random distribution of the protons in the transverse plane and a gradual decreasing of the transverse magnetization $\vec{M}_{xy}$ [41]. As time elapses, more and more proton spins are out of phase and the transverse magnetization $\vec{M}_{xy}$ returns to 0 [47].

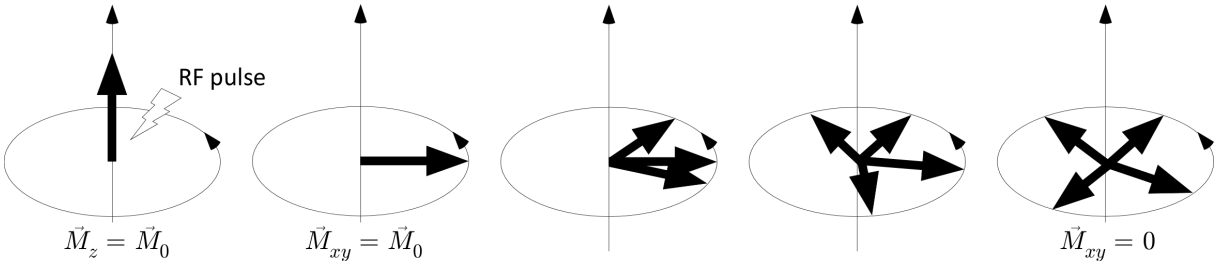


Figure 2.15.: T2 relaxation process

The time that is required for the transverse magnetization $\vec{M}_{xy}$ to decay by 63 percent of its initial value is called T2 relaxation time. The decay of the transverse magnetization follows also an exponential process [39]:

$$M_{xy}(t) = M_0 \cdot e^{-\frac{t}{T2}}$$

where T2 is the time constant describing the rate of decrease.

The T2 relaxation process is illustrated by following figure, where the transverse magnetization is maximal after a RF pulse and has decreased at $t = T_2$ by 63 percent [47].

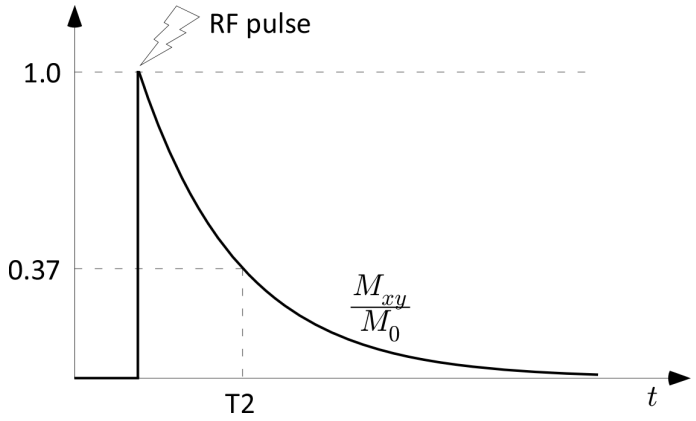


Figure 2.16.: T2 relaxation curve

The T1 and T2 relaxation times can both be measured and vary from one tissue to the next. In addition, the T1 time is dependent on the strength of the external magnetic field $\vec{B}_0$. As listed in the table below the average T2 relaxation time occurs in most tissues in the first 40 to 100 msec., while the T1 relaxation occurs within 200 to 900 msec. [39].

Tissue Type	**T2 [msec.]**	**T1 [msec.] at 0.5 tesla**	**T1 [msec.] at 1.5 tesla**
Muscles	47	550	870
Liver	43	330	500
Gray matter	101	660	920
White matter	92	540	790
Fat	84	210	260
CSF	200	1700	2200

Table 2.2.: Values of T1 and T2 relaxation times in different tissues [50]

In total, the signal intensity of an MRI image and therefore the image contrast is affected by three tissue-specific properties [41]:

- The T1 time of a tissue describes the time which is required for the recovery of the longitudinal magnetization of the proton spins by dissipating energy to the lattice. Images whose contrast is mainly determined by the T1 time are called T1-weighted (T1w) images.
- The T2 time of a tissue is the time it takes for the transverse magnetization of the proton spins to decay by exchanging energy with each other. Images whose contrast is mainly determined by the T2 time are called T2-weighted (T2w) images.
- The proton density ρ of a tissue is the number of excited hydrogen atoms per unit volume. Images whose contrast is mainly determined by the proton density are called proton density-weighted (PDw) images and can be obtained by minimizing the T1 and T2 weighting.

In order to generate an MRI image, the protons must be excited and the resulting magnetization and relaxation times recorded many times. A combination of several RF pulses in a row to influence the hydrogen atoms is called MRI sequence [39]. A commonly used sequence in MRI is the Spin Echo (SE) sequence, which has at least two RF pulses, a 90° excitation pulse and one or more 180° refocusing pulses that generate the spin echos [47]. The signal of a SE sequence depends on the three tissue-specific parameters (T1, T2 and proton density) and on two recording parameters, called Repetition Time (TR) and Echo Time (TE) [40, 49]:

$$S_{SE} = \rho \cdot \left(1 - e^{-\frac{TR}{T1}}\right) \cdot e^{-\frac{TE}{T2}}$$

The Repetition Time (TR) is the length of the relaxation between two 90° excitation RF pulses and therefore controls mainly the T1 weighting of the resulting MRI image [47]. If the TR interval is long most of the excited protons in different tissues have recovered and regained their longitudinal magnetization. Hence these tissues can no longer be distinguished and the T1 weighting is low. On the other hand, if a short TR is selected, tissues with a short T1 relaxation time regain most of their longitudinal magnetization and thus produce a stronger signal than tissues with a long T1 relaxation time. Therefore, a short T1 time usually results in a bright appearance on an MRI image while tissues with a long T1 time appear dark [39, 41].

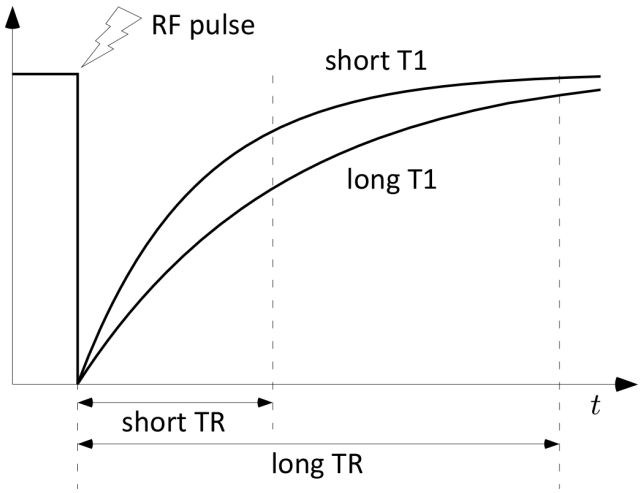


Figure 2.17.: Relationship between TR and T1

The Echo Time (TE) is the time between the 90° RF pulse and the collection of the signal after the 180° refocusing pulses and is therefore crucial for the T2 weighting [49]. If a short TE is used, the T2 relaxation has only just started and the decay of the transverse magnetization in different tissues is low. A long TE, however, causes tissues with a short T2 relaxation time to lose most of their transverse magnetization, which results in a weak signal. These tissues appear dark. In contrast, tissues with a longer T2 time appear bright, because they produce a stronger signal at the time of echo collection [39, 51].

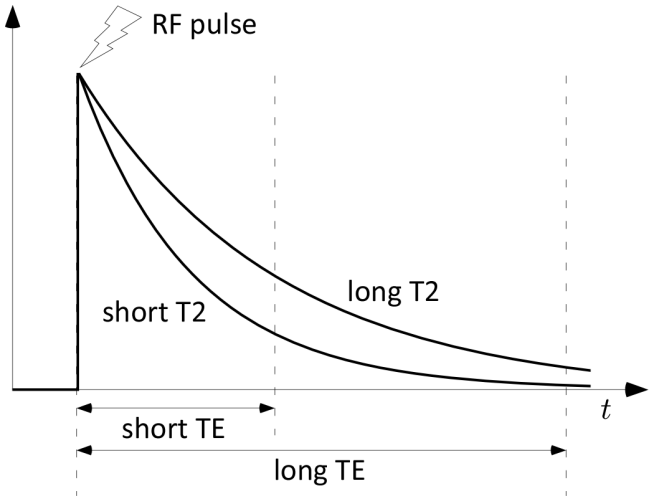


Figure 2.18.: Relationship between TE and T2

To acquire a T1w image using a SE sequence, TE (the time that affects the T2w) should be much shorter than the usual T2 relaxation times (Table 2.2). This can also be verified by the equation of the SE sequence [49]:

$$TE \ll T2 \quad \rightarrow \quad S_{SE} = \rho \cdot \left(1 - e^{-\frac{TR}{T1}}\right) \cdot e^{0} = \rho \cdot \left(1 - e^{-\frac{TR}{T1}}\right)$$

A T1w image depends therefonly only on T1 and the proton density ρ.

In contrast, a T2w image can be acquired with a TR that is longer than the general T1 relaxation times (Table 2.2). For the SE equation it follows [49]:

$$TR \gg T1 \quad \rightarrow \quad S_{SE} = \rho \cdot (1 - e^{-\infty}) \cdot e^{-\frac{TE}{T2}} = \rho \cdot e^{-\frac{TE}{T2}}$$

By combining the T1w and T2w effect, an MRI image can be obtained that only depends on the proton density ρ [49]:

$$TE \ll T2 \quad TR \gg T1 \quad \rightarrow \quad S_{SE} = \rho \cdot (1 - e^{-\infty}) \cdot e^{0} = \rho$$

The following figure compares a T1w and a T2w MRI image of the lumbar back and lists the relative brightness levels for the main anatomical structures.

Figure 2.19.: T1w and T2w image of the lumbar back [52]

According to the weighting of an MRI image, each tissue type of the human body is represented by a characteristic intensity distribution. For example, in a T1w scan, fat and fat-containing tissue such as bone marrow appears hyperintense or bright [53]. The diagnostic benefit of this property is that T1w images are well-suited for visualizing the anatomical structures of the body. Consequently, they are often created routinely, even if a clear finding has not been established [54]. The reverse is true for T2w images, in which water- and fluid-containing tissue is hyperintense and bright, while fat-containing tissue appears dark [39]. Since damaged tissue tends to develop edema, it has a different water concentration and a different intensity distribution than its surroundings. For the diagnostic use, this means that T2w images are more sensitive for pathology, and generally able to distinguish pathologic tissue from normal tissue. Therefore, T2w images are generated if findings or symptoms indicate a particular disease, such as lower back pain indicating a degenerated intervertebral disc, since it permits a more specific search and a conclusive confirmation of the causes [54].

3. Materials and Methods

In recent years, the relevance of Computer-Aided Diagnosis (CAD) in assisting radiologists in the evaluation of medical images has increased constantly. Improved imaging techniques and imaging quality are mainly responsible for the rapid development and application of new methods of computer-assisted image analysis. CAD is fundamentally based on image processing algorithms to interpret medical images automatically and make better use of them [55]. Among the large variety of image processing techniques, an automatic classification or segmentation of an image into different regions is one of the most critical steps in image understanding and processing [56]. For example in computer aided diagnosis, it is helpful to segment medical images into different tissues so that a radiologist can easily verify a finding and reduce the risk of complications while planning a surgery.

The following section provides an overview of the software and toolkits used and of the main principles in designing a simple segmentation module for 3D Slicer, a CAD software application for computer scientists and clinical researchers. Based on these principles the modules in Chapter 4 are developed.

3.1. Software and Toolkits

3D Slicer, or often simply referred to as Slicer, is a free, open-source software package for medical image visualization and analysis. The application was initiated in 1998 by the Surgical Planning Laboratory at the Brigham and Women's Hospital, Boston, Massachusetts, USA, and the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. Since then, the software has evolved into an end-user application to support a wide range of clinical applications and functionality such as segmentation, registration and three-dimensional visualization of image data [57].

The software infrastructure of Slicer is separated into a base system and a variety of extendable modules which can be included through the "Slicer 3 Execution Model" and get dynamically loaded during the start. This plugin mechanism represents an easy way for developers to add new functionality and to incorporate a command line program as a Slicer module. While the main system is implemented in C++, modules can also be developed using other programming languages like Python or Tcl [57].
The main application is built on two frameworks: ITK, to support image analysis, and VTK, for three-dimensional graphics and visualization [58].

The Insight Toolkit (ITK) was developed by the US National Library of Medicine in 1999 as an open-source, object-oriented software library for analyzing medical images. It provides a variety of image processing algorithms for filtering, segmentation and registration, which can be connected together to design a complex processing pipeline [59]. The ITK library is written in C++ in a generic programming style and uses templates to apply already implemented code and algorithms to other classes and objects. Such templating enables the reuse of code, providing high efficiency and performance both in developing and program execution [60]. Because the main purpose of ITK is image processing, it does not provide visualization methods, which are realized within Slicer by using the Visualization Toolkit (VTK).

VTK was created in a research department of General Electric in 1993 and primarily used for scientific visualizing of three-dimensional images. Since 1998, Kitware Inc., founded by the developers of VTK, distributes the toolkit, which is today the prime visualization system in the world [61]. The core is implemented as a compiled C++ class library and supports an automated wrapping into Python, Java and Tcl, so that VTK applications may also be written using these programming languages [62]. The individual components or modules of VTK provide surface and volume rendering as well as basic image processing techniques. The modules perform algorithmic operations on the data and can be connected to a processing pipeline; within this, an executed module passes its results further to the next module. The pipeline execution is controlled in response to the demands for data (demand-driven) or in response to the user input (event-driven) [63]. Therefore, the different VTK models are flexible and can be quickly adapted to different data types or new algorithmic implementations.

Because the source codes of ITK and VTK are under an open source license, the development of both toolkits is mainly driven by their respective communities and allows developers all over the world to contribute new algorithms and increase the quality and quantity of both toolkits [59, 61].

Applications with a medical context in particular are often built using both toolkits to get the image processing functionality of ITK and the visualization and rendering features of VTK. This can be done through conversion classes to enable access to the ITK library wrapped in VTK classes and to obtain the advantages of both toolkits. The modules in Chapter 4 rely on this combination, which is also demonstrated in the example module of the following section.

3.2. Main principles of developing a Slicer module

By implementing a thresholding algorithm in C++ and using ITK for reading the input and writing the output and VTK to process the thresholding, classes of both toolkits are attached together to a single pipeline to perform a simple segmentation. The thresholding pipeline and the corresponding ITK and VTK classes are shown in the following figure.

Figure 3.1.: A thresholding pipeline using ITK and VTK

The reason that a thresholding operation is implemented in this module is based on the fact that it is the simplest method of segmenting an image [55] and is therefore best suited to explaining the principles of developing a segmentation command line program. A threshold labels all pixels in an image either as “foreground” pixels if their intensity is between two certain values (called lower and upper threshold) or as “background” pixels otherwise. Typically, the foreground is given a value of 1 (illustrated as violet in Figure 3.1) while the background pixels are given a value of 0. The output is a binary image, because it consists of just two values, representing the segmentation [64, 65]. Due to the marking or labeling of the pixels, the term labelmap is also sometimes used for the output.

The module to realizing this segmentation is called `BinaryThresholding` and consists of the following three files, which are also at least part of any other command line program. The complete code of these files is available in the Appendix.

`BinaryThresholding.cxx` The CXX file specifies the functional logic of an application written in the programming language C++. This file contains the complete source code of the algorithm shown in Figure 3.1, and must therefore be modified and checked for errors during the development and debugging process very frequently.

`BinaryThresholding.xml` Slicer modules are described by using an XML file which has two purposes: Firstly, it specifies the parameters that are necessary for the CXX file (in this case for example the input image and the thresholding values) and secondly it constructs the Graphical User Interface (GUI) of the module. Therefore, a developed Slicer module can be used as a standalone executable, controlled by entering the parameters as command lines in a DOS or UNIX terminal, or incorporated into Slicer and controlled via the generated GUI. Figure 3.3 shows the appearance of the `BinaryThresholding` module as a command line module as well as a Slicer module.

`CMakeLists.txt` This is the configuration file to specify the build parameters of the module like the path of the CXX and XML file, and of the ITK and VTK libraries required for the processing pipeline. This file is used by the open-source build manager CMake and defined using the CMake configuration language. However, CMake does not directly build the final software, but instead uses this operating system independent configuration file to generate build files (named `CMakeFiles`) such as makefiles on UNIX or projects/workspaces in Windows Visual Studio. Once the build files are created the module can be compiled easily by native build tools on the corresponding platform.

These three files are the starting point of a Slicer module illustrated also by the figure below. After the `CMakeFiles` are created by CMake (this step must usually be performed once), the executable can be generated by a native compiler and be used as a command line program or, as mentioned above, included into Slicer as a Slicer module. Afterwards, modifications to the CXX and/or the XML file only require a re-compilation to affect the executable and the Slicer module.

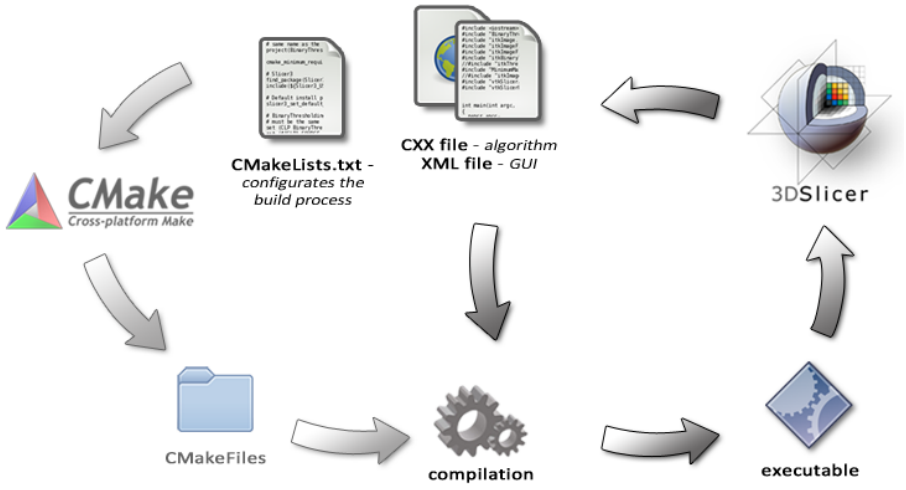


Figure 3.2.: Developing process of creating a Slicer module

In the following, the content of the three files will be explained in more detail, starting with the `CMakeLists.txt` file and continuing with the XML and CXX file.

To configure a project with CMake either to build a command line program or a Slicer module, the name of the project must first be specified in the `CMakeLists.txt` file, in this case `BinaryThresholding`.

```
1 project(BinaryThresholding)
```

Secondly, it is necessary to include the Slicer libraries for the application. In a typical configuration, it is hardly possible to specify in which location the Slicer header files are and which libraries are required. The location of the header files and libraries varies from install to install and from platform to platform. However, there are usually some standard places to look and CMake automates this search using the following macros:

```
5 find_package(Slicer3 REQUIRED)
6 include(${Slicer3_USE_FILE})
```

These macros detect the location of the Slicer header files and libraries and set up the most common libraries that are linked to in a typical Slicer project. If CMake cannot find the header files, the path to the Slicer-build directory must be specified manually during the configuration and generating process of the project (as it can be seen in the `Slicer3_DIR` value of Figure 3.6).

The following block sets a variable CPL, which stands for Command Line Processing, to the name of the project and, the location of the source code to the `BinaryThresholding.cxx` file. The `generateclp` command finally generates the header file `BinaryThresholdingCLP.h`, which contains forward declarations of variables and parameters specified in the XML file.

```
10 set (CLP BinaryThresholding)
11 set (${CLP}_SOURCE ${CLP}.cxx)
12 generateclp(${CLP}_SOURCE ${CLP}.xml)
```

In addition, CMake needs to know the type of the module to be built from the source code, like an executable and/or a library. Based on the file extension of the module type, CMake automatically figures out the correspondent compiler on the native platform. Here an executable will be created, named `BinaryThresholding.exe`.

```
14 add_executable(${CLP} ${${CLP}_SOURCE})
```

According to the Slicer header files, specified above, this application also requires several ITK and VTK classes for the intended pipeline. Before adding these classes to the `CMakeLists.txt` file, the term library should be briefly discussed. Both toolkits are organized in so-called libraries, which can be thought of as a folder containing different classes and algorithms. If a particular class is needed in the CXX file, the corresponding ITK or VTK library must be specified in the `CMakeLists.txt` file. For example to use the `vtkImageThreshold` class in the source code, the library `vtkImaging` must be linked against them. All ITK and VTK libraries could be specified here, as being of any concern during the implementation, but this example intends to provide a minimum of code to realize the shown pipeline.

```
17 target_link_libraries (${CLP}
18   vtkImaging       #vtkImageThreshold.h
19   ITKIO            #itkImageFileReader.h, itkImageFileWriter.h
20   ITKCommon        #itkOrientedImage.h
21   ITKBasicFilters  #itkChangeInformationImageFilter.h
22  )
```

The terminology of the XML file is based on the XML 1.0 Specification produced by the World Wide Web Consortium (WC3), and uses markups, tags and attributes to describe the module and its parameters. The complete XML file of this project is listed in the Appendix, and here only some important principles are explained.

The string characters which make up an XML document are divided into markups, usually the characters `<` and `>`, as well as content, which is surrounded by a start-tag, such as `<name>`, and an end-tag, like `</name>`. Tags that can be used for a Slicer model are defined and interpreted by the Slicer 3 Execution Model and are either required or optional.

An XML document generally starts with the declaration of some information about itself, like the XML version and the encoding type of the Unicode characters.

```
1 <?xml version="1.0" encoding="utf-8"?>
```

Furthermore, the XML file is separated into two parts. The first part contains a description of the module and requires at least the tags `<title>` and `<description>`.

```
4   <title>BinaryThresholding</title>
5   <description>ITK reading and writing, processing as VTK</description>
```

The second part consists of one or more groups to specify the module parameters (or arguments). All parameters could actually be packed into a single group, but for clarity it is useful to distinguish between input/output parameters and module parameters, which affect the processing. Parameter groups require a `<label>` for naming the group and a `<description>`, which appears as a balloon tooltip in the GUI.

```
7   <parameters>
8     <label>Input and Output</label>
9     <description>The input and output parameters</description>
      ...
24  </parameters>

26  <parameters>
27    <label>Binary Thresholding Parameters</label>
28    <description>The parameters of the binary threshold</description>
      ...
43  </parameters>
```

The type of the parameters varies from scalar types (`<integer>`, `<float>`, etc.) to `<boolean>` and `<image>`. The `<image>` tag is a special parameter type that indicates that a file name has to be entered to specify an image. Whether the image is an input or an output parameter is set by the `<channel>` tag. The optional attribute `type="label"` for an image means, that the output image should be a segmentation or a label/labelmap. The following lines show the definition of the output image.

```
18    <image type="label">
19      <name>MyOutputVolume</name>
20      <index>1</index>
21      <label>Output Volume</label>
22      <channel>output</channel>
23    </image>
```

All parameters require a unique `<name>` tag by which the parameter can be addressed in the CXX file. This means that the C++ variable of the output image is `MyOutputVolume`. While the `<label>` has the same purpose as above and affects only the name of the parameter in the GUI, the `<index>` tag indicates an integer value starting at zero and specifies the order of the arguments in the command line module. In this project a total of four parameters are used,

named `MyInputVolume`, `MyOutputVolume`, `MyLowerThreshold` and `MyUpperThreshold`, with the corresponding index numbers 0 to 3.

The following figure shows the call of the resulting module described by the XML file as a command line module in the terminal and incorporated as a Slicer module.

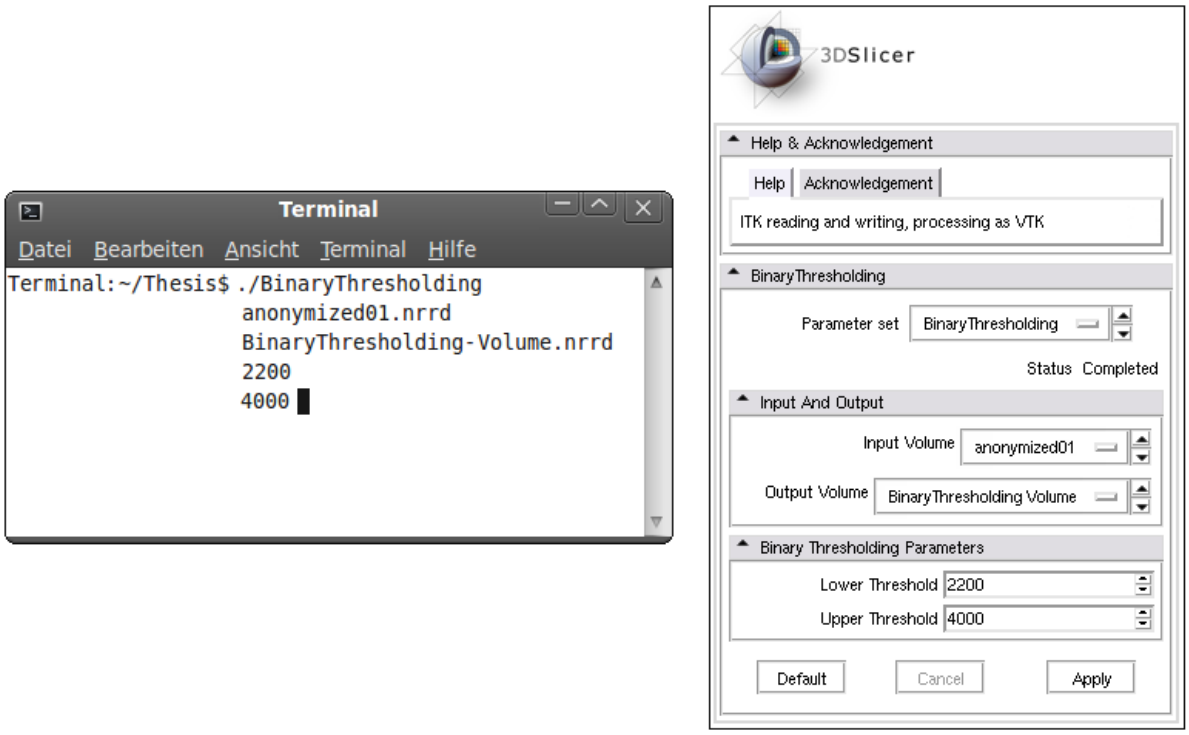


Figure 3.3.: `BinaryThresholding` as a command line module and as the corresponding GUI in Slicer

The complete algorithmic source code of the project is located in the CXX file, where first the `#include` statements of the necessary ITK and VTK header files must be specified.

```
 2 #include "BinaryThresholdingCLP.h"
 3 // ITK classes
 4 #include "itkOrientedImage.h"
 5 #include "itkImageFileReader.h"
 6 #include "itkImageFileWriter.h"
 7 #include "itkChangeInformationImageFilter.h"
 8 // VTK classes
 9 #include "vtkImageThreshold.h"
10 // converter classes
11 #include "itkImageToVTKImageFilter.h"
12 #include "itkVTKImageToImageFilter.h"
```

The first line in the main procedure has to be the `PARSE_ARGS` statement.

```
14 int main(int argc, char * argv [])
15 {
16   PARSE_ARGS;
       ...
64 }
```

As mentioned in the `CMakeList.txt` file, the command `generateclp` generates the header file `BinaryThresholdingCLP.h` by reading the XML description. If this header is included in the CXX file (as at line 2) and the `PARSE_ARGS` macro is specified, the defined model parameters (e.g. `MyInputVolume`, `MyLowerThreshold`) can be accessed in the source code.

Section 3.1 showed, that the programming style of ITK differs from VTK. While ITK uses the techniques of generic programming in its implementation to create an object, VTK waives this kind of templating. The differences can be comprehended in the following code snippets:

```
22  typedef itk::ImageFileReader<OrientedImageType> ReaderType;
23  ReaderType::Pointer reader = ReaderType::New();
      ...
34  vtkImageThreshold *imageThreshold = vtkImageThreshold::New();
```

The definition of an object in ITK, for example an image reader, must be performed by instantiating the `ImageFileReader` class with the `typedef` statement. This creates an instance of the `ImageFileReader` class named `ReaderType`. The concrete object `reader` is afterwards created by invoking the `New()` operator from the corresponding class and assigning the result to the pointer class of the `ReaderType`.
In contrast, the VTK object `imageThreshold` is defined directly with the call of the `New()` operator, which allocates memory and returns a pointer to the newly created object.

Once the necessary objects are created, the data can be passed from the output of one object to the input of the next by using the operators `SetInput` and `GetOutput`. Thus, the different objects which all perform simple operations can be attached together to a single pipeline and to a more complex task. The passing of the input image, defined in the XML file as `MyInputVolume`, from the reader to the writer though the ITK to VTK conversion, the thresholding operation and the conversion back is performed by the following code lines:

```
24  reader->SetFileName(MyInputVolume.c_str());
      ...
30  convertITKtoVTK->SetInput(reader->GetOutput());
      ...
35  imageThreshold->SetInput(convertITKtoVTK->GetOutput());
      ...
44  convertVTKtoITK->SetInput(imageThreshold->GetOutput());
      ...
51  changeInformationFilter->SetInput(convertVTKtoITK->GetOutput());
      ...
60  writer->SetFileName(MyOutputVolume.c_str());
```

The module parameters `MyLowerThreshold` and `MyUpperThreshold`, which have to be entered by the user and affect the processing, are used by the thresholding method `ThresholdBetween`.

```
36  imageThreshold->ThresholdBetween(MyLowerThreshold, MyUpperThreshold);
```

The pixels of the image with an intensity in the range between these two values are labeled with `InValue` 1, or otherwise with `OutValue` 0, set in the next code lines.

```
37  imageThreshold->SetInValue(1);
38  imageThreshold->SetOutValue(0);
```

The call of the `Update()` operator finally performs the thresholding operation and creates the output, which can be used as an input of the following object.

```
39  imageThreshold->Update();
```

A very important point to note regarding the convention between ITK, VTK and back is that the individual toolkits use different anatomical coordinate systems (also known as patient coordinate systems). In general this system describes a continuous three-dimensional space with respect to the anatomical position of a human and consists of three different planes:

- An axial plane which is parallel to the ground and separates the head (superior) from the feet (inferior)
- A coronal plane which is perpendicular to the ground and separates the front (anterior) from the back (posterior)
- A sagittal plane which separates the left from the right

Additionally to these anatomical planes (illustrated by Figure 3.4) it is specified that all axes have their notation in a positive direction, meaning that for example the negative superior axis is represented by the inferior axis.

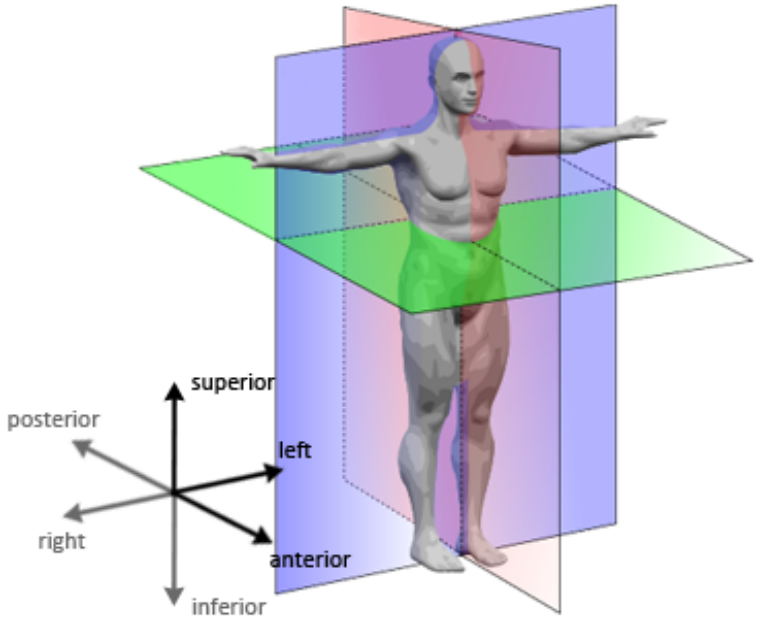


Figure 3.4.: Anatomical coordinate system

However, the order of the axis directions is not defined, which results in different specifications to describe the anatomical position of a human. The specification used by ITK is the LPS (left, posterior, superior) direction system and defined as follows:

$$LPS = \left\{ \begin{array}{c} \text{from right towards left} \\ \text{from anterior towards posterior} \\ \text{from inferior towards superior} \end{array} \right\}$$

On the other hand, VTK uses RAS (right, anterior, superior) directions, which is quite similar to LPS but has the first two axes flipped.

$$RAS = \left\{ \begin{array}{c} \text{from left towards right} \\ \text{from posterior towards anterior} \\ \text{from inferior towards superior} \end{array} \right\}$$

Both systems are consistent, equally useful and logical. Unfortunately, the conversion from ITK to VTK and back ignores that the direction system has changed and results in a flipped output.

Therefore, an `itkChangeInformationImageFilter` class has to be used after the conversion back to ITK to set the direction to the same value as of the input image.

```
53   changeInformationFilter->SetOutputDirection(reader->GetOutput()->GetDirection());
```

If this step is not performed, the output, in this case the segmentation, would be flipped and would not fit to the input image, as shown by the figure below.

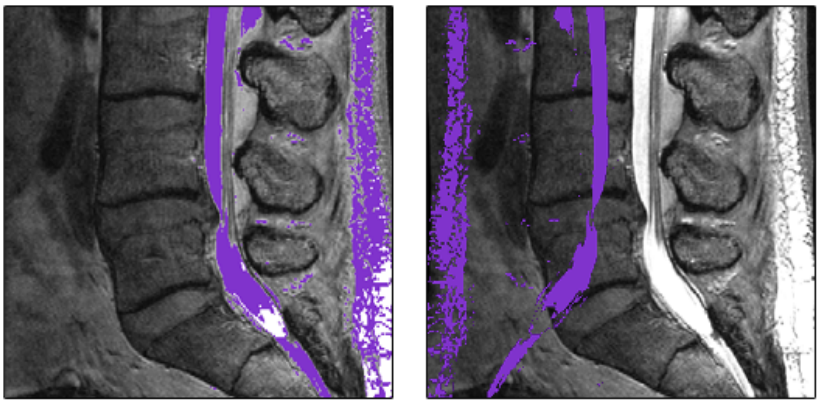


Figure 3.5.: Segmentation with and without a reset of the direction

The storage of the thresholded image with the changed direction is performed by the `itkImageFileWriter` class, where first the output of the `changeInformationFilter` is set to the input of the writer, second the filename is set to the labelmap `MyOutputVolume` defined in the XML file and third the `Write()` operator is used to handle the release of the output.

```
59   writer->SetInput(changeInformationFilter->GetOutput());
60   writer->SetFileName(MyOutputVolume.c_str());
61   writer->Write();
```

If at least the XML and the `CMakeLists.txt` files are implemented, the CXX file must be created but not yet written completely, the module can be generated by using CMake. This requires only the specification of the directory where the `CMakeLists.txt` file is located and where the resulting binaries should be saved.

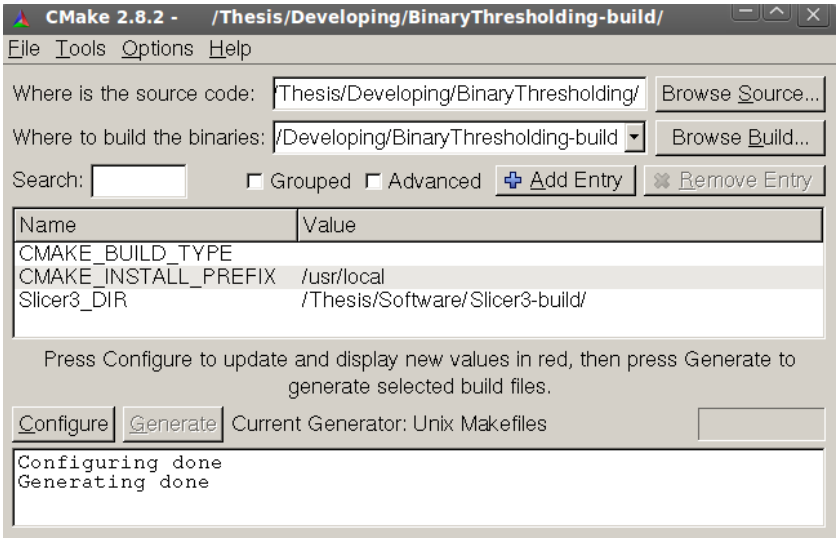


Figure 3.6.: CMake configuration and generating of the `BinaryThresholding` project

Afterwards, the executable can be compiled by accessing the path to the binaries and entering the UNIX-command `make` in the terminal. The compilation output summarizes that a header file `BinaryThresholdingCLP.h` is generated, that two parameter groups are created, consisting of the input and output image as well as of the two thresholding parameters, and that an executable `BinaryThresholding.exe` is built.

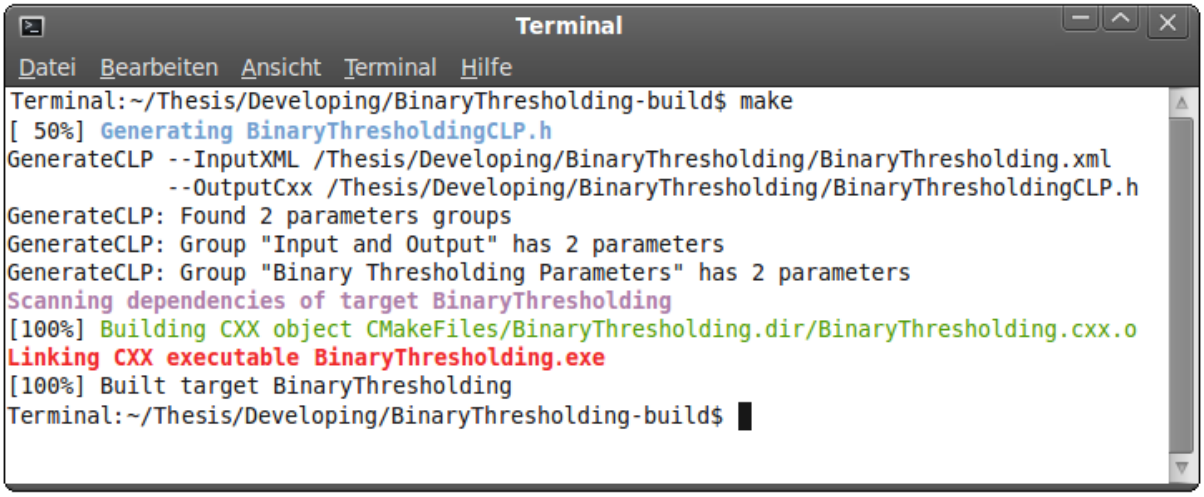


Figure 3.7.: Compilation of the `BinaryThresholding` project

Any changes to one or more of the three files only requires a re-compilation of the executable by re-entering the `make` command.

The execution of the `BinaryThresholding` module, either as a Slicer module or as a command line module, finally creates the image segmentation, illustrated by the following figure with dataset *anonymized01.nrrd* and a thresholding range between 2200 and 4000.

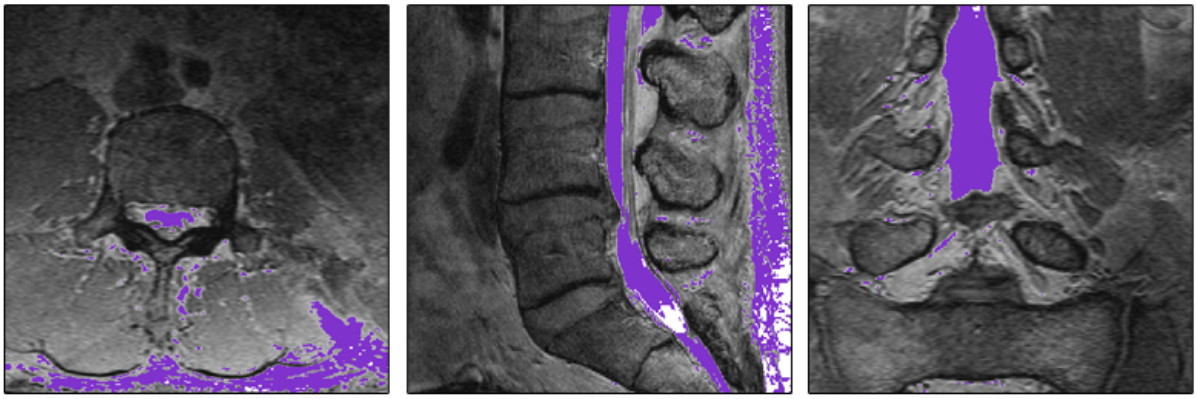


Figure 3.8.: Result of the `BinaryThresholding` module

3.3. Datasets

A total of four datasets are available for segmenting the CSF and validating the developed modules. These datasets are T2w MRI images and have been acquired at the Brigham and Women's Hospital, Boston, USA, in 2009, visualizing the vertebra, discs and CSF of the lumbar region. The registered patients, three males and one female, were aged between 32 and 59 years and had in three cases a bulged intervertebral disc.

The image parameters are largely similar in all datasets which all have a high axial and sagittal resolution. This high resolution is a result of the image dimensions and spacing. While the image dimensions correspondent to the number of slices recorded in the axial, sagittal and coronal anatomical plane, the spacing specifies the distance between two pixels. The smaller the spacing is, the higher is the image resolution and more small structures can be detected on the recordings. Usually, MRI images are taken with a spacing of 0.5 mm, but in these cases a slice thickness of less than 0.3 mm was used.
Out of the spacing and the image dimensions, the actual length in millimeters can be calculated.

The following table shows a detailed listing of all four datasets including the patient's gender, finding, image dimensions, spacing, length, and pixel intensity range.

anonymized01.nrrd

Gender	Male	*Dimensions*	$512 \times 512 \times 192$
Finding	Disc herniation	*Spacing*	$0.27 \times 0.27 \times 0.7$
		Length	$138.2 \times 138.2 \times 134.4$
		Intensity range	$0 \ldots 13090$

anonymized02.nrrd

Gender	Male	*Dimensions*	$512 \times 512 \times 172$
Finding	No herniation	*Spacing*	$0.27 \times 0.27 \times 0.7$
		Length	$138.2 \times 138.2 \times 120.4$
		Intensity range	$0 \ldots 14726$

anonymized03.nrrd

Gender	Male	*Dimensions*	$512 \times 512 \times 180$
Finding	Disc herniation	*Spacing*	$0.43 \times 0.43 \times 0.7$
		Length	$220.1 \times 220.1 \times 126.0$
		Intensity range	$0 \ldots 10577$

anonymized04.nrrd

Gender	Male	*Dimensions*	$512 \times 512 \times 80$
Finding	Disc herniation	*Spacing*	$0.27 \times 0.27 \times 1.4$
		Length	$138.2 \times 138.2 \times 112.0$
		Intensity range	$0 \ldots 10485$

4. Results

The four datasets described in Section 3.3 have the characteristics that the CSF is the most recognizable and brightest component within the vertebral column. Although the nerve fibers are primarily affected by a herniated or a bulged intervertebral disc and cause most of the symptoms of lower back pain, they constitute just a tiny, less visible structure on the datasets. Therefore, they are hardly possible to separate from the CSF, vertebra and discs, even by a radiologist. The only way to visualize the damaged disc is to segment the CSF, since a herniated disc always leads to a displacement of the fluid in the subarachnoid space.

There are several algorithmic possibilities to realize a registration and segmentation of the CSF. Fundamentally, a fully-automatic and a semi-automatic approach can be differentiated between. While a fully-automatic algorithm has to scan and analyze the dataset first before performing the segmentation, the user can interact with the semi-automatic method for example by setting a seed point, which often results in a less time-consuming execution.

In this chapter both approaches are presented with the appropriate algorithmic steps first and afterwards applied to the four T2w MRI datasets, which results can be found at the end of each algorithmic description.

4.1. Fully-Automatic Algorithm

The fully-automated algorithm is divided into two main parts, of which the greater part is the automatic detection of the CSF, also referred to as region of interest (ROI). Once the ROI is found on the dataset, the CSF can be separated from its surrounding structures and stored as a labelmap. The detection part is based on the observations and remarks in Section 2.2 that on T2w MRI images the CSF appears very bright and the other tissues less bright in the case of fat or even dark in bones and intervertebral discs. In addition, the CSF, or rather the subarachnoid space where the CSF is located, has a tubular structure, which extends approximately perpendicular in the lumbar region between the first lumbar and first sacral vertebra. The following figure shows the classification of the ROI where the bright CSF is labeled as green and its dark surroundings as red.

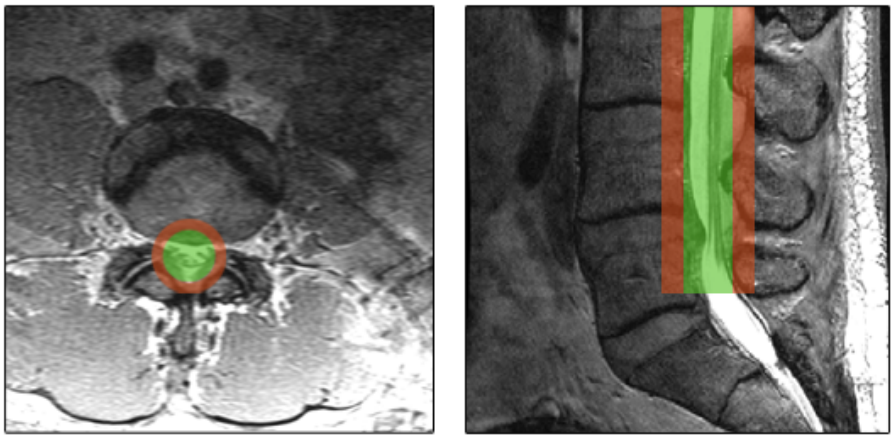


Figure 4.1.: Classification of the ROI

If these two facts, a perpendicular and tubular ROI, are taken together, the CSF can be interpreted mathematically as a cylinder with a diameter in the same range as the extents of the CSF. In analogy, the dark tissue around the CSF can also be modeled as a cylinder, or rather as a hollow cylinder, which surrounds the CSF-representing cylinder in its middle. Due to the facts that fluid and fat differ only slightly from each other in their signal intensity on T2w images and that fat is the closest tissue to the CSF, the two cylinders cannot be directly adjacent to each other. Therefore, a gap between both cylinders is required, realized by setting the radius of the cylinder in the middle to a smaller value than the inner radius of the hollow cylinder. This results in the consequence, that the area between both cylinders can be ignored during the automatic detection process, because it cannot be classified with certainty as belonging to the CSF or to the dark area around it. An exploration where different cylinder parameters were examined has shown which radius, gap and height are best suited to detect the CSF. These parameters are listed in the figure below, which also illustrates the arrangement of the two cylinders.

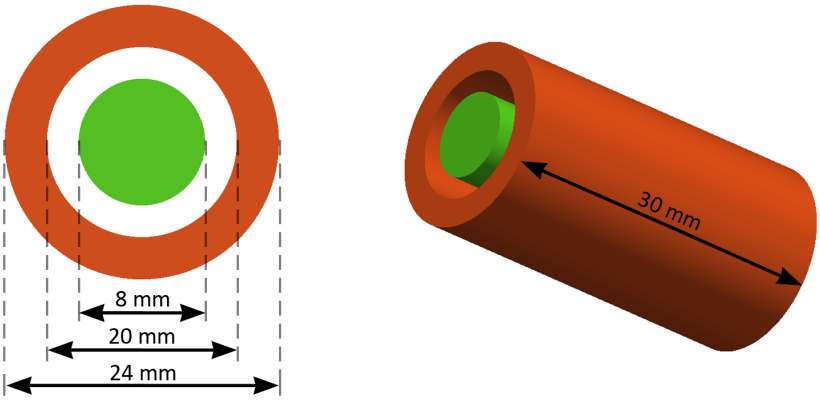


Figure 4.2.: Cylinder-based model of the CSF and its surrounding area

However, it should be noticed that the cylinders are not geometric shapes as shown in Figure 4.2, but are composed by a collection of points which are arranged cylindrically. These points are generated with respect to the spacing of the dataset, meaning that the distance between two points is the same as the distance between two pixels, or rather between two voxels, because of the three-dimensionality of the cylinders and datasets. Consequently, every point of a cylinder is located within exactly one voxel. For dataset *ananymized01.nrrd* for example the inner cylinder consists of 45120 points and the outer of 141120 points, so that 45120 voxels are inside the inner cylinder and 141120 voxels inside the outer cylinder. In the figure below the actual appearance of both cylinders is shown.

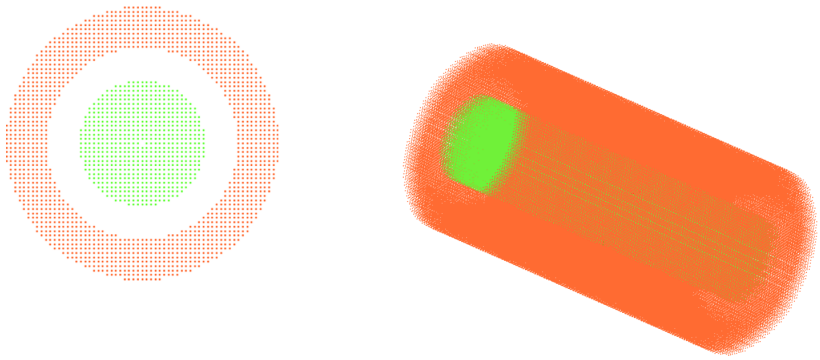


Figure 4.3.: Cylinder-based model consisting of a set of points

The main idea of this fully-automatic algorithm is to move both cylinders simultaneously over the dataset and to analyze the voxels which are located within the inner cylinder and within the outer hollow cylinder. The area of the CSF has the property that the analyzed voxels inside the inner cylinders are very bright, while the voxels around the CSF, and therefore within the hollow cylinder, are dark. In all options to place both cylinders on the dataset, there is only one in which the voxel intensity between the inner and outer cylinder differs largest from each other. In this case, the position of the inner cylinder corresponds to the position of the CSF.

As illustrated by the following figure the first pair of cylinders is placed in the top left corner of the dataset so that the left-most point and the upper-most point of the outer cylinder touches the left and upper border of the dataset.

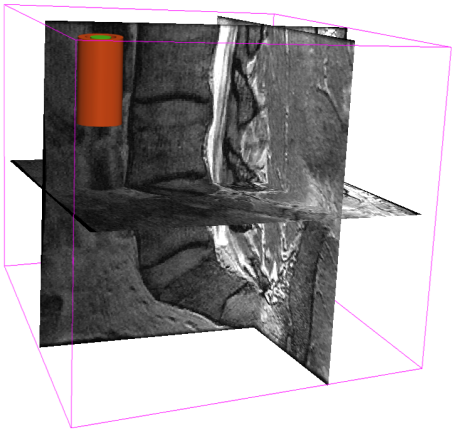


Figure 4.4.: Start position of the cylinder pair

The movement of the cylinders over the dataset is dependent on a parameter, called step size, which defines the distance between two pairs of cylinders. To get a continuous movement the step size in three directions must have the same value as the spacing of the corresponding anatomic plane. This results in a voxel by voxel movement from the top left corner to the bottom right corner. Such a step size would dramatically increase the number of options to place both cylinders on the dataset and therefore the computation time, but it would also increase the probability that the inner cylinder fits exactly to the position of the CSF.
For example, dataset *ananymized01.nrrd* has a dimension of $512 \times 512 \times 192$ pixels and a spacing of 0.27, 0.27, 0.7 millimeter per pixel. Therefore the lengths in millimeters along the axial/sagittal plane and along the coronal plane are:

$$512\,\text{px} \cdot 0.27\,\frac{\text{mm}}{\text{px}} = 138.2\,\text{mm} \qquad 192\,\text{px} \cdot 0.7\,\frac{\text{mm}}{\text{px}} = 134.4\,\text{mm}$$

If each pair of cylinders with a diameter of 24 mm and a height of 30 mm is placed tightly together, whereby the end of one pair matches to the beginning of the next pair, 5 pairs are possible to the left, 5 pairs in a posterior direction and 4 pairs in an interior direction, which results in a total of 100 pairs. In this case the step size would be 24 mm, 24 mm, 30 mm. The mathematical calculation is the following, where the operator $\lfloor x \rfloor$ is called a floor function to map a real number to the largest previous integer.

$$\left\lfloor \frac{138.2\,\text{mm}}{24\,\text{mm}} \right\rfloor \cdot \left\lfloor \frac{138.2\,\text{mm}}{24\,\text{mm}} \right\rfloor \cdot \left\lfloor \frac{134.4\,\text{mm}}{30\,\text{mm}} \right\rfloor = 100$$

A continuous movement however would mean a step size of 0.27 mm, 0.27 mm, 0.7 mm, and leads to

$$\left\lfloor \frac{138.2\,\text{mm} - 24\,\text{mm}}{0.27\,\text{mm}} \right\rfloor \cdot \left\lfloor \frac{138.2\,\text{mm} - 24\,\text{mm}}{0.27\,\text{mm}} \right\rfloor \cdot \left\lfloor \frac{134.4\,\text{mm} - 30\,\text{mm}}{0.7\,\text{mm}} \right\rfloor = 26\,534\,516$$

options to place both cylinders on the dataset.

The following figure illustrates the cylinder movement, but for clarity only in the axial plane with a step size of 24 mm on the left and a continuous step size of 0.27 mm on the right.

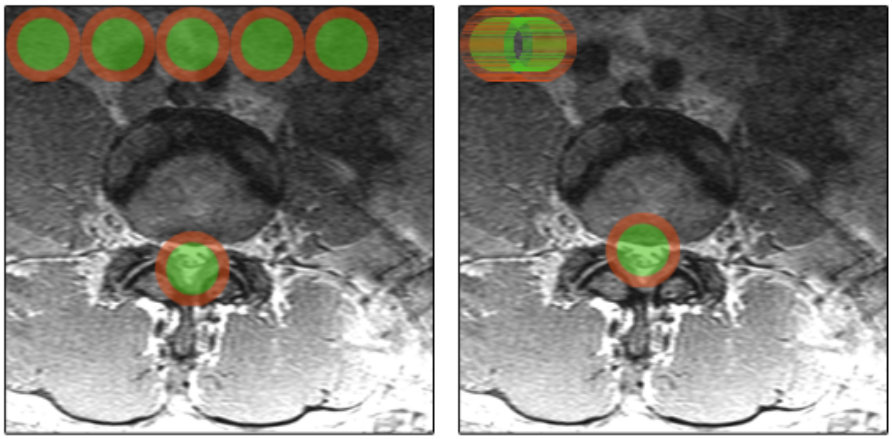


Figure 4.5.: Cylinder movement with different step sizes

The best match of the cylinders in Figure 4.5 shows that for a large step size the CSF is not completely covered by the inner cylinder. Therefore, there could be another pair of cylinders where the voxel intensity between the inner and outer cylinder differs larger from each other than at those cylinders near the CSF. The computation would fail, because of a too large step size and the outcome would be located at a different position than the CSF really is.

The step size of the cylinder movement is the most crucial part of the fully-automatic algorithm and determines whether the ROI is detected or not. During the same parameter exploration which was mentioned above, different step sizes were examined with the result that a cylinder movement between 5 and 15 mm nearly always detects the ROI and produces a result in an acceptable computation time of 50 seconds to two minutes. If the step size is less than 5 mm, for example has the same value as the spacing of the dataset, the ROI will definitely be detected, but would lead to a computation time within the range of hours. Nevertheless, a large step size could also result in a good detection by coincidence.

Before the dataset is scanned by the cylinders, the intensity values of the voxels are normalized to a range between 0 and 255. The initial intensity range of dataset *ananymized01.nrrd* is between 0 and 13090, where the highest intensity values are not in the region of the CSF, but rather at the adipose tissue of the skin and therefore at the border of the dataset. If only the area of the CSF is considered, the maximum intensity is approximately 8500. A normalization process reduces the high intensity values of the skin while the contrast between the CSF and its surroundings only slightly degrades. The figure below compares dataset *ananymized01.nrrd* with its normalized version, on which the whole computation is performed.

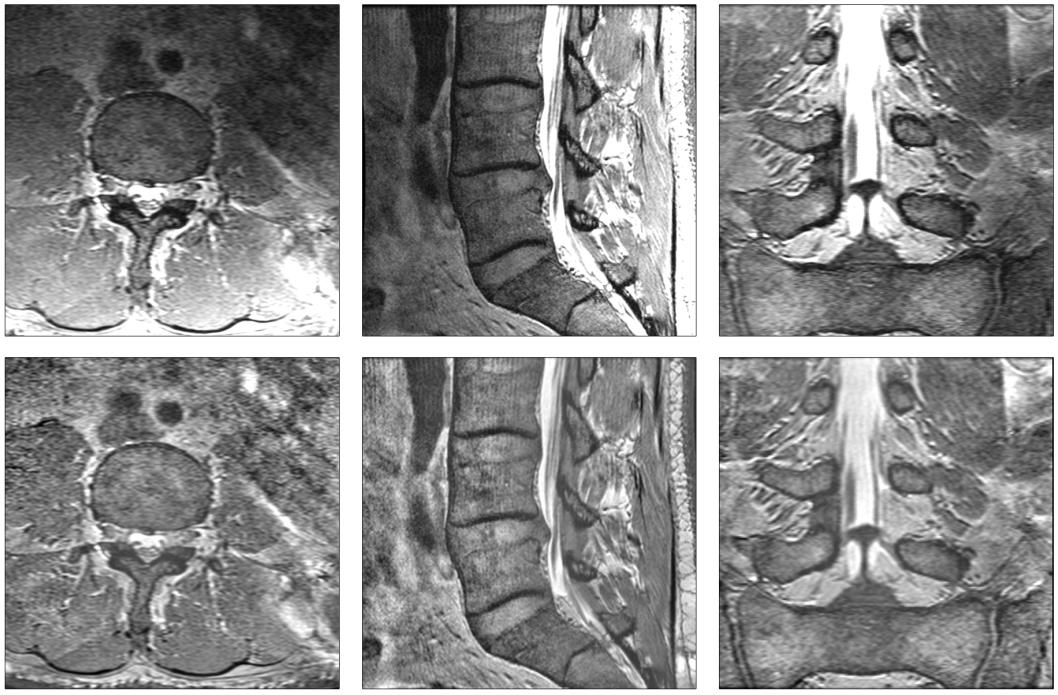


Figure 4.6.: Dataset *ananymized01.nrrd* before and after the normalization of the intensity values

In the following the algorithmic steps of the computation with a step size of 18 mm, 18 mm, 30 mm are described. This step size results in a total of 144 and 36 different options to place a pair of cylinders on the dataset or on an axial plane, respectively. During the movement of the cylinders the corresponding voxels, which are either located within the inner cylinder or within the outer cylinder, are collected and sorted according to their intensity values. This representation builds up an image histogram on which the number of voxels is mapped to their intensities. The following figure illustrates the image histogram of the inner cylinder of pair number 47, meaning that this pair has already moved 30 mm in an inferior direction and is now in the second row and fifth column of the axial planes between 30 and 60 mm.

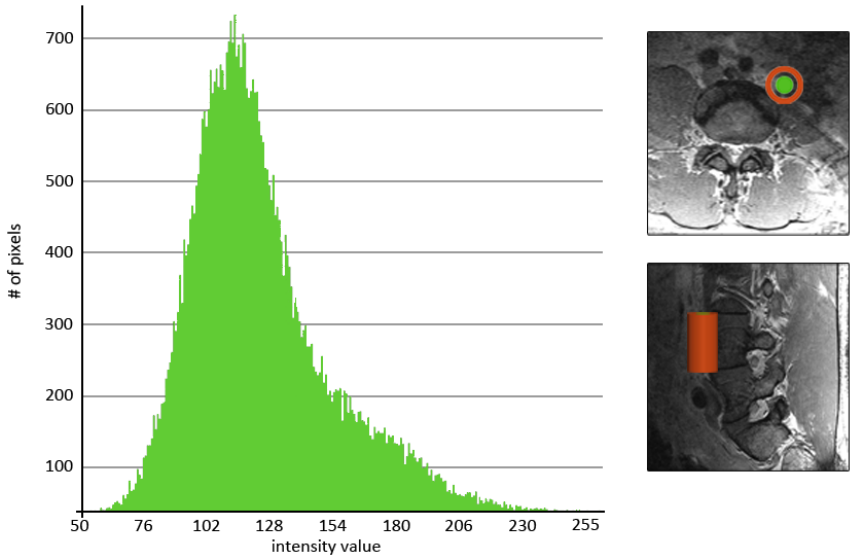


Figure 4.7.: Image histogram of the inner cylinder of pair number 47 and its position on the dataset

To determine the largest difference between the voxel intensity of the inner and outer cylinder, the maximum intensities of the inner cylinder and the minimum intensities of the outer cylinder are of interest. For the inner cylinder this would be 253, but as only one voxel may have this intensity, this value would not be representative of the voxels analyzed by the inner cylinder. Therefore, the extreme values are removed by calculating the cumulative histogram out of the image histogram and detecting the intensity value, whose cumulative voxel number is higher than 90 percent of the total voxel number. In analogy to the inner cylinder, for the outer cylinder the intensity value is determined whose cumulative number is less than 10 percent of the voxel number. This means that the image histograms of both cylinders are trimmed by 10 percent to avoid extreme values, either from the right in the case of inner cylinders or from the left for outer cylinders. Consequently, the detected maximum intensity value is actually the 90 percent intensity value and the minimum intensity value the 10 percent intensity value.
The cumulative histogram of the inner cylinder of pair number 47 is shown below as well as the 10 and 90 percent limit values of the trimmed histogram. As mentioned above, the inner cylinder on dataset *ananymized01.nrrd* consists of 45120 points and therefore the intensity values of 45120 voxels are detected. This number can also be traced in the last bin of the cumulative histogram. In addition, it should be noted that this figure illustrates both limit values, even though only the upper limit is calculated for the inner cylinder (here it would be 167) and only the lower limit for the outer cylinder.

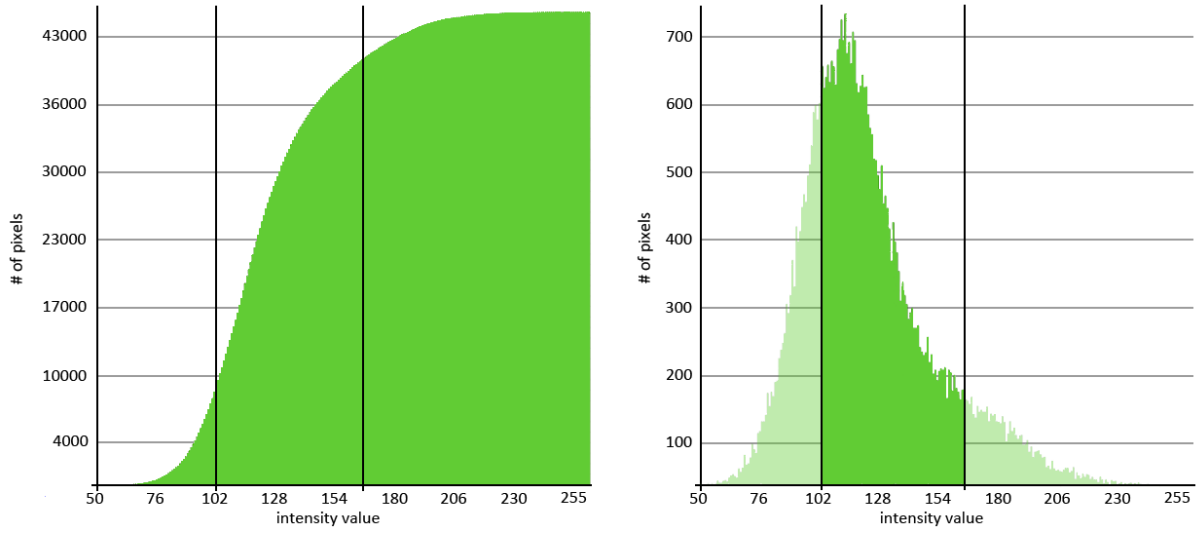


Figure 4.8.: Cumulative and trimmed image histogram of the inner cylinder of pair number 47

The calculation of the image histogram and cumulative histogram is performed by all pairs of cylinders, for both the inner and the outer cylinder, to obtain the corresponding maximum or minimum intensity values. These values (167 for the inner cylinder and 95 for the outer – not illustrated – cylinder) are stored in two newly created three-dimensional images, named maxIn and minOut, which represent the maximum intensities detected by the inner cylinders and the minimum intensities detected by the outer cylinders. Both images have the same dimensions as options to place the cylinders on the dataset and the same spacing as the step size. Therefore, each voxel of one of the two images is associated with the corresponding cylinder and allows the inferring to its position on the dataset.

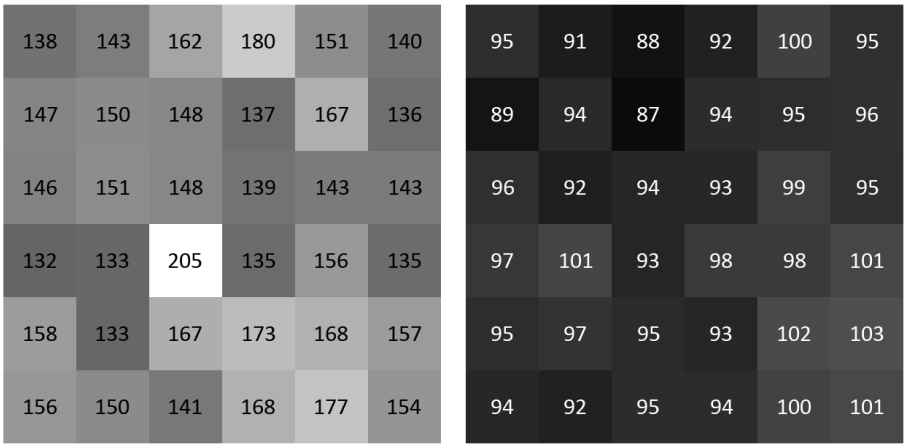


Figure 4.9.: Maximum and minimum intensity image

On the maximum intensity image of Figure 4.9 the position of the bright CSF can already be spotted. The algorithm performs this step by comparing the intensity values of both images voxel by voxel and calculating the difference between the maxIn and minOut image. Those voxels where the difference is largest, here between 205 and 93, finally corresponds to the position of that cylindrical pair, which is located within the CSF.

Since there is the possibility that the cylinders are a bit shifted and do not overlap the complete CSF, a cuboid is created with a length and depth of 30 mm and a height of 110 mm around the pair of cylinders which has the highest intensity difference. The dimensions of the rectangular box ensure that the ROI is completely covered even if the step size of the cylinders coincidentally means that only half of the CSF is inside the inner cylinder. In this case, two cylinder pairs would have a high intensity difference, but only the position of the highest difference would be marked as CSF. By increasing the dimensions of the detected location, the nearby pair is also included within the rectangular box. The generated boundary box around the detected ROI is show in the figure below.

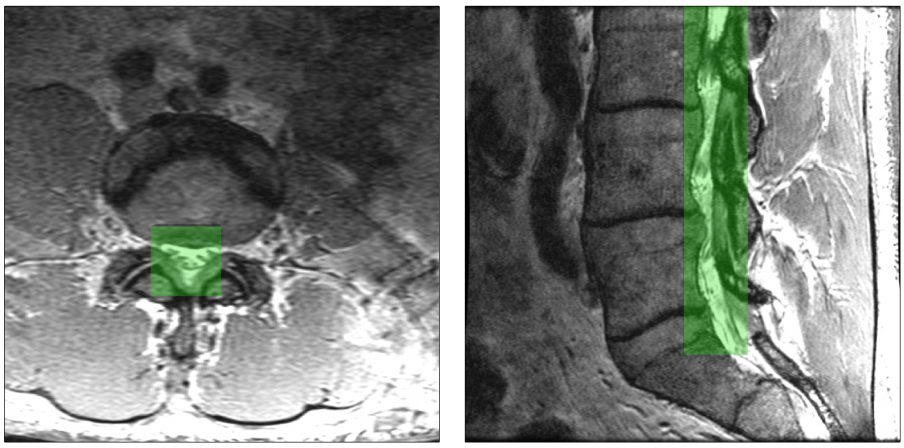


Figure 4.10.: Boundary box around the detected ROI

In order to segment the ROI out of the boundary box, an image histogram is generated from the corresponding voxels. This histogram consists of two large clusters of bright and dark intensities, which are originated either from the CSF or from the vertebrae and discs, which are also covered by the boundary box. This condition appears in the histogram as two hills, one on the left, dark side and the other on the right, bright side. Between both histogram hills is a valley, since most of the voxels belong to one of the two hills. Therefore, the intensity value where the valley is located can be used as a threshold to separate the hill on the right side, representing the CSF,

from the dark tissues on the left. The algorithmic implementation to realize this task, is to start at the mean intensity of the voxels within the boundary box and to detect those two peaks which have the largest voxel numbers on the left and right side. In a second step, the smallest voxel number between both peaks is detected, and the corresponding intensity value used for an image thresholding operation. In the following figure the image histogram of the boundary box is illustrated as well as the detected threshold value.

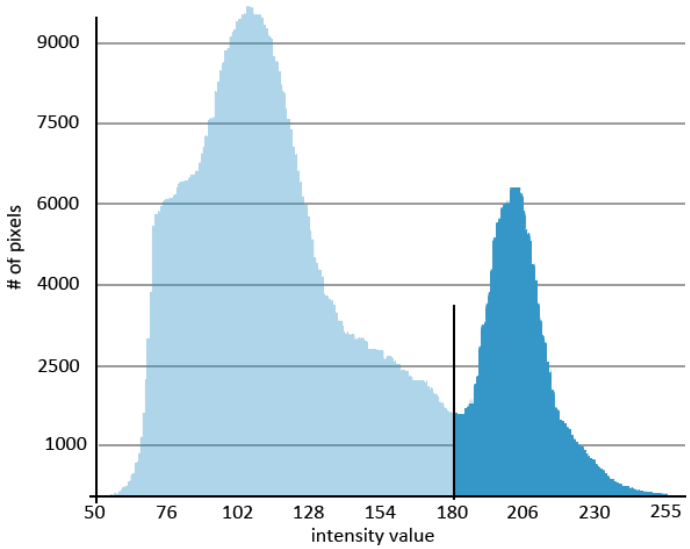


Figure 4.11.: Image histogram of the boundary box and threshold value

The output of the threshold is a labelmap on which the ROI has a value of 1 and the background voxels a value of 0. This labelmap represents the segmentation of the CSF and is therefore the first generated result of the fully-automated algorithm. Furthermore, a three-dimensional polygonal mesh of the segmentation is created by using the marching cubes algorithm of the VTK class library. On those datasets which were registered from patients with a bulged intervertebral disc, the model permits the easy recognition of the pathological location and can thus be used for planning further surgical interventions.

In total, the presented fully-automatic algorithm generates two outcomes, a labelmap which appears superimposed on the dataset within the slice view of Slicer, and a three-dimensional model whose display can be modified for example by rotating and clipping.

The application of the fully-automated algorithm to the four T2w MRI datasets of Section 3.3 and the corresponding segmentation results as well as the three-dimensional models are illustrated on the following pages in detail. It should be noted that the pictures at the top and bottom of these pages represent the appearance of the two outcomes in the slice view and in the three-dimensional view of Slicer. The axial images in the middle of the figures were created by zooming in the dataset and can be used to visually determine the accuracy of the segmentation.

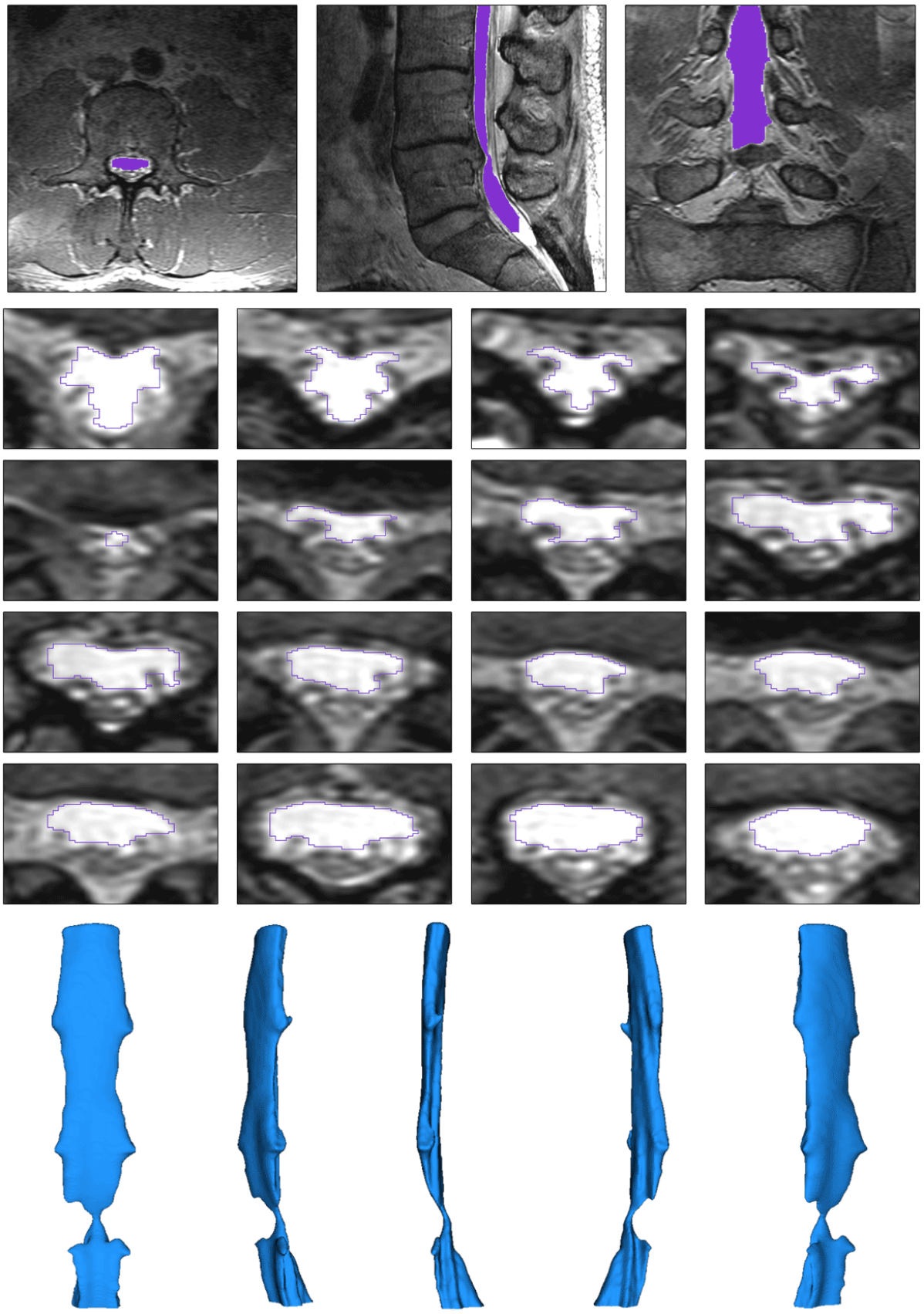


Figure 4.12.: Fully-automatic segmentation result of dataset *anonymized01.nrrd*

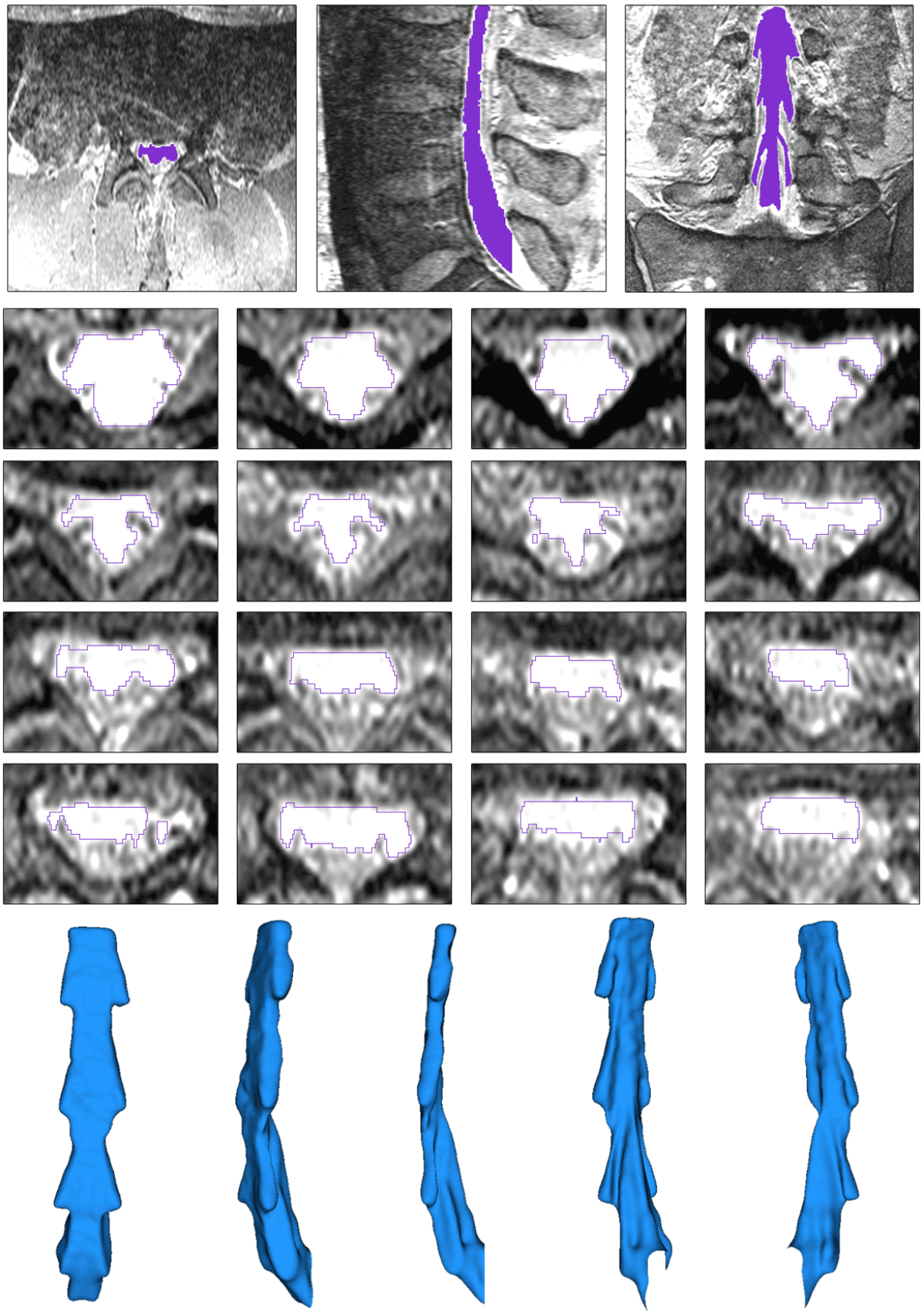


Figure 4.13.: Fully-automatic segmentation result of dataset *anonymized02.nrrd*

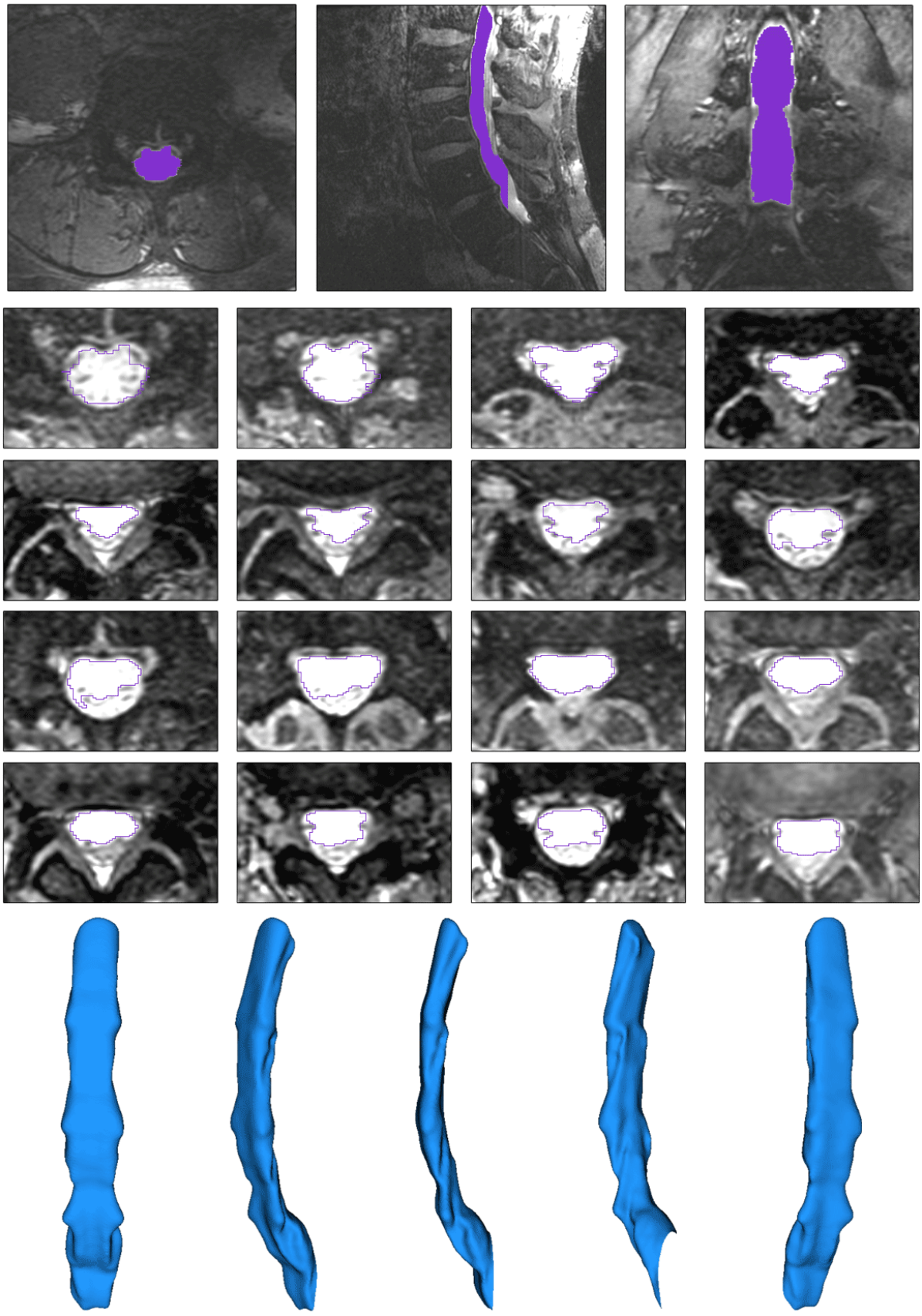


Figure 4.14.: Fully-automatic segmentation result of dataset *anonymized03.nrrd*

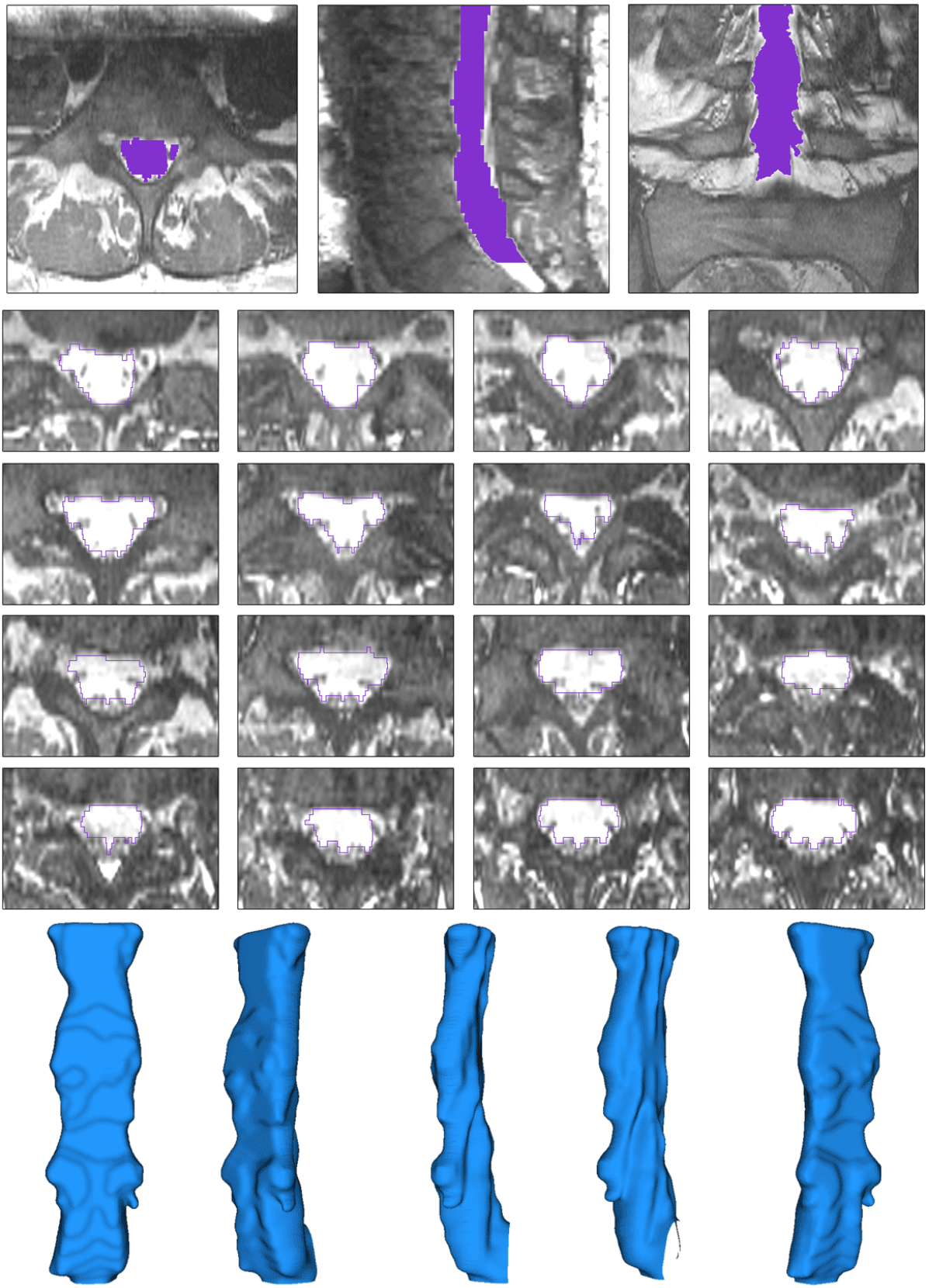


Figure 4.15.: Fully-automatic segmentation result of dataset *anonymized04.nrrd*

4.2. Semi-Automatic Algorithm

The main focus of the fully-automated algorithm is the automatic detection of the ROI on the datasets, realized by emulating the CSF as a three-dimensional model. The actual segmentation on the other hand is only implemented by a thresholding filter, which refers to the whole volume inside the boundary box. In this case, the conducted method of calculating the threshold value out of the image histogram is the best way to get a proper segmentation of the CSF, but it cannot be as accurate as when performed manually by a radiologist. This is in particular due to the fact that a radiologist would segment slice by slice and not a volume at once.

The second approach sets the detection of the ROI aside and gives the user the ability to mark the location of the CSF on the dataset, which assumes that the user has knowledge of the anatomic structures. The segmentation is afterwards performed individually on each axial plane by separating the CSF from its surroundings.

The user interaction takes place before the algorithm performs its appropriate steps and consists of setting a point on the dataset by using the Fiducials Module of Slicer. This point has to be located within the bright area of the CSF and serves as a seed point for the algorithmic execution. The figure below shows the Fiducials Module of Slicer as well as a seed point placed by the user.

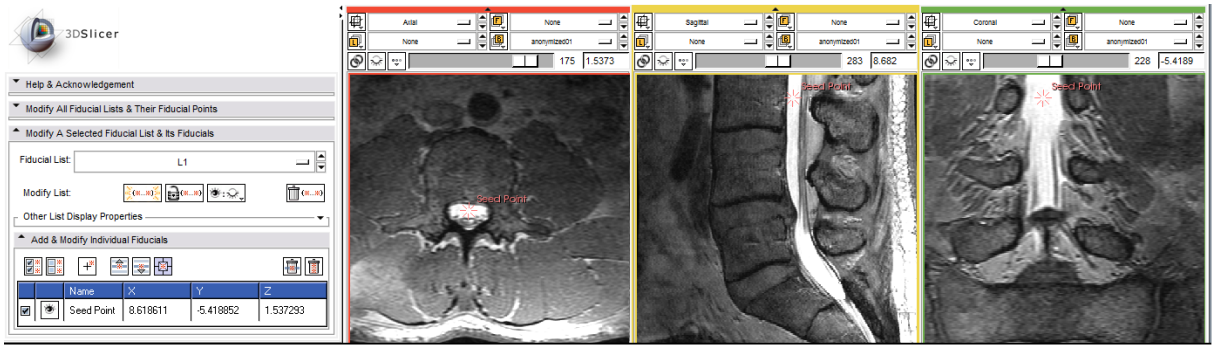


Figure 4.16.: Fiducials Module and the location of the seed point inside the CSF

The coordinates of the seed point which can be found in the Fiducials Module are 8.61, -5.41 and 1.53. These coordinates refer to the RAS coordinate system as described in Section 3.2, but since the dataset is read by using the `ImageFileReader` class of the ITK library, it is present in the LPS coordinate system. Therefore, the seed point has to be converted first from RAS to LPS by changing the algebraic sign of the first two coordinates (the right direction is changed into the left direction and the anterior direction into the posterior direction). Because the segmentation is performed in layers, it is also necessary to determine the image coordinates of the seed point. The image coordinate system uses the indices i, j and k to describe the position of a voxel on an image, starting at the upper left corner, which has the image coordinates 0, 0, 0. Voxels which are right next to the first voxel increase the i value by one; those which are below increase the j value and those which are behind increase the k value. To obtain the appropriate image coordinates of the seed point, the LPS coordinates must be transformed, which requires two more parameters of the recorded dataset, named origin and spacing. While the term spacing has been described previously, the origin of a dataset refers to the geometric coordinates of the first voxel. For dataset *ananymized01.nrrd* for example the origin in the LPS coordinate system is 68.418, 67.55 and -120.49. Using the origin and spacing, the position of each voxel in image coordinates can be calculated from its corresponding LPS anatomical coordinates, as shown for the index i by the following equation.

$$i = \frac{|L - \text{origin}_L|}{\text{spacing}_L}$$

where L is the left coordinate of the seed point, in this case -8.618, due to the changed algebraic sign resulting from the conversion from RAS to LPS.

Since pixel coordinates support only integer numbers, the results would be 282. Furthermore, it should be considered that the LPS coordinate system specifies the anatomic planes, meaning that the L coordinate is related to the sagittal plane, which separates the left part of the body from the right. Therefore, the i value 282 is associated with the sagittal slice number of the seed point. The other calculated image coordinates of this seed point are $j = 227$ for the coronal slice number and $k = 174$ for the axial slice number. These values can also be traced on the slice view of Figure 4.16, but it should be noted that within Slicer the indicated slice numbers start at 1 and not as common for the ijk coordinate system at 0.

Once the image coordinates of the seed point are calculated, the appropriate axial plane is extracted from the dataset and used as the first layer on which the segmentation of the CSF is performed. The segmentation is implemented as a region-based segmentation algorithm and uses only classes and methods of the VTK library. The property of this segmentation method, often referred to as region growing, is that it starts at the position of the seed point, or rather at the position of the equivalent voxel, and examines the neighboring voxels depending on a region membership criterion. If an adjacent voxel meets the criterion, it is added to the region and serves as a kind of new seed point, meaning that it is now considered whether its neighbors should be added to the region or not. In this way the region iteratively extends around the initial seed point until no more neighboring voxels meet the criterion of the region membership.

The algorithm performs this task by first creating a duplicate of the axial plane in which the seed point is located. This duplicate plane serves as a labelmap and is initially filled only with "background" voxels which all have an intensity of 0. As the seed point is placed by the user, it can be assumed that the corresponding voxel on the axial plane belongs to the region and is therefore marked as a "foreground" voxel on the labelmap. In a second step, a list is generated in which the coordinates of the neighboring voxels are stored. Since only one axial plane k is considered by the algorithm, the coordinates i and j are sufficient to describe clearly the position of an adjacent voxel.

The first iteration is that the eight neighbors, which are located left, right, above, below and diagonal of the initial voxel are determined. Afterwards, these neighbors are stored as candidates in the previously created list, and examined candidate by candidate whether they meet the criterion of the membership. If a candidate meets the criterion, the corresponding voxel on the labelmap is marked as "foreground" and a new iteration is started in which the eight neighbors of the labeled candidate are detected. However, before these new neighbors are stored in the list, it is first checked if they have already been added by a previous iteration. This has to be performed because a voxel which is not located at the border of the dataset can insert exactly eight candidates to the list, and can therefore be detected in eight different iterations, which means that the already examined voxel has to be checked once again. By avoiding duplicate entries in the list, it can be ensured that an infinite loop is prevented and each candidate is listed exactly once. Unique candidates are subsequently added to the end of the list and examined by a later iteration.

After the second iteration, a total of 13 candidates are present in the list, of which eight are from the first iteration and five from the second, under the condition that the first candidate has met the criterion. If a candidate does not meet the criterion, the corresponding voxel on the labelmap remains as "background" and the candidate search is skipped.

This process allows the collection of all neighbors around the initial seed voxel and marks the position on the labelmap as related to the region or not, which is illustrated in the figure below on an axial plane of dataset *ananymized01.nrrd* at different iteration steps.

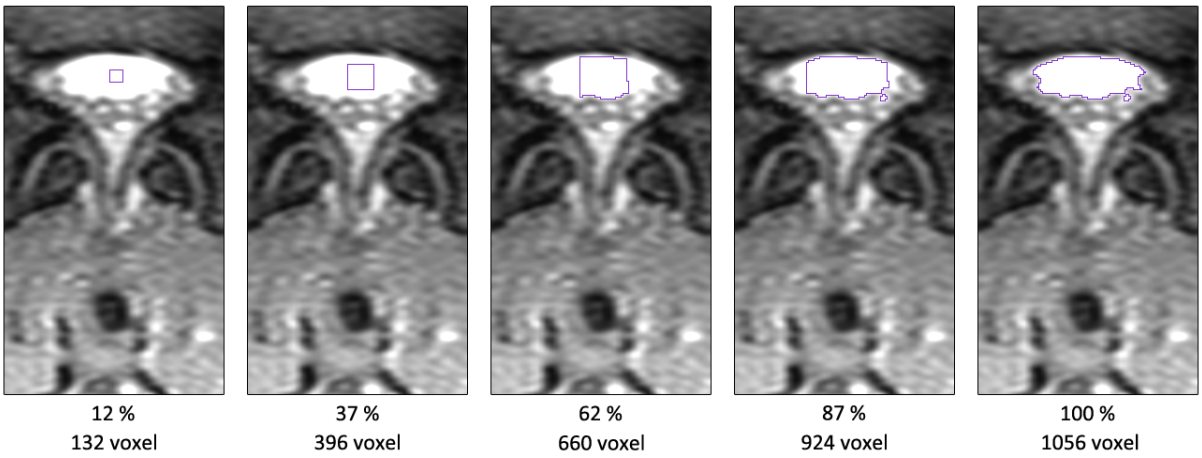


Figure 4.17.: Region growing at different iteration steps

The classification of a candidate is particularly dependent on the region membership criterion. To determine an appropriate criterion, several representative axial planes were examined and the intensity threshold incrementally increased. Based on the present gold-standard segmentation of dataset *ananymized01.nrrd*, in which the CSF was segmented accurately with the assistance of a physician, the threshold was raised manually until the extents of the thresholded area were equivalent to the gold-standard segmentation. Therefore, the detected threshold represents the optimal value of this particular plane. In the following figure, the outline of the gold-standard segmentation is shown in green and the removed voxels whose intensity value is below the various thresholds are illustrated as red. The optimal threshold of this axial plane is 2400.

Figure 4.18.: Manual increase of the threshold in an axial plane

By repeating this process in different axial planes, a correlation could be noticed between the average intensity of the voxels within the CSF and those voxels which have the lowest intensity values and are mainly in the border area of the CSF. Accordingly, voxels are still associated with the CSF as long as their intensity is 20 percent below the average intensity of the CSF.

The following figure illustrates the intensities in the area around the CSF and the histogram of the image section. In this example, the CSF has an intensity range from 2400 to 3612 and the average intensity of the voxels within the CSF is approximately 3020. An intensity of 80 percent of the average intensity, which is 2416, would therefore be suitable to obtain the minimum CSF intensity.

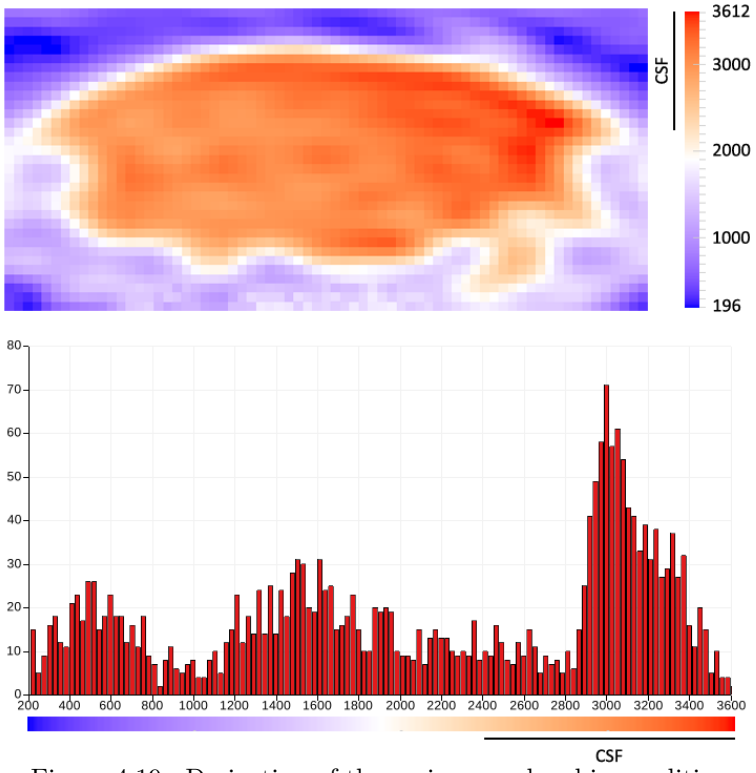


Figure 4.19.: Derivation of the region membership condition

The segmentation of the neighboring voxels in Figure 4.17 uses this condition and calculates the average intensity of all previously marked voxels. In the case of the first iteration the average intensity has the same value as the voxel of the initial placed seed point, but changes if new voxels of the candidate list meet the criterion.
Therefore, during the iteration process not only the candidate list grows, but also the average intensity continuously decreases, as the segmentation starts from high intensities in the middle and spreads towards the lower intensities in the border area. The 80 percent condition is the best compromise between preventing an "overflow" of the region, meaning that the region extends to one side and segments voxels out of the CSF, and a complete coverage of the ROI.

One feature of the region growing algorithm is unfavorable for a complete segmentation of the CSF, which is that the region must be continuous and the voxels connected with each other. However, the nerve fibers within the CSF are on some axial planes relatively thick and consist of very dark intensities on T2w MRI images, so that the region may be divided and no longer fully connected. The figure below illustrates this situation, in which on the first axial plane a nerve fiber at the lower right edge of the CSF is so thin that its low intensity does not affect the high intensity of the CSF and results in a segmentation of the CSF beyond that fiber. On the directly consecutive plane, the fiber is becoming thicker and its intensity lower, which means that the region growing algorithm is able to segment the main accumulation of CSF but misses the separated region. Since this region, which is marked as green on Figure 4.20, has been classified previously to belong to the ROI, it has to be segmented too.

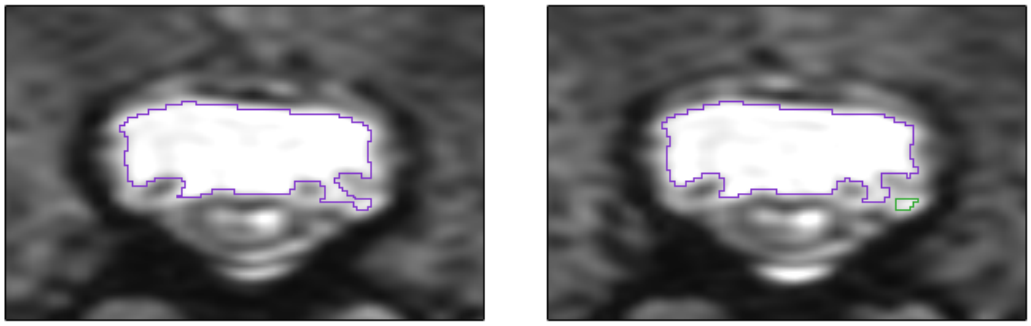


Figure 4.20.: Disadvantage of the region growing algorithm

Furthermore, there is also the case that liquid is separated from the main accumulation throughout all axial planes, and can therefore never be detected by the region growing algorithm, as for example the bright spot just below the CSF. In order to get a complete segmentation of the CSF, this spot must also be covered by the labelmap. However, not all bright spots around the CSF are liquid, but can also be fat deposits, which do not differ in their intensity values from the CSF.
If all these facts are taken together, an accurate segmentation for the axial plane on the right side of Figure 4.20 would be the following, where the area detected by the region growing algorithm is illustrated as violet, the missing areas as green and the fat deposit, which must be excluded from the segmentation, as red.

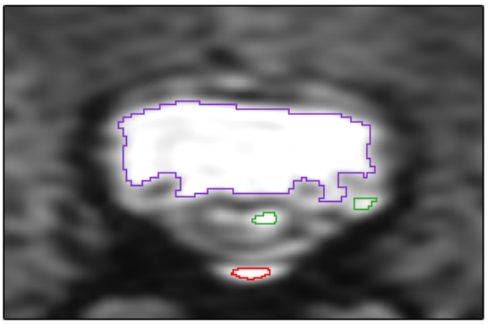


Figure 4.21.: Complete coverage of the CSF

With the help of the gold-standard segmentation, all spots which were identified as CSF were determined and analyzed for common characteristics. Thereby, it could be noticed that all external regions are located posterior to the CSF or below of the main accumulation in an axial plane. In addition, the fat deposits, whose intensities are in the range of the fluid spots and can therefore not be used to distinguish them, are always a few millimeters further in a posterior direction than the liquid and border on the bony tissue of the vertebra. Anatomically, this can be explained by the fact that the nerve fibers and the CSF in the lumbar region are surrounded by two meninges (the arachnoid and the dura mater), which separate them from the fat deposits. Even if these meninges are very thin, they allow together with the posterior extending nerve fibers a separation of the subarachnoid space and the epidural space on the present datasets.

In the algorithmic implementation, the procedure is therefore that first the main accumulation of the CSF is detected by using the region-based segmentation as described above. After executing the region growing algorithm, the dimensions of the segmented region along the i and j axes can be calculated and used to create a bounding box around the CSF which covers the complete region. This box is afterwards extended by 20 percent of its height in a posterior direction, i.e. along the j axis. The extension by 20 percent represented in all datasets a good coefficient to capture the fluid spots and leave out the fat deposits, as all structures located within the

extended frame are close enough to the CSF to be still liquid. The following segmentation refers only to this box and is realized by a threshold operation with the same intensity criterion as previously calculated for the region growing. This means that those structures are segmented within the extended box whose intensities are higher than 80 percent of the average intensity of the main CSF accumulation.
In the figure below the bounding box of Figure 4.21 is shown in blue and its 20 percent extension in yellow. It can be noted that only the bright spot below the CSF is covered by the extension, while the fat deposit is left out.

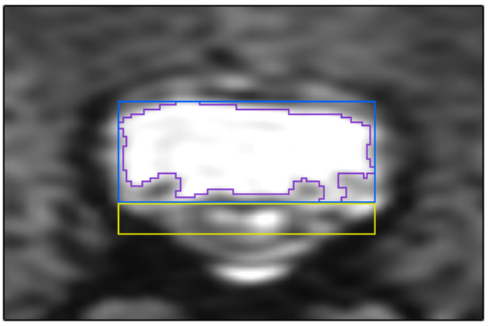


Figure 4.22.: Bounding box of the extended segmentation

Although the result of the region growing is actually rejected by the threshold, it is still necessary to determine the extent of the segmentation and, more importantly, to provide an individual and optimal threshold value for each axial plane. A comparison of both segmentation results at different axial planes of dataset *ananymized01.nrrd* is illustrated by the following figure, in which the first row shows only the region growing result and the second row the same axial plane with the extended segmentation.

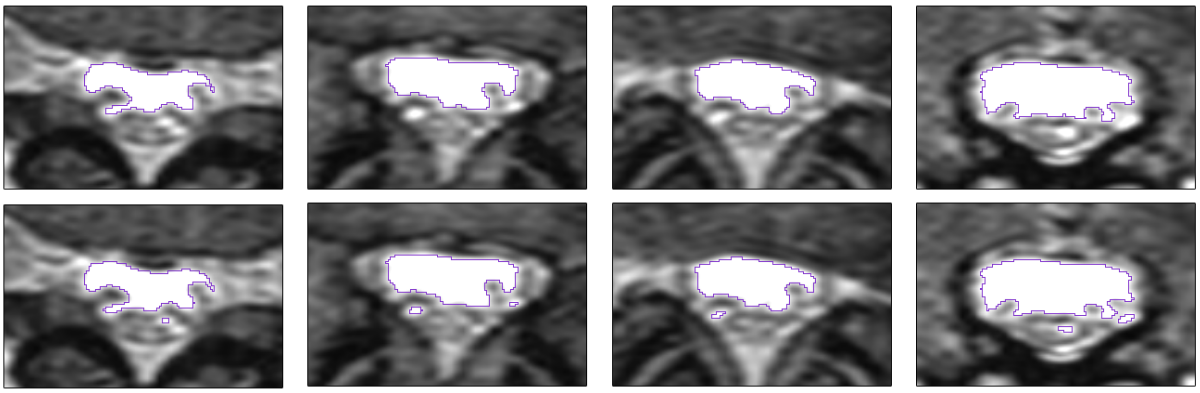


Figure 4.23.: Comparison of the region growing segmentation and the extended segmentation

The previously described algorithm is only able to perform the segmentation in exactly that axial plane in which the seed point has been placed by the user, meaning that a segmentation of other planes is only possible if a seed exists in these planes too. Since a dataset consists of approximately 200 planes, a manual setting of all seeds is not a realistic option. Therefore, an automatic calculation is used, which is based on the fact that the CSF has over a wide range of the dataset a similar and self-containing elliptical shape, so that a point in the center of a segmentation is quite suitable to serve as the starting point for the subsequent plane. The i and j coordinates of the geometric center are calculated according to the following equations.

$$\bar{i} = \frac{1}{N} \sum_{m=0}^{i_{max}} \sum_{n=0}^{j_{max}} i \qquad \bar{j} = \frac{1}{N} \sum_{m=0}^{i_{max}} \sum_{n=0}^{j_{max}} j$$

where N is the total number of segmented voxels by the region growing algorithm and m/n two control variables, which start at the first voxel of the labelmap and move row- and column-wise to the last.

In this way, the sum of the corresponding i and j coordinates, which are marked as "foreground" on the labelmap, is determined and used to get the i and j mean values of the segmented voxels. Afterwards, these coordinates are transferred to the under- or overlying axial plane (in the case of the initial axial plane to both) and used as a seed point of the region growing and extended segmentation algorithms, which are executed again as previously described.

In total, the algorithm is divided into three parts, whereby the first part consists of segmenting the axial plane that has been selected by the user through setting the seed point. After the segmentation has been performed and the geometric center calculated in the initial plane k, a for-loop uses the geometric center as the new starting point and repeats the execution in a superior direction, meaning that it starts at plane $k+1$ and ends at the top-most plane. In the third part, a second for-loop processes similarly, but from the plane below of the initial plane $k-1$ and continues in an inferior direction. The following figure shows eight consecutive axial planes in a superior direction with their extended segmentations as well as the position of its seed point, which corresponds to the calculated geometric center of the previous plane.

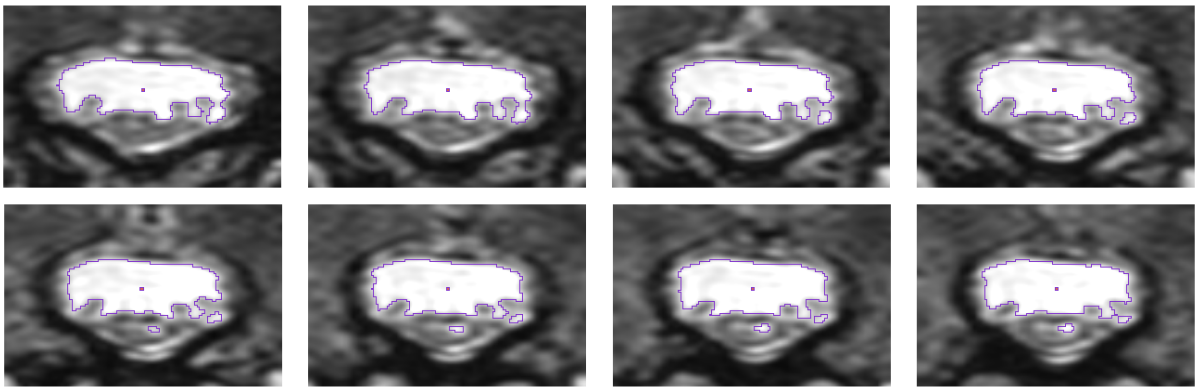


Figure 4.24.: Eight consecutive axial planes and their seed points

The final algorithmic step is now to combine the segmentations of the individual planes into a single labelmap, which appears as the first result of the semi-automatic algorithm superimposed on the dataset within the slice view of Slicer. Following the description in Section 4.1, a three-dimensional model is generated in addition to the segmentation by using the marching cubes method of the VTK class library. Since this model has the same purpose as the model of the fully-automatic algorithm, it can be used in those datasets with a bulged intervertebral disc to identify the pathological region easily. Even though the extended and more accurate segmentation leads to a model that is not as smooth as the model of the fully-automatic algorithm, it further permits the display of the separated fluid accumulations.

On the following pages the outcomes of the algorithmic execution are illustrated for the T2w MRI datasets. As mentioned above, the pictures at the top and bottom of these pages correspond to the slice and three-dimensional view of Slicer and the detailed axial views in the middle allow a visual determination of the segmentation accuracy.

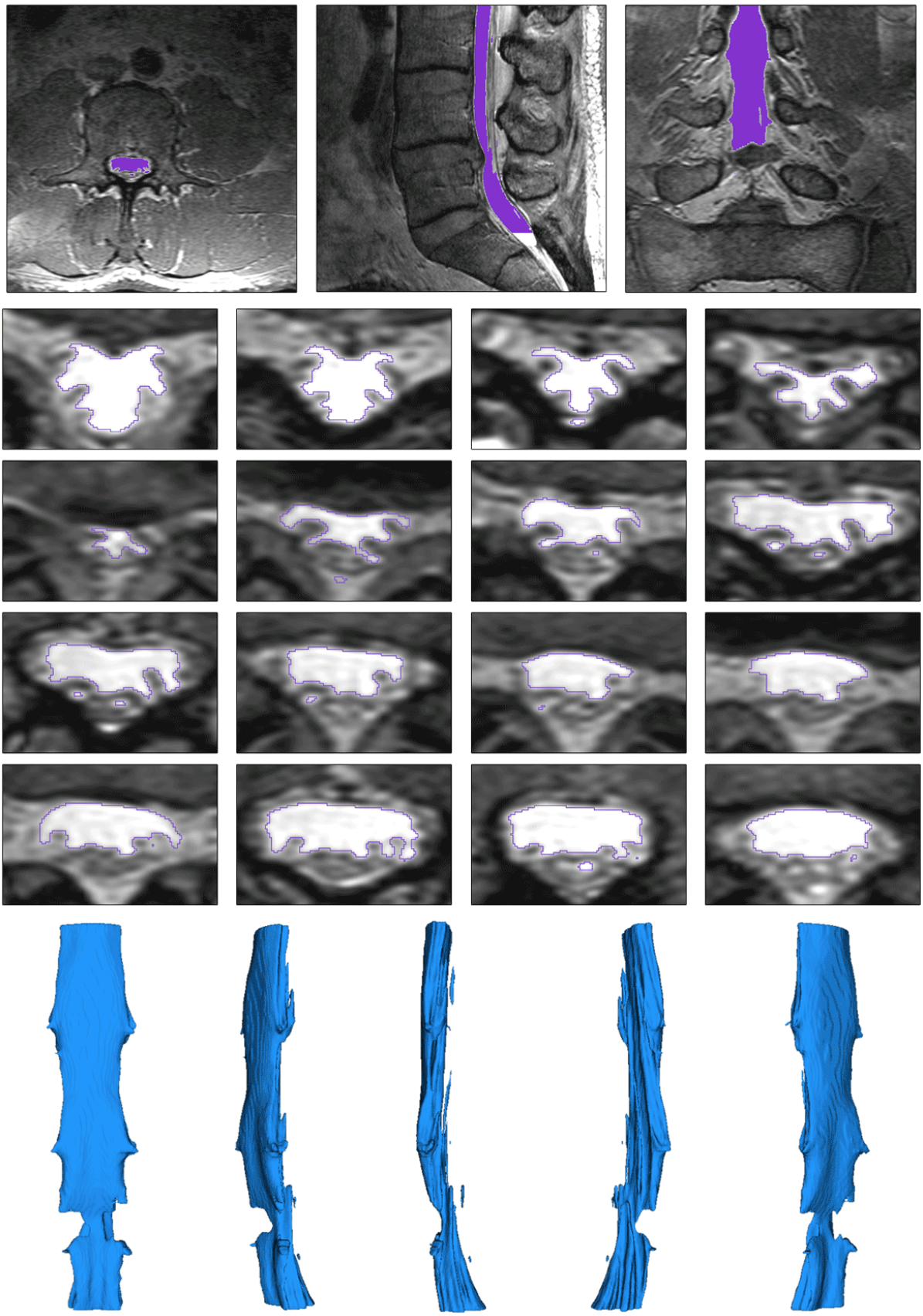


Figure 4.25.: Semi-automatic segmentation result of dataset *anonymized01.nrrd*

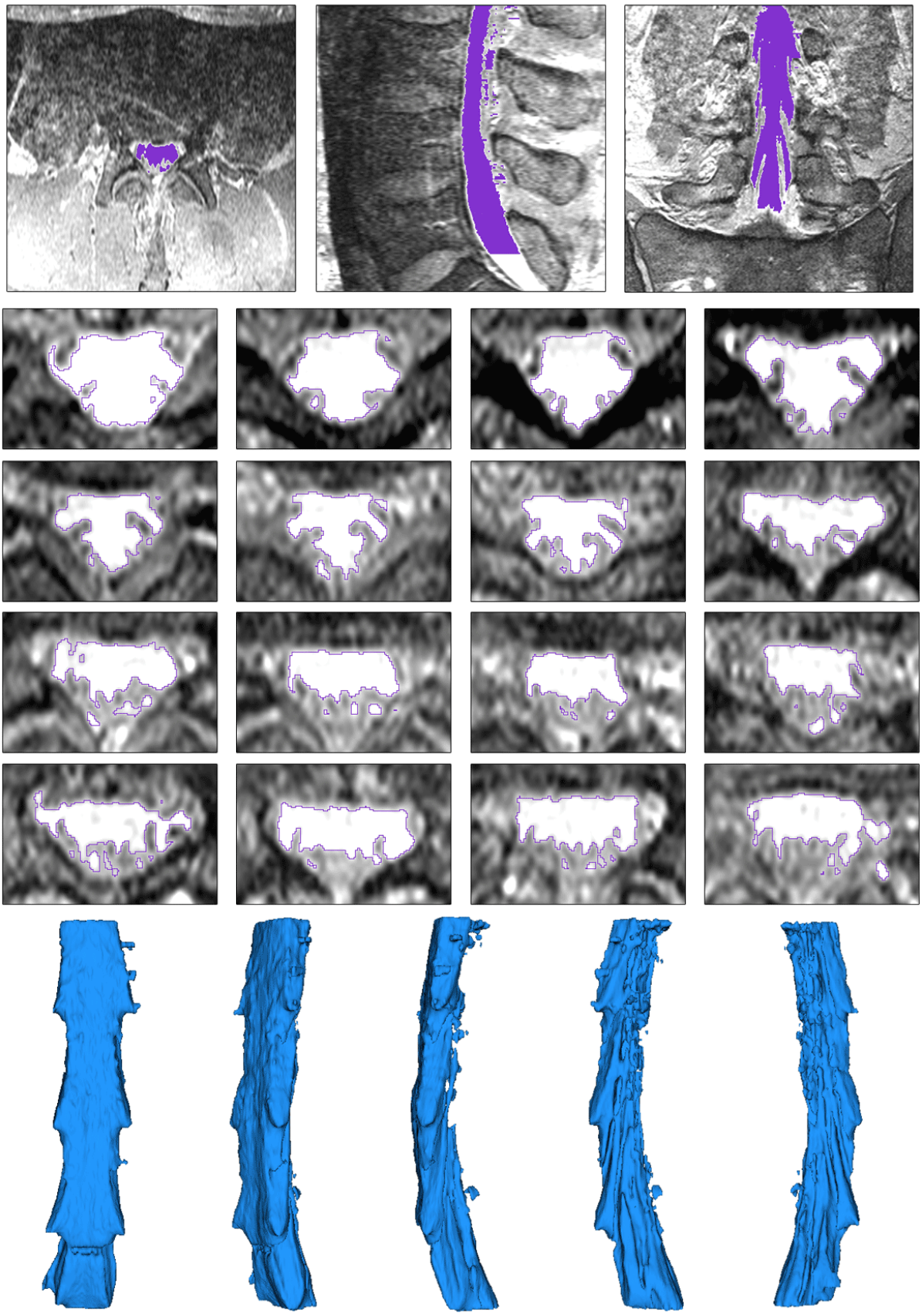


Figure 4.26.: Semi-automatic segmentation result of dataset *anonymized02.nrrd*

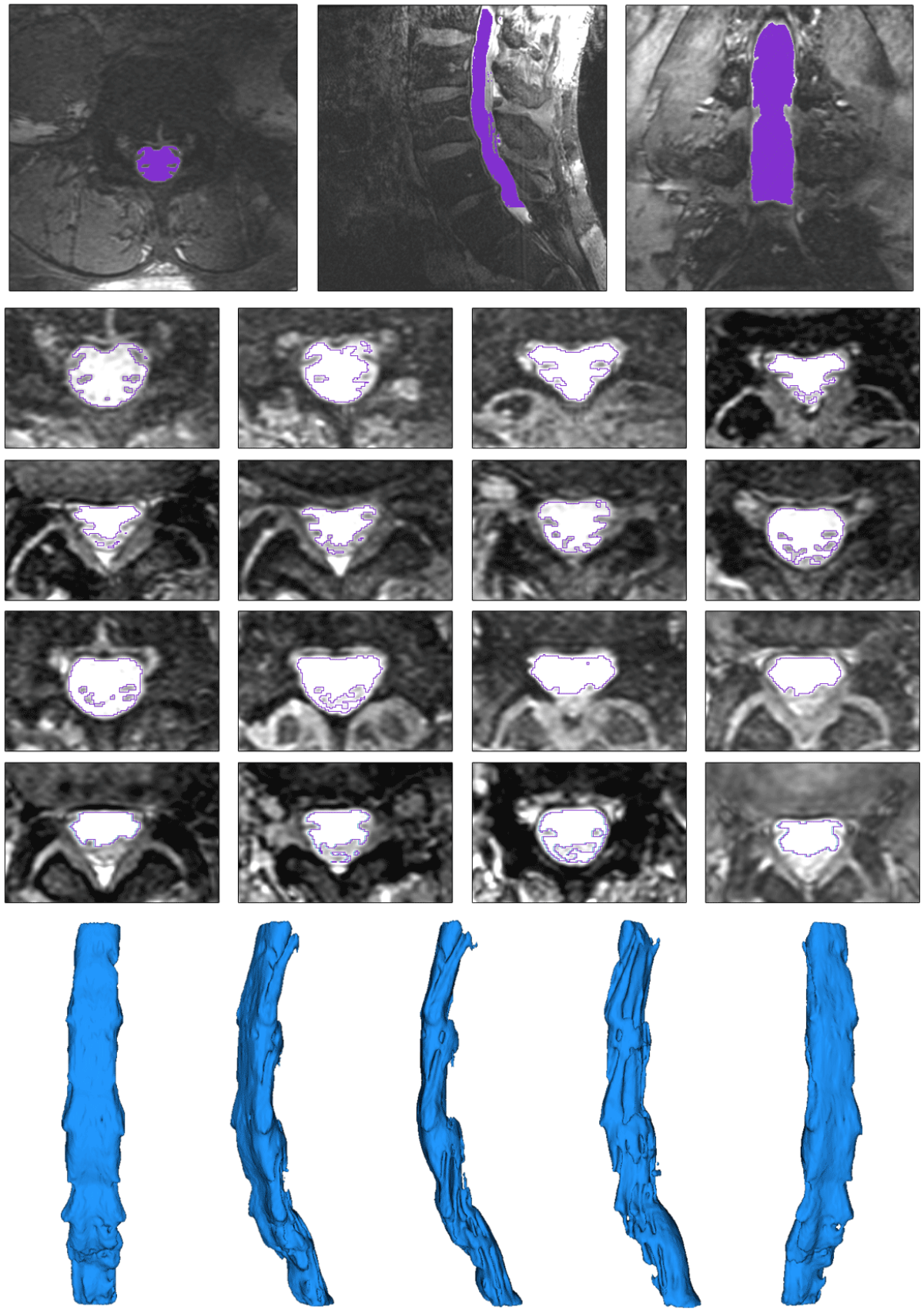


Figure 4.27.: Semi-automatic segmentation result of dataset *anonymized03.nrrd*

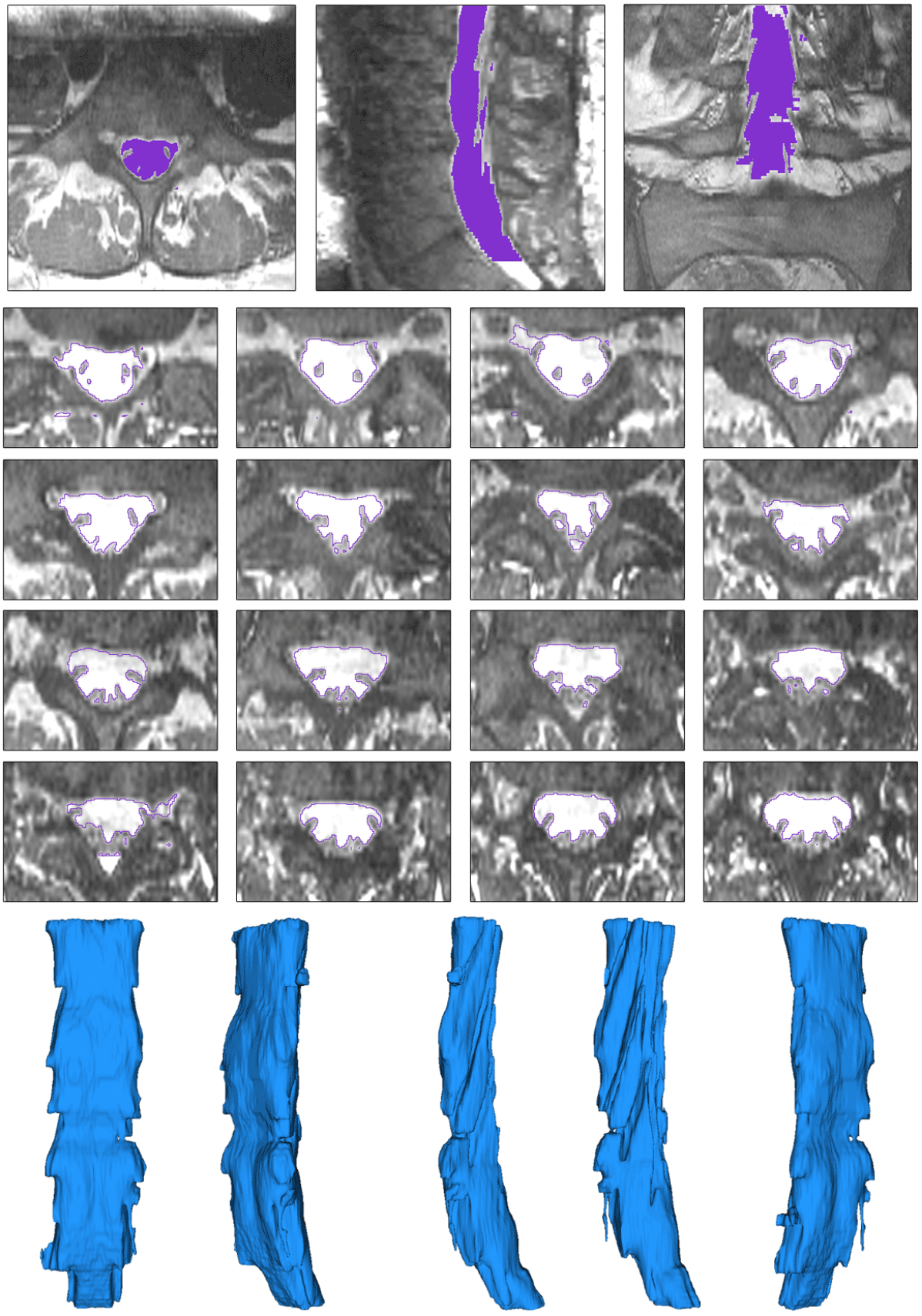


Figure 4.28.: Semi-automatic segmentation result of dataset *anonymized04.nrrd*

5. Discussion

A total of two completely different automatic algorithms have been developed to perform a segmentation of the CSF in the lower lumbar region of the vertebral column. However, the intended approach of using the nerve fibers of the cauda equina as ROI could not be realized, since the imaging quality of the present T2w MRI datasets is not adequate enough for an automatic detection and segmentation of the nerve fibers. While it is hardly possible at dataset *anonymized01.nrrd*, which has by far the best imaging quality, to recognize the nerve fibers with the naked eye, even this procedure cannot be applied to the other records. This is in particular due to the fact that the nerve fibers are too small and therefore too low in contrast compared with the bright CSF to separate them manually from their surroundings. As the present datasets are inadequate for a manual segmentation of the cauda equina, an automatic detection cannot be considered. Therefore, both algorithms are limited to segmenting the CSF. Although the CSF is not of the highest priority for a physician during the surgical treatment of a disc herniation, it is also affected by the deformed intervertebral disc and can serve as an appropriate visualization of the pathological region. Furthermore, an automated detection of the CSF is a good starting point to segment the cauda equina within the CSF, once datasets have been acquired with novel MRI sequences and increased visibility of the nerve fibers.

In the fully-automatic approach no user interaction is required but only a one-button click to scan the dataset volume-based and to detect a bright tubular region which is surrounded by dark tissue. Hereby, the fundamental issue is that the automatic analysis of a dataset, consisting of approximately 50 million voxels, requires a time-consuming execution if it is performed on a single or multiple central processing unit (CPU). Even a multi-threaded calculation with for example four threads could only process a maximum of four voxels at a time and reduce the computation time by at most a quarter. An alternative and a dramatic speed-up would have been to perform the calculation on a video card with for example 1024 shader units and therefore 1024 simultaneous processed voxels. However, since the GPU acceleration is a recent development and so far only supported by a few ITK and VTK classes, the conventional CPU computation had to be used in both algorithms. This fact had in particular an influence on the fully-automatic algorithm and the choice of an appropriate cylinder step size. A too low step size has indeed the property that the CSF will be detected for sure, but leads also to an inacceptable computation time within the range of hours. On the other hand, a higher step size could reduce the number of options to place the cylinders on the dataset and therefore the execution time, but has the disadvantage that the moving cylinders could simply skip the ROI and end up in a region which lies apart from the actual CSF. As the computation time is correlated with the step size of the cylinders, a value with a high success probability and a low computation time had to be used to perform the execution. Nevertheless, this step size can only guarantee that the CSF will be detected with certainty on those four datasets which were present during the development process and on which the algorithm was applied. In a further implementation of the algorithm with GPU acceleration a smaller step size could be used, as the computation time would be reduced significantly, while the sensitivity to provide correct results would increase. From the current project state, it has to be stated that the fully-automatic algorithm depends too heavily on the step size of the cylinders and is therefore sometimes uncertain whether the ROI is detected or not.

Since the detection of the fully-automatic algorithm is volume-based, realized by emulating the CSF as a three-dimensional model, this approach has been retained for the segmentation method, which also refers to a volume. The implemented thresholding filter, together with the evaluation of the image histogram and the calculation of an adaptive threshold, represents a simple but effective way of separating the CSF from its surroundings within a volume. Because the threshold is determined from the intensities of several planes at once, it corresponds to the average thresholding value, which is optimal for the whole volume but not for an individual plane. As a result, a loss of precision in the resulting segmentations has to be accepted.

As an alternative to the volume-based segmentation method of the fully-automatic approach, a semi-automatic algorithm was designed. This algorithm relies on a seed point placed by the user inside the ROI, so that the automatic detection of the ROI was no longer relevant. Instead, a more accurate segmentation was intended, performed individually on each axial plane by a region-based segmentation algorithm. Thereby, all voxels adjacent to the seed point are labeled as related to the ROI until no more neighboring voxels meet the region membership criterion. Through a suitable choice of the criterion it is ensured that only those voxels are added to the region that have a high intensity and thus belong to the CSF. Even if the region membership criterion was identified based on the gold-standard segmentation of dataset *anonymized01.nrrd*, it has proven to be robust and reliable also on the other datasets. With the use of a thresholding filter, which is applied to a small region in posterior direction of the main CSF accumulation, the disadvantage of a region growing algorithm has been resolved, meaning that the region no longer has to be continuous or the voxels connected with each other. In order to avoid that the result of the region growing is rejected by the thresholding operation, the same limit as previously calculated for the region membership criterion is used by the threshold. This procedure guarantees that all fluid accumulations which are separated from the main accumulation but are close enough to the CSF and have the same intensity as the main accumulation are also include in the segmentation. Overall, the combination of a region growing algorithm and a thresholding filter conducted to each axial plane results in a segmentation that is more precise than the outcome of the volume-based approach.

The semi-automatic algorithm has shown how little effort is necessary to realize the detection of the ROI by a simple user interaction compared to the fully-automated approach and which alternatives allow a more accurate and thus more useful segmentation. Although it would have been possible to connect both algorithms, so that the detection is performed automatically by the usage of the cylinders and the segmentation by the region growing method, this option was abandoned during the development process. The reason for this decision was that even if the fully-automatic algorithm would obtain a better segmentation procedure, the issue with the step size still exists. Consequently, there are currently two independent algorithms which can be connected with each other once the step size and/or the computation time of the fully-automatic algorithm is reduced, for example by a GPU acceleration.

The application of the fully- and semi-automatic algorithm to the four T2w MRI datasets and the illustration of the segmentation results in the corresponding sections so far provide only a visual evaluation of the segmentation accuracy. In order to obtain a better assessment, the outcomes for dataset *anonymized01.nrrd* of both algorithms are now individually contrasted and compared with the present gold-standard segmentation. The labelmap is more suitable than the generated three-dimensional model for this purpose, since the image coordinates of the marked and unmarked voxels can be used to clearly identify to what extent the labelmap of the fully- or semi-automatic algorithm is identical to the labelmap of the gold-standard segmentation.

Therefore, the voxels of each labelmap are considered as a set and compared slice by slice according to the principles of a Venn diagram. Moreover, appropriate metrics can be determined for the number of segmented voxels which are present either in both labelmaps or only in one.

In the following figure a Venn diagram represents the logical relations between the segmented voxels of the gold-standard segmentation and the segmented voxels of the semi-automatic algorithm. It should be noted that the Venn diagram for a comparison between the gold-standard segmentation and the fully-automatic algorithm would appear in the same way.

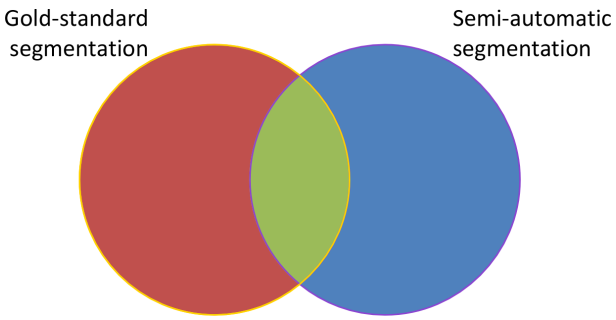


Figure 5.1.: Venn diagram of the comparison between the gold-standard segmentation and the semi-automatic segmentation

In total, three regions of the Venn diagram, which are visualized as green, red and blue in Figure 5.1, are of importance to compare the gold-standard segmentation either with the fully- or with the semi-automatic segmentation:

- The green region corresponds to those voxels which are segmented by both the gold-standard segmentation as well as the automatic segmentation, meaning that both labelmaps have these voxels in common.
- The red region indicates those voxels which are only segmented by the gold-standard segmentation, meaning that the automatic algorithm has failed at this point and segmented too few voxels.
- The blue region represents the opposite of the red region and corresponds to those voxels which are only segmented by the automatic algorithm, meaning that the automatic algorithm has failed at this point and segmented too many voxels.

In the ideal case, the number of common voxels is equal to the number of voxels segmented by the automatic algorithm and as a result the number of too few and too many segmented voxels is zero. In conclusion, the labelmap of the automatic segmentation would be identical to the labelmap of the gold-standard segmentation.

This concept is now applied to all axial planes of the labelmap, whereby the result for the comparison between the gold-standard segmentation of dataset *anonymized01.nrrd* and the fully-automatic segmentation corresponds to the figure below. As already assumed, the loss of precision due to the calculation of a volume-based threshold affects most of the axial planes in the way that too few voxels are segmented. While on average the number of too many segmented voxels is only 2.5 percent, 22.4 percent of the voxels are absent in the labelmap of the fully-automatic algorithm.

As Figure 5.2 indicates, the generated labelmap has its greatest inaccuracies between the 25th and 45th plane, which corresponds on dataset *anonymized01.nrrd* to the location of the disc herniation. In this region the number of too few segmented voxels is 48.2 percent, which has to be considered as too high to serve as an adequate support during a surgical intervention.

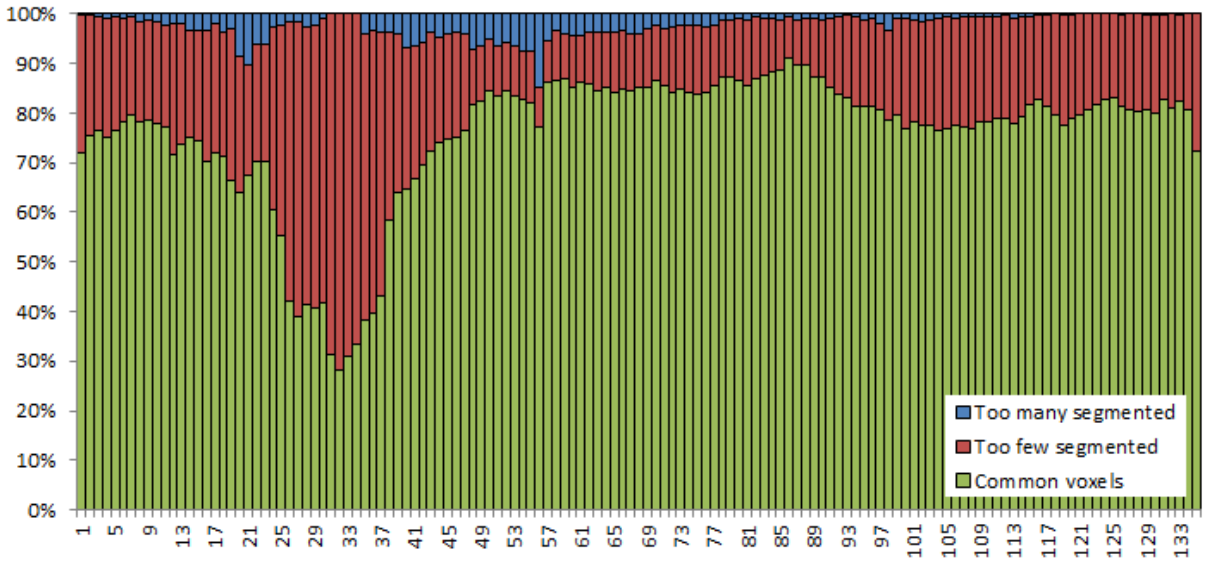


Figure 5.2.: Plane-wise comparison between the gold-standard segmentation and the fully-automatic segmentation of dataset *anonymized01.nrrd*

In contrast, the plane-wise comparison between the gold-standard segmentation and the semi-automatic segmentation of dataset *anonymized01.nrrd* appears as follows:

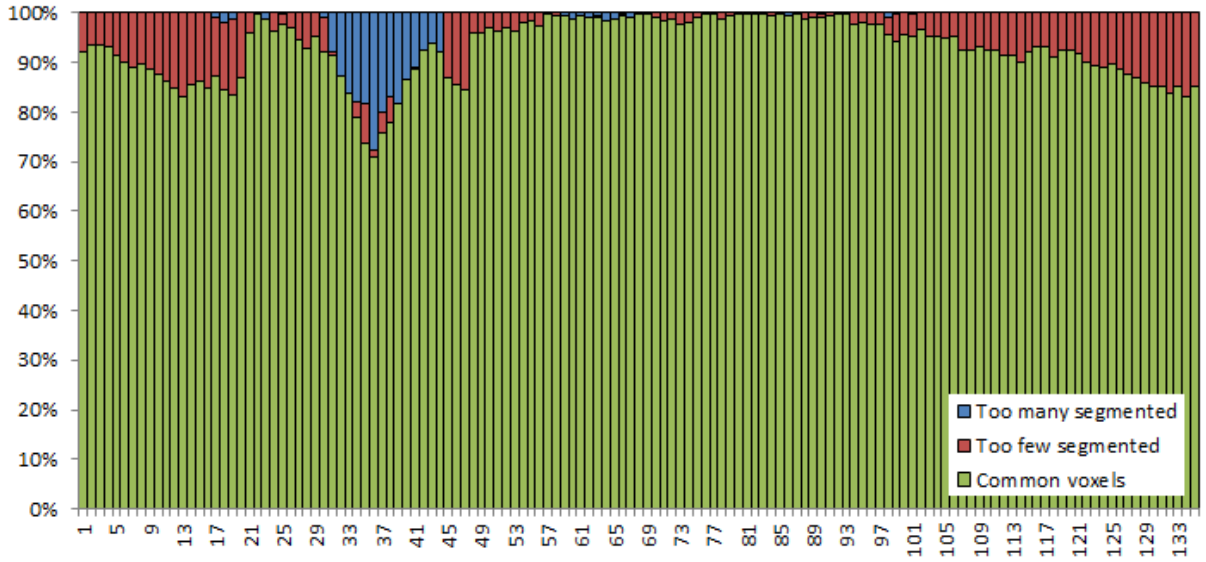


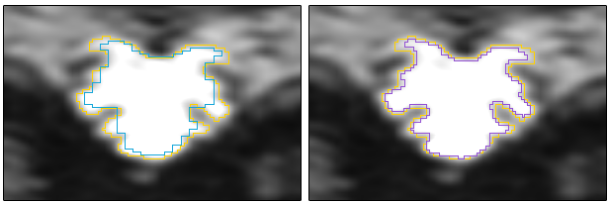
Figure 5.3.: Plane-wise comparison between the gold-standard segmentation and the semi-automatic segmentation of dataset *anonymized01.nrrd*

Here it can be noticed that the number of too few segmented voxels is consistently below 15 percent, and on average only 5.5 percent. In addition, just 1.9 percent of the voxels are segmented by the semi-automatic algorithm and are not included in the gold-standard segmentation.

The pathological region between axial plane number 25 and 45 in particular shows that the segmentation of the semi-automatic algorithm has its advantages and is much more accurate than the labelmap of the fully-automatic approach. This region has on average only 3.3 percent missing voxels in the semi-automatic case, compared to 48.2 percent for the fully-automatic algorithm. In contrast to this, the number of too many segmented voxels is 8.7 percent and therefore about 6 percent larger than in the fully-automatic segmentation.

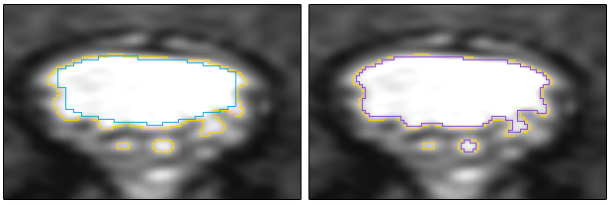
A closer look at some representative planes shows in which areas of the labelmap both algorithms have inaccuracies and how these can be interpreted and explained.

Axial layer number 8



Number of voxels		
Gold-standard	1138	1138
Fully/semi-automatic	925	1020
Common voxels	906 (79%)	1020 (90%)
Too few segmented	232 (20%)	118 (10%)
Too many segmented	19 (2%)	0

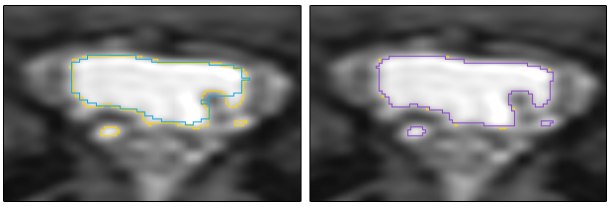
Axial layer number 119



Number of voxels		
Gold-standard	1192	1192
Fully/semi-automatic	925	1102
Common voxels	924 (77%)	1102 (93%)
Too few segmented	268 (23%)	90 (8%)
Too many segmented	1 (0%)	0

In the segmentation results of the lower and upper part of the lumbar vertebra, it can be recognized that the labelmap of the semi-automatic algorithm has the same shape as the gold-standard segmentation and differs only in one or two voxels at the border. The situation in the labelmap of the fully-automatic algorithm is, however, that indentations of the CSF are not covered at all, as the intensity in these areas is not as high as of the main CSF accumulation. The same applies for the separated fluid accumulations as for example in the labelmap of the axial plane number 119. Since not even the semi-automatic algorithm has detected all relevant voxels, the region membership criterion could be reduced from 80 percent of the mean intensity to 75 percent. However, this could mean that afterwards too many voxels are segmented, so that the original criterion is considered to be appropriate, as the plane-wise comparison in the middle part of the lumbar vertebra indicates.

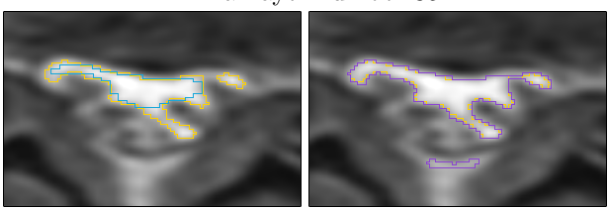
Axial layer number 71



Number of voxels		
Gold-standard	873	873
Fully/semi-automatic	795	858
Common voxels	769 (88%)	858 (98%)
Too few segmented	104 (12%)	15 (2%)
Too many segmented	26 (3%)	0

As mentioned above, the greatest inaccuracies occur in both algorithms in the pathological region between the 25th and 45th axial plane. While in this range the fully-automatic algorithm segments far too few voxels (48.2 percent), there are too many in the semi-automatic case (8.7 percent).

Axial layer number 35



Number of voxels		
Gold-standard	190	190
Fully/semi-automatic	84	215
Common voxels	76 (40%)	172 (91%)
Too few segmented	114 (60%)	18 (9%)
Too many segmented	8 (4%)	43 (22%)

The loss of precision in this region can be explained by the fact that the herniated disc causes a narrowing of the CSF, so that it no longer has its physiological shape but a much lower intensity

than the accumulations of the other axial planes. The fully-automatic algorithm fails to segment most of the relevant voxels, because more "healthy" than pathological planes are included in the considered volume, which results in an averaged threshold that is too large to detect the low intensity areas.
In the case of the semi-automatic algorithm it has to be noted that the extension of the segmentation proves to be unsuitable for the pathological region. This is due to fact that the CSF passes atypically far in the posterior direction, so that parts of the fat deposits are also covered by the labelmap. A segmentation performed exclusively by the region growing algorithm, without an expansion, would therefore be more reasonable in this region. However, since this would require further user interaction it has been avoided, particularly because the segmentation in this region is still accurate and the number of too many segmented voxels below 9 percent.

To be able to evaluate the performed segmentations and to compare the similarity and diversity of the labelmaps, the Jaccard index J was calculated for each segmented axial plane according to the equation:

$$J = \frac{|G \cap A|}{|G| + |A| - |G \cap A|}$$

where $|G|$ and $|A|$ are the numbers of voxels that are present in either the labelmap of the gold-standard segmentation or in the labelmap of the automatic segmentation and $|G \cap A|$ the number of voxels which both labelmaps have in common. The index ranges from 0 to 1 and the value 1 implies an absolute similarity of the two sets G and A.

According to the Jaccard index, a plane-wise comperation between the gold-standard segmentation and the labelmaps generated by the fully- and semi-automatic algorithms for dataset *anonymized01.nrrd* results in the following similarity curves. The pathological region is slightly highlighted and illustrates clearly the advantage of the semi-automatic algorithm compared with the fully-automatic algorithm.

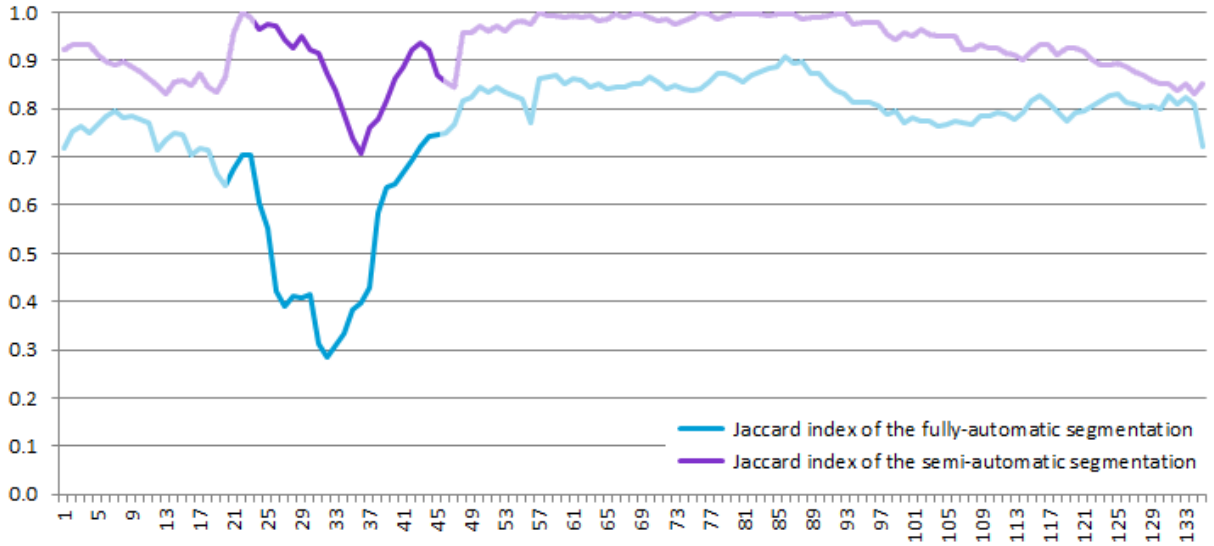


Figure 5.4.: Jaccard index of the plane-wise comparison between the gold-standard segmentation and the fully-automatic segmentation as well as between the gold-standard segmentation and the semi-automatic segmentation for dataset *anonymized01.nrrd*

Overall, it can be stated that the semi-automatic segmentation is in all planes more accurate than the fully-automatic segmentation, as the Jaccard index is higher throughout. Furthermore, the semi-automatic algorithm has on average a similarity coefficient of 0.931, meaning

that 93.1 percent of the labelmap is identical to the labelmap of the gold-standard segmentation. For the fully-automatic algorithm the similarity of the whole segmentation is on average 78.7 percent. Since the segmentation of the CSF in the lumbar region of the vertebral column is a new approach, the similarity coefficients can only be compared to studies which perform a related segmentation on the brain. An automated segmentation of the intracranial CSF, published by Louis Lemieux in 2003, obtained a CSF similarity with the gold standard of 86 percent, which is between the obtained results in the lumbar region [23]. Accordingly, a similarity of 93.1 percent in the case of the semi-automatic algorithm is similar to the results of other studies and represents an acceptable value for a semi-automatic segmentation.

A review which includes only the pathological region, since this is of the greatest importance during a surgical intervention, indicates on average a similarity coefficient of 56.6 percent for the fully-automatic algorithm. On the other hand, the semi-automatic algorithm obtains in this region a similarity of 87.8 percent, which is considered a good result. Therefore, it can be stated conclusively that the segmentation approach of the semi-automatic algorithm is much better suited to achieving more accurate and higher-quality results, which can be used to support computer-assisted surgeries of lumbar disc herniations.

Bibliography

[1] NINDS. Low back pain fact sheet. National Institute of Neurological Disorders and Stroke, July 2003.

[2] R.A. Deyo, S.K. Mirza, and B.I. Martin. Back pain prevalence and visit rates: estimates from u.s. national surveys, 2002. *Spine*, 31:2724–2727, November 2006.

[3] Chin-Teng Chung, Chun-Fu Wang, Chrong-Song Chou, and Haw-Chang Lan. Single photon emission computed tomography (spect) for low back pain induced by extension with no root sign. *Journal of the Chinese Medical Association*, 67:349–354, July 2004.

[4] Veerle Hermans. Research on work-related low back disorders. Institute for Occupational Safety and Health, January 2000.

[5] AANS. Back pain: Understanding causes and treatment. Section On Pain 2008 Biennial Meeting, April 2008.

[6] T.S. Carey, A.T. Evans, N.M. Hadler, G. Lieberman, W.D. Kalsbeek, and A.M. Jackman. Acute severe low back pain. a population-based study of prevalence and care-seeking. *Spine*, 21:339–344, 1996.

[7] Christopher M. Williams, Christopher G. Maher, Mark J. Hancock, James H. McAuley, and Jane Latimer. Low back pain and best practice care. *Archives of Internal Medicine*, 170:271–277, 2010.

[8] G.B. Andersson. Epidemiological features of chronic low-back pain. *Lancet*, 354:581–585, 1999.

[9] Robert Bohinski, Jody Hessel, and Bobbie Ryan. Herniated lumbar disc. Mayfield Spine Institute, April 2009.

[10] Walter de Gruyter. *Pschyrembel Klinisches Wörterbuch*. Gruyter, 260 edition, 2004. 2046 pp.

[11] LSI. Spine conditions that cause back and neck pain. Laser Spine Institute, 2010.

[12] S. Craig Humphreys and Jason C. Eck. Clinical evaluation and treatment options for herniated lumbar disc. American Academy of Family Physicians, February 1999.

[13] Edwin Clarke and C.D. O'Malley. *The human brain and spinal cord*. Norman Publishing, 2 edition, 1996. 804 pp.

[14] Jeffrey G. Jarvik and Richard A. Deyo. Diagnostic evaluation of low back pain with emphasis on imaging. *Annals of Internal Medicine*, 137:586–597, 2002.

[15] E.S. Gibson. Review: clinical findings should determine choice of imaging test for patients with low back pain in a primary care setting. *Evidence-Based Medicine*, 8:62, 2003.

[16] James N. Weinstein, Tor D. Tosteson, and Jon D. Lurie. Surgical vs nonoperative treatment for lumbar disk herniation. *Journal of the American Medical Association*, 296:2441–2450, 2006.

[17] H. Osterman, R. Sund, S. Seitsalo, and I. Keskimäki. Risk of multiple reoperations after lumbar discectomy: a population-based study. *Spine*, 15:621–627, 2003.

[18] Zhigang Peng, Jia Zhong, William Wee, and Jing huei Lee. Automated vertebra detection and segmentation from the whole spine mr images. *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, 27: 2527–2530, January 2006.

[19] Y. Kim and D. Kim. A fully automatic vertebra segmentation method using 3d deformable fences. *Computerized Medical Imaging and Graphics*, 33(5):343–352, July 2009.

[20] C.L. Hoad and A.L. Martel. Segmentation of mr images for computer-assisted surgery of the lumbar spine. *Physics in Medicine and Biology*, 47:3503–3517, 2002.

[21] A. Wong, A. Mishra, J. Yates, P. Fieguth, D.A. Clausi, and J.P. Callaghan. Intervertebral disc segmentation and volumetric reconstruction from peripheral quantitative computed tomography imaging. *Biomedical Engineering, IEEE Transactions on*, 56:2748–2751, November 2009.

[22] Steven P. Cohen, Charles E. Argoff, and Eugene J. Carragee. Management of low back pain. *British Medical Journal*, 337:100–1006, 2008.

[23] L. Lemieux, A. Hammers, T. Mackinnon, and R.S. Liu. Automatic segmentation of the brain and intracranial cerebrospinal fluid in t1-weighted volume mri scans of the head, and its application to serial cerebral and intracranial volumetry. *Magnetic Resonance in Medicine*, 49:872–884, May 2003.

[24] S. Kobashi, T. Takae, Y. Hata, Y.T. Kitamura, T. Yanagida, and O.Ishikawa amd M. Ishikawa. Automated segmentation of the cerebrospinal fluid and the lateral ventricles from human brain mr images. *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, 4:1961–1966, July 2001.

[25] Urs E. Ruttimann, Eileen M. Joyce, Daniel E. Rio, and Michael J. Eckardt. Fully automated segmentation of cerebrospinal fluid in computed tomography. *Psychiatry Research: Neuroimaging*, 50:101–119, June 1993.

[26] Andriy Fedorov, Ron Kikinis, Ginger Li, Chris Wyatt, Martin Styner, Kilian Pohl, Hans Johnson, and Marcel Prastawa. Measuring alcohol and stress interaction and vervetmrilongitudinalanalysis. *NA-MIC NCBC Collaboration*, 10:1, January 2010.

[27] S. Jacob. *Atlas of Human Anatomy.* Churchill Livingstone, 2002. 255 pp.

[28] Kenneth Saladin. *Human Anatomy.* McGraw-Hill, 2007 edition, 2007. 807 pp.

[29] Frank Henry Netter. *Atlas of Human Anatomy.* Saunders, 4 edition, July 2006. 640 pp.

[30] Renate Huch and Christian Bauer. *Mensch, Körper, Krankheit.* Urban & Fischer, 4 edition, 2003. 502 pp.

[31] Elaine Nicpon Marieb and Katja Hoehn. *Human anatomy & physiology.* Pearson Education, Inc., 7 edition, 2007. 347 pp.

[32] Henry Gray. *Anatomy of the human body.* Philadelphia, 2000. 1396 pp.

[33] Linda J. Vorvick, C. Benjamin Ma, and David Zieve. Cauda equina. A.D.A.M., Inc., 2009.

[34] SCC. Primary cns lymphoma treatment. Siteman Cancer Center, 2008.

[35] H. Weber. Lumbar disc herniation. a controlled, prospective study with ten years of observation. *Spine*, 8:131–140, 1983.

[36] Nouzhan Sehati. Lumbar discectomy. sehati.org, 2010.

[37] Richard G. Fessler, Robert E. Isaacs, and Laurie Rice-Wyllie. Lumbar discectomy: Minimally invasive spine surgery. SpineUniverse.com, 2010.

[38] eOrthopod. A patient's guide to lumbar discectomy. eOrthopod.com, 2003.

[39] Josef F. Bille and Wolfgang Schlege. *Medizinische Physik 2: Medizinische Strahlenphysik.* 2. Springer, 1 edition, May 2002. 548 pp.

[40] E. Mark Haacke, Robert W. Brown, Michael R. Thompson, and Ramesh Venkatesan. *Magnetic Resonance Imaging: Physical Principles and Sequence Design.* John Wiley & Sons, 1 edition, July 1999. 914 pp.

[41] Dominik Weishaupt, Victor D. Koechli, and Borut Marincek. *How does MRI work?: An Introduction to the Physics and Function of Magnetic Resonance Imaging.* Springer, 2 edition, July 2006. 170 pp.

[42] Joseph P. Hornak. The basics of mri. cis.rit.edu, 2010.

[43] DRH and Associates. Mri specialists in neurology, spine and orthopaedics. DRH & Associates, Inc, 2010.

[44] Paul C. Lauterbur. Image formation by induced local interactions: Examples employing nuclear magnetic resonance. *Nature*, 242:190–191, 1973.

[45] R. Damadian, M. Goldsmith, and L. Minkoff. Nmr in cancer: Xvi. fonar image of the live human body. *Physiological Chemistry and Physics*, 9:97–100, 1977.

[46] W. S. Hinshaw, P. A. Bottomley, and G. N. Holland. Radiographic thin-section image of the human wrist by nuclear magnetic resonance. *Nature*, 270:722–723, 1977.

[47] Mark A. Brown and Richard C. Semelka. *MRI: Basic Principles and Applications.* Wiley-Blackwell, 4 edition, May 2010. 264 pp.

[48] Sunder S. Rajan. *MRI: a conceptual overview.* Springer, 1 edition, October 1997. 167 pp.

[49] M. Reiser and W. Semmler. *Magnetresonanztomographie.* Springer, 1 edition, January 1992. 706 pp.

[50] Paul A. Bottomley, Thomas H. Foster, Raymond E. Argersinger, and Leah M. Pfeifer. A review of normal tissue hydrogen nmr relaxation times and relaxation mechanisms from 1-100 mhz: dependence on tissue type, nmr frequency, temperature, species, excision, and age. *Medical Physics*, 11(4):425–448, 1984.

[51] Scott A. Huettel, Allen W. Song, and Gregory McCarthy. *Functional Magnetic Resonance Imaging.* Palgrave Macmillan, 2 edition, February 2009. 510 pp.

[52] S. Sasaki, K. Kaji, and K. Shiba. Upper thoracic disc herniation followed by acutely progressing paraplegia. *Spinal Cord*, 43:741–745, July 2005.

[53] MR-TIP. Magnetic resonance - technology information portal, August 2010.

[54] B. Ostendorf, A. Scherer, and M. Schneider. Spezielle diagnostische techniken. Deutsche Gesellschaft für Rheumatologie, May 2000.

[55] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing.* Prentice Hall, 2 edition, January 2002. 793 pp.

[56] Gregory A. Baxes. *Digital Image Processing: Principles and Applications.* John Wiley & Sons, 1 edition, September 1994.

[57] S. Pieper, M. Halle, and R. Kikinis. 3d slicer. *Proceedings of the 1st IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 1:632–635, April 2004.

[58] S. Pieper, B. Lorensen, W. Schroeder, and R. Kikinis. The na-mic kit: Itk, vtk, pipelines, grids and 3d slicer as an open platform for the medical image computing community. *Proceedings of the 3rd IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 1:698–701, April 2006.

[59] Luis Ibanez and Will Schroeder. *The ITK Software Guide - The Insight Segmentation and Registration Toolkit.* Kitware, Inc., 2.4 edition, November 2005. 787 pp.

[60] T. S. Yoo, M.J. Ackerman, W.E. Lorensen, W. Schroeder, V. Chalana, S. Aylward, D. Metaxas, and R. Whitaker. Engineering and algorithm design for an image processing api: a technical report on itk–the insight toolkit. *Stud Health Technol Inform*, 85:586–92, 2002.

[61] Inc. Kitware. *The VTK User's Guide - Install, Use and Extend The Visualization Toolkit.* Kitware, Inc., 5 edition, September 2006. 382 pp.

[62] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit - An Object-Oriented Approach to 3D Graphics.* Kitware, Inc., 4 edition, December 2006. 528 pp.

[63] William J. Schroeder, Kenneth M. Martin, and William E. Lorensen. The design and implementation of an object-oriented toolkit for 3d graphics and visualization. *Proceedings of the 7th conference on Visualization '96*, 1:93–102, 1996.

[64] Azriel Rosenfeld, Avinash C. Kak, and A. C. Kak. *Digital Picture Processing*, volume 1. Morgan Kaufmann, 2 edition, August 1982. 435 pp.

[65] Hartmut Dickhaus. *Bildverarbeitung (Lecture Notes).* Medizinische Informatik Dickhaus, 2008.

[66] Jang-Gyu Cha. 3d merge: A robust aid to spinal imaging. *GE Healthcare MR publication*, Spring 2009:26–29, 2009.

[67] Adrian Christian. *Living with spinal cord injury.* Demos Medical Publishing, Inc., 1 edition, 2004. 181 pp.

[68] eMedicineHealth. Lumbar laminectomy - facts about back pain.

[69] D.T. Gering, A. Nabavi, R. Kikinis, W.E.L. Grimson, N. Hata, P. Everett, F.A. Jolesz, and W.M. Wells III. An integrated visualization system for surgical planning and guidance using image fusion and interventional imaging. *Int Conf Med Image Comput Comput Assist Interv.*, 2:809–819, September 1999.

[70] Jeffrey G. Jarvik. Imaging of adults with low back pain in the primary care setting. *Neuroimaging Clinics of North America*, 2:13, 2003.

[71] Andre A. Lighvani and Elias R. Melhem. Advances in high-field mr imaging of the spine. *Applied Radiology*, 38:18–27, 2009.

[72] Ken Martin and Bill Hoffman. *Mastering CMake - A Cross-Platform Build System.* Kitware, Inc., 4 edition, February 2008. 385 pp.

[73] NIH. Handout on health: Back pain. National Institutes of Health, July 2009.

[74] Xenophon Papademetris and Alark Joshi. *An Introduction to Programming for Medical Image Analysis with the Visualization Toolkit.* BioImage Suite, 2 edition, November 2009. 283 pp.

[75] Sidney M. Rubinstein and Maurits van Tulder. A best-evidence review of diagnostic procedures for neck and low-back pain. *Best Practice & Research Clinical Rheumatology*, 22:12, 2008.

[76] Ehud J. Schmidt, Ajit Shankaranarayanan, Sylvain Jaume, Giovanna Danagoulian, Srinivasan Mukundan, and Krishna S. Nayak. Wide-band steady state free precession with small diffusion gradients for spine imaging: Application to superior nerve visualization. *Proceedings of ISMRM 2010*, 1:2, 2010.

[77] Amy Henderson Squillacote. *The ParaView Guide - A Parallel Visualization Application.* Kitware, Inc., 3 edition, February 2008. 366 pp.

A. Appendix

```
 1 project(BinaryThresholding)

 3 cmake_minimum_required(VERSION 2.6)

 5 find_package(Slicer3 REQUIRED)
 6 include(${Slicer3_USE_FILE})

 8 slicer3_set_default_install_prefix_for_external_projects()

10 set (CLP BinaryThresholding)
11 set (${CLP}_SOURCE ${CLP}.cxx)
12 generateclp(${CLP}_SOURCE ${CLP}.xml)

14 add_executable(${CLP} ${${CLP}_SOURCE})
15 slicer3_set_plugins_output_path(${CLP})

17 target_link_libraries (${CLP}
18   vtkImaging       #vtkImageThreshold.h
19   ITKIO            #itkImageFileReader.h, itkImageFileWriter.h
20   ITKCommon        #itkOrientedImage.h
21   ITKBasicFilters  #itkChangeInformationImageFilter.h
22  )

24 enable_testing()
```

Listing A.1: `CMakeLists.txt`

```
<?xml version="1.0" encoding="utf-8"?>

<executable>
  <title>BinaryThresholding</title>
  <description>ITK reading and writing, processing as VTK</description>

  <parameters>
    <label>Input and Output</label>
    <description>The input and output parameters</description>

    <image>
      <name>MyInputVolume</name>
      <index>0</index>
      <label>Input Volume</label>
      <channel>input</channel>
    </image>

    <image type="label">
      <name>MyOutputVolume</name>
      <index>1</index>
      <label>Output Volume</label>
      <channel>output</channel>
    </image>
  </parameters>

  <parameters>
    <label>Binary Thresholding Parameters</label>
    <description>The parameters of the binary threshold</description>

    <integer>
      <name>MyLowerThreshold</name>
      <index>2</index>
      <label>Lower Threshold</label>
      <default>2200</default>
    </integer>

    <integer>
      <name>MyUpperThreshold</name>
      <index>3</index>
      <label>Upper Threshold</label>
      <default>4000</default>
    </integer>
  </parameters>

</executable>
```

Listing A.2: `BinaryThresholding.xml`

```
#include <iostream>
#include "BinaryThresholdingCLP.h"
// ITK classes
#include "itkOrientedImage.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"
#include "itkChangeInformationImageFilter.h"
// VTK classes
#include "vtkImageThreshold.h"
// converter classes
#include "itkImageToVTKImageFilter.h"
#include "itkVTKImageToImageFilter.h"

int main(int argc, char * argv [])
{
  PARSE_ARGS;

  typedef itk::OrientedImage<short, 3> OrientedImageType;
  OrientedImageType::Pointer orientedImage = OrientedImageType::New();

  // read ITK image
  typedef itk::ImageFileReader<OrientedImageType> ReaderType;
  ReaderType::Pointer reader = ReaderType::New();
  reader->SetFileName(MyInputVolume.c_str());
  reader->Update();

  // convert ITK image to VTK
  typedef itk::ImageToVTKImageFilter<OrientedImageType> ConvertITKtoVTKType;
  ConvertITKtoVTKType::Pointer convertITKtoVTK = ConvertITKtoVTKType::New();
  convertITKtoVTK->SetInput(reader->GetOutput());
  convertITKtoVTK->Update();

  // threshold VTK image
  vtkImageThreshold *imageThreshold = vtkImageThreshold::New();
  imageThreshold->SetInput(convertITKtoVTK->GetOutput());
  imageThreshold->ThresholdBetween(MyLowerThreshold, MyUpperThreshold);
  imageThreshold->SetInValue(1);
  imageThreshold->SetOutValue(0);
  imageThreshold->Update();

  // convert VTK image to ITK
  typedef itk::VTKImageToImageFilter<OrientedImageType> ConvertVTKtoITKType;
  ConvertVTKtoITKType::Pointer convertVTKtoITK = ConvertVTKtoITKType::New();
  convertVTKtoITK->SetInput(imageThreshold->GetOutput());
  imageThreshold->Delete();

  // change ITK image direction
  typedef itk::ChangeInformationImageFilter<OrientedImageType> ChangeInformationType;
  ChangeInformationType::Pointer changeInformationFilter=ChangeInformationType::New();

  changeInformationFilter->SetInput(convertVTKtoITK->GetOutput());
  changeInformationFilter->ChangeDirectionOn();
  changeInformationFilter->SetOutputDirection(reader->GetOutput()->GetDirection());
  changeInformationFilter->Update();

  // write ITK image
  typedef itk::ImageFileWriter<OrientedImageType> WriterType;
  WriterType::Pointer writer = WriterType::New();
  writer->SetInput(changeInformationFilter->GetOutput());
  writer->SetFileName(MyOutputVolume.c_str());
  writer->Write();

  return EXIT_SUCCESS;
}
```

Listing A.3: `BinaryThresholding.cxx`

Declaration

I hereby certify that I have written this thesis independently, solely based on the literature and other sources listed in the bibliography and properly cited in the text. This document will only be submitted to the University of Heidelberg and the University of Heilbronn in partial fulfillment of the requirements for the degree of "Diplom-Informatiker der Medizin".

Heidelberg, 17th November 2010

Martin Löpprich

Student ID HN: 164264
Student ID HD: 2462091