Supervised Machine-Learning Framework and Classifier Evaluation for Automated Three-dimensional Medical Image Segmentation based on Body MRI

Master Thesis

Patrick Frischmann, BS

(Patrick.Frischmann@edu.fh-kaernten.ac.at)

ID: 1110310002

Klagenfurt, October 2013

Performed at the

Fachhochschule Kärnten/Carinthia University of Applied Sciences

School of Medical Information Technology

and

The University of Texas MD Anderson Cancer Center

Department for Diagnostic Radiology

supervised by

Marvin D. Hoffland, MS

Naveen Garg, MD

(Patrick Frischmann)

Abstract

This paper presents a supervised machine-learning framework for automated segmentation of prostate tissue to include localization and the further zonal segmentation of the prostate gland within an MRI volume of the human body. The proposed image-processing algorithm in combination with high detection accuracy of prostate structures offered by MRI enables accurate evaluations of prostate volume; thereby improving prostate cancer assessment and treatment planning. Volumetric measurements following this technique are superior to an estimation of prolate ellipsoid volume based on orthogonal measurements, while simultaneously time-consuming manual segmentation of prostate zones is eliminated.

Segmentation is performed utilizing a pattern recognition approach based on classification of features that are extracted from the input data. Method accuracy was compared to prostate contours outlined by expert radiologists and median Dice similarity coefficients of 0.84 for the central gland and 0.70 for the peripheral zone were recorded. On average, the predicted volume matches the ground truth by 97% with a fluctuation range of less than ±15%. Results are averaged over a data set of 100 random patient cases that are subject to variations in magnetic field strength, image resolution and usage of endorectal coils. The proposed completely automated algorithm enables fast prediction times for prostate zone labels and is generalizable to other anatomical structures in the human body as well.

Key words: automated prostate zone segmentation, prostate volume estimation, neural network, random forest, three-dimensional feature classification

(Patrick Frischmann)

Kurzfassung

Diese Masterarbeit beschreibt ein Anwendungs-Framework zur automatischen Segmentierung von Prostatagewebe in MRT-Schnittbildern. Der Ansatz basiert auf überwachtem maschinellen Lernen und unterstützt die Lokalisierung und weiterführende Zonensegmentierung der Prostata. Der vorgestellte Bildverarbeitungsalgorithmus ermöglicht in Kombination mit der genauen Darstellung von Prostatastrukturen durch die Magnetresonanztomographie eine präzise Evaluierung des Volumens der Prostata. Dadurch werden die medizinische Beurteilung und Behandlungsplanung von Prostatakrebs unterstützt. Die vorgestellte volumetrische Messmethode ist der Volumenschätzung basierend auf einem gestreckten Rotationsellipsoid im Genauigkeitsgrad überlegen und zudem sind keine zeitintensiven manuellen Segmentierungen der Prostata erforderlich.

Das Segmentierungsproblem wird durch Klassifikation von Merkmalen gelöst, welche aus den Eingangsbildern extrahiert werden. Die Genauigkeit dieser Methode wird über den Vergleich mit manuellen Segmentierungen (durchgeführt von Radiologen) beschrieben, wobei mittlere Dice-Koeffizienten von 0,84 für die zentrale Zone und 0,70 für die periphere Zone erreicht wurden. Im Mittel stimmte das prognostizierte Volumen zu $97\% \pm 15\%$ mit den Ground Truth-Daten überein. Diese Resultate wurden über einen Datensatz von 100 zufällig gewählten Patientenakten gemittelt, wobei sich die einzelnen MRT-Aufnahmen unter anderem durch die genutzte magnetische Feldstärke, Bildauflösung und Verwendung von endorektalen Spulen unterscheiden. Der vorgestellte vollautomatische Segmentierungsalgorithmus ermöglicht schnelle Prognosen für die Bestimmung von Prostatazonen und ist generalisierbar für die Anwendung auf andere anatomische Gewebestrukturen im menschlichen Körper.

Suchbegriffe: automatische Segmentierung von Prostatazonen, Volumenschätzung der Prostata, Neuronales Netzwerk, Random Forest, dreidimensionale Merkmalserkennung

Acknowledgements

I would like to express my gratitude to my supervisors Marvin D. Hoffland and Naveen Garg for their useful comments, remarks and engagement throughout the project work of this master thesis. I really appreciate the support and guidance, which helped me a lot during my research and writing of this thesis.

In addition, I would like to thank the Austrian Marshall Plan Foundation[1] and Industriellenvereinigung Kaernten[2] for their trust and financial support. This research project was made possible due to the Marshall Plan scholarship and Exzellenzstipendium 2013 that I thankfully received.

[1]Austrian Marshall Plan Foundation, Marshallplan Jubilaeumsstiftung, Walcherstrasse 11A, 1020 Vienna, Austria, `http://www.marshallplan.at`, date accessed: October 2013.

[2]Industriellenvereinigung Kaernten, Dr.-Franz-Palla-Gasse 21, 9020 Klagenfurt, Austria, `http://www.iv-kaernten.at/`, date accessed: October 2013.

Contents

Chapter 1

Introduction

The research described in this paper is concerned with the development of a supervised machine learning framework for segmentation purposes. An approach to achieve automated medical image segmentation tasks based on magnetic resonance (MR) images of the human body is presented. The motivation for this research in terms of medical and technical aspects as well as related studies are addressed in this introductory chapter.

Due to increased interest of the radiology department of the MD Anderson Cancer Center and the participation at the 2013 ISBI[1] Grand Challenge for *Automated Segmentation of Prostate Structures*[2], this paper strongly focuses on prostate tissue. Therefore, Chapter 2 provides insights into medical prostate imaging and prostate gland anatomy. Despite focusing on a specific organ throughout the development, the presented approach is generalizable to other structures in the human body as well, because the input transformation and resulting representation are universally applicable. Chapter 2 also provides a fundamental background for image segmentation, classification and supervised learning techniques. These theoretical background sections should improve the reader's understanding for upcoming sections and proposed methodology.

Materials and methods utilized to achieve automatic segmentation tasks are described step-by-step in Chapter 3. The integration of the whole segmentation chain into a framework enables high usability for further research purposes. Functionality includes a viewer for medical images that offers the possibility to draw contours for regions of interest. Various classifiers are implemented and methods for error metric calculations are provided to offer objective, reliable and reproducible evaluations.

Experiments have been made with various classifier implementations, different settings and multiple test sets. The respective results are discussed in Chapter 4, utilizing several plots for output visualization. This chapter also includes evaluations of volumetric measurements.

The thesis finalizes with a result discussion and conclusion in Chapter 5. Benefits of this automatic segmentation approach are explained and limitations are demonstrated. An outlook on future possibilities is provided and the work is summarized.

[1]2013 ISBI Grand Challenges, International Symposium on Biomedical Imaging, `http://www.biomedicalimaging.org/2013/program/isbi-challenges/`, date accessed: June 2013.

[2]National Cancer Institute, The Cancer Imaging Archive research projects, `https://wiki.cancerimagingarchive.net/x/8QRp`, date accessed: June 2013.

1.1 Medical background of the prostate gland

Prostate cancer is the most frequently diagnosed cancer in men after skin cancer and the second leading cause of cancer deaths in men in the United Kingdom and the United States [1],[2],[3]. Based on incidence rates, an estimated 238,590 new cases of prostate cancer are expected to occur in the United States in 2013 and according to the American Cancer Society 29,720 men will die of the disease in 2013 [1]. The death rate has dropped since the mid 1990s, which may be a result of performing early screening and treatment improvements [2], [4]. The 5-year survival rate at local and regional stages amounts to 100% but drops significantly to 28% in case of distant metastases [1]. The slow natural disease progression and difficulty in accurate staging make prostate cancer management a complex and controversial issue [5]. There are multiple treatment options including radical prostatectomy, hormonal treatment and various forms of radiation therapy [2], [5]. Similar to other types of diseases, prostate cancer is most effectively treated when diagnosed early [4], [6].

The commonly used methods for early prostate cancer detection are prostate-specific antigen (PSA) screening and/or digital rectal examination (DRE) [2], [4]. If either the serum PSA value is elevated or DRE findings are abnormal, the presence of carcinoma is indicated and performing a biopsy for confirmation is highly recommended [2], [7]. The universally approved method and current gold standard for prostate cancer diagnosis is a transrectal ultrasound (TRUS)-guided biopsy followed by a histopathological examination [2], [6], [8]. Due to the limited visualization capabilities of ultrasound imaging, a sextant approach is commonly used to increase the likelihood of obtaining actual cancer tissue and thus reducing the need of repeated biopsies [2], [6]. Nevertheless, studies showed that approximately 20% of repeated biopsies show cancer in men with elevated PSA levels above 4.0 ng/mL following negative initial biopsy results [8], [9], [10]. It has also been found that biopsy-detected prostate cancer is not rare among men with normal levels of serum PSA (less than 4.0 ng/mL) [11], [12]. Overall it is estimated that the false negative rate of TRUS-guided biopsies lies between 15% and 34% [6]. In these cases where pathological diagnosis cannot be confirmed despite indications of cancer by PSA/DRE tests, magnetic resonance imaging (MRI) was found to be helpful [6], [8].

MRI offers advantages over ultrasound and is considered to be the most sensitive imaging modality for displaying anatomical regions of the prostate [6], [13]. On multiplanar T2-weighted images the zonal anatomy is well shown and tumor-suspected areas within the peripheral zone are identifiable [2], [3], [8]. The clear picture can also show if the cancer has spread outside of the prostate into the seminal vesicles or other structures that are located nearby [4]. By incorporating MRI, the prostate cancer detection sensitivity for predicting positive biopsies lies between 57% and 100% with specificity values in the range of 44% to 96% [8]. For cancer detection, MRI offers notably higher sensitivity compared to TRUS; however, low specificity values limit its use and are the reason that transrectal biopsy currently cannot be replaced as a diagnosis step [6], [8]. Taking this into consideration, MRI still has a high value in treatment planning and as a staging tool [3]. At the same time it helps to detect prostate cancer in patients with prior tumor-negative TRUS-guided prostate biopsies [3], [8], [14]. Since 2000, MRI-guided biopsies have gained increasing popularity and studies have shown that prostate cancer was detected in more than 40% of patients with one or more previous negative TRUS-guided biopsy sessions (compared to 20% to 35% from conventional repeated biopsies) [8].

Due to its high detection accuracy of prostate structures compared to ultrasound, MR imaging allows a better assessment of prostate volume (PV) [6], [15], [16]. Prostate volume is an important parameter especially as an indicator for treatment outcome and as a component of treatment planning for radiation therapy [6], [17], [18]. With the current transrectal ultrasound standard, three orthogonal measurements are performed and the PV is estimated according to a formula for ellipsoid volume calculation [15]. In contrast to this method, which is subject to erroneous discrepancies, MRI offers the possibility of accurate segmentations [15]. However, manual segmentation is time consuming and rather subjective to the respective radiologist and is therefore prone to bias and drift [15], [19], [20]. Thus, automatic segmentation approaches of the prostate in MRI is of increasing interest and has already shown better correlations to manual expert segmentations compared to automated evaluations involving ultrasound imaging [15], [16].

1.2 Technical background and related work

Abnormality detection in medical images plays a key role in image interpretation and hence provides vital information for the ability to make a diagnosis [21]. However, the detection of abnormalities in images is a high-level image processing step that is enabled by performing previous tasks, e.g. object segmentation [22]. Segmentation denotes the separation of an object from the background in an image [13]. For medical diagnosis, tissue segmentation is the first step for image analysis applications [19]. The easiest and at the same time most reliable way to obtain segmentation results of regions of interest is manual segmentation through visual observation by medical doctors [23]. This manual task is very time consuming and therefore practically not feasible in the majority of patient cases due to the high number of imaging examinations. Hence, it is important to provide software applications that are capable to achieve automated medical image segmentation [19], [23], [24]. If regions of interest are segmented within an image, properties that describe the structure can be extracted. Those characteristics can be compared to what are considered normal values and abnormalities can be detected [22]. The validity of the final result strongly depends on the accuracy and robustness of the initial segmentation that enabled the high-level image analysis [23].

In this work a supervised machine-learning framework for medical image segmentation is proposed. Hence, research has been conducted to enable a good segmentation capability within magnetic resonance images in order to either provide a basis for high-level image processing tasks or to directly support medical doctors in making their diagnosis or evaluation. To be useful in practical clinical applications, an automated algorithm must be accurate similar to an expert, has to offer reasonably fast processing time and has to be robust in terms of diverse datasets [16], [19]. However, computer-aided prostate segmentation (whether automated or semi-automated) is a very challenging task due to strong inter-patient variabilities in shape, size and deformations of the prostate gland [18], [20]. In addition, noise, imaging artifacts and inhomogeneities in intensity values inside the prostate regions are introduced by the imaging device [20]. Moreover, research publications in this field are limited as well [16].

Despite ultrasound being the dominant imaging modality for prostate diagnosis, approaches for MRI have been developed [13]. For example, Zwiggelaar *et al.* in [25] proposed a semi-automated technique to perform 2D contouring of MRI slices in polar-transform space (to take advantage of the elliptical shape of the prostate). Likewise, Vikal *et al.* in [26] utilize a 2D approach to find a region of interest on each slice and afterwards form a 3D surface. However, one major part of the problem of prostate segmentation in MR images is achieving a shape constraint of the final segmentation result [13]. Some authors have tried to solve this problem through the use of active shape models. Zhu *et al.* published a hybrid approach employing a 2D active shape model and 3D optimization to achieve segmentation, described in [27] and [28]. Allen *et al.* extended the technique to combine 3D shape modeling with voxel classification in [15] and [29], similar to Makni *et al.* who in addition use Markov fields to describe contextual information [30]. Extended approaches involving deformable models are also used recently, amongst others proposed by Gao *et al.* in [13] and Toth *et al.* in [31].

Over the last years, reasonable results have been achieved in the field of atlas-based prostate segmentation as well, e.g. by Martin *et al.* in [16] and [32], Dowling *et al.* in [33] and Klein *et al.* in [34]. These approaches involve the construction of an atlas from training shapes of the prostate and the subsequent matching of this atlas under local constraint criteria, which is an image registration problem.

More recently, progress has been made in the field of utilizing machine learning (ML) approaches for medical image analysis. Machine learning is a field of computer science that is concerned with the development of systems that are capable to learn from a provided input. The goal is that knowledge gained from observed samples contributes to improve decision-making tasks for similar new input in the future [35]. ML is extensively used in face detection/recognition [35] and speech recognition [36] tasks. In order to be able to make class assignments, algorithms for classification tasks need to learn to automatically recognize complex patterns [35]. Therefore, this method performs well as long as pattern recognition is feasible in the input (feature) space and class labels are sufficiently distinguishable [37].

Experimental research for medical segmentation problems has been performed with various classifiers. Decision forests are utilized for example by Ghose *et al.* in [20] and Geremia *et al.* in [38]. Also support vector machines (SVMs) [39], [40] and neural networks [19], [41] have been applied to solve medical segmentation problems. So far, none of the classifiers shows superior performance over another and comparisons are hard to achieve as classification results almost solely rely on input selection. However, it can be said that supervised learning based segmentation methods that use manually segmented training data as a reference show superior performance and are computationally efficient [37].

This work is important because automated prostate structure segmentation provides information about the size, shape, position and volume of the prostate. These characteristics increase the knowledge of the gland, which in turn could affect multiple clinical routines like prostate cancer treatment selection and treatment planning procedures. Besides, the prostatic volume correlates with the presence of prostate cancer and is important for pathologic staging [42], [43]. Moreover, interventional techniques such as MR-guided biopsies could be influenced and diagnostic uncertainties might be reduced [6].

Chapter 2

Theoretical imaging and image processing framework

In this chapter, the basic theoretical background of the issues addressed in this thesis is provided. Firstly, a brief overview of prostate gland anatomy is given for better understanding of zonal segmentation goals. Moreover, the MR imaging appearance of prostate tissue is illustrated in Section 2.1. Secondly, Section 2.2 concentrates on explanations of machine learning principles and the basics of segmentation and classification, including the mathematical background. Lastly, the concepts of two major classifiers, namely neural networks and random forest are explained.

2.1 Prostate gland anatomy and MR imaging appearance

This section addresses the anatomy of the prostate gland as this knowledge is crucial for image interpretation. Figure 2.1 shows a schematic illustration of the gland's zonal anatomy. The prostate can be separated into glandular and non-glandular regions. The main non-glandular elements are the anterior fibro-muscular stroma and the prostatic urethra that runs vertically through the prostate from its base to apex. The glandular prostate is subdivided into an inner component, which is basically comprised of the transitional zone and into outer components. The outer components consist of central and peripheral zones [3].

As it can be seen in Figure 2.1, the peripheral zone (PZ) is the largest part of the prostate, making up 70% of its volume in young men. The peripheral zone is located predominantly lateral and posterior to the central zone, which is the second largest component and accounts for about 25% of prostate tissue in young men. The remaining 5% form the transition zone. Due to the fact that prostate zones are defined histologically, many prostatic diseases have a zonal distribution. Seventy percent of adenocarcinomas arise in the peripheral zone, 20% in the transition zone and only 10% in the region of the central zone [2], [3], [44].

On T1-weighted MR images, prostate zones cannot be distinguished because the normal prostate gland is represented with uniform intermediate-to-low signal intensity

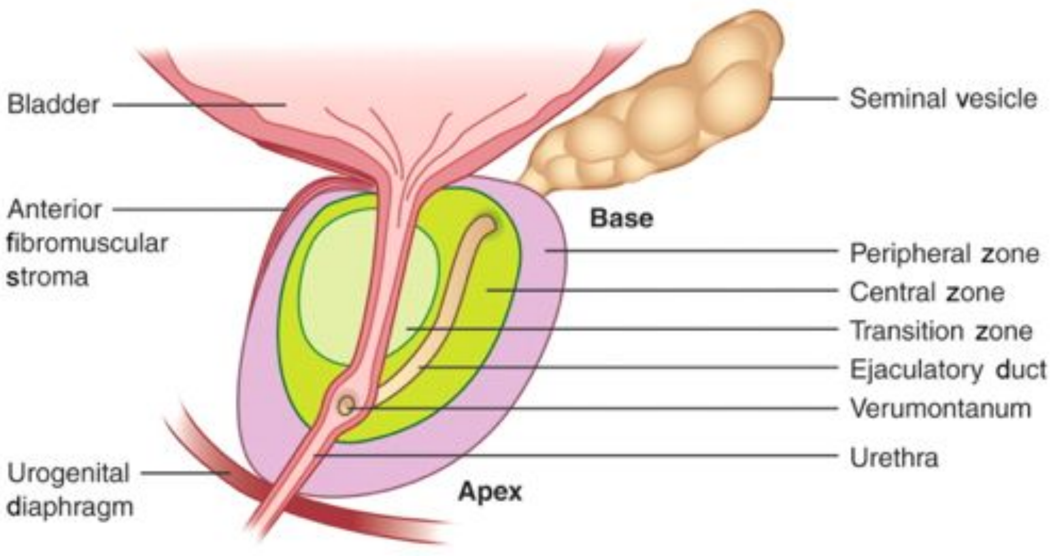


Figure 2.1: Schematic illustration of prostate zonal anatomy in sagittal view. [2]

[2], [3], [5]. Similar to computed tomography (CT), the soft-tissue contrast resolution of T1-weighted MR imaging is insufficient to visualize the intra-prostatic anatomy [3]. However, on a T2-weighted image the prostatic zones are well depicted, mainly because of the shorter T2 relaxation time of the central zone [2], [3], [5], [7].

Figure 2.2 shows a series of T2-weighted MR images from one patient case. Throughout this paper the terms *image stack* (referring to the series of images comprising one case) and *image slice* (denotes one image out of the stack) are used. The peripheral zone demonstrates high T2 signal intensity and is subject to an age-related intensity increase [2]. A thin rim of low signal intensity, referred to as the anatomic or true capsule surrounds the peripheral zone [2], [3]. Compact smooth muscle fibers and sparse glandular elements are the reason for short T2 relaxation times that lead to low signal intensities of the central and transition zones [2], [44]. They both have similar signal intensities and therefore can be identified best when considering their anatomical locations [3]. Due to the respective MR imaging appearance in further sections of this work, the terms central gland (CG), which refers collectively to the periurethral, central, and transition zones, and peripheral gland (includes only the peripheral zone) are used. An additional reason for this region grouping is that for clinical use the anatomical distinction between central and peripheral glands is of most importance [44].

The current clinical standard is to perform prostate MRI at magnetic field strengths of at least 1.5 tesla (T) [2]. Endorectal coils are used to obtain high-resolution images for accurate localization and staging of prostate cancer [2]. For cancer assessment, usually a conventional T1 and T2-weighted MRI is combined with functional techniques such as MR spectroscopy, diffusion-weighted imaging or dynamic contrast-enhanced MRI to improve sensitivity and specificity [2], [3], [7].

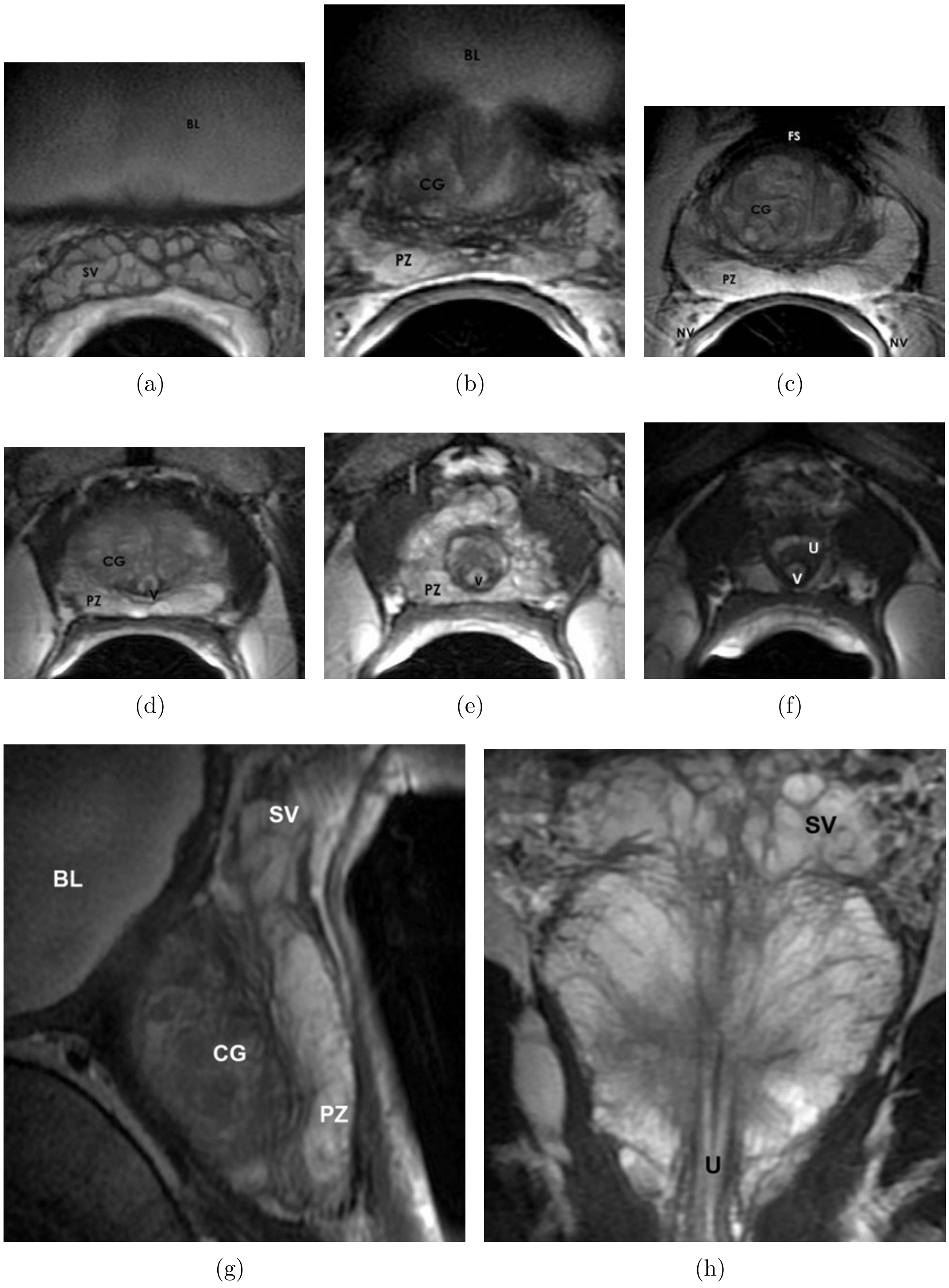


Figure 2.2: MR prostate imaging example of a 50-year-old man. SV = seminal vesicle, BL = urinary bladder, CG = central gland (central zone and transition zone), PZ = peripheral zone, FS = anterior fibro-muscular stroma, NV = neurovascular bundle, V = verumontanum, U = urethra. (a)-(f): axial T2-weighted images, (g): parasagittal T2-weighted image, (h): coronal T2-weighted image. Zonal anatomy is shown at the level of seminal vesicles in (a), base of prostate gland in (b), mid gland in (c) and (d), apex of gland in (e) and membranous urethra in (f). [2]

2.2 Segmentation, classification and supervised learning

Segmentation is the process of dividing an image into regions with reasonably similar properties such as gray-level or texture [24]. In the field of medical image analysis, several objectives for performing a segmentation can be identified, such as

- study of anatomical structures
- identification of a certain region of interest (e.g. abnormalities like tumors)
- assistance in treatment planning (e.g. in radiation therapy to calculate dose of radiation)
- measurement of treatment outcome (e.g. tissue volume that reflects tumor size).

Automatic segmentation of medical images is a difficult task because structures in the human body rarely have any homogeneous properties when they are recorded with an imaging device. There is a high chance that soft-tissue regions which belong to different anatomical structures have very similar intensity values. Moreover, the presence of artifacts and the partial volume effect further increase the problem complexity [24].

Classification in contrast to segmentation aims at the assignment of a tissue class to each point in the image, where the classes are specified in advance. In cases of the brain for example, the classes gray matter, white matter and cerebrospinal fluid can be identified. The two problems of segmentation and classification are closely linked to each other as a classifier implicitly segments an image and a segmentation implies a classification. In this work, a segmentation result is achieved by classification of image pixels. Every pixel is categorized according to the properties that identify it as a member of a particular tissue class. This subsection explains the basic concept of utilizing machine learning for classification problems. The theoretical background provided in this section is strongly based on the books of C. M. Bishop [45] and R. O. Duda *et al.* [46].

In the field of pattern recognition, computer algorithms are used to automatically discover regularities within the data used. These regularities are then used to partition the data into a finite number of different categories. Despite uncertainties that arise during the partitioning process, optimal decisions should be made in every situation. Suppose an input vector $\mathbf{x}$ and corresponding target vector $\mathbf{t}$ are available. The targets represent the different categories, also called class labels. Uncertainties associated with these variables are described by the joint probability distribution $p(\mathbf{x}, \mathbf{t})$. The classification goal is to predict $\hat{t}$, given a new value $\hat{x}$. Predicting $\hat{t}$ is equivalent to assigning $\hat{x}$ to one of K discrete classes C_k, where $k = 1, ..., K$.

Consider, for example, the recognition of handwritten digits. A program should be developed that takes an input image $\mathbf{x}$ representing a handwritten digit and assigns classes $0, ..., 9$ as an output. Due to the high variability of handwriting this is a very hard problem. Hence, many handcrafted rules based on shapes need to be combined to achieve proper differentiations; however, in practice such an approach is not feasible and emerging exceptions of rules may lead to errors and poor results.

2.2.1 Supervised learning

Given these kind of problems, better results can be obtained through the implementation of a machine learning approach. The purpose of ML is to generate an adjustable *model* that represents the input data and alter its parameters using a large *training set*. Suppose $\mathcal{X}$ denotes the input space and $\mathcal{T}$ is the output space. Assume that a set $\mathcal{D}$ of training examples of N observations of x, written $\mathbf{x} = (x_1, ..., x_N)^T \in \mathcal{X}$, is given together with corresponding targets $\mathbf{t} = (t_1, ..., t_N)^T \in \mathcal{T}$:

$$\mathcal{D} = \{(x_1, t_1), ..., (x_N, t_N)\} \tag{2.1}$$

Applications where the training data are comprised of a set of input vectors along with their respective targets are called *supervised learning* problems. In order to obtain the correct categories represented by targets t_n, usually a manual labelling process needs to be performed through individual inspections of inputs x_n.

The purpose of a learning algorithm is to find a decision function $g : \mathcal{X} \rightarrow \mathcal{T}$ out of a space of possible functions $\mathcal{G}$, usually called *hypothesis space*. This function g represents the characteristics of the input-to-target mapping. It can also be represented by a scoring function $f : \mathcal{X} \times \mathcal{T} \rightarrow \mathbb{R}$ to define g through the returning value of t that gives the highest score:

$$g(x) = \arg\max_{t}\ f(x, t) \tag{2.2}$$

Once a decision function is found utilizing the training set, predictions of a target value $\hat{t}$ for a new input value $\hat{x}$ can be made. These new inputs that differ from the training set are referred to as the *test set*. Due to the high variability of input vectors in practical applications, a training set will only cover a small fraction of all possible inputs. Thus, it is important to achieve *generalization* to correctly classify novel examples out of a test set [36], [45].

2.2.2 Feature extraction

To reduce the variability within the input data, typically a preprocessing step is introduced to transform the original input values to a new space, illustrated in Figure 2.3. The goal of this transformation ϕ is to ease the pattern recognition problem by reducing the dimensionality when operating in the new space. This step is referred to as *feature extraction*. It involves finding appropriate representations for real world objects in order to map them into the *feature vector space*. It is essential to choose the right characteristics for input representation to encourage the formation of groups (or clusters) in the feature space. An ideal feature extractor would result in a representation that makes the classifier job trivial. It is important to note that once following this technique, also the test data need to be preprocessed by performing the same steps used for the training data. In addition to reducing the complexity of the classification problem, preprocessing is also performed to increase the computational efficiency [36], [45].

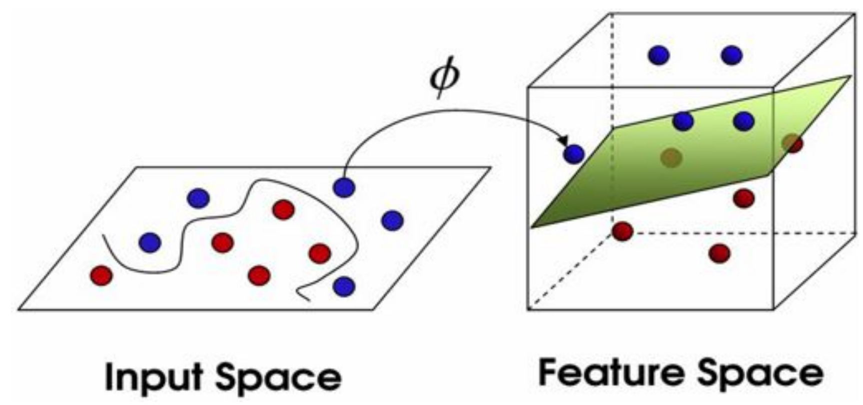


Figure 2.3: Illustration of input to feature space mapping for a two-class problem. By enforcing the transformation ϕ, inputs are converted to the feature space where classification tasks ideally are easier to achieve. In this case an optimal hyperplane for linear separation is found that solves the originally non-linear decision problem. [47]

2.2.3 Inference and decision-making

The remaining problem is to partition the feature space into separate regions by making optimal input assignments. This decision problem can be solved in one of three ways: using generative models, discriminative models or discriminative functions. Basically, classification is a task of recovering the model that generated some kind of explorable patterns; thus, the usability of different techniques depends on the type of underlying models. Statistical properties of the patterns are generally expressed in probability densities illustrated in Figure 2.4.

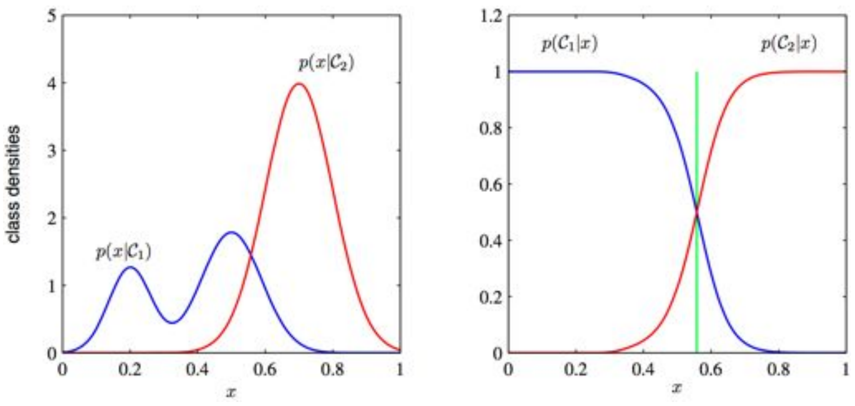


Figure 2.4: Example of class-conditional densities for two classes having a single input variable x (left plot) together with corresponding posterior probabilities (right plot). The green vertical line in the right plot represents the optimal decision boundary. [45]

1. **Generative models**
   With the use of generative models, the classification problem is basically solved in three steps. Within the *inference* step the training data are used to determine class-conditional densities $p(\mathbf{x}|C_k)$ and class priors $p(C_k)$. Afterwards, Bayes' theorem is used to find the posterior class probabilities $p(C_k|\mathbf{x})$. Figure 2.4 illustrates probabilities for a two-class case with one input value. In the *decision* step, optimal class

assignments can be made by using the learned posterior probabilities. This is the most demanding approach because it involves finding the joint probability $p(C_k, \mathbf{x})$. In addition, the training set has to be very large for high-dimensional data in order to determine the class-conditional probability densities to reasonable accuracy.

2. **Discriminative models**
   If only classification decisions should be made, it can lead to a waste of computational resources to obtain class-conditional densities as they might contain many structures that have little effect on posterior probabilities. Hence, it might be enough to solve the inference problem of finding the posterior class probabilities $p(C_k|\mathbf{x})$ and directly model them to class labels.

3. **Discriminant functions**
   It is also possible to solve these multiple steps at once. For this approach the training data are used to determine a *discriminant function* that directly maps inputs $\mathbf{x}$ to class labels C_k. In this case no probability estimation is necessary. In cases of one-dimensional data this corresponds to finding a certain threshold value that gives the minimum error rate classification (finding the x-value marked with a green line in Fig. 2.4).

This work primarily uses neural networks and decision trees for classification purposes; hence, the main focus lies on discriminant models and functions. The principles of the linear discriminant function will be described in the following paragraphs.

2.2.4 Linear basis function model

The simplest form of a linear function model is comprised of a linear combination of input variables

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + ... + w_D x_D \tag{2.3}$$

where $\mathbf{x} = (x_1, ..., x_D)^T$ is the input vector. The fact that y is modeled as a linear function of the input variables x_i poses a limitation to the model. This constraint can be bypassed through a model extension, where linear combinations of fixed nonlinear functions of the input $\phi(\mathbf{x})$ are formed resulting in

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}). \tag{2.4}$$

The total number of parameters is given with M. The parameter w_0 that is excluded from the sum represents a constant offset in the data. The offset w_0 is also referred to as *bias* parameter. If ϕ is nonlinear, y can now be a nonlinear function. The fixed quantity of nonlinear functions $\phi_j(\mathbf{x})$ is referred to as *basis functions*. If the basis function is defined in a way that $\phi_0(\mathbf{x}) = 1$, the model can be defined as

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) \tag{2.5}$$

where $\mathbf{w} = (w_0, ..., w_{M-1})^T$ and $\phi = (\phi_0, ..., \phi_{M-1})^T$. The basis functions $\phi_j(\mathbf{x})$ denote the feature extraction step, where the original variables form the vector $\mathbf{x}$. If nonlinear basis functions are used, the function $y(\mathbf{x}, \mathbf{w})$ is a nonlinear function of the input vector $\mathbf{x}$. However, $y(\mathbf{x}, \mathbf{w})$ is a linear function in $\mathbf{w}$ and therefore called a linear model.

2.2.5 Generalized linear model

Considering the model prediction $y(\mathbf{x}, \mathbf{w})$ from Equation 2.4 for the case where it is **linear in the input variables**, the model can be written as

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 \tag{2.6}$$

where y results in a real number. However, for classification problems discrete class labels should be predicted; thus, y needs to be mapped to discrete values. This transformation is achieved using a nonlinear *activation function* $f(\,\cdot\,)$, resulting in

$$y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + w_0). \tag{2.7}$$

Models of this class are called *generalized linear models*. They are no longer linear in the parameters $\mathbf{w}$ because of the transformation introduced by the nonlinear function $f(\,\cdot\,)$.

2.2.6 Linear discriminant function

The goal of implementing a discriminant function is to directly assign each input vector $\mathbf{x}$ to a specific class C_k. A linear discriminant function $y(\mathbf{x})$ for a **two-class problem** is obtained using the linear model of the input variables from Equation 2.6, where $\mathbf{w}$ is the *weight vector* and w_0 is the bias (also called *threshold*). Class assignments of the input vector are performed according to the function output. If $y(\mathbf{x}) \geq 0$, $\mathbf{x}$ is assigned to class C_1 and otherwise to C_2.

This formulation of linear discriminants can be extended to $K > 2$ classes by encompassing K linear functions in the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x} + w_{k0} \tag{2.8}$$

where $k = 1, ..., K$. The input $\mathbf{x}$ is then assigned to class C_k, if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$, thus

$$\hat{C}_k = \arg \max_{k=1,...,K} \; y_k(\mathbf{x}). \tag{2.9}$$

Hence, the classifier can be represented as a network that computes multiple discriminant functions and selects an output category corresponding to the largest discriminant. An illustration of such a network is provided in Figure 2.5. To reach more complex decision surfaces, additional terms need to be added to the decision function. The original linear discriminant function can be written as

$$y(\mathbf{x}) = w_0 + \mathbf{w}^T\mathbf{x} = w_0 + \sum_{i=1}^{D} w_i x_i \tag{2.10}$$

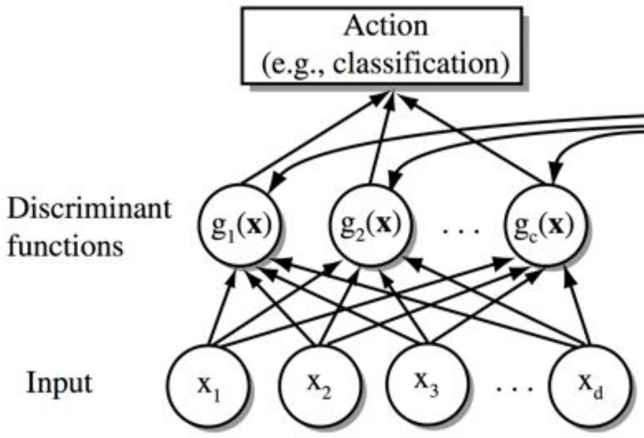


Figure 2.5: Functional structure of a general statistical pattern classifier that includes d inputs and c discriminant functions $g_i(\mathbf{x})$. Input categorization is performed by obtaining the maximum of the discriminant values incorporating costs of decision errors. The arrows show the information flow direction. [46]

with w_i being the respective components of the weight vector $\mathbf{w}$. The addition of terms involving the products of pairs of components of $\mathbf{x}$ results in the *quadratic discriminant function*

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D}\sum_{j=1}^{D} w_{ij} x_i x_j. \tag{2.11}$$

The decision surface of the quadratic discriminant function is a hyperquadratic surface. The class of *polynomial discriminant functions* can be obtained by continuing to add terms like $w_{ijk}x_ix_jx_k$.

2.2.7 The perceptron

One example for a linear discriminant model is the perceptron (cf. Rosenblatt, 1962). It is based on linear combinations of fixed basis functions. The perceptron is a two-class model similar to the one described in Equation 2.7. The difference is that in this case a nonlinear transformation is used to transform the input vector $\mathbf{x}$ to the feature vector $\phi(\mathbf{x})$. This results in the generalized linear form of the perceptron model

$$y(\mathbf{x}) = f(\mathbf{w}^T\phi(\mathbf{x})) \tag{2.12}$$

where $f(\,\cdot\,)$ is a nonlinear activation function. A bias $\phi_0(\mathbf{x})$ is typically included in the feature vector $\phi(\mathbf{x})$. The activation function $f(\,\cdot\,)$ is a step function of the form

$$f(a) = \begin{cases} +1 & \text{if } a \geqslant 0 \\ -1 & \text{if } a < 0. \end{cases} \tag{2.13}$$

The perceptron is a learning machine that learns from training vectors. During the training phase the weights $\mathbf{w}$ are altered according to minimize an error function, known as the *perceptron criterion*. If the output error is minimal, the optimal weights are found.

However, in order for the perceptron algorithm to converge, the target classes need to be linearly separable. A drawback of the perceptron is that it does not provide probabilistic outputs; hence, information about the degree of certainty or uncertainty is lost. Moreover, it does not generalize to $K > 2$ classes. Aside from these properties, the perceptron is also subject to the same limitation as the previously discussed models, which is that they are based on linear combinations of fixed basis functions. The difficulty is that the basis functions $\phi_j(\mathbf{x})$ are fixed prior to an observation of the training set. Hence, M basis functions along each dimension of the D-dimensional input space require a total of M^D basis functions. This is known as the *curse of dimensionality*.

2.2.8 Neural networks

For real-world problems involving demanding applications, classification models that comprise linear combinations of basis functions are not general enough. There are many problems for which the use of linear discriminants is not sufficient to achieve results with minimal error. In order to minimize the error in such cases, nonlinear basis functions need to be used [46]. The parameters defining those functions are learned and adjusted during training [45]. The *feed forward neural network* is an example for these kind of models. In the following, the basic neural network model is explained.

A neural network can be represented similarly to a linear model but with generalized basis functions $\phi_j(\mathbf{x})$ as

$$y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\mathbf{x})\right). \tag{2.14}$$

Neural networks use a fixed number of nonlinear basis functions in parametric form. During the network training, the parameters of the basis functions $\phi_j(\mathbf{x})$ are adjusted together with the coefficients $\{w_j\}$. The parametric nonlinear basis functions are constructed in a way that they follow the basis function form in Equation 2.14. Every nonlinear function $\phi_j(\mathbf{x})$ is a linear combination of D inputs. The coefficients in the linear combination are adaptive parameters. This leads to the definition of the first network layer.

Given the input variables $x_1, ..., x_D$, a total of M linear combinations of the input are constructed in the form

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2.15}$$

where $j = 1, ..., M$ and the superscript $(\,\cdot\,)$ denotes the network layer. In this case, (1) indicates that the corresponding parameters are part of the first layer of the network. Similar to the nomenclature used to describe the linear discriminant function, $w_{ji}^{(1)}$ refers to the weights and $w_{j0}^{(1)}$ represents the biases.

A differentiable, nonlinear activation function $h(\,\cdot\,)$ is then used to transform the activations a_j, which results in

$$z_j = h(a_j). \tag{2.16}$$

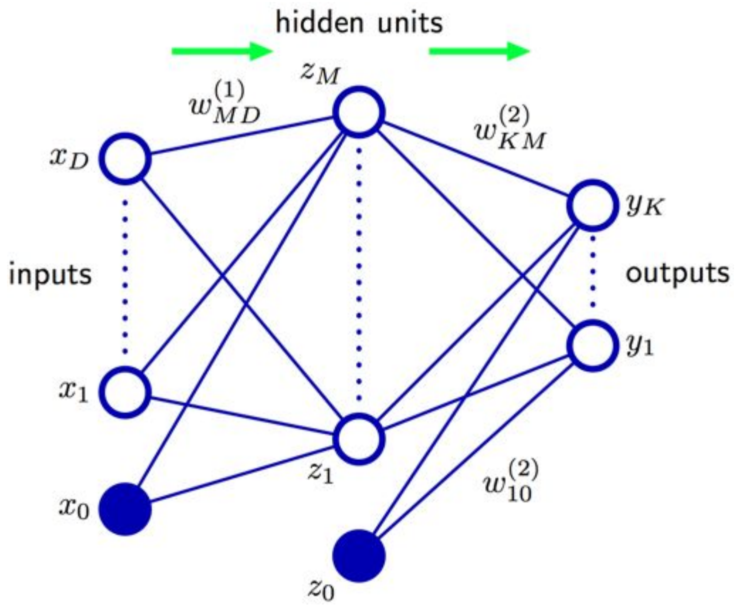


Figure 2.6: Network diagram for a two-layer neural network corresponding to Equation 2.22. The input variables x_d, hidden variables z_m and output variables y_k are represented by nodes. All weight parameters w are illustrated by links between the nodes. Bias parameters are represented by links originating from additional input and hidden variables x_0 and z_0. The green arrows denote the flow of information through the network during forward propagation. [45]

The quantities z_j are called hidden units. These hidden units correspond to the outputs of the basis functions $\phi_j(\mathbf{x})$ in Equation 2.14. Other than using a step function as in the perceptron model (cf. Equation 2.13), sigmoidal functions are used for the nonlinear activation functions $h(\,\cdot\,)$.

The hidden units are followed by K output units. First, the output unit activations a_k are obtained by linear combinations of the hidden unit outputs z_j. This gives

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{2.17}$$

where $k = 1, ..., K$. These hidden unit transformations are performed in the second network layer, denoted by (2). In order to determine the final network outputs y_k, the activations a_k are again transformed utilizing a proper activation function:

$$y_k = \sigma(a_k) \tag{2.18}$$

The type of activation function $\sigma(\,\cdot\,)$ is chosen depending on the underlying dataset and target value distribution. For binary classification tasks a *logistic sigmoid function* is used where

$$\sigma(a) = \frac{1}{1+\exp(-a)}. \tag{2.19}$$

For classification problems involving multiple classes (i.e. $K > 2$), a *softmax* activation function is used. The softmax function is a generalization of the logistic function to multiple variables. It basically provides an approximation of the maximum operation and at the same time ensures that the output values are between 0 and 1 and also sum up to 1. Given an input $\mathbf{a}$ and K nodes in the softmax layer, the activation function is given by

$$\sigma(\mathbf{a}, k) = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)}. \tag{2.20}$$

The smoothed version of the maximum function is achieved because if $a_k \gg a_j$ for all $j \neq k$, then $y_k \simeq 1$ and $y_j \simeq 0$.

All of these previous stages can be combined to form the overall network function. For sigmoidal output unit activations, this results in

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{2.21}$$

where the vector $\mathbf{w}$ represents the set of all weight and bias parameters. Both activation functions $\sigma(\ \cdot\ )$ and $h(\ \cdot\ )$ are incorporated. This nonlinear neural network function basically maps a set of inputs $\{x_i\}$ to a set of output variables $\{y_k\}$. This mapping is controlled by the vector $\mathbf{w}$ of adjustable parameters. Figure 2.6 illustrates the network diagram that represents the forward propagation through the network (i.e. evaluation of Equation 2.21). By incorporating bias values into new variables x_0 and z_0 as shown in Figure 2.6, the overall network function becomes

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right). \tag{2.22}$$

The network diagram shown in Figure 2.6 illustrates a neural network topology that consists of two processing steps that are defined by Equation 2.22. Each of these steps corresponds to the perceptron model represented by Equation 2.12. Therefore, the neural network model is also referred to as the *multilayer perceptron*. However, there is an important difference between the processing stages of a neural network and a perceptron. The perceptron uses step-function nonlinearities whereas the neural network implements continuous sigmoidal nonlinearities in the hidden units (cf. Equation 2.16). This variation is significant because it makes the neural network function differentiable with respect to the network parameters.

2.2.9 Decision trees and random forest

Decision trees are an alternative model to solve classification problems. The goal of implementing a classification tree is to create a model that predicts a target class based on sequential decisions on a number of input variables (observations). As illustrated in Figure 2.7, these series of successive binary selections are made when traversing through the tree structure. A tree consists of a top node called the root element, followed by leaves and branches. In Figure 2.7, leaves are denoted $A - E$ and represent class labels. They are the outcome associated with combinations of features that lead to those labels, represented by branches.

Figure 2.8 shows a two-dimensional input space that has been partitioned into five regions using the binary tree illustrated in Figure 2.7. Algorithms for decision trees usually work from top to bottom by choosing a "best" feature variable at each step that splits the input in the most optimal way. Various different approaches of determining the "best" feature at each node exist depending on the algorithm. In this example, first the entire input space is divided into two regions, depending on if $x_1 \leq \theta_1$ or $x_1 > \theta_1$. θ_1 is a model parameter that represents a threshold for the feature x_1. This threshold initially creates two subregions that are in further consequence treated separately. Each of these independent subregions are then split up further. The region $x_1 \leq \theta_1$, for example, is subdivided depending on whether $x_2 \leq \theta_2$ or $x_2 > \theta_2$, resulting in an assignment to regions A and B. The region assignment for a new input $\mathbf{x}$ is made by starting at the root node of the tree and following down the branches according to the decision criteria until a certain leaf node is reached. Each region is then assigned to a target class.

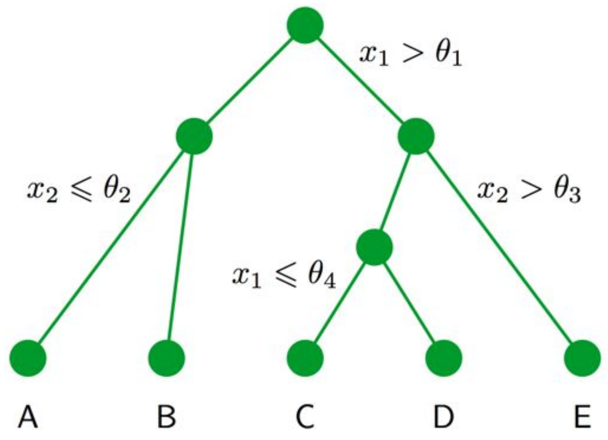


Figure 2.7: Binary decision tree for input partitioning corresponding to Figure 2.8. Through sequential decisions inputs are assigned to classes A - E, whereby x and θ represent inputs and thresholds, respectively. [45]

During the training step the structure of the tree model needs to be learned. This involves choosing a certain input feature variable x_i as well as a threshold θ_i for the split criteria at each node. Given a D-dimensional vector of input feature variables $\mathbf{x} = (x_1, ..., x_D)^T$, the goal is to predict a single target t. The training data are comprised of input vectors $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ together with corresponding target labels $\{t_1, ..., t_N\}$.

The problem is how to determine the structure of the decision tree (including feature and threshold selection at any given node) that optimizes the assignment of classes. Minimizing a performance error for a tree consisting of a fixed number of nodes is usually not feasible due to the large amount of different input combinations. Therefore, a different solution is generally used. A tree for the whole input space is grown one node at a time, starting at the root node. Every time a node is generated, one input feature variable as well as the respective threshold value are chosen. This selection step is repeated for all of the D input variables at that node and for each of the possible combinations the performance error for the current configuration is calculated. The optimal input feature variable at that respective node is the one out of D variables that splits the input best, and thus, results in the smallest error.

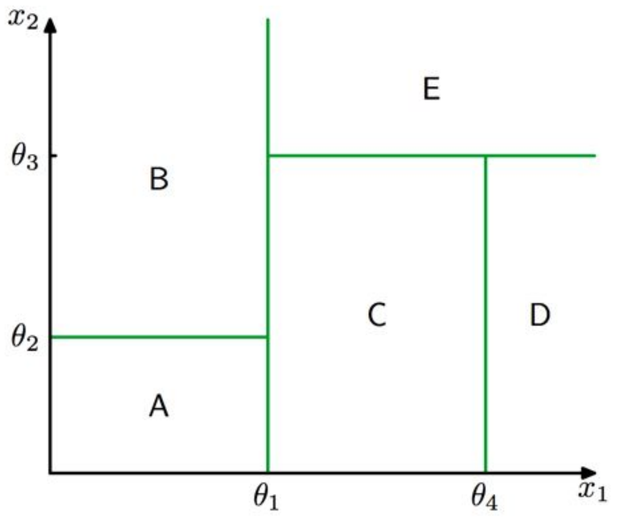


Figure 2.8: Illustration of partitioning of a two-dimensional input space into five regions using a binary decision tree. Note that axis-aligned boundaries are used for the spacing. [45]

The issue that remains when using this technique is when to stop adding nodes to the tree. A simple approach is to use an error threshold as a stopping criterion. However, it has been found empirically that often even if the overall error remains almost constant despite adding new nodes, after some additional splits, a significant error reduction still occurs [45].

Therefore, the common practice is to grow a large tree (whereas the size depends on the characteristics of the input data) and afterwards prune the resulting tree. The criterion used for this pruning step is based on balancing the error against the overall complexity of the model. Suppose the initially created tree is denoted by T_0. Then the subtree $T \subset T_0$ is gained by eliminating nodes from T_0, thus combining the respective regions. Leaf nodes are denoted with τ, where $\tau = 1, ..., |T|$ and $|T|$ represents the total amount of leaves. A leaf node τ represents the region $\mathcal{R}_\tau$ of an input space with N_τ data points. For a given region $\mathcal{R}_\tau$, the optimal prediction outcome is then given by

$$y_\tau = \frac{1}{N_\tau} \sum_{\mathbf{x}_n \in \mathcal{R}_\tau} t_n. \tag{2.23}$$

For classification trees, two performance error measures are commonly used: the cross-entropy and the *Gini index*. If $p_{\tau k}$ is defined as the proportion of data points that lie in region $\mathcal{R}_\tau$ and are assigned to class k, where $k = 1, ..., K$, then the cross-entropy is defined as

$$Q_\tau(T) = \sum_{k=1}^{K} p_{\tau k} \ln p_{\tau k} \tag{2.24}$$

and the Gini index as

$$Q_\tau(T) = \sum_{k=1}^{K} p_{\tau k}(1 - p_{\tau k}). \tag{2.25}$$

Both measures have a maximum at $p_{\tau k} = 0.5$ and vanish for $p_{\tau k} = 0$ and $p_{\tau k} = 1$. The cross-entropy and the Gini index are both differentiable, which enables the application of gradient based optimization methods.

Random Forest

A *Random Forest* (cf. Breiman, 2001 [48]) classifier grows multiple decision trees in order to improve the classification rate. Trees are grown using a random set of the input variables (with the same distribution for all trees). To achieve a classification result, a new input vector is traversed through each of the trees in the forest. Every single tree then provides a class assignment. It is said that a tree "votes" for a class. The overall result of the forest is then chosen according to the class that received the most votes. The definition of a random forest by L. Breiman in [48] is as follows:

> ”*A random forest is a classifier consisting of a collection of tree-structured classifiers* $\{h(\mathbf{x}, \Theta_k), k = 1, ...\}$ *where the* $\{\Theta_k\}$ *are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input* $\mathbf{x}$.” [48]

The procedure of growing a tree is as follows:

1. Given an original training set including N cases, randomly sample N cases with replacement. These sampled cases will be the training set for growing the tree.

2. Suppose the input vector $\mathbf{x}$ consists of M feature variables. For the k-th tree T_k, a random vector Θ_k is generated by selecting $m < M$ out of the M variables. The random vector Θ_k is thereby independent of the past random vectors $\Theta_1, ...\Theta_{k-1}$. At each node the one variable out of m that best splits the data is selected. The value of m specifying the amount of randomly chosen feature variables remains constant for the whole forest growing process.

3. Each tree is grown to the largest extent that is possible and no pruning is performed.

Due to the fact that a different random sample of input variables is used for the construction of each tree, there is no need for cross-validation or a separate test set to get an unbiased error estimate. For the construction of the k-th tree, about one-third of the input cases are left out. Those cases internally serve as the test set and are then used to obtain a classification result. The resulting error that can be computed is called the **out-of-bag (oob) error estimate**. Similarly, the **variable importance** of feature variables can be assessed. Using the oob cases the number of correct class votes is counted. This procedure is repeated with randomly permuted values of variable m. The two resulting scores (number of votes in each case) are subtracted and averaged over all trees to obtain the raw variable importance score.

Chapter 3

Materials and methods

In this chapter materials and methods used for the segmentation framework development are addressed. The segmentation problem is solved utilizing a supervised learning approach; hence, it is important to have a set of training data available in order to be able to train a classifier. For the application targeted by this research, a proper training set consists of images and corresponding ground truth markups of the region of interest. The properties of the provided MR image recordings are described firstly in Section 3.1. Section 3.2 then addresses the development of a multi-purpose image viewer that allows the user to view MR image slices as well as draw manual contours within these images. Section 3.3 focuses on the main part of the project, which is the development of the segmentation algorithm. The algorithm is thereby described step-by-step, starting at preprocessing and concluding with postprocessing. This chapter also includes remarks on the creation of a final framework in Section 3.4. In addition, Section 3.5 explains the scoring metrics that are used for segmentation accuracy evaluations. Code development for this project was performed using *MATLAB*.[1]

3.1 Characteristics of the MR image database

The segmentation algorithm was developed and tested using 50 anonymized patient cases with biopsy confirmed cancer acquired as T2 weighted magnetic resonance axial pulse sequence. An imaging example is illustrated in Figure 2.2. Half of the patient cases were obtained at the Boston Medical Center with a Philips Achieva at 1.5 T using an endorectal receiver coil. The other half at 3 T with a Siemens TIM and a surface coil at Radboud University Nijmegen Medical Center, Netherlands. Datasets acquired with a field strength of 3 T consist of 15+ slices with 4 mm thickness. 1.5 T recordings have 28+ adjacent axial cross-section cuts with 3 mm slice thickness. For the training set, markups in Nearly Raw Raster Data (NRRD) [2] format defining the central gland (CG) and peripheral zone (PZ) of the prostate are available in addition to the respective Digital Imaging and Communications in Medicine (DICOM) image files. The NRRD library and file format is designed to support scientific visualizations and image processing involving

[1]MATLAB and Image Processing Toolbox Release 2013a, The MathWorks, Inc., Natick, Massachusetts, United States.

[2]NRRD - Nearly Raw Raster Data, `http://teem.sourceforge.net/nrrd/`, date accessed: August 2013.

N-dimensional raster data. Each file contains a multi-line text header including meta information and the current raster data. The header also describes the orientation of the raster grid in respect to some kind of surrounding space. In this case, the relative orientation of the MRI volume to the patient coordinate system is specified. NRRD files are imported to MATLAB using the *NRRD Format File Reader* provided by Jeff Mather [3]. Patient cases are available from the National Cancer Institute via The Cancer Imaging Archive.[4] An example image slice with expert labeled prostate structures is illustrated in Figure 3.1.

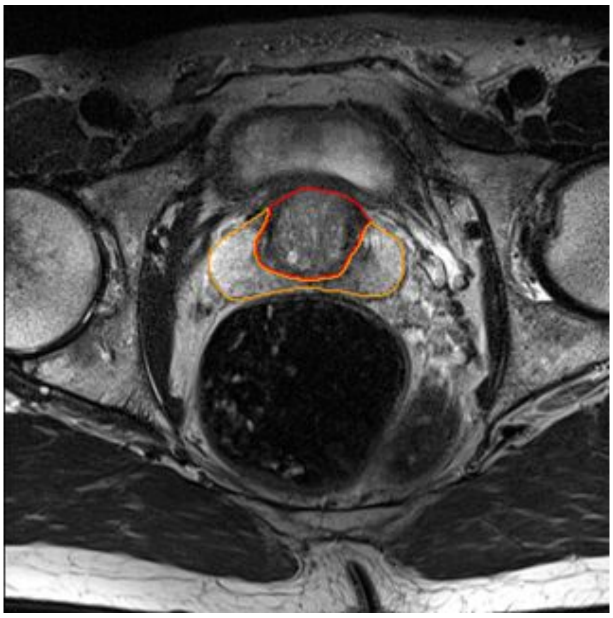


Figure 3.1: 3 T MR image slice with expert labeled CG (red) and PZ (orange) structures

3.1.1 Remarks on variations between 1.5 T and 3 T image sets

The utilization of different magnetic field strengths during image recording introduces variations in image characteristics. In addition to differing resolution and intensity values, additional disparities that affect the development of a segmentation algorithm are present. During the development it was found that the position of the prostate within an MR image stack is subject to variations. It seems that the two institutions which provided the anonymized magnetic resonance recordings used a different axial range for the imaging procedure. This circumstance in addition to the smaller slice thickness is the reason why on average 1.5 T cases consist of twice as much slices than 3 T cases. Considering the anatomical space (patient coordinate system), this means that there are significant shifts within the axial plane. On average and compared to 3 tesla recordings, axial slices of 1.5 tesla recordings start with an offset to the inferior end of the human body. They also monitor a broader region of the body which again results in a stretch into the direction of the superior end. This extended range affects the relative number of image slices that

[3]NRRD Format File Reader, `http://www.mathworks.com/matlabcentral/fileexchange/34653-nrrd-format-file-reader`, date accessed: August 2013.

[4]National Cancer Institute, The Cancer Imaging Archive, `https://wiki.cancerimagingarchive.net/x/8QRp`, date accessed: April 2013.

actually show prostate tissue.

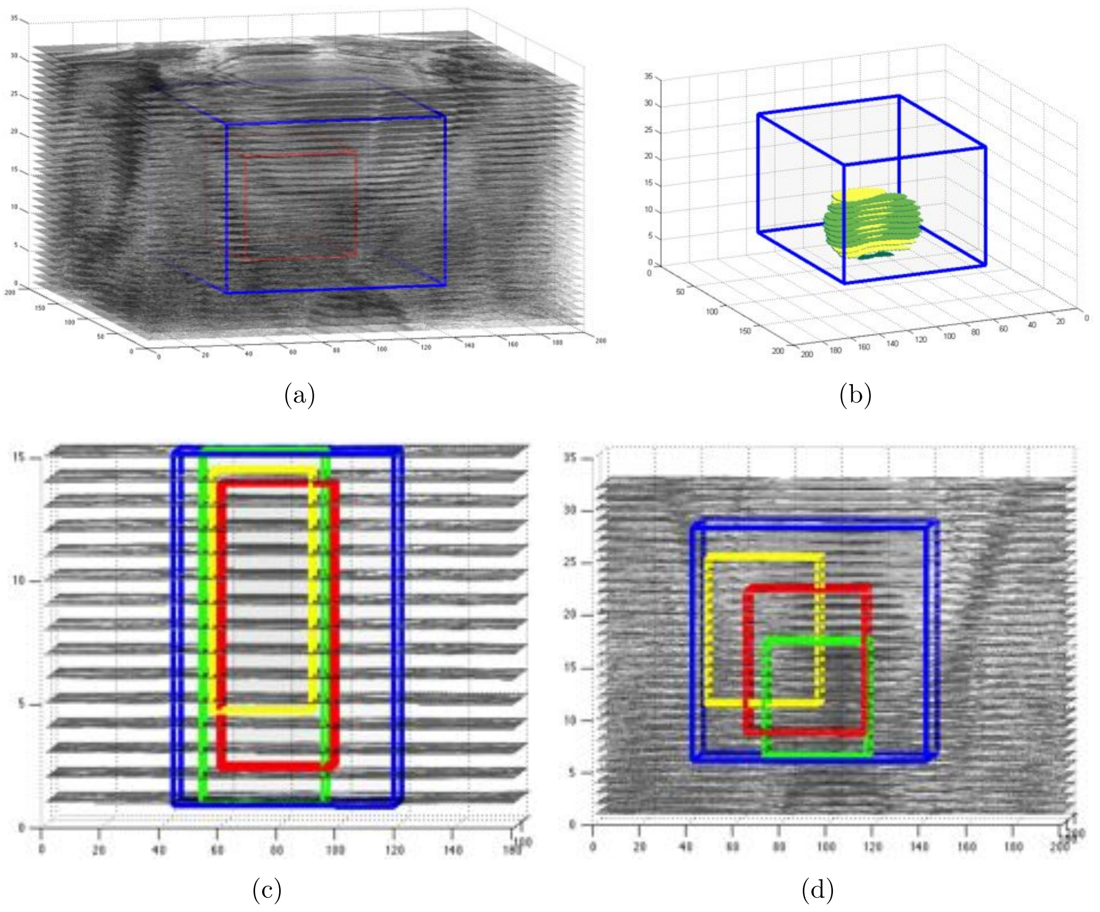


Figure 3.2: Visualization of MR image slices and prostate bounding boxes. Axial image slices are plotted in z-axes direction. Fig. (a) shows the same 1.5 T case as Fig. (d) with a different viewing angle. Fig. (c) shows image slices of a chosen 3 T patient case. Fig. (b) illustrates ground truth labels for the prostate central gland (yellow) and peripheral zone (green) corresponding to (a). A maximum bounding box for the prostate structures is outlined in blue color in all figures. A red bounding box indicates an average prostate position. Green and yellow cuboids mark prostate positions of single cases.

Figure 3.2 illustrates exemplary image stacks out of the training cases. An average three-dimensional prostate position has been calculated for both 1.5 T and 3 T recordings throughout all respective training cases (indicated as a red box in Figures 3.2(c) and 3.2(d)). The prostate position is denoted by a normalized three-dimensional bounding box. This bounding box applied to a 15-slices patient case out of the 3 T set shows that the prostate is on average visible on slices 2 to 14 (see Fig. 3.2(c)). This means that the prostate is visible starting at the second slice throughout to the second last slice, which equals 86% of all slices. Plotting the bounding box for patient cases recorded at a magnetic field strength of 1.5 T shows a different situation. Having 32 image slices, Figure 3.2(d) indicates that only slices 9 to 22 actually visualize the prostate. This basically shows that in these cases, less than half of the available images contain prostate tissue.

Similarly to the average position, a maximum three-dimensional bounding box is calculated that includes the entire prostate structure on every image slice throughout all training cases. Given any image slice out of the training cases, this means that if prostate tissue is visible on that specific slice, the entire prostate tissue region will be embedded in the maximum bounding box. The plots of this bounding box in Figures 3.2(c) and 3.2(d) also indicate that in contrast to 3 T cases certain slices of the 1.5 T images never contain prostate tissue.

Suppose that out of all training cases and without considering the magnetic field strength, one image slice is chosen to serve as the current input to the algorithm. Furthermore, assume that this image is the 10th slice out of a certain case. Considering the 3 T training images, there is a 100% chance that prostate tissue is visible on that image slice. However, if this slice was taken from the 1.5 tesla set, chances are that there is no prostate tissue included at all. In fact, in some of the 1.5 T cases the first image that includes one of the prostate zones is the 11th slice (see yellow box in Fig. 3.2(d)). Even then, obviously only few pixels represent prostate tissue as the inferior part of the gland (apex) viewed as a cross-sectional cut is small in diameter (compare Fig. 2.1).

3.2 Development of an image viewer for visualization of image and label data

An important initial part of the project was the development of a graphical user interface (GUI) that serves as an image viewer. As mentioned in Section 3.1, patient files are available in the DICOM format. The DICOM standard is described in [49]. A DICOM file contains a header and the image data. The included header stores information about the type of scan, the image position and dimension, slice thickness, pixel spacing and other data referring to the properties of the imaging device, setup of the recording and the patient.

The viewer enables the radiologist to load and display all DICOM images out of an available patient case. Images are thereby shown slice by slice whereas it can be scrolled through them by pressing a key on the keyboard or by clicking on the next image in the list. If the respective information is available in the header, all MR images are sorted by *SliceLocation*. Otherwise their *InstanceNumber* is used to provide a proper arrangement. This sorting routine was necessary to avoid errors introduced by inconsistencies in the naming of files (image files were not always named in succession).

The GUI offers basic image processing functions that affect the visibility of the images. These options include gray-value adjustment and histogram equalization. Furthermore, the window level can be adjusted to enhance the visibility of certain tissues. Window level settings can be saved to ensure maximum usability.

Another very important functionality in addition to the image viewing capability is that manual contours can be drawn using this GUI. As described in Section 2.2.1, the training data for a supervised learning algorithm consists of input vectors along with their

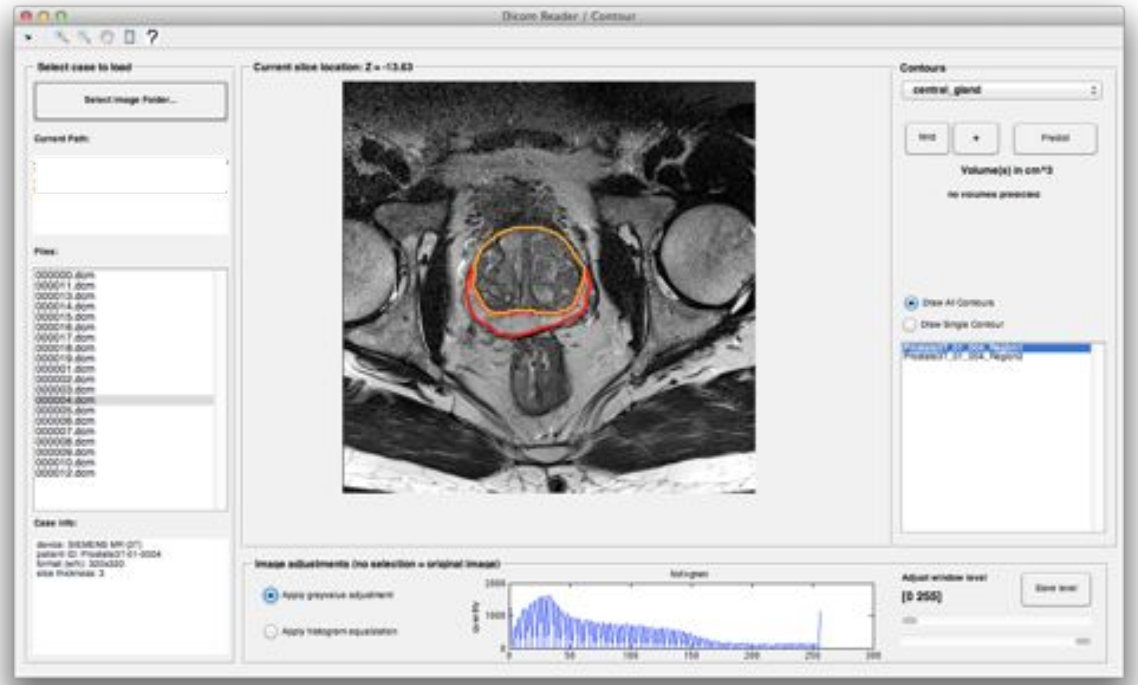


Figure 3.3: GUI of image viewer.

corresponding targets. To collect proper targets for this application, tissue contours need to be drawn by an expert. In case of the prostate, a radiologist needs to mark central gland and peripheral zone on every image slice that shows these regions throughout all patient cases. The outcome of this process for one slice is shown in Figure 3.1.

The image viewer that has been developed allows the radiologist to name all new contours and save them to *MAT*-files as well as store them in NRRD format. Existing contours can be edited as well. This editing process is simplified by providing the ability to add or subtract parts to/from the marked region. These options prevent the need to redraw the whole contour.

Tissue markups are visualized as an overlay of the image, whereas only region boundaries are shown (illustrated in Figure 3.3). All available contours for a specific image slice can be displayed at the same time whereas certain ones might be hidden as well. However, it is only possible to edit one contour at a time. The current choice is specified through a selection in the drop-down menu. The final contours need to be stored as a single NRRD file in the same folder together with the images of the corresponding patient case.

3.3 Development of the segmentation algorithm

Having a complete training set available, a segmentation algorithm based on a supervised machine learning approach was developed. In this section, the most important considerations for the development of this algorithm are described. A segmentation result is achieved by processing three major steps: preprocessing, feature extraction and classification. Every step is explained in the following subsections. Afterwards the algorithm pipeline is illustrated and in addition the reasons for implementing a multi-layer topology are explained in Section 3.3.4.

3.3.1 Preprocessing of input data

The first major part of the algorithm is a preprocessing step. This step is necessary to gain prior knowledge of the input dataset characteristics and to perform adaptations on the input that facilitate subsequent computations. The input dataset in this case denotes a collection of one or more patient cases. Each patient case consists of multiple MR image slices. As indicated in Section 3.1, the amount of slices thereby varies depending on the imaging device and setup of the recording. Similarly, differences in image resolution are present as well throughout the sets of patient cases. These and other variations within the training set that have been addressed in Section 3.1.1, increase the problem complexity. In case of the training set, expert labels of the prostate structure are available in addition to the MR images.

The goal of preprocessing is to reduce variations within the image sets. Figure 3.4 illustrates the basic steps that are performed throughout this procedure. In order to reach this goal it was found necessary to apply image adjustments and cropping methods. In terms of image adjustments, mainly gray-level corrections are performed in order to reach a certain homogeneity level of intensity values throughout different patient cases. Such window level corrections are applied to all images by reading *windowCenter* and *windowWidth* parameters from the respective DICOM headers. The respective level settings depend on the configuration of the imaging device used for the recordings. Furthermore, a rescaled and smoothed representation of every image is formed in order to eliminate noise artifacts and to further reduce data size. To suppress noise while preserving edges a median filter with a kernel size of 3×3 pixels is used for the smoothing process.

The cropping step on the other hand is mainly introduced to reduce the amount of input data for subsequent computations. In addition, it helps to compensate the problem of prostate position variations. Within this sequence, a maximum three-dimensional bounding box is calculated that includes the whole prostate structure on every image slice throughout all training cases (see Fig. 3.2). The resulting normalized cuboid position and size are used to crop every image stack to reject unimportant structures. As visualized in Figure 3.2(d), the cropping will especially affect all 1.5 T cases. The resulting image stack has the same dimensions as the blue bounding box in Figure 3.2(d) and is illustrated in Figure 3.4.
After this sequence the relative, averaged prostate position of patient cases recorded at 1.5 tesla better matches with the positions in 3 T cases. This technique significantly contributes to data reduction as well, because the prostate region of interest on body MRI is commonly only visible on small sections of the image (e.g. see Fig. 3.1 where only the outlined regions are of concern). The remaining parts of the image are unimportant in the context of this application and referred to as background pixels throughout this work.

Consider for example a patient case with image stack dimensions of $500 \times 500 \times 34$. After the cropping step with the bounding box, the stack is reduced to a size of

160 × 140 × 14. The subsequent filtering and down-sampling process again reduces the stack dimensions to a final size of 80 × 70 × 14. In this exemplary case the number of pixels in the input stack is reduced from 8,500,000 to 78,400, which is a hundredfold reduction.

Besides input size reduction, important information for succeeding feature extraction is obtained throughout this preprocessing step. In addition to the bounding box that includes the prostate structure, an average three-dimensional weighted center of the

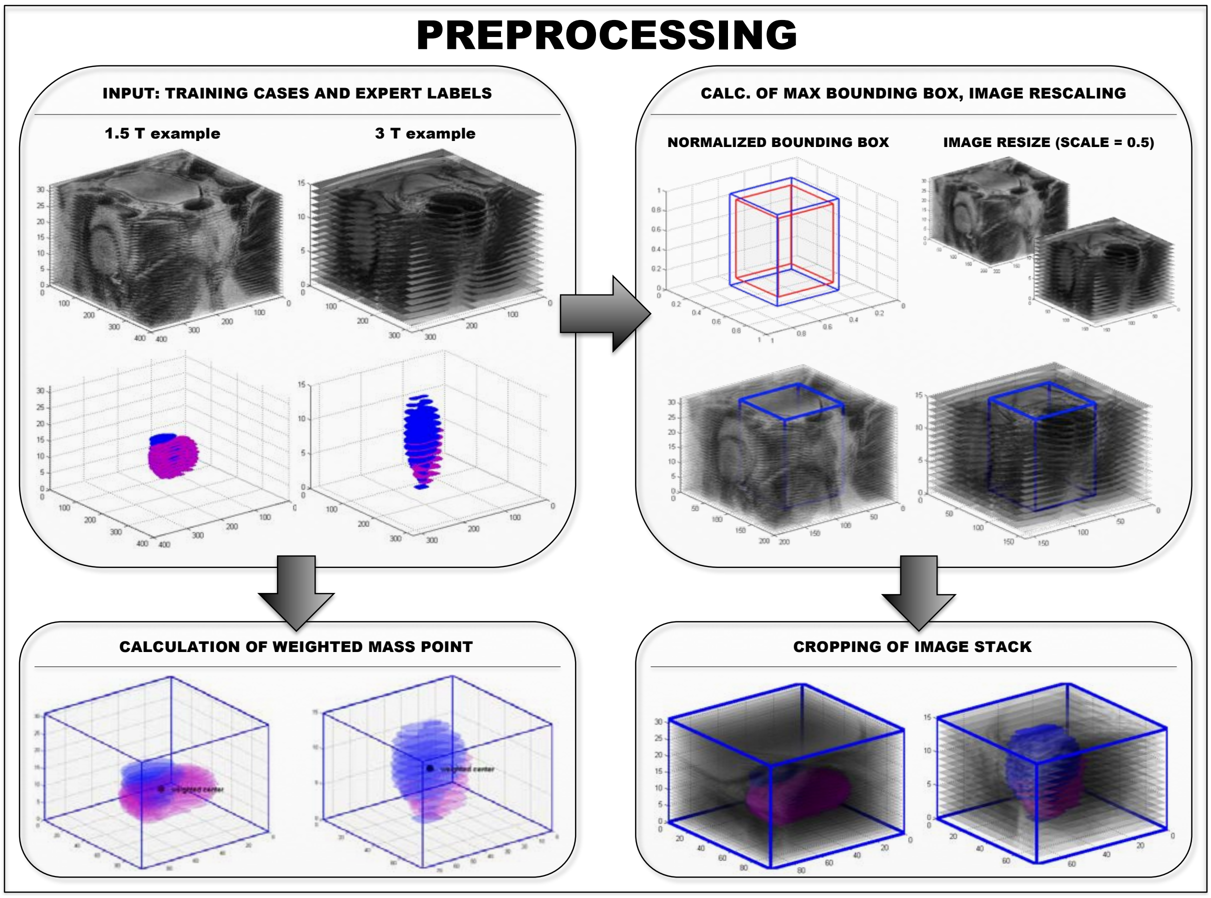

PREPROCESSING
INPUT: TRAINING CASES AND EXPERT LABELS
1.5 T example
3 T example
CALC. OF MAX BOUNDING BOX, IMAGE RESCALING
NORMALIZED BOUNDING BOX
IMAGE RESIZE (SCALE = 0.5)
CALCULATION OF WEIGHTED MASS POINT
weighted center
CROPPING OF IMAGE STACK

Figure 3.4: Preprocessing pipeline. The upper left figure shows the structure of the input dataset. Exemplary cases out of the 1.5 T as well as 3 T set are shown. Training cases are provided together with corresponding expert labels for the prostate central gland (CG) and peripheral zone (PZ). In this illustration, the CG is colored in blue and the PZ is represented in magenta. Utilizing the training labels, a maximum bounding box for the structure of interest is calculated (shown in the upper right figure). The red cuboid represents the original calculation but in regard of future testing cases that differ from the training set, a tolerance is added resulting in the blue maximum bounding box. All images are resized and smoothed to suppress noise artifacts. In order to minimize the amount of input data, all pixels outside the bounding box are rejected as they are considered unimportant for this application. The outcome of the preprocessing step are cropped versions of the image stacks and the results for the weighted center point for each prostate structure in the training set.

prostate is computed. This center point is essential as a reference point for calculation of distance measures (as described in Section 3.3.2).

All of the operations described above are possible because ground truth expert labels are available for all training cases. The outputs of the preprocessing step are:

- For each original image stack, a cropped and smoothed version of every image in this stack. A maximum three-dimensional bounding box (with added tolerance) that includes the prostate on every patient case is calculated to obtain the cropping limits. Only the cropped version of the stack is used for subsequent computations.
- A weighted center for each prostate structure in the training set.

For the generation of the test set, image stacks are cropped utilizing the constraints of the bounding box obtained through the training step.

3.3.2 Feature extraction and input transformation

Feature extraction described in this section is important to create a good internal representation of the visual information presented at the input. As no low-dimensional clustering appears in the original input space (raw pixel values) it is not feasible for the classifier to partition the image into the desired anatomical regions (cf. Fig. 2.3). Therefore, a feature representation is created where nonlinear classification is made possible for the learning algorithm. The algorithm should ideally be capable to label every pixel with the category of the structure it belongs to. This process is known as scene labeling or scene parsing [50].

To be able to later assign a class to a certain pixel, a feature vector for every input pixel needs to be formed. The original pixel information is solely its gray-value intensity. However, labeling each image pixel by only looking at it or a small region around it is difficult [50]. The category of a pixel may depend on various additional information including its orientation in space and broader neighborhood context. Therefore, additional properties that are relevant for a description of prostate zones are added to each image pixel. These properties are referred to as features.

Features are extracted utilizing a feature extraction function $f^{\mathbf{I}}$ on each pixel within the image stack $\mathbf{I}_q(i, j, l)$ as denoted in Equation 3.1. The feature extraction function is applied to both train and test cases to generate a consistent input for a certain classifier object.

$$\mathbf{X} = f^{\mathbf{I}_q}\left(\sum_{q=1}^{Q} \mathbf{I}(i, j, l)\right) \tag{3.1}$$

Q indicates the total number of cases. The final feature matrix then has a dimension of the total number of pixels in all training cases u times the amount of features v. The steps necessary to obtain this matrix are illustrated in Figure 3.5. The generated $u \times v$ feature matrix is denoted as $\mathbf{X}$. The output $\mathbf{T}$ is either a u-dimensional vector or a $u \times K$

dimensional matrix depending on the target coding scheme, which is addressed in Section 3.3.3. **X** and **T** are denoted in Equation 3.2.

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_u^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1v} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2v} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{u1} & x_{u2} & x_{u3} & \cdots & x_{uv} \end{pmatrix} \qquad \text{and } \mathbf{T} = \begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \vdots \\ \mathbf{t}_u \end{pmatrix} \tag{3.2}$$

X and **T** need to have the same amount of rows for the training phase. The final training set $\mathcal{D}$ is generated by a combination of **X** and **T**, resulting in

$$\mathcal{D} = \left\{ \left(\mathbf{x}_i^T, \mathbf{t}_i^T \right) \right\}_{i=1}^{u} \qquad \text{or } \mathcal{D} = (\mathbf{X}, \mathbf{T}) \,. \tag{3.3}$$

This approach results in a very large representation, which is why decreasing the amount of input pixels during the preprocessing step accelerates all following computations. All features are manually chosen and implemented, whereby all extraction methods are combined in one feature extraction function $f^{\mathbf{I}}$.

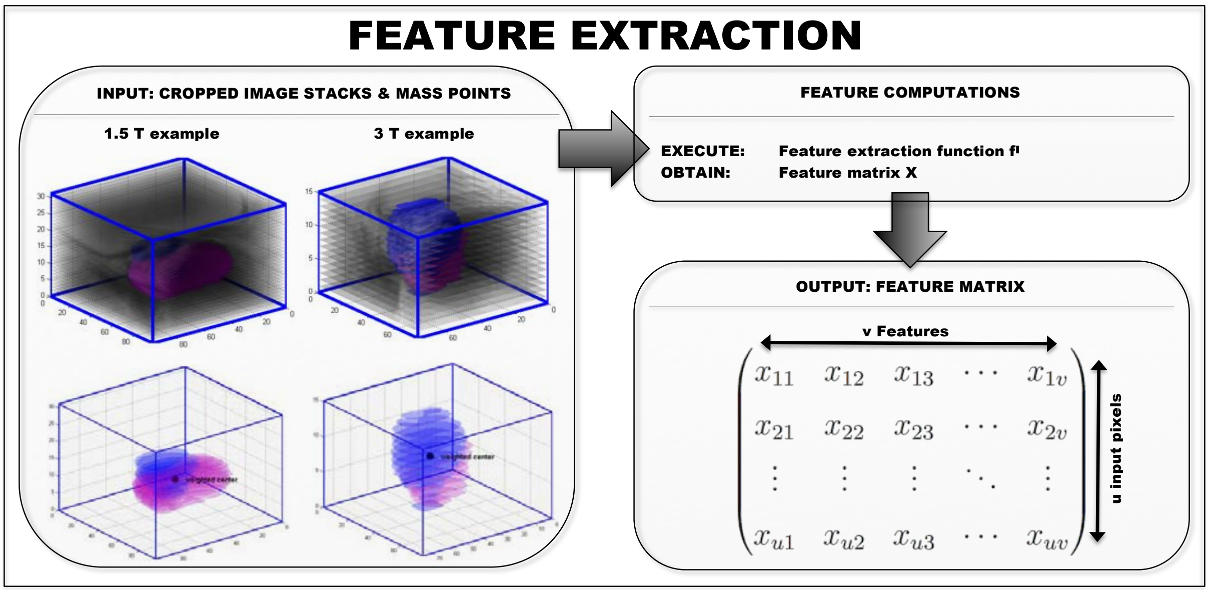


Figure 3.5: Feature extraction pipeline. The figure to the left shows the structure of the input dataset, obtained by performing the preprocessing step. Exemplary cropped cases and center points out of the 1.5 T as well as 3 T set are shown. The feature matrix **X** is obtained by executing the feature extraction function $f^{\mathbf{I}}$. Every row in **X** represents a single image pixel p_i, where $i = 1, ..., u$. u denotes the total amount of image pixels, summed over all cases Q in the train or test set, respectively. A total of v features are calculated per pixel p_i, resulting in the final matrix with feature values x.

The remaining part of this section is dedicated to explain how these individual features are obtained and why they are considered important to enable input clustering. Pixel characteristics that were implemented in this work include statistical parameters, two-dimensional and three-dimensional neighborhood properties, distance and angle measures. All feature values are normalized to reach an invariance in terms of image resolution, image stack dimension (amount of slices) and gray-value distribution/range. The most important features are described in the following paragraphs.

a) A priori and a posteriori probability maps

Considering all training cases, a normalized probability distribution for every target label is calculated. A priori class probabilities for every image pixel are mapped onto a cuboid and are averaged throughout all training cases. The size of the cuboid is determined by the average size of the image stacks used for training. A certain value (between 0 and 1) in the cuboid for the central gland structure represents the average probability for a pixel on this position to belong to the central gland.

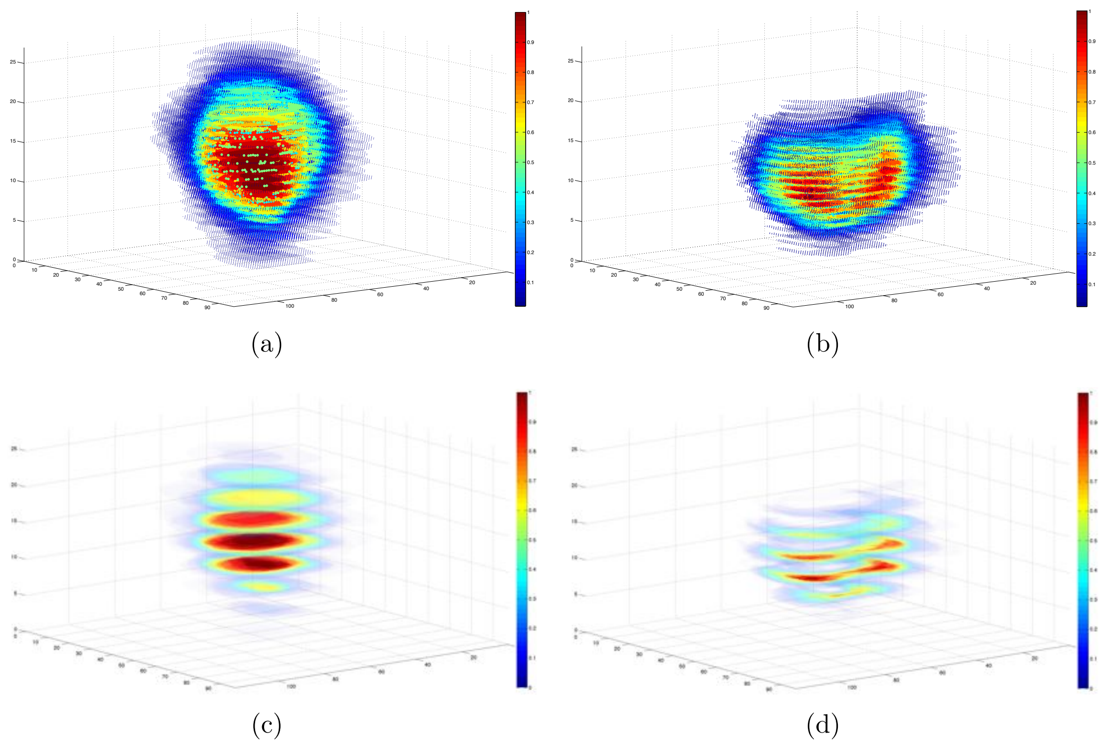

(a) (b) (c) (d)

Figure 3.6: Probability map for central gland (left) and peripheral zone (right) prostate structures, averaged over 50 training cases. (c) and (d) correspond to (a) and (b), respectively, with some slices being hidden for better visibility. Red indicates high and blue represents low class probabilities.

In Figure 3.6 exemplary probability map cuboids are illustrated. Obviously, if a pixel is located near the weighted center of the structure, it has a very high probability to belong to that structure. On the other hand, picking a pixel near the object boundary results in lower probabilities as structure boundaries vary within the training cases which in turn introduces uncertainties.

To use this feature for one specific case for the first classification layer, the normalized cube is resized in three dimensions to fit the current image stack. The second layer extends the influence of this feature as a weighting of a priori probabilities is performed. Weighting becomes possible by utilizing additional a posteriori class probabilities obtained from the prediction result of the first layer. For detailed information on the layer architecture see Section 3.3.4. This weighting step improves the accuracy of the probability map for the current case by still maintaining average structure likelihoods to enhance the overall prediction outcome.

Magnetic field strength

A value between zero and one is assigned according to the magnetic field strength of the device used for image recordings. In this case this value is either 0 or 1 as only two cases exist. Including this feature is important due to the differences in image characteristics introduced by variations in magnetic field strength (cf. Sec. 3.1.1).

3D distance measures

During the preprocessing step described in Section 3.3.1, a weighted mass point for all prostate structures included in the training set is determined. This center point information is utilized to calculate distance features. 3D distance measures include the Euclidean distance of the current pixel's position to the weighted center of prostate structures in millimeters. The conversion from pixels to millimeters is essential because of variabilities in pixel spacing within the patient cases.

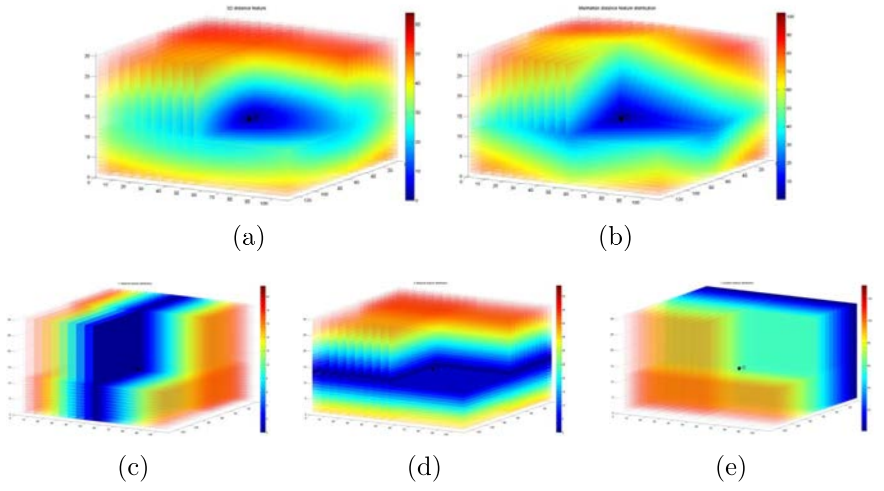


Figure 3.7: Distance and position feature distributions. (a): Euclidean distance, (b): Manhattan distance, (c) and (d): y- and z distances, (e): Y position. Red indicates high values at a large distance whereas blue represents small ones.

This distance measure was found to be helpful due to the approximately spherical shape of the prostate. x, y and z distances from the center are also taken into account to learn the average position of the prostate within an image stack. Besides measures with respect to the center point which are all subject to symmetrical properties it was found helpful to also include x and y positions according to the Cartesian coordinate axes as well as the current slice number. The symmetrical distance properties lead to problems in properly detecting the peripheral zone position. To additionally enhance the accuracy, the Manhattan distance is added which sums up x and y displacements. Representative distributions of distance and position features are illustrated in Figure 3.7.

The mode of these distance calculations is different depending on the input, which is directly related to the algorithm condition. Basically, the two main states of the algorithm are training and testing.

Within the training step, expert labels that are considered ground truth are available corresponding to the images. For each case, the weighted center of the prostate can therefore directly be deduced from the labels and the distance measures can be computed with maximum accuracy.

In contrast to this situation, no labels are provided with the testing cases. This makes sense as the goal is to predict labels utilizing an already trained algorithm. In this case, the center points obtained within the preprocessing step during training are averaged to gain a reference point for distance computations. Measurements calculated utilizing this normalized, median center point are less accurate considering individual cases.

Spherical coordinates

In addition to distance measurements, spherical coordinates are used to describe the position of a pixel in reference to a center point. The position of a pixel P is thereby specified by three numbers: the radial distance r of that point from a fixed origin and its angular displacement specified by elevation angle and azimuth angle (see Figure 3.8). The elevation is thereby measured from the x-y plane, that means if elevation = 0, the point is in the x-y plane. These parameters are particularly helpful for the detection of the peripheral zone and are normalized in a way to profit from its common symmetrical properties.

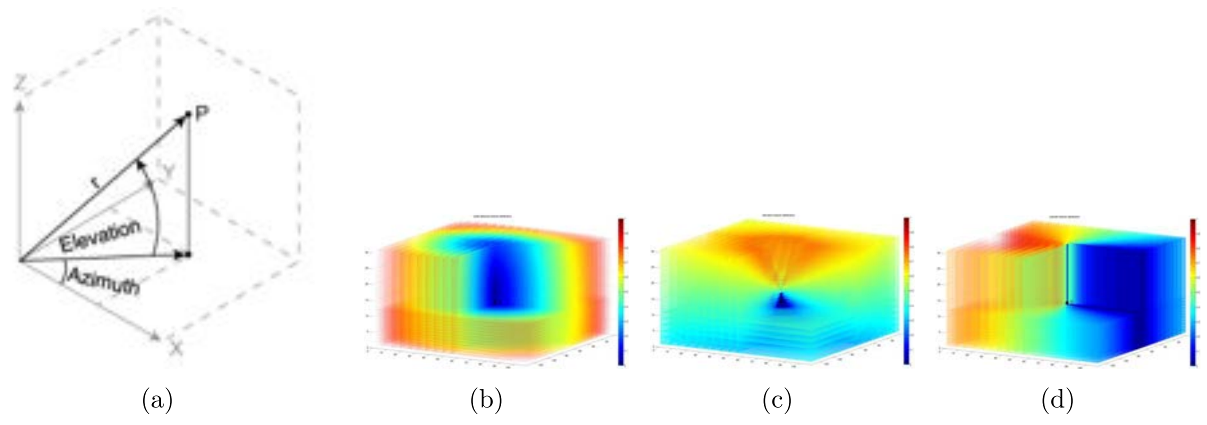


Figure 3.8: Illustration of spherical coordinate feature distributions. (a): mapping from three-dimensional Cartesian coordinates to spherical coordinates.[5]For the purpose of this work the origin was translated to the weighted center point. (b): radial distance, (c): elevation angle, (d): azimuth angle. Angles have been normalized to promote certain symmetrical properties.

Pixel gray-values

Within the preprocessing step described in Section 3.3.1, the original image stack is resized and cropped resulting in an image stack with smaller dimensions. For the purpose of this explanation this stack is denoted Ξ_1. In addition to dimensionality reductions, window level adjustments are performed. This windowed version of the stack is labeled Ξ_2.

In addition to Ξ_1 and Ξ_2 that have been obtained during preprocessing, a third image stack Ξ_3 is now created. MATLAB's `medfilt2` function is used to apply median filtering in a 3×3 neighborhood on Ξ_2. The reason for choosing a median filter is that it reduces "salt and pepper" noise while at the same time it preserves edges.

The gray-value of one pixel is then taken from the original but resized stack Ξ_1, the windowed version Ξ_2 and the filtered version Ξ_3 of the image stack, resulting in a total of three intensity values for one pixel.

3D neighborhood median and standard deviation

Beside single pixel gray-values, statistical parameters within a certain three-dimensional neighborhood (e.g. $3 \times 3 \times 3$ pixels) are considered. For this purpose, standard deviation and median in a predefined pixel neighborhood are determined. Due to the cropping of image slices during preprocessing the homogeneity of the prostate zones mostly exceeds the homogeneity of surrounding tissue regions, leading to classification improvements when implementing these neighborhood features.

In addition to the statistical representations it was found helpful to also include raw intensity values of neighborhood pixels in the feature vector. Suppose a $3 \times 3 \times 3$ pixel neighborhood is chosen, then an additional 27 values are added to the feature vector of the current pixel.

Feature adjustment and rounding

In a last step, a histogram equalization is applied to all characteristics in the final feature vector to achieve equally distributed values utilizing the maximum range. Moreover, all values are rounded to a certain amount of decimals.

3.3.3 Classification in the feature space

To achieve classification in the feature space, a classifier needs to be trained on pixel features computed from the 50 cases training set. The features that are utilized are described in detail in the previous Section 3.3.2. Besides features that represent the classifier input $\mathbf{X}$, also targets $\mathbf{T}$ are required for a supervised classifier training. A target vector maps pixel characteristics (feature patterns) to the desired labels. Targets are represented using a 1-of-K coding scheme resulting in a vector $\mathbf{t}$ of length K. If the correct class is given with C_j, then all elements t_k of $\mathbf{t}$ are zero except element t_j, which

[5] MATLAB Release 2013a Documentation, The MathWorks, Inc., Natick, Massachusetts, United States., http://www.mathworks.com/help/matlab/ref/cart2sph.html, date accessed: May 2013.

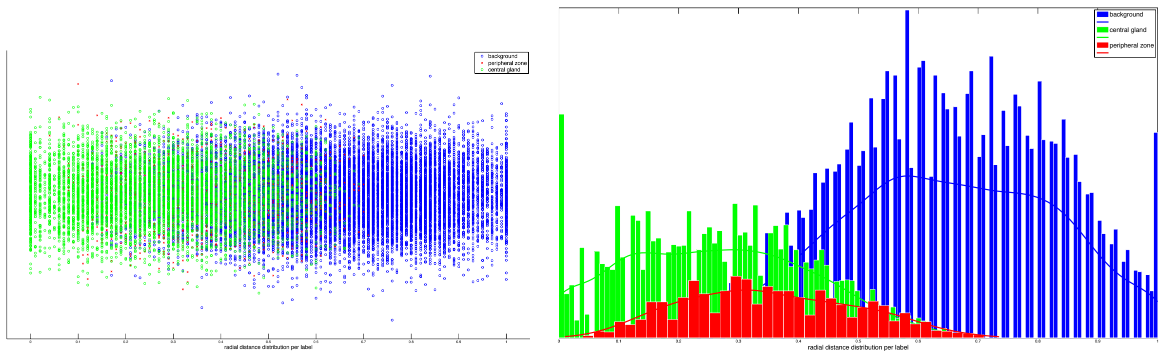
is assigned value 1. Suppose $K = 3$ classes, namely *background* (class 1), *peripheral zone* (class 2) and *central gland* (class 3) are available. Each pixel is a member of exactly one of these regions. For instance, a pixel representing peripheral zone tissue (class 2) would then be assigned the target vector

$$\mathbf{t} = (0, 1, 0)^T. \tag{3.4}$$

While some of the classifiers require the usage of this 1-of-K coding scheme, others need a single-valued target. In this case that would result in

$$\mathbf{t} = 2. \tag{3.5}$$

The main task of the classifier is to determine patterns amongst the features that characterize pixels. Pixels that are members of different classes should be distinguishable by varying feature patterns. The level of disparity between feature patterns of different class members directly influences the decision strength of the classifier. This property is influenced by the manual feature selection process which is why the type of implemented characteristics were chosen according to their validity.



(a) (b)

Figure 3.9: Distribution of radial distance feature for one case in the second layer, illustrated as a scatter plot in (a) and a histogram plot in (b). In both plots blue visualizes background, green represents central gland and red denotes peripheral zone class membership. During the feature extraction step the radial distance is calculated for every image pixel. The resulting values for all pixels within one patient case are plotted on the horizontally-aligned x-axis. Values are normalized and range from 0 to 1. Every dot in Figure (a) represents an image pixel that is colored according to its target label and vertically positioned depending on its radial distance value. A noise offset in vertical y-axis direction has been added for better visibility. Similarly, in Figure (b) the amount of pixels having a certain radial distance value are summed up, resulting in a histogram plot. In both plots it can be seen that pixels having large radial distance values are most likely representatives of the class background whereas small distance measures indicate prostate membership.

For example Figure 3.9 illustrates the distribution of the radial distance feature for one patient case out of the 3 T training set. Radial distances for all pixels are calculated with respect to the mass point and mapped to a range of zero to one. The resulting values are plotted and colored depending on their target class. It appears that pixels located away from the mass point, thus having a large radial distance, belong to the background (colored blue in Figure 3.9). As desired, pixels located close to the mass point are most likely a member of the prostate gland.

In Figure 3.9, points plotted in green belong to the central gland and points colored in red are associated with peripheral zone tissue. Both classes mainly appear in the lower distance range. Due the relatively low overlap of background and prostate gland pixels it is assumed that the radial distance is a good feature for the classification process. It helps to detect pixels belonging to the prostate; however, it is less useful for distinguishing between prostate zones. This is the reason why multiple features are added together to achieve optimal conditions for the classifier input.

The goal of the classifier training phase is to learn a model or decision function g that maps the input to the correct output. The form of the model is determined based on the training set $\mathcal{D}$. Subsequently, the function g can be applied to novel input $\mathbf{X}'$ where no label data (class assignment) are available. This procedure is referred to as prediction step (see Equation 3.6). The ability to predict outputs given unknown input data is made possible through generalization of the model, which is an important goal in this case. The accuracy of the prediction output depends amongst others on the amount of samples for each class that are included in $\mathcal{D}$.

$$\text{training: } \mathcal{D} = (\mathbf{X}, \mathbf{T}) \rightarrow g \qquad \text{prediction: } \hat{\mathbf{T}} = g(\mathbf{X}') \tag{3.6}$$

Transferred to this application, the goal is to train a model based on existing MR patient cases and corresponding expert labels to enable predictions of prostate zones on novel patient cases. To achieve this goal, various classifiers have been implemented and an extensive evaluation is provided in Chapter 4. In the following, a brief explanation of classifier implementations is provided.

Random Forest

The random forest classifier is implemented using MATLAB's `TreeBagger` class to create an ensemble of decision trees for either classification or regression. To construct the class, a number of parameters need to be specified. For this application, the `TreeBagger` class is constructed and trained as follows:

Listing 3.1: Construction of `TreeBagger` object

```
% train
rf = TreeBagger(...
    NumTrees,...                                % number of decision trees grown
    trainingset.features,...                    % features
    trainingset.targetsSingleColumn,...         % targets
    'Method', 'classification',...              % method used by trees
    'NPrint', NumPrint, ...                     % for console output (# cycles)
    'NVarToSample', sNVarToSample,...           % number of random variables at split
    'Options', paroptions, ...                  % options structure
    'oobpred', LogPredictionConvergence, ...    % set either 'on' or 'off'
    'oobvarimp', LogVariableImportance...       % set either 'on' or 'off'
    );

% predict
[YFIT, scores] = predict( rf, testcase.features );
```

The `NumTrees` parameter hereby specifies the number of decision trees that should be constructed in the ensemble. Computed features and targets coded using scheme (3.5) are contained in the training set and accessed through `trainingset.features` and `trainingset.targetsSingleColumn` respectively.

Another very important parameter is `NVarToSample`. It specifies the number of features that are selected at random for each decision split. By default, this value is set to the square root of the total number of feature variables (cf. V_1-V_N in Fig. 3.5).

The `Options` structure specifies computational options. In this case the Parallel Computing Toolbox™ is used to open a `matlabpool` to compute decision trees in parallel, following the code in Listing 3.2.

`OOBPred` and `OOBVarImp` parameters are used to assess out-of-bag class probabilities and to estimate out-of-bag feature importance in the ensemble (described in Sec. 2.2.9). Both values are only set to `'on'` for evaluation purposes. For more information regarding out-of-bag estimates it is referred to L. Breiman's descriptions in [48].

Listing 3.2: Parallel options

```
matlabpool( ...       % open a MATLAB pool by specifying
    'local', ...      % matlabpool profile and
    12 ...            % matlabpool size
    )
paroptions = statset( 'UseParallel', true );

% ... construct and train classifier ...

matlabpool close
```

Neural network

The framework includes three different implementations of neural networks. For fast prototyping the current version of this software application makes use of either the *DeepLearnToolbox* (DLT)[6] developed by R. B. Palm [51] or MATLAB's Neural Network Toolbox™. Using the *DeepLearnToolbox*, a multilayered feedforward backpropagation neural network is initialized as shown in Listing 3.3.

Parameter values in Listing 3.3 do not reflect values of the final setup; however, they provide a general idea of the configuration process. One major part of setting up the neural network is to specify its topology. The first layer contains neurons equal to the amount of input features and is called the input layer. If, for example, the pattern for one pixel consists of 40 features, this layer is comprised of 40 neurons.

Similarly, the number of nodes in the output layer are determined by the amount of target classes. In case of this application, three target classes exist. The only neuron count that is specified by the user is the amount of hidden neurons as well as the number of hidden layers. In Listing 3.3 three hidden layers, each with either 150, 100 or 50 neurons are defined. There is no universally applicable rule to determine the "best" number of layers and neurons, respectively. The network topology that works best for a specific application can only be assessed by performing experiments. Evaluations for this work can be found in Chapter 4.

[6] The DeepLearnToolbox by R.B. Palm is available at `https://github.com/rasmusbergpalm/DeepLearnToolbox`, date accessed: July 2013.

Listing 3.3: Neural network implementation using the DLT

```
% configure number of neurons in each layer
inputsize = size(trainingset.features,2); outputsize =
size(trainingset.targetsMultiColumn,2); NumHidden = [150 100 50];
NeuronsPerLayer = [inputsize NumHidden outputsize];

% set up neural network
nn = nnsetup( NeuronsPerLayer );

nn.normalize_input = 0;            % input normalization disabled; already normalized
nn.learningRate = 1;               % learning rate parameter
nn.activation_function = 'sigm';   % Sigmoid activation function
opts.numepochs = 4;                % Number of iterations through the training data
opts.batchsize = 20;               % Take a mean gradient step over this many samples

% train network
nn = nntrain( ...
    nn, ...                                % network that has been preconfigured
    trainingset.features, ...              % features
    trainingset.targetsMultiColumn, ...    % targets in 1-of-K coding scheme
    opts );

% predict
prediction = nnpredict(nn, testcase.features); scores = prediction.scores;
labels = prediction.i;
```

Network training is performed during multiple epochs defined by `opts.numepochs` with the goal of reducing the overall mean squared error. The DLT requires the target vector to be formatted in the 1-of-K coding scheme; therefore, the `trainingset.targetsMultiColumn` set is used.

When configuring the `batchsize` parameter it is important to know that the total number of input samples has to be divisible by the `batchsize` value with a remainder of 0. However, the number of samples varies depending on the composition and amount of training cases. Therefore, the developed framework only allows a desired value for the batch size to be set and internally determines the maximum size that can be used and treats the desired value as a limit. Thus the actually used `batchsize` might be smaller than intended by the user.

Neurons use sigmoidal activation functions and weights are adjusted by back-propagating the error at the output units. Label prediction is accomplished by executing a feed-forward step with pixel features for current patient cases as network inputs. Final labels are assigned according to the maximum of resulting class probabilities. One drawback of this toolbox is that it does not directly support the use of parallel computing.

In addition to a feed-forward network, a Deep Belief Network (DBN) is implemented using the DLT. The configuration process of a DBN is shown in Listing 3.4. For the usage in this application, the main difference to the feed-forward network is the network setup. As it can be seen in Listing 3.3, the feed-forward net is set up using the `nnsetup` method. As a result, neurons within all layers are initialized with random weights. Thus, each time a feed-forward network is initialized the network parameters are different and might produce a different solution. Introducing the additional step of training a DBN on the other hand ideally leads to better initializations of weights.

Notice that the deep belief network in Listing 3.4 is set up and trained by basically utilizing only the features as input. This type of network is capable of recognizing patterns and performing input clustering without requiring any targets and thus represents an unsupervised learning approach. In fact, this is the strength of a deep belief network. In many applications it replaces the time-consuming manual feature selection process and enables input clustering using raw input data. However, within the application described in this work, the network input is comprised of the feature matrix obtained during the feature extraction step. The goal here is to obtain a better pre-initialization of the weight matrix. Using `dbnunfoldtonn`, which replaces `nnsetup`, the DBN network is unfolded to create a feed-forward network and the following steps remain unchanged.

Listing 3.4: Deep belief network implementation using the DLT

```
% configure number of neurons in hidden layers
NumHidden = [100 100];

% set up deep belief network
dbn.sizes = NumHidden;              % set up a 100-100 hidden unit DBN

dbn.normalize_input = 0;            % input normalization disabled; already normalized
dbn.learningRate = 1;               % learning rate parameter
dbn.activation_function = 'sigm';   % Sigmoid activation function
opts.numepochs = 4;                 % Number of iterations through the training data
opts.batchsize = 20;                % Take a mean gradient step over this many samples
opts.momentum = 0; opts.alpha = 1;

% perform DBN setup and training
dbn = dbnsetup( dbn, trainingset.features, opts ); dbn = dbntrain( dbn,
trainingset.features, opts );

% use DBN weights to initialize a NN
nn = dbnunfoldtonn( dbn, size(trainingset.targetsMultiColumn, 2) );
nn.activation_function = 'sigm';

% train neural network
nn = nntrain( nn, trainingset.features, trainingset.targetsMultiColumn, opts
);

% predict
prediction = nnpredict(nn, testcase.features); scores = prediction.scores;
labels = prediction.i;
```

In addition to the implementations above using the *DeepLearnToolbox*, a neural network is created with the MATLAB toolbox for neural networks (cf. Listing 3.5). The process is similar but with the advantage of native support for parallel computing.

A `patternnet` is related to a `feedforwardnet` with the exception that it uses a hyperbolic tangent sigmoid transfer function in the last layer. A network is trained according to `nn.trainFcn` and `nn.trainParam`. The training stops if the maximum number of epochs specified through `nn.trainParam.epochs` is reached or a performance goal is met. Validation is performed internally.

Listing 3.5: Neural network implementation using the Neural Network Toolbox™

```
% configure number of neurons in hidden layers
NumHidden = [100 100];

% set up patternnet
nn = patternnet(
    NumHidden, ...                    % set up a 100-100 hidden unit NN with
    'trainscg' );                     % Scaled Conjugate Gradient training function

nn.trainParam.epochs = 2000;          % maximum number of training iterations
nn.trainParam.show = 1;
nn.trainParam.time = 2000;            % maximum training time
nn.trainParam.showWindow = 1;
```

```

% train neural network
matlabpool( 'local', 12 )              % open matlabpool

nn = train(nn, ...                     % preconfigured network
    trainingset.features', ...         % features (transposed)
    trainingset.targetsMultiColumn',... % targets (transposed)
    'useParallel','yes',...            % calculation on parallel workers
    'showResources','yes',...          % command line resource usage output
    'useGPU','yes');                   % calculations on gpu device if suported

matlabpool close;                      % close matlabpool

% predict
scores = nn( testcase.features' ); labels = vec2ind(scores)';
```

K-means

In Listing 3.6 the creation of a *k*-nearest neighbor classification object is shown. The distance metric and the number of neighbors can be specified using key-value argument pairs of `Distance` and `NumNeighbors` parameters, respectively. Predictions are made by calling the `predict` method. Note that the `ClassificationKNN` object contains all the data used for training.

Listing 3.6: K-means implementation

```
% train
km = ClassificationKNN.fit( ...
    trainingset.features, ...              % features
    trainingset.targetsSingleColumn, ...   % targets in single column format
    'BreakTies', 'nearest', ...            % if multiple classes have the same smallest
        cost
    'Distance', 'euclidean', ...           % distance metric
    'DistanceWeight', 'equal', ...         % specifies distance weighting function.
    'NumNeighbors', 1, ...                 % number of nearest neighbors
    'prior', 'empirical' );                % prior probabilities for each class

% predict
[labels,scores] = predict( km, testcase.features );
```

Naive Bayes

With the training samples this method estimates the parameters of the probability distribution of features given the corresponding class labels. After the classifier has been trained, it can compute posterior class probabilities for unseen test samples. Test samples are classified according to the largest posterior probability. The general implementation process is described shortly in Listing 3.7.

Listing 3.7: Naive Bayes implementation

```
% train
nb = NaiveBayes.fit( ...
    trainingset.features, ...              % features
    trainingset.targetsSingleColumn, ...   % targets in single column format
    'Distribution', 'normal', ...          % distribution for data modeling
    'Prior', 'empirical' );                % prior probabilities for each class

% predict
labels = nb.predict( testcase.features ); scores = posterior( nb,
testcase.features );
```

3.3.4 Motivation for a two layer topology

This section presents the algorithm pipeline and addresses details in terms of topology. Considering the features described in Section 3.3.2, one can see that many of them are distance-based. Hence, the computation strongly relies on the prostate mass point and bounding box calculations and subsequent image stack cropping performed during preprocessing (cf. Section 3.3.1). Due to the fact that the prostate gland is subject to strong inter-patient variabilities in size and shape, the variance within the distance feature values of different patient cases is significant.

Consider for example the radial distance feature. If the prostate gland is small relative to the bounding box, a normalized radial distance of 0.4 will not include prostate tissue any more. On the contrary, in case of a large gland, even a radial distance value of 0.8 may include one of the prostate zones. This characteristic leads to difficulties within the classifier training phase, because in one case a feature value of 0.4 indicates a member of the background class, whereas in the other case this value is far from representing background.

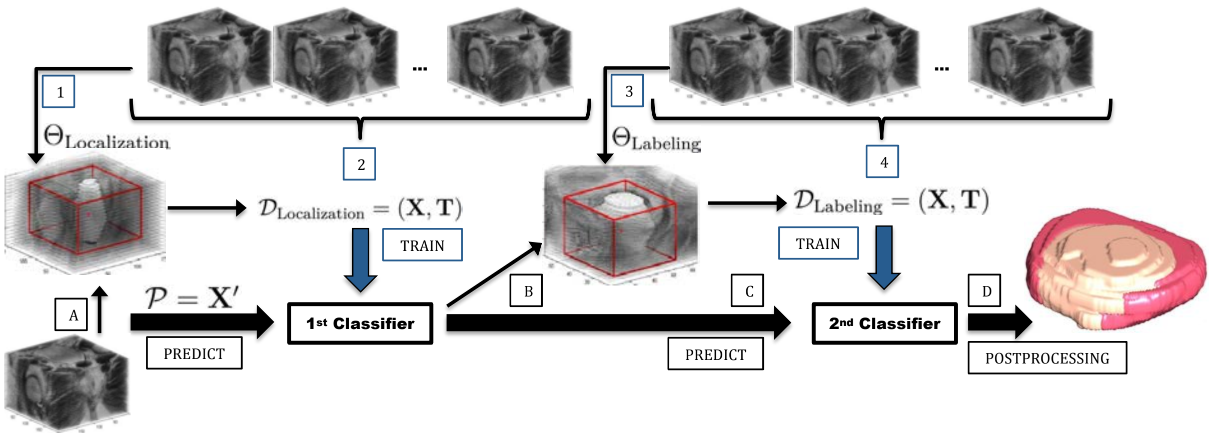

1
$\Theta_{\text{Localization}}$
2
$\mathcal{D}_{\text{Localization}} = (\mathbf{X}, \mathbf{T})$
TRAIN
3
Θ_{Labeling}
4
$\mathcal{D}_{\text{Labeling}} = (\mathbf{X}, \mathbf{T})$
TRAIN
A
$\mathcal{P} = \mathbf{X}'$
PREDICT
1st Classifier
B
C
PREDICT
2nd Classifier
D
POSTPROCESSING

Figure 3.10: This figure illustrates the algorithm pipeline and two layer topology. In a first preprocessing step, dataset knowledge is extracted in the form of an average mass point and a maximum bounding box (1). This information is bundled in $\Theta_{\text{Localization}}$ and the available training cases are then used to generate the training set $\mathcal{D}_{\text{Localization}}$ (2). Afterwards, the first classifier is trained utilizing $\mathcal{D}_{\text{Localization}}$. In an analogous manner the classifier of the labeling layer can be trained (3, 4). The main difference is in computing Θ_{Labeling}. The second classifier requires a more accurate bounding box based on a previous localization of the prostate. This localization is simulated by an exact bounding box computation based on the ground truth labels. Once the classifiers are trained, novel input can be processed. Steps A to D indicate this procedure. Firstly, a new input stack is cropped based on the previously determined maximum bounding box and a feature matrix $\mathbf{X}'$ is calculated that serves as an input for the first classification layer (A). Based on the output results, a new bounding box is obtained that is more accurate (B). Feature values are adjusted and a label prediction is performed through the second classifier (C). After postprocessing, the final segmentation output is obtained (D).

During testing it was found that one classification layer does not produce satisfying results. It is considered that distance features have a high impact on the decision-making process; however, the calculations suffer from the problems described above. To overcome this inaccuracy, a second classification layer is designed that works on the results of the first layer, resulting in a hierarchical topology. The basic concept is illustrated in Figure 3.10. The aim of the first classifier is to locate the prostate within the image stack, therefore it is called *localization layer*. Due to its responsibility for outlining the prostate zones, the succeeding classifier is then denoted *labeling layer*.

It is important to note that now both of the classifiers need to be trained, based on varying training sets. The most important aspect is that the first layer uses a maximum bounding box (calculated over all training cases) for the cropping procedure, which has an impact on all distance features. On the other hand, training of the second classifier is based on an exact bounding box for each case that is obtained from the ground truth labels. This step is necessary as it simulates prior knowledge of the prostate location. In the prediction phase this knowledge is determined by the output result of the localization layer. Besides the bounding box, the mass point computation is affected by this hierarchical model as well (average mass point vs. true mass point for each case).

Steps A to D in Figure 3.10 describe the prediction phase of the algorithm that can be performed once all classifiers are properly trained. The proposed algorithm is capable of executing a prediction for one case at a time. Hence, for n cases steps A-D have to be repeated n times. A new image stack $\mathbf{I}(i,j,l)$ is cropped utilizing the maximum bounding box obtained during preprocessing in the training phase, resulting in $\mathbf{I}'(i,j,l)$. In addition to cropping, other preprocessing steps are executed including gray-level transformations and down-sampling. Next, feature extraction $\mathbf{X}_1' = f^{\mathbf{I}}\left(\mathbf{I}'(i,j,l)\right)$ is performed to generate the input $\mathbf{X}_1'$ for the first classifier and consequently to produce a prediction result $\hat{\mathbf{T}}_1 = g_1(\mathbf{X}_1')$.

Utilizing the bounding box computation based on $\hat{\mathbf{T}}_1$, the original input $\mathbf{I}(i,j,l)$ is cropped obtaining $\mathbf{I}''(i,j,l)$. $\mathbf{I}''(i,j,l)$ is then smaller in terms of dimensions than $\mathbf{I}'(i,j,l)$ of the first layer. Moreover, the actual mass point of the predicted prostate structure in $\hat{\mathbf{T}}_1$ is determined. Features are then extracted from $\mathbf{I}''(i,j,l)$ by additionally using the information of the prediction result to gain $\mathbf{X}_2' = f^{\mathbf{I}}\left(\mathbf{I}''(i,j,l)\right)$. $\mathbf{X}_2'$ serves as input for the second classifier in order to predict $\hat{\mathbf{T}}_2 = g_2(\mathbf{X}_2')$. After a minor postprocessing routine the final prediction result $\hat{\mathbf{T}}$ is obtained.

The proposed algorithm does not require user interaction of any kind and consequently represents a fully automated approach for prostate zonal segmentation. The framework is capable of simulating different topologies; thus, algorithm modes have been developed. Figure 3.10 shows the *A2 mode*, which identifies an automated mode incorporating two classification layers. For comparison purposes, an *A1 mode* has been preserved where only the first localization layer is used. Naturally, this mode provides the least accurate results due to the variabilities of the prostate discussed before. However, the A1 mode is still very important as the result of the first layer directly affects the second classifier input. Hence, special focus is directed to the robustness of the first layer and the A1 mode therefore enables a separate simulation.

The third mode represents an interactive approach and is called *I mode*. It is common practice amongst radiologists at the MD Anderson Cancer Center to mark the extent of the prostate structure within an MR image stack with a measurement on the sagittal, coronal and transversal planes. This tree-axis linear dimension measurement of expert markups directly corresponds to the bounding box computation performed for the automated modes. Consequently, this knowledge can be incorporated to crop every image stack with high accuracy. Hence, novel test cases can be cropped precisely due to the additional information regarding the prostate position. Note that cropping for the A2 mode is based on the prediction outcome of the first layer and therefore is subject to inaccuracies. The I mode was denoted interactively because in case these dimensional markups are not available, they could be added on the fly by a user (assuming the GUI supports this kind of interaction). This minimal interaction results in the most accurate segmentation outcome. This can be seen in Section 4.2.2, which provides a result comparison for the three algorithm modes.

3.3.5 Postprocessing of the prediction result

The proposed algorithm solves the segmentation problem by means of pixel classifications. The output $\hat{\mathbf{T}}$ therefore consists of raw pixel assignments. To obtain a smooth surface or volume, minor postprocessing tasks are necessary. The predicted image stack includes labels for central gland ("CG") and peripheral zone ("PZ"). In a first step, two binary image stacks $\mathbf{I}_{\text{cg}}$ and $\mathbf{I}_{\text{pz}}$ are created that include either central gland labels or peripheral zone assignments, respectively.

$$\mathbf{I}_{\text{cg}}(i,j,l) = \begin{cases} 1 & \text{if } \hat{\mathbf{T}}(i,j,l) = \text{"CG"} \\ 0 & \text{otherwise} \end{cases} \quad \mathbf{I}_{\text{pz}}(i,j,l) = \begin{cases} 1 & \text{if } \hat{\mathbf{T}}(i,j,l) = \text{"PZ"} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

To remove pixel outliers and to close holes within the predicted structure of the respective prostate zone, morphological erosion and dilation are applied to each individual slice l in $\mathbf{I}_{\text{cg}}$ and $\mathbf{I}_{\text{pz}}$. Secondly, it was found that the overall robustness of the output result could be increased by smoothing the zonal shape of the prostate. Hence, a three-dimensional Gaussian smoothing is performed on both binary image stacks utilizing a $3 \times 3 \times 3$ dimensional filter kernel. Afterwards, the processed $\mathbf{I}_{\text{cg}}$ and $\mathbf{I}_{\text{pz}}$ are again merged to form one label stack. This output is then rescaled to the original image dimensions to undo the size reductions performed during cropping and down-scaling (see Section 3.3.1).

The result represents the final algorithm segmentation output and is then saved as NRRD file for further consideration (e.g. inspection with the image viewer presented in Section 3.2). The NRRD save function was implemented in MATLAB and developed for the purpose of this work.

3.4 Framework development

Upon completion of the algorithm development, the segmentation routine has been applied and tested on a variety of datasets. It has been found that the algorithm code is very inconvenient to use and to configure for all kinds of situations. Hence, the development was continued to generate a framework that incorporates the segmentation algorithm and image viewer presented in sections 3.3 and 3.2, respectively. The goal is to provide an easy-to-use interface to test and adopt the segmentation routine on novel data.

property name	type	description
PATH_TRAINING_CASES	char	Path to the respective training images and labels.
PATH_TESTING_CASES	char	Path to the respective testing images (if needed).
CALCVOLUME	logical	Enable/Disable volume calculation.
CLASSIFIER_MODE	char	Algorithm mode ('A1','A2' or 'I').
CLASSIFIER_TYPE	char	Type of classifier (e.g. 'RF' = Random Forest).
CLASSIFIER_SETTING	structure	Contains settings for the chosen classifier type (e.g. number of trees).
FEATURE_SELECTION_LOCALIZATION	structure	Incorporated features and neighborhood size for the localization layer.
FEATURE_SELECTION_LABELING	structure	Same as above but for the labeling layer.
FRACTION_TRAINING	numeric	Fraction training/validation for cross-validation purposes (e.g. 0.85).
NUM_ITERATION_EVALUATION	numeric	Number of cross-validation cycles.
ALL_REGIONS_LOCALIZATION	logical	Should the local. layer distinguish between structure regions or take the whole object.

Table 3.1: Properties of the `MedObjSegmentation` class.

Basically, the classification algorithm and all input and output generation processes have been integrated into one `MedObjSegmentation` class. During the construction of an object of the class `MedObjSegmentation`, default property values and paths are set. In addition, public properties then allow the configuration of this object and tasks can be executed via public function calls.

Table 3.1 lists primary class properties that are used mainly for configuration purposes. Public methods are listed in Table 3.2. Most of the class methods are designed for evaluation tasks, for example, cross-validation and error computation functions. They are used to test the algorithm performance on different datasets to find a robust setting. Once this setting is found, a classifier object can be generated, trained on a training set and saved to the hard drive (or external storage device).

function name	description
generateTrainingSet	Generate a training set (involves feature extraction using images and corresponding labels).
loadTrainingSet	Loads an existing training set (a training set can be saved).
generateTestingSet	Generate a testing set.
loadTestingSet	Loads an existing testing set.
loadExistingClassifier	Loads a pretrained classifier object.
performClassifierTraining	Perform classifier training using a generated training set as input.
performPrediction	Predict labels for a certain test set.
performCVStep	Perform one cross-validation step.
performPassThrough	Combines train and test set generation, classifier training and output prediction.
loadResult	Loads an output result for further processing.
plotResult	Plots an output result showing all respective MR image slices.
getPredictionError	Computation of error metrics (requires ground truth labels).
evaluateClassifier	Performs certain amount of CV steps optionally with varying training set size.
plotCase	Plots one case with three-dimensional volume view.
saveNRRDs	Save output result (labels) as NRRD file.

Table 3.2: Methods of the `MedObjSegmentation` class.

The image viewer is able to create or work with instances of the `MedObjSegmentation` class. For instance, it can load images of a patient case and call prediction functions that use the pretrained classifier object to generate an output. The output result can then be viewed directly with the image viewer as contours of the predicted structure regions are plotted onto each image slice. The short overview of the most important public class methods listed in Table 3.2 together with the large variety of configurations through class properties demonstrates the possibilities of the framework for further research purposes.

3.5 Description of error metrics

Despite visual inspection being considered as the best evaluation process, the calculation of various error metrics has been implemented to enable reliable result comparison. It is important to note that these calculations can only be performed for the training set because ground truth labels need to be available. Hence, the metrics are used mainly for cross-validation purposes. Computations are applied to both of the predicted prostate zones following the scheme illustrated in Algorithm 1.

Suppose the ground truth targets $\mathbf{T}$ and predicted class assignments $\hat{\mathbf{T}}$ are available. Both of them consist of the label stacks $\mathbf{L}_{\mathrm{q}}(i,j,l)$ or $\hat{\mathbf{L}}_{\mathrm{q}}(i,j,l)$, whereby $q = 1,...,Q$ and Q represents the total amount of predicted patient cases. Each of these label stacks can then be split according to the class memberships to generate $\mathbf{L}_{\mathrm{q,k}}(i,j,l)$ or $\hat{\mathbf{L}}_{\mathrm{q,k}}(i,j,l)$ with $k = 1,...,K$. K thereby indicates the total number of class labels.

This partitioning is performed to evaluate the segmentation outcome for the two prostate zones separately. Error metrics are then calculated for the central gland and peripheral zone class labels throughout all test cases. The final score for the Dice similarity coefficient (DSC) $\eta_{\mathrm{k,dice}}$ per label k is obtained by averaging the individual case scores $\eta_{\mathrm{q,k,dice}}$ over all patient cases. Analogous, other scores are determined. The following subsections describe the calculation of individual error metrics.

Algorithm 1 Error calculation process.

```
Require: $\mathbf{T}$, $\hat{\mathbf{T}}$        ▷ ground truth and prediction result for patient cases 1 to Q

  $\mathbf{T} \rightarrow \sum_{q=1}^{Q} \mathbf{L}_{\mathrm{q}}(i,j,l) \rightarrow \sum_{q=1}^{Q} \sum_{k=1}^{K} \mathbf{L}_{\mathrm{q,k}}(i,j,l)$
  $\hat{\mathbf{T}} \rightarrow \sum_{q=1}^{Q} \hat{\mathbf{L}}_{\mathrm{q}}(i,j,l) \rightarrow \sum_{q=1}^{Q} \sum_{k=1}^{K} \hat{\mathbf{L}}_{\mathrm{q,k}}(i,j,l)$

  for $q \leftarrow 1$ to $Q$ do                       ▷ for all patient cases
      for $k \leftarrow 1$ to $K$ do                   ▷ for all class labels
          $\eta_{\mathrm{q,k,dice}} \leftarrow$ DiceSimilarity2DImage($\mathbf{L}_{\mathrm{q,k}}$, $\hat{\mathbf{L}}_{\mathrm{q,k}}$)
          $\eta_{\mathrm{q,k,Hausdorff}} \leftarrow$ HausdorffDist($\mathbf{L}_{\mathrm{q,k}}$, $\hat{\mathbf{L}}_{\mathrm{q,k}}$)
          ...
      end for
  end for

  $\eta_{\mathrm{k,dice}} \leftarrow \frac{1}{Q} \sum_{q=1}^{Q} \eta_{\mathrm{q,k,dice}}$      ▷ average error over all patient cases
  $\eta_{\mathrm{k,Hausdorff}} \leftarrow \frac{1}{Q} \sum_{q=1}^{Q} \eta_{\mathrm{q,k,Hausdorff}}$
  ...
```

3.5.1 Dice coefficient

The Dice coefficient is used to compare the similarity of ground truth segmentation and prediction result. Given two images X and Y, the index η_{dice} is calculated as

$$\eta_{\text{dice}}(X, Y) = \frac{2\,|X \cap Y|}{|X| + |Y|}. \tag{3.8}$$

In terms of false positive (FP), false negative (FN), true negative (TN), and true positive (TP) counts, this formula may also be written as

$$\eta_{\text{dice}} = \frac{2 \times TP}{(FP + TP) + (TP + FN)}. \tag{3.9}$$

The higher the value for η_{dice}, the better is the match between the segmentation result and the manually generated expert labels. A value of 0 denotes no overlap and a value of 1 indicates perfect agreement. In terms of this work, the Dice coefficient is computed following Dr. Rex Cheung's implementation[7] (cf. Listing 3.8).

Listing 3.8: Dice coefficient computation

```
% Copyright (c) 2012, Rex Cheung
% All rights reserved.
function DiceCoef = DiceSimilarity2DImage( img1, img2 )
    %1. set one image non-zero values as 200
    img1(img1>0)=200;

    %2. set second image non-zero values as 300
    img2(img2>0)=300;

    %3. set overlap area 100
    OverlapImage = img2-img1;

    %4. count the overlap100 pixels
    [r,c,v] = find(OverlapImage==100);
    countOverlap100=size(r);

    %5. count the image200 pixels
    [r1,c1,v1] = find(img1==200);
    img1_200=size(r1);

    %6. count the image300 pixels
    [r2,c2,v2] = find(img2==300);
    img2_300=size(r2);

    %7. calculate Dice Coef
    DiceCoef = 2*countOverlap100/(img1_200+img2_300);
end
```

3.5.2 Hausdorff distance

The Hausdorff distance represents a measure of the greatest of all the spatial distances from a point in one set to the closest point in the other set in a Euclidean metric space. Suppose set $A = \{a_1, ..., a_p\}$ contains all points of the prediction result and set $B = \{b_1, ..., b_q\}$ all points of the ground truth segmentation. For one point a_1 in A, the

[7] `DiceSimilarity2DImage(img1, img2)` by Dr. Rex Cheung is available at `http://www.mathworks.com/matlabcentral/fileexchange/36322-inspire-utility-to-calculate-dice-coefficient`, file version: April 23, 2012, date accessed: July 2013.

distances to all points b_j in B ($1 \leq j \leq q$) are calculated. Amongst the resulting values, the minimum distance is found and saved. This process is repeated for all other points a_i in A, where $2 \leq i \leq p$. Following this, the overall largest of these minimum distances then determines the value of $h(A, B)$, referred to as the directed Hausdorff distance [52].

In mathematical terms, the directional Hausdorff distance is defined as

$$h(A, B) = \max_{a \in A} \left(\min_{b \in B} (\|a - b\|) \right). \tag{3.10}$$

As desired, the function $h(A, B)$ finds the point a within the prediction set A that is farthest from any point in B and measures the distance from a to its nearest neighbor in B. Analogous, $h(B, A)$ is computed finding the point $b \in B$ that is farthest from its corresponding point (point of minimal distance) in A. The final score $\eta_{\text{Hausdorff}}$ for the Hausdorff measure is then defined as

$$\eta_{\text{Hausdorff}} = \max(h(A, B), h(B, A)). \tag{3.11}$$

The Hausdorff distance quantifies how well prostate boundaries in a prediction output correspond to boundaries in the ground truth segmentation by indicating worst case deviation. Calculation of the Hausdorff distance is performed using a MATLAB implementation by Zachary Danziger[8].

3.5.3 Sensitivity

The sensitivity measure indicates the fraction of true positives that were included in a segmentation and is calculated using the formula

$$\eta_{\text{sensitivity}} = \frac{TP}{TP + FN}. \tag{3.12}$$

The best score would be 1, suggesting that all pixels in the ground truth segmentation were also included in the segmentation result. While this measure basically represents the fraction of true positives that were correctly detected, it does not consider false positives and true negatives. Due to this characteristic, sensitivity should never be used without a specificity measure to describe the segmentation quality.

3.5.4 Specificity

Specificity measures the fraction of negatives that are correctly detected determined by the formula

$$\eta_{\text{specificity}} = \frac{TN}{TN + FP}. \tag{3.13}$$

For the prostate segmentation task described in this work, a score of 1 for specificity indicates that all non-prostate pixels were labeled as background; a value of 0 suggests

[8]`HausdorffDist(P,Q,lmf,dv)` by Zachary Danziger is available at `http://www.mathworks.com/matlabcentral/fileexchange/26738-hausdorff-distance`, file version: April 3, 2013, date accessed: July 2013.

that all non-prostate pixels were labeled as either one of the prostate zones. Again, the specificity measure should be accompanied by a sensitivity score to achieve reliable and informative results. Both sensitivity and specificity are computed using `classperf` as shown in Listing 3.9.

Listing 3.9: Evaluating classifier performance using MATLAB's `classperf`

```
% create classifier performance object and extract error measures of interest
CP = classperf( groundTruthLabels , classifierPrediction ) % Evaluate classifier
    performance

% Correctly Classified Positive Samples / True Positive Samples
Sensitivity = CP.Sensitivity;

% Correctly Classified Negative Samples / True Negative Samples
Specificity = CP.Specificity;

% Correctly Classified Positive Samples / Positive Classified Samples
PositivePredictiveValue = CP.PositivePredictiveValue;

% Incorrectly Classified Samples / Classified Samples
ErrorRate = CP.ErrorRate;
```

Chapter 4

Results

An evaluation of the first prototype of the initial one-layer framework (basically A1 mode described in Section 3.3.4) was performed through the participation at the *2013 NCI-ISBI Grand Challenge on Automated Segmentation of Prostate Structures*. This approach used a feed-forward neural network for classification purposes. The classification algorithm was pretrained using patient cases of the training set described in Section 3.1. A testing set of 10 new prostate MRI cases was provided for label predictions during the on-site challenge. The execution of training and prediction routines was automated, thus requiring absolutely no user interaction.

In total, five error metrics were considered for comparison and accuracy measurement of label prediction results and ground truth (expert) markups. All metrics were averaged over the 10 testing cases to obtain final scores. The developed algorithm ranked third out of five international approaches, reaching average DSCs of 0.38 for the PZ and 0.68 for the CG. For consideration of worst case scenarios and outliers the Hausdorff distance of boundaries was computed, which resulted in values between 9 mm to 10 mm for both of the prostate zones. Specificity results were 0.997 for PZ and 0.999 for CG, representing the overall robustness of the algorithm.[1]

In further course of the development the accuracy of the algorithm was enhanced amongst others by adding a second network layer (cf. Section 3.3.4) and the interactive (I-) mode. Moreover, additional features were incorporated to improve the quantity of descriptive information that is extracted from the input data. An extensive evaluation of the current algorithm performance is provided in the following sections. At first, an overview of the experimental setup for all framework evaluations is provided. This includes a description of the cross-validation concept, computer hardware and result visualization properties.

Afterwards, the suitability of various implemented classifiers for prostate zonal segmentation in magnetic resonance images is proven based on different training and testing data. All classifiers offer certain options for configuration purposes; thus, test runs were performed to ascertain a robust setting with maximum prediction accuracy. These experiments include an analysis of the number of random variables at each decision split in

[1]MIDAS Digital Archiving System provided by Kitware Inc., NCI-ISBI 2013 Prostate Challenge, `http://challenge.kitware.com/midas/community/5`, date accessed: April 2013.

a decision tree, presented in Section 4.2.1. In addition, Section 4.2.2 evaluates the label prediction accuracy for each of the three different algorithm modes, termed A1, A2 and I, based on a random forest classifier.

In terms of neural networks, one of the major properties that can be specified by the user is the hidden layer topology. This setting as well as the proper choice for the network train function are addressed in Section 4.3.1 and Section 4.3.2, respectively. Lastly, sections 4.4 and 4.5 assess the applicability of k-means and Naive Bayes classification methods for the segmentation approach presented in this work. Furthermore, other aspects to judge the practicality of this approach, such as temporally extending the algorithm training and label prediction as well as computing hardware requirements are discussed in this chapter.

4.1 Evaluation setup and characteristics of result visualization

To generate reliable and reproducible results, the setup for all experiments is described in this section. A cross-validation concept is implemented to estimate the model accuracy and to assess the result generalization to independent datasets. A repeated random sub-sampling validation is used, whereby the training data $\mathcal{D}$ are randomly partitioned into two subsets. For this purpose, the subset $\mathcal{D}_{\text{train}} \subset \mathcal{D}$ comprised of 85% of the total training cases is created. In every iteration, the data in $\mathcal{D}_{\text{train}}$ are used to fit a model. The remaining 15% of the training data $\mathcal{D}$ produce the validation set $\mathcal{D}_{\text{validate}}$. Patient cases out of $\mathcal{D}_{\text{validate}}$ are used to assess the prediction accuracy of the previously generated model. The splitting step as well as model training and prediction tasks are repeated for each cross-validation cycle. A total of 10 iterations were performed for every experiment and error metrics were then averaged to obtain the final scores.

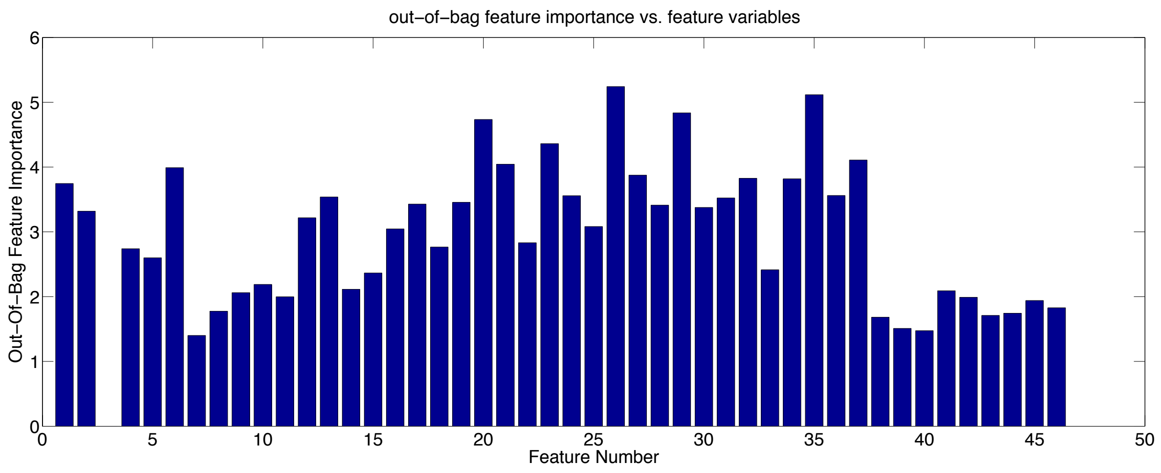


Figure 4.1: Out-of-bag feature importance vs. feature number in a random forest. The horizontal axis represents the specific features, in this case 46 pixel characteristics. The height of the bars indicates the importance of the feature for the decision process. This experiment was performed using 25 cases of the 3 tesla training set. The third feature is the magnetic field strength, which is not important at all because only the 3 T set was used.

The repeated random sub-sampling process is preferred over k-fold cross-validation to make the fraction for the training/validation split independent of the number of cycles. However, it is important to note that when applying this method, some patient cases may never be selected for validation purposes (i.e. may never be contained in $\mathcal{D}_{\text{validate}}$). At the same time, other cases may be selected more than once. This characteristic may lead to result variations when the cross-validation step is repeated with different random subsets. However, due to the relatively small amount of training cases (max. 100 patient cases) and the high amount of iterations, the influence of these variations on the final result are not considered to be critical.

It is important to note that the calculation of the Hausdorff distance was performed in the down-sampled version of the images; thus, results need to be multiplied by factor 2. The resulting score then reflects the distance in pixels of the worst outlier.

Despite the possibility of a specific feature selection enabled by the framework (see class properties in Table 3.1), all experiments incorporated the same features (with the exception of the experiment in Section 4.2.3). Throughout various test runs it was found that all features are of similar importance (see Figure 4.1). Future tests may focus on an enhanced feature selection process, which was out of the scope of this work.

All calculations were performed on a datacenter computer that runs Windows Server 2008 R2 Enterprise (64-bit) as an operating system. In terms of computational hardware, it featured four quad-core AMD Opteron™ processors (model 8356), each clocked at a frequency of 2.31 GHz. Moreover, it was equipped with 128 GB of physical Random Access Memory (RAM). This computer setup profits especially from parallel operations; hence, highly demanding computation tasks were run in parallel whenever possible. This performance aspect primarily concerns classifier training. As this machine was dedicated exclusively to this project, all experiments were conducted in a way to utilize the maximum amount of resources available and therefore reduce algorithm training times to a minimum.

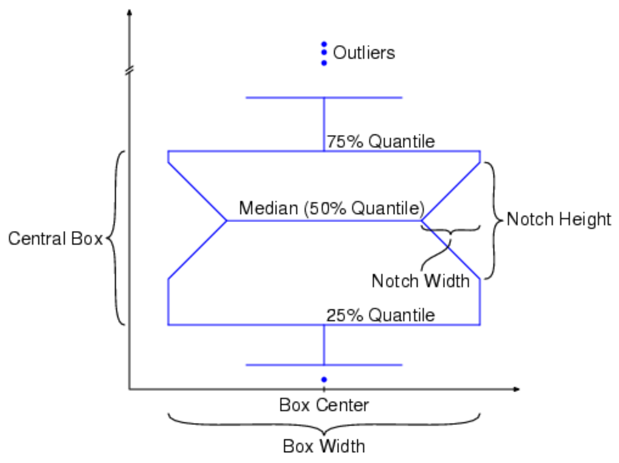


Figure 4.2: Properties of the box plot that is used for result visualization.[2]

To visualize the result in terms of error metrics, a box plot shown in Figure 4.2 is used. Blue boxes indicate error measures for central gland tissue, whereas red boxes denote scores for the peripheral zone of the prostate. On each box, the central line marks the median value (averaged over every validation case throughout 10 cross-validation cycles). The upper and lower edges of the box are specified by the 25th and 75th percentiles. While outliers are plotted individually, two vertical lines that extend from the central box (denoted as whiskers) indicate remaining data points that are extreme values but not considered outliers. Outliers are defined as values that are more than 1.5 times the distance between top and bottom of the box away from the top or bottom of the box, respectively.

4.2 Random forest evaluation

The results presented in this section were achieved using a random forest classifier. The classifier object was created following the description in Section 3.3.3. Classifier training was performed on a parallel MATLAB cluster utilizing the maximum amount of workers possible. The maximum amount of workers on the datacenter computer that was used is given with a value of 12, whereby this number is additionally limited by memory requirements for the generation of the specific decision tree ensemble. The memory requirements mainly increased with the amount of trees and size of the input data. Figure 4.3 shows an example of the resource utilization.

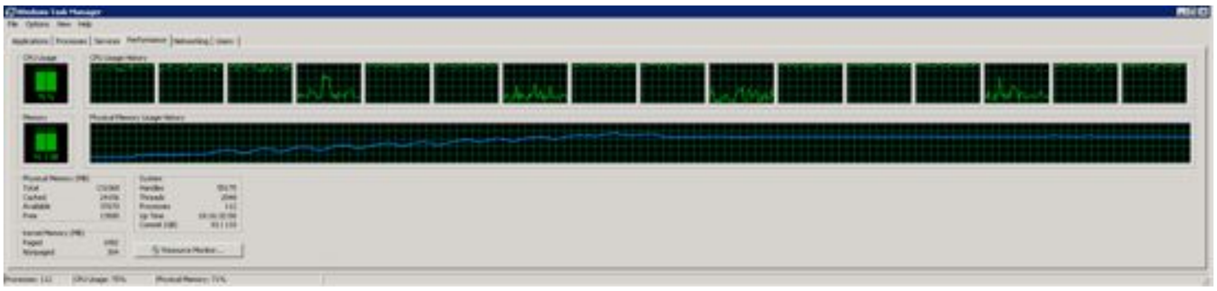


Figure 4.3: Illustration of random forest computing resource requirements during training. A forest with 100 trees was grown using 50 patient cases and 12 parallel workers in algorithm I mode. 12 of the total of 16 CPU cores were equally worked to capacity, which directly corresponds to the amount of workers used. This resulted in an overall CPU usage of 75%. In the plot showing the physical memory usage history it can be seen that the memory requirements increased step-by-step as workers joined the computational ensemble and capped at 91 GB.

For the majority of experiments, the random forest was comprised of an ensemble of 100 decision trees. Growing more than 100 trees resulted in a strong increase in memory requirements, while it did not significantly decrease the overall mean squared error. This correlation is visualized in Figure 4.4.

[2]MATLAB Release 2013a Documentation, The MathWorks, Inc., Natick, Massachusetts, United States., `http://www.mathworks.com/help/symbolic/mupad_ref/plot-boxplot.html`, date accessed: August 2013.

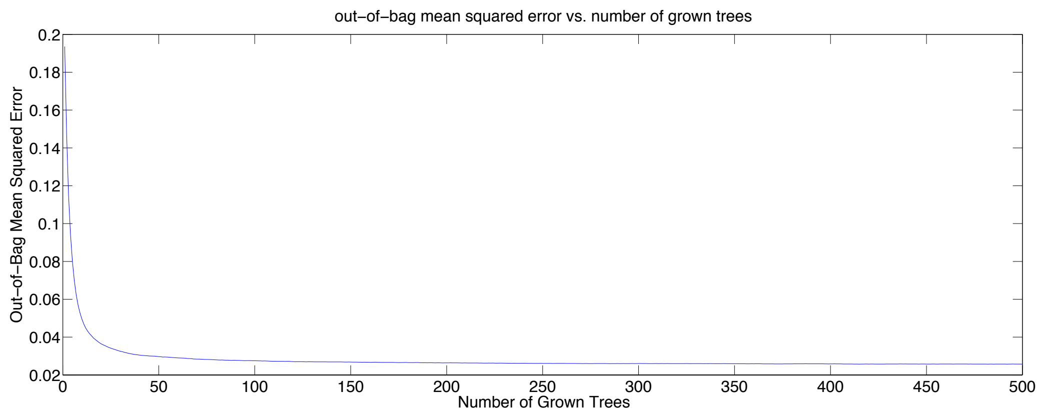


Figure 4.4: Out-of-bag mean squared error vs. number of grown trees in a random forest. This experiment was performed using 25 cases of the 3 T training set. Working with a larger training set might increase the need for additional decision trees.

Unless otherwise noted, the number of random variables `NVarToSample` for each decision split was set to the square root of the amount of features that were used. This amount was found favorable for the purpose of this application, determined by the experiment described in Section 4.2.1. The I mode is compliant with its design the most accurate algorithm mode, which is proven through a trial covered in Section 4.2.2.

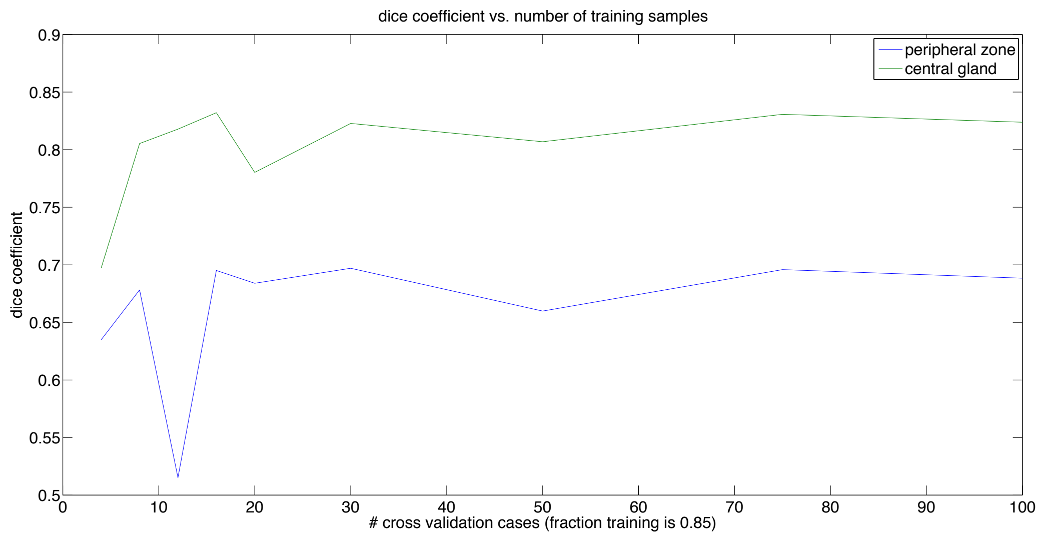


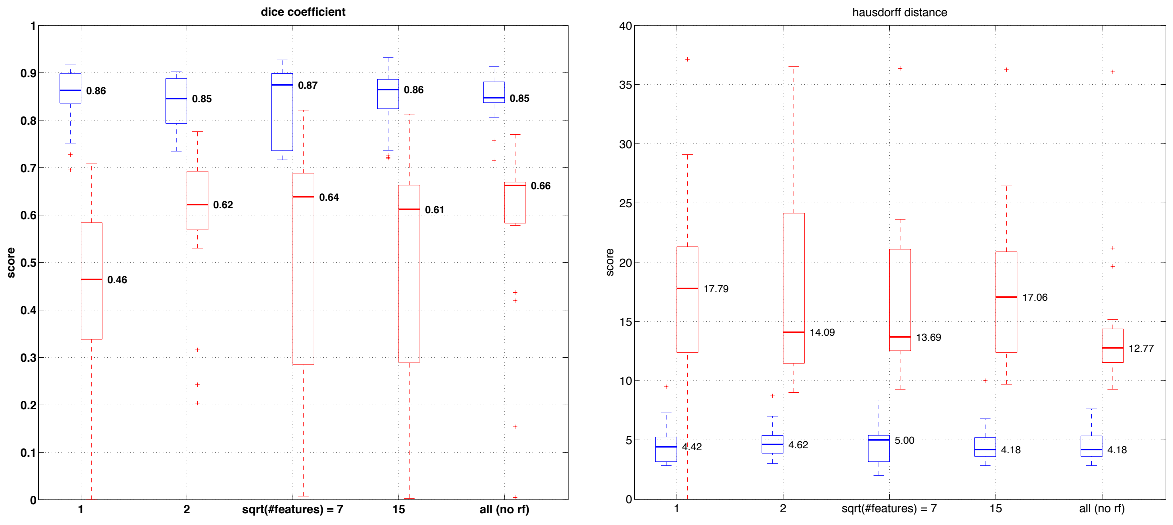
Figure 4.5: This figure illustrates the influence to the Dice coefficient by the amount of training data. A test run with 10 cross-validation iterations at a time was performed on a varying amount of training cases. The number of cases for a certain test run is plotted on the horizontal axis. For each case, the corresponding score of the DSC was calculated. This experiment was conducted using the algorithm I mode and 100 patient cases provided by the MD Anderson Cancer Center. There are hardly any fluctuations if the algorithm training is completed with more than 25 cases, as the accuracy of the label predictions converges to a limit.

It is assumed that if the algorithm learns from a higher number of training cases, the output results will improve. This presumption is true up to a certain limitation. A trial was conducted, where the Dice coefficient was calculated for a varying amount of training cases. The result is illustrated in Figure 4.5.

This experiment shows that it is sufficient to train the learning algorithm on an amount of 25 to 30 patient cases. Incorporating more cases generally leads to a similar outcome. A training set of 25 cases seems small; however, it is not insignificant considering that the classifier makes raw pixel predictions. The final segmentation result is obtained through simple morphological operations on raw pixel classifications. Therefore, the current amount of input values can be estimated by taking the dimensions of the input image stack times the number of cases. After image cropping within the preprocessing step, the resulting stack has average dimensions of 80 px × 70 px × 14 slices. Taking 25 of these training cases, this results in a total of almost 2,000,000 input variables, which is assumed to be adequate for this application. Thus, following this knowledge the majority of evaluation test runs were performed based on 25 training cases to increase the efficiency.

4.2.1 Number of random variables to sample at each decision split in a tree

By means of this experiment, the optimal setting for the `NVarToSample` parameter of the random forest was found. Only 25 cases out of the 3 T training set were used to speed up the computations by reducing resource requirements. The result is illustrated in Figure 4.6.



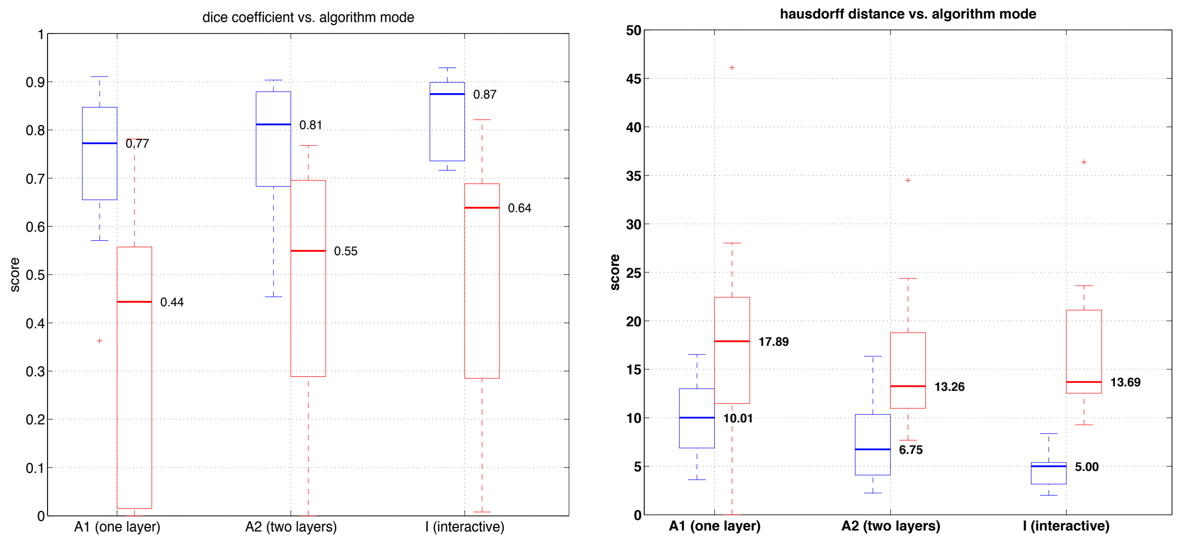
(a) (b)

Figure 4.6: Evaluation of amount of random variables for the random forest classifier. The segmentation algorithm was run in I mode with a varying amount of random variables for each decision split at the generated trees. Figure (a) shows scores for the Dice coefficient and Figure (b) illustrates scores for the Hausdorff distance, respectively. Blue = CG, red = PZ.

The resulting scores for Dice coefficient and Hausdorff distance show that the best accuracy is achieved using either all feature variables or the square root of the number of features (in this case 7 variables). However, the average training time using all feature variables equalled 57 minutes while the usage of 7 random features only took approximately 11 minutes. While the training time is of minor importance for practical applications, it is important when multiple experiments are conducted that involve a training step. When 10 cross-validation cycles are performed, the difference in terms of calculation time amounts to approximately 460 minutes. Hence, considering the minor deterioration in performance, all following random forest experiments were performed with the reduced amount of variables for efficiency purposes.

4.2.2 Comparison of algorithm modes

This experiment compares the segmentation outcome given different algorithm modes. Results are shown in Figure 4.7. The 3 T training set was used for this test run. Both, Dice coefficient as well as Hausdorff distance scores improve throughout the various modes. As desired, the I mode is the most accurate one amongst the three options, where especially the scores for the peripheral zone advance.



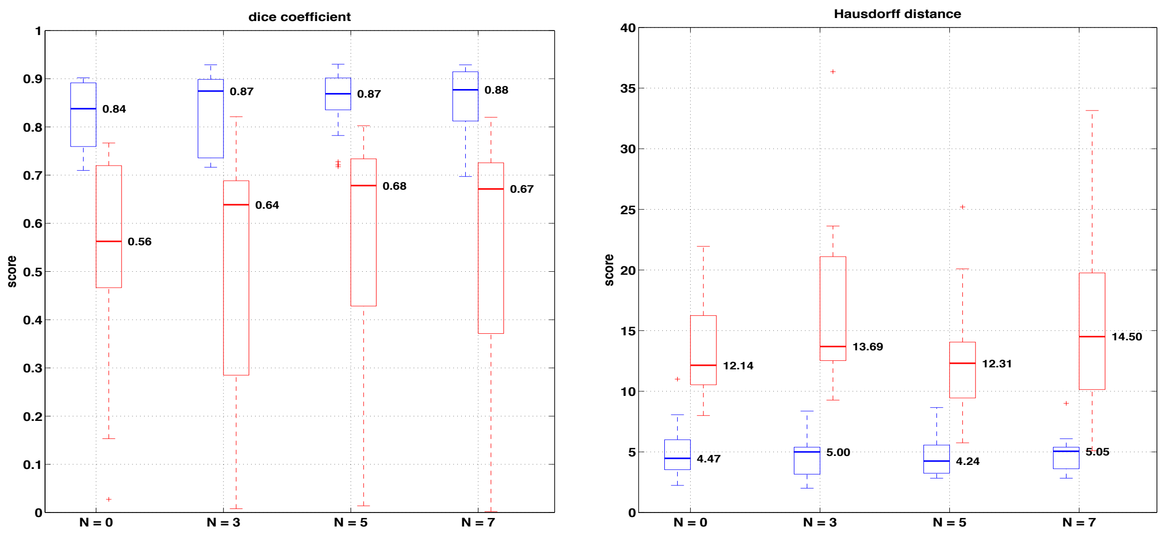
(a) (b)

Figure 4.7: Evaluation of algorithm modes using the random forest classifier. The segmentation algorithm was run in either A1, A2 or I mode. Figure (a) shows scores for the Dice coefficient and Figure (b) illustrates scores for the Hausdorff distance, respectively. Blue = CG, red = PZ.

4.2.3 Assessment of the influence of the neighborhood size for regional features

The goal of this experiment is to compare the prediction outcome of the segmentation algorithm based on the neighborhood size for features that describe regional properties. Section 3.3.2 describes the process of extracting features to achieve a better representation of the original input. In addition, the most important features that are being used for a representation are introduced. Amongst others, they also include characteristics that provide information about a certain pixel neighborhood as well as raw intensity values of the respective neighbors. For each classification layer, the size of the neighborhood that is considered for these computations can be specified in the `FEATURE_SELECTION_LOCALIZATION` and `FEATURE_SELECTION_LABELING` properties of the `MedObjSegmentation` class.

The default value is $N = 3$. Changing this size affects the neighborhood in all three dimensions; thus, a size of $N = 5$ would result in a window of $5\times5\times5$. In comparison to a $3\times3\times3$ window, the amount of pixels involved in the computations then increases from 27 to 125. Thereby, this small change drastically influences the computational effort. At the same time, the integration of raw intensity values of the respective neighborhood pixels into the feature vector also strongly affect its dimensions. The size of the feature vector in turn is responsible for the duration of classifier training times. These consequences need to be considered when changes are made to the size of the neighborhood.



(a) (b)

Figure 4.8: Evaluation of optimal neighborhood size for regional features. 25 patient cases of the 3 T training set were used for this experiment and the segmentation algorithm was run in I mode, incorporating a random forest with 100 trees. The horizontal axis denotes the size N of the neighborhood. A neighborhood size of 0 thereby indicates that no regional features were used at all. Figure (a) shows scores for the Dice coefficient and Figure (b) illustrates scores for the Hausdorff distance, respectively. Blue = CG, red = PZ.

Within the experiment that was performed, N was set to either 0, 3, 5 or 7. The results are illustrated in Figure 4.8. An inspection of Dice coefficient and Hausdorff distance scores suggests that the optimal neighborhood size is 5. However, due to similar results where the inaccuracies were not in relation with the increased computational effort, the default value of $N = 3$ was retained for further experiments. Nevertheless, in the case of a practical application a value of $N = 5$ should be preferred.

4.3 Neural network evaluation

Results in this section were computed utilizing neural networks. The classifier objects were created following the description in Section 3.3.3. The MATLAB Neural Network Toolbox™ supports parallel computing; thus, network training was performed in parallel to enhance efficiency. Again, the amount of workers involved were limited by resource requirements. However, they were defined in order to reach the maximum performance possible given this specific computing hardware. This setup enables a comparison with other classifiers in terms of training time intervals.

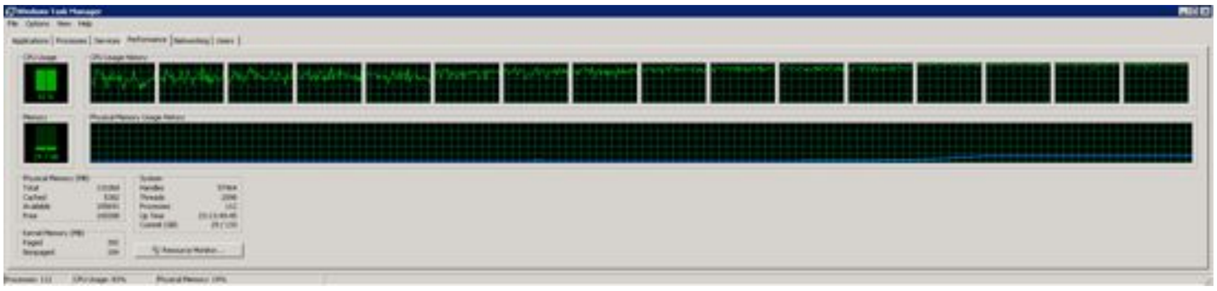


Figure 4.9: Illustration of neural network computing resource requirements during training. A network with 100 hidden neurons in one hidden layer was trained on 25 patient cases using the `trainscp` train function and 12 parallel workers. The CPU usage amounted to 83% while physical memory usage was low with around 25 GB.

Figure 4.9 illustrates resource requirements for neural network training. It can be seen that in comparison to random forest training (cf. Fig. 4.3) a similarly high CPU workload was recorded, while the memory usage was kept low. If the network is set up using the *DeepLearnToolbox*, only serial training is possible.

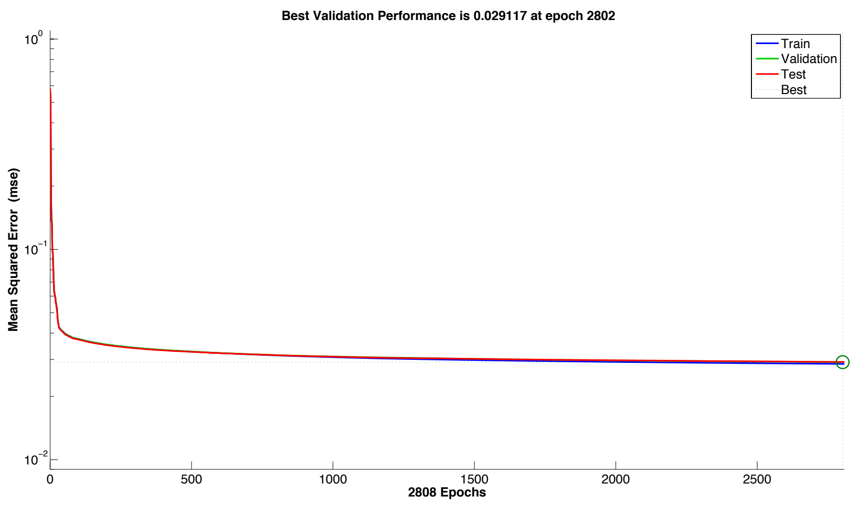


Figure 4.10: Neural network validation performance vs. mean squared error. The validation performance reached a minimum at epoch 2802. The training then continued for 6 more iterations before it stopped. No major problems within the training were indicated, as the validation and test curves were very similar. This plot corresponds to the result presented in Figure 4.13.

Network training was performed over a number of epochs and the validation error was recorded and used as a stopping criterion. Figure 4.10 shows the progression of the mean squared error in relation to the amount of training epochs.

4.3.1 Evaluation of hidden layer topology

The goal of this experiment is to find the optimal hidden layer topology for the neural network. The amount of neurons in the input layer (cf. x_0 to x_D in Fig. 2.6) is determined by the amount of input feature variables. Similar, the number of output neurons are defined with the target classes. The quantity of neurons in the hidden layer(s) as well as the hidden layer count can be specified by the user. However, assorted topologies lead to different output results. There is no obvious right or wrong topology. The best setup generally depends on the specific application, and thus, has to be assessed by experiments.

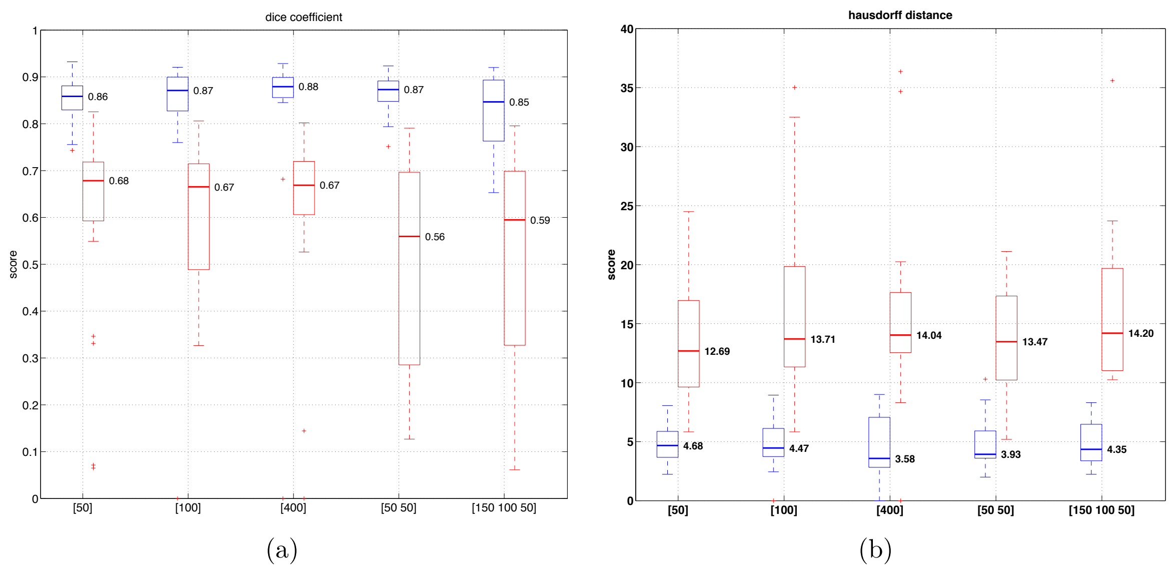


Figure 4.11: Evaluation of hidden layer topology using a neural network. These results were computed using 25 patient cases of the 3 T training set. The algorithm was set to I mode. Figure (a) shows scores for the Dice coefficient and Figure (b) illustrates scores for the Hausdorff distance, respectively. Blue = CG, red = PZ. The horizontal axis denotes the network topology, whereas the digit indicates the amount of neurons in each layer. For example, [50 50] describes a network with two hidden layers, both comprised of 50 neurons. The maximum amount of training epochs was set to 1000.

Figure 4.11 shows an excerpt of these experiments. Generally speaking, a higher amount of network layers does not implicitly lead to a better prediction accuracy. Quite the opposite is true. The best results for this application were achieved using only one hidden layer. Scores remained stable, despite variations in the number of neurons.

The respective training times for different amounts of neurons are outlined in Table 4.1. Using 100 instead of 50 neurons almost doubled the time span required for network training. This effect was increased to a factor of 9, if 400 neurons were used. As the final result variations were small, a topology of one single hidden layer comprised of either 50 or 100 neurons is suggested for further experiments.

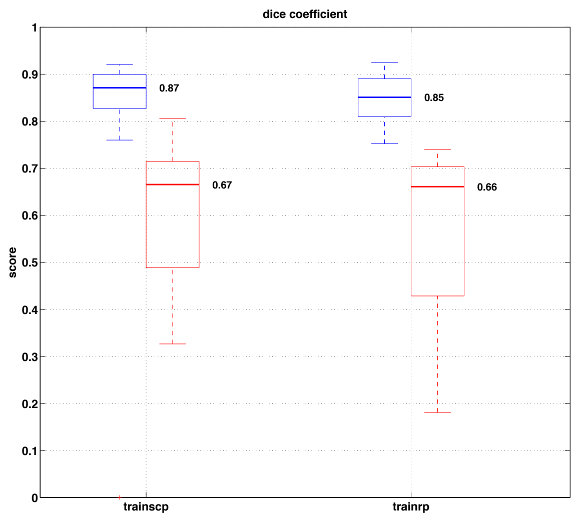
# hidden neurons	training time
50	15 min
100	27 min
400	134 min

Table 4.1: Neural network training time vs. number of hidden neurons. The training time is averaged over 10 cross-validation iterations.

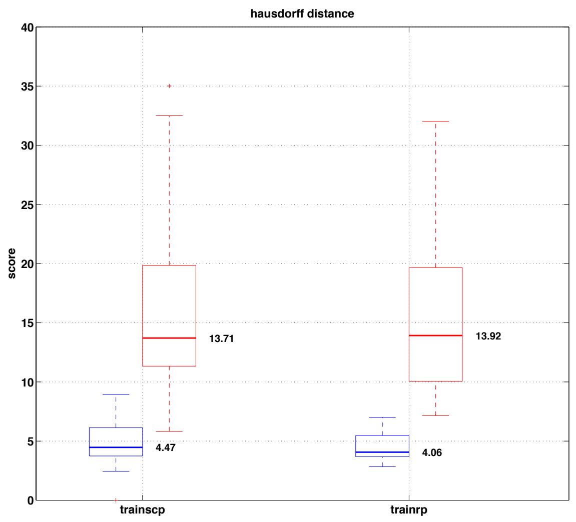
4.3.2 Evaluation of neural network train function variation

Within this experiment the effects of choosing a different `trainFcn` setting for the neural network are evaluated. The respective results for Dice coefficient and Hausdorff distance are plotted in Figure 4.12.

The default train function for other experiments incorporating the neural network `patternnet` was `trainscg`, which updates weight and bias values according to the scaled conjugate gradient method. For comparison purposes, the classifier output using the `trainrp` function was tested using 10 cross-validation cycles. In case of `trainrp` training, updates are performed following the resilient backpropagation algorithm (RPROP). However, it can be seen that the scores show no significant deviation, indicating that a change of the network training method has no effect on the classification outcome.



(a)



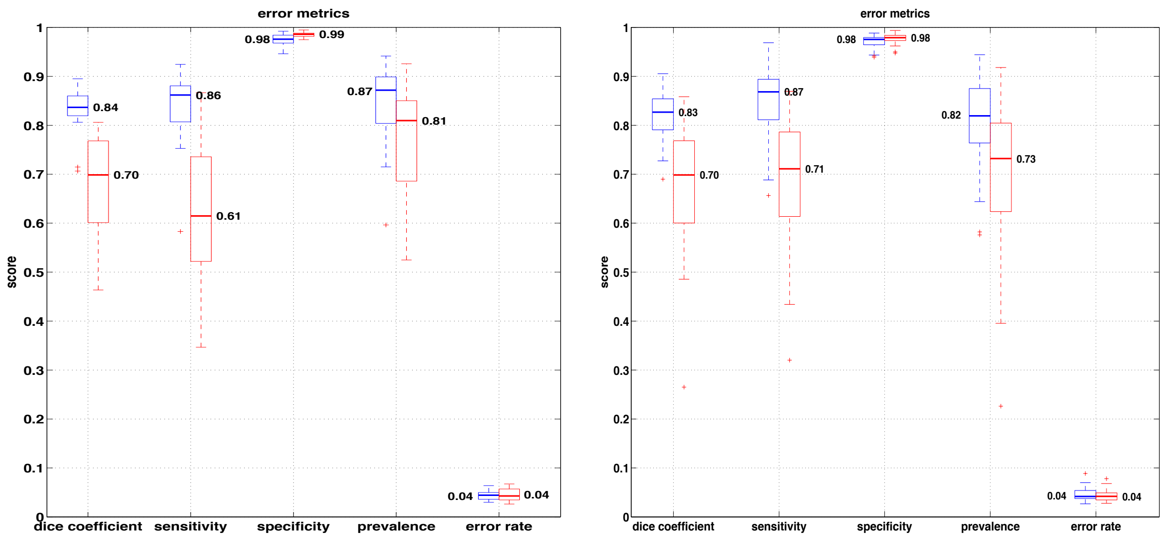
(b)

Figure 4.12: Evaluation of neural network train function variation. The results were obtained using the 3 T training set with the segmentation algorithm in I mode. 100 hidden neurons in a single layer were utilized with the maximum amount of epochs set to 2000. Figure (a) shows scores for the Dice coefficient and Figure (b) illustrates scores for the Hausdorff distance, respectively. Blue = CG, red = PZ. The horizontal axis denotes the respective training function.

4.3.3 Results with final neural network setting

Due to findings of the conducted experiments, which are described in previous sections, a robust setting for the neural network classifier was found. It is considered that a network topology with one single hidden layer incorporating 100 neurons is suitable for this application, discussed in Section 4.3.1. In addition, the comparison in Section 4.3.2 shows that the network can be trained using `trainscg` as a proper train function. All experiments were conducted utilizing 25 patient cases of the 3 T training set due to performance benefits of a smaller training set.

With a robust parameter set for both the MATLAB Neural Network Toolbox™ and the *DeepLearnToolbox*, a final test run was performed on 100 patient cases provided by the MD Anderson Cancer Center. The results are presented in Figure 4.13. Both of the neural network toolboxes showed similar results with a median peripheral zone DSC of 0.70. The score for the central gland DSC indicates only little variance and a median value of 0.84 or 0.83, respectively. These outcomes are very satisfying, considering the high variability of prostate structures throughout 100 patient recordings. These results after 10 cross-validation cycles prove the overall robustness of the segmentation framework.



(a) (b)

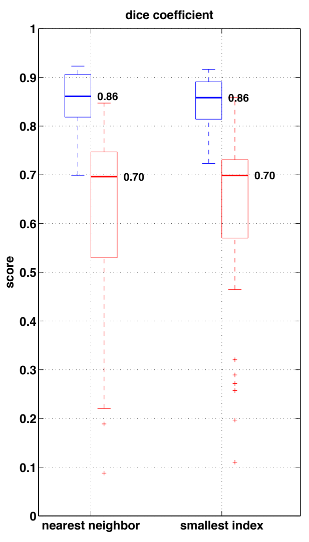
Figure 4.13: Final neural network result for 100 MD Anderson Cancer Center cases. The segmentation algorithm was set to I mode and 100 hidden neurons in a single layer were utilized. Figure (a) shows scores obtained with the Neural Network Toolbox™ and Figure (b) illustrates scores for the *DeepLearnToolbox*. Blue = CG, red = PZ.

An evaluation of the validation performance using the Neural Network Toolbox™ is illustrated in Figure 4.10. The plot was recorded during the final neural network evaluation and compares the mean squared error to the amount of training epochs. It can be seen that a small amount of iterations already sufficiently reduces the overall error. Training was stopped after 6 succeeding and successful validations with consistent mean squared error.

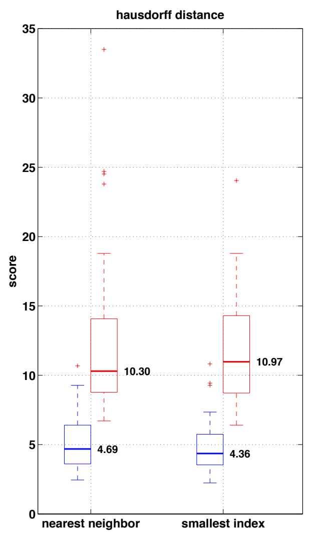
4.4 K-means evaluation

This section summarizes the findings using k-means classification for prostate zonal segmentation. The classifier objects were generated as shown in Listing 3.6. Experiments were conducted with varying `BreakTies` and `NumNeighbors` properties. The `BreakTies` parameter specifies the method that is used to break ties in case multiple classes have the same smallest cost.[3] In this test run this argument was set to either `'nearest'`, i.e. multiple classes have the same number of nearest points among the K nearest neighbors, or `'smallest'`. If `'smallest'` is specified, the smallest index amongst tied groups is used.

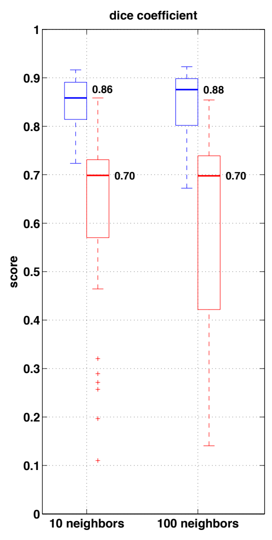
The `NumNeighbors` parameter specifies the amount of nearest neighbors to find within the input data for the classification of each pixel during the prediction step. An experiment was made comparing values of 10 and 100 for this property. The results of the k-means evaluation are illustrated in Figure 4.14.



(a)



(b)



(c)

Figure 4.14: K-means result evaluation obtained using the 3 T training set with the segmentation algorithm in I mode. Figures (a) and (b) show DSC and Hausdorff scores for varying `BreakTies` parameters. Figure (c) illustrates a Dice coefficient comparison for a different setting of the number of neighbors, whereas in this case the smallest index is used to break ties. Blue = CG, red = PZ.

It can be seen that despite various changes in the values of the classifier properties, the result scores remained stable. Error scores are satisfying with Dice coefficients of 0.86 and 0.88 for the central gland and 0.70 for the peripheral zone. However, there are two main disadvantages of utilizing this classification method. The first one is that the `ClassificationKNN` object can get very large in terms of data size. All the data required for the training step are stored in the object, which makes it an unfavorable choice in

[3]MATLAB Release 2013b Documentation, The MathWorks, Inc., Natick, Massachusetts, United States., `http://www.mathworks.com/help/stats/classificationknnclass.html`, date accessed: September 2013.

terms of storing the trained classifier on a hard drive. Secondly, prediction times are very long as the developed segmentation routine uses a high-dimensional feature vector (cf. Section 3.3.2). Throughout all experiments and therefore regardless of the configuration, the prediction time amounted to approximately 16 hours (averaged over 5 test runs, each with 10 cross-validation cycles). These two characteristics are the reason why the k-means classifier is found to be unsuitable for practical applications involving the approach presented in this work.

4.5 Naive Bayes evaluation

Lastly, the usage of the Naive Bayes classifier for prostate segmentation purposes is discussed and evaluated. A Naive Bayes classifier object was created following the implementation in Listing 3.7. The most important property of the `NaiveBayes` object is `'Distribution'`. It describes which distribution is used to model the input data.[4] The results of employing a Gaussian distribution and a multivariate multinomial distribution are compared and illustrated in Figure 4.15. In case of the multivariate multinomial distribution it is assumed that each feature follows a multinomial model within a class. For this experiment, the `Prior` property was set to `'empirical'`; hence, prior class probabilities were estimated from the relative class frequencies in the training data.

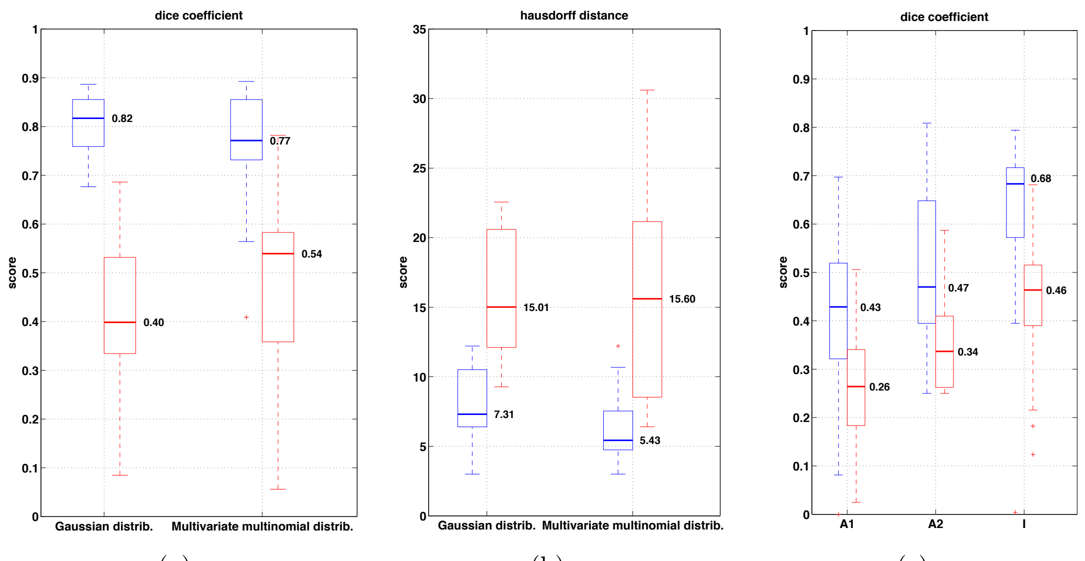


Figure 4.15: Naive Bayes result evaluation. Figures (a) and (b) show DSC and Hausdorff scores for varying distributions settings, obtained using the 3 T training set and the algorithm in I mode. Figure (c) illustrates a Dice coefficient comparison for different algorithm modes. The respective scores were computed utilizing 100 MD Anderson patient cases and the multivariate multinomial distribution presumption. Blue = CG, red = PZ.

It is important to note that this classifier type requires a variance greater than

[4]MATLAB Release 2013b Documentation, The MathWorks, Inc., Natick, Massachusetts, United States., `http://www.mathworks.com/help/stats/naivebayes.fit.html`, date accessed: September 2013.

0 within the feature matrix. Suppose all patient cases for an experiment are taken from the 3 T training set. In this example, the magnetic field strength feature cannot be incorporated in the feature matrix as every sample would have the same value, i.e., the variance for this specific feature would be 0. Obviously, in this case this circumstance does not affect the classification outcome as the magnetic field strength would not contribute to the decision-making capability anyway. However, it has to be kept in mind to switch off this feature in case the Naive Bayes classifier is used.

The error measurements plotted in Figure 4.15 indicate that the result outcomes using a Naive Bayes classifier are not comparable to other classification methods discussed in this chapter. This classification method might not be suitable for this application, because it strongly relies on the validity of the distribution assumption. However, not much effort was expended on a proper configuration. The integration of this classifier type was mainly motivated based on the intention to construct a complete framework, rather than the focus on this specific type of application.

4.6 Volumetric measurements and comparison to other techniques

This section addresses the overall segmentation outcome in terms of prostate volume. To enable a volumetric calculation, surface points of the predicted label stack are firstly transformed to a three-dimensional grid. It is important to note that this transformation takes x, y and z spacing (pixel/mm) of the MR recording into account. The resulting 3D point cloud is then triangulated utilizing a Delaunay triangulation. The final volume is defined by the sum of all tetrahedrons, which are obtained through the connection of surface triangles with a point inside of the object (e.g. mass point of the structure). This approach also offers a volume estimation for the peripheral zone of the prostate, as it provides a robust solution for concave objects. This volume calculation method is described in [53] and its accuracy has been validated on artificial objects.

The volume that was computed using the process described above is an estimate based on the prediction of the segmentation algorithm and is denoted as $\hat{V}$. To determine an accurate measurement of this predicted volume, the same calculation method was applied to the ground truth label stack to obtain V_{gt}. This ground truth volume is considered to be the most accurate measurement, because it is determined using expert contours of the prostate zones on each magnetic resonance image slice.

However, in clinical practice the commonly preferred technique for PV measurement is prolate ellipse volume calculation [54], [55]. This method involves only three measurements of the largest diameters in three planes, making it universally available, fast and still precise enough for routine clinical applications. The prolate ellipse volume is calculated as follows:

$$\tilde{V} = d_1 \times d_2 \times d_3 \times \frac{\pi}{6} \tag{4.1}$$

The values for d_1, d_2 and d_3 thereby correspond to height, length and width measurements of the prostate gland. Width is defined as the maximal transverse diameter at mid-gland level and length is the distance measurement from the proximal external sphincter to the urinary bladder. The height of the prostate gland is given by the anteroposterior diameter, which can be measured either in axial or sagittal plane. All three measurements d_1, d_2 and d_3 are obtained through expert markups.

#case	d_1 [cm]	d_2 [cm]	d_3 [cm]	$\tilde{V}$ [cm^3]	$V_{\mathbf{gt}}$ [cm^3]	$\hat{V}$ [cm^3]	$V_{\mathbf{gt}}/\tilde{V}$	$\hat{V}/V_{\mathbf{gt}}$
21	5.9	4.5	3.1	43.1	32.5	31.3	0.75	0.96
22	5.0	4.7	3.5	43.1	30.8	30.9	0.71	1.00
23	2.7	4.5	4.3	27.4	21.7	19.9	0.79	0.91
26	5.3	3.1	4.8	41.3	28.4	25.6	0.69	0.90
29	5.0	1.9	3.8	18.9	14.8	10.0	0.78	0.68
31	5.9	4.4	3.1	42.2	32.0	26.0	0.76	0.81
33	4.7	5.0	3.3	40.6	30.9	26.5	0.76	0.86
310	2.0	3.3	2.8	9.7	7.1	9.2	0.74	1.30
311	5.0	4.0	2.6	27.2	21.8	23.4	0.80	1.07
313	4.1	5.2	2.8	31.3	21.0	25.9	0.67	1.23
314	5.6	5.5	3.1	50.0	35.1	38.4	0.70	1.10
315	5.0	2.8	4.1	30.1	21.2	18.8	0.70	0.89
317	4.4	3.6	2.4	19.9	13.5	13.3	0.68	0.98
319	5.1	4.2	3.4	38.2	25.6	27.3	0.67	1.07
321	5.0	4.3	3.0	33.8	21.4	18.2	0.63	0.85
324	2.7	5.1	4.4	31.7	26.1	27.7	0.82	1.06
103	4.5	2.5	4.2	24.8	15.0	13.8	0.61	0.92
291	4.6	2.5	4.2	25.3	21.3	18.1	0.84	0.85
293	4.4	5.0	3.1	35.7	31.2	26.8	0.87	0.86
297	4.5	3.0	4.7	33.2	20.0	20.9	0.60	1.05

Table 4.2: Volumetric measurement data for 20 test cases. The case number is listed for identification purposes. The values d_1, d_2 and d_3 represent the prostate diameter measurements of height, length and width. $\tilde{V}$ was calculated following the formula for the prolate ellipse volume, specified in Equation 4.1. V_{gt} denotes the ground truth volume obtained by manual segmentation and $\hat{V}$ indicates the volume based on the prediction result of the automated segmentation algorithm. The proportion of $V_{\text{gt}}/\tilde{V}$ is listed for comparison purposes and shows that the prolate ellipse volume calculation overestimates the prostate volume. $\hat{V}/V_{\text{gt}}$ indicates the ratio of ground truth volume and prediction estimate.

The resulting data for this experiment were obtained using the MD Anderson Cancer Center training set comprised of 100 patient cases. The value for the `FRACTION_TRAINING` property was set to 0.9; thus, 90 cases were used for algorithm training and 10 for algorithm testing. Training and prediction steps were repeated twice, each time utilizing different test data. The resulting 20 anonymized testing cases were provided together with expert dimension measurements for the prostate gland's diameters. These measurements are available for all MD Anderson Cancer Center patient cases as it is a standard procedure for radiologists at this institution to mark the extent of the prostate on the respective image planes.

In terms of classifiers, the *DeepLearnToolbox* was used to implement a neural network incorporating one hidden layer with 50 neurons. The segmentation framework was configured to work with the I mode. Volume calculations were performed on the basis of ground truth markups as well as on the predicted label stack. An exemplary prediction result of case 21 is illustrated in Figure 4.16. All respective result values from the volume calculation as well as diameter measurements performed on the test set are listed in Table 4.2.

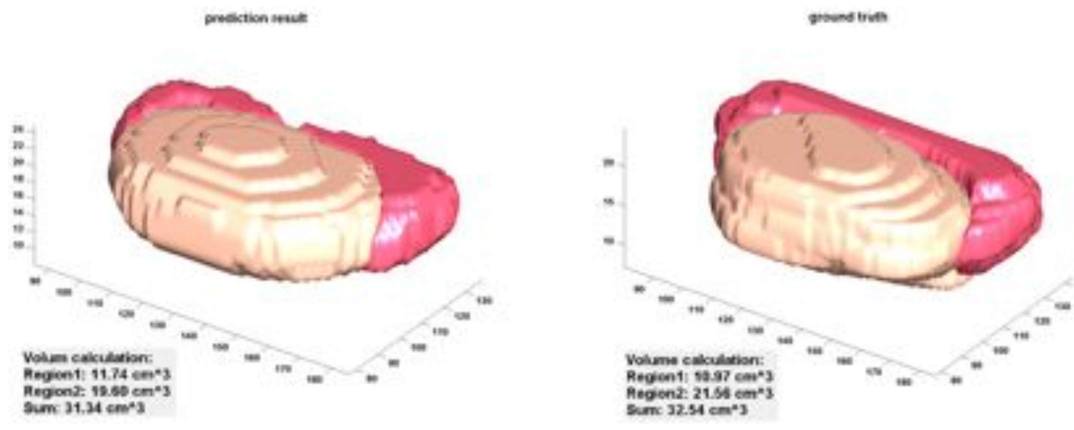


Figure 4.16: Exemplary three-dimensional segmentation output with volumetric measurements for the prediction result (left) and ground truth markups (right). In this case, *Region1* represents the peripheral zone and *Region2* indicates the central gland of the prostate.

The results in Table 4.2 show that in comparison to the ground truth volume V_{gt}, the volume $\tilde{V}$, which was obtained through the three diameter measurements, overestimated the actual PV. This characteristic is also indicated by the ratio $V_{\text{gt}}/\tilde{V}$. On average, the ground truth amounted $72.9 \pm 7.6\%$ of the volume computed with the formula given in Equation 4.1. Despite the overestimation, a computation of Pearson's correlation coefficient ρ indicates a strong linear relationship between the two modes of volume determination. For the volumes $\tilde{V}$ and V_{gt} listed in Table 4.2, the correlation coefficient that measures the strength of the relation between two series of measurements equals $\rho_{\tilde{V},V_{\text{gt}}} = 0.947$.

However, the interesting aspect for the purpose of this experiment is the accuracy of the prediction result $\hat{V}$ in comparison to the ground truth volume V_{gt}. On average, the ratio $\hat{V}/V_{\text{gt}}$ was $96.7 \pm 14.7\%$. Thus, the respective results are considered similar with an average fluctuation range of around $\pm 15\%$. Again, a correlation coefficient of $\rho_{V_{\text{gt}},\hat{V}} = 0.92$ suggests a strong linear relation.

These results prove that the automated segmentation approach introduced in this work provides accurate results for volumetric evaluations. Median DSCs of 0.84 (CG) and 0.70 (PZ) with the final neural network setting tested on the MD Anderson Cancer Center patient cases (cf. Figure 4.13) were sufficient to achieve more precise values for the prostate volume compared to calculations involving three diameter measurements.

Chapter 5

Discussion and conclusion

This paper presents an approach for completely automated zonal segmentation of prostate structures in magnetic resonance images of the human body. Segmentation of a region of interest in medical image recordings enables an intensified study of anatomical structures and assistance in treatment planning. Information on the shape and location of the prostate gland is essential for surgical planning for prostatectomy or minimally invasive therapies and for dose calculations in radiation therapy.

In addition, automated segmentation from MR images has the potential to deliver precise estimates of volume change [15]. The determination of prostate volume in conjunction with other parameters facilitates an assessment of prostate cancer. It can help to estimate the pathological stage of disease as well as to predict treatment response. As an additional information to prostate-specific antigen (PSA) levels, the PSA density can be derived by incorporating PV calculations to support clinical decisions. However, currently used techniques for prostate volume measurements show high variability and generally lack accuracy. This poses a limitation to the usefulness of the prostate-specific antigen density in clinical practice because its value is directly influenced by the PV value.

In terms of clinical relevance, automated segmentation of the prostate could also be used to provide more standardized and objective radiology reports with unified PV calculations. Manual markups are rather subjective to the respective radiologist ([20]), which is also shown in [56] where a certain variance is recorded between contours marked by three experts.

The method proposed in this research uses a low-level feature representation of image pixels as input for a feedforward backpropagation neural network. This approach shows promising results with average Dice similarity coefficients (DSCs) of 0.84 and 0.70 (illustrated in Figure 4.13) for the central gland and peripheral zone, respectively. Similar result accuracy is achieved with a random forest implementation where DSC scores equal 0.87 and 0.64 for the respective prostate zones (shown in Figure 4.7). A representation of results in terms of error metrics is important for the overall validity and comparability to other segmentation techniques. A representative visualization of the resulting segmentation boundaries plotted on MR image slices is illustrated in Figure 5.1 and Figure 5.2.

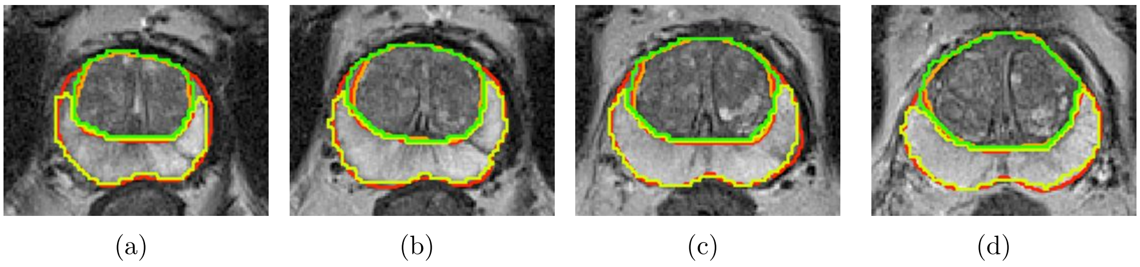


Figure 5.1: Successive MR image slices of the midsection level of the prostate with representative segmentation boundaries. Orange = CG ground truth, red = PZ ground truth, green = CG prediction, yellow = PZ prediction.

Due to the high variability of prostate tissue within different patient cases, the generalization introduced in the classification step implicitly leads to errors affecting the overall segmentation result when calculating mean values. However, results for specificity and Hausdorff distance of boundary surfaces representing worst-case scenarios, prove the overall robustness of this method. Experiments in Section 4.2.2 and Section 4.3.3 show that specificity values of 0.98 are reached and Hausdorff distances equal 10 px and 27 px for the CG and PZ. Most other papers report the 95th percentile of the Hausdorff distance, but in this paper the distance of the one worst pixel prediction is stated. The qualification and accuracy of prostate volume measurements utilizing the framework proposed in this paper are demonstrated in Section 4.6. Overall, the discrepancies between manual and automatic segmentation seem to have minor impact on the prostate volume calculation. On average, the predicted volume matches the ground truth by 97% with a fluctuation range of less than ±15%.

Still, box plot result visualisations indicate that the variance throughout label predictions of different patient cases is not negligibly. While median scores representing average prediction accuracy are satisfying, some outliers might still be present. Consider, for example, the DSC scores for a neural network implementation utilizing the *DeepLearnToolbox* illustrated in Figure 4.13. A median result of 0.70 for the peripheral zone is considered accurate as this zone is subject to strong shape variabilities. Nevertheless, within 10 cross-validation cycles, this value drops below 0.30 for a certain test set, which is marked as an outlier in Figure 4.13. A volume prediction based on this segmentation outcome will most likely significantly differ from the ground truth. Hence, it is important to note that individual predictions might be unreliable, despite the effort spent on improving the overall reliability of the segmentation routine.

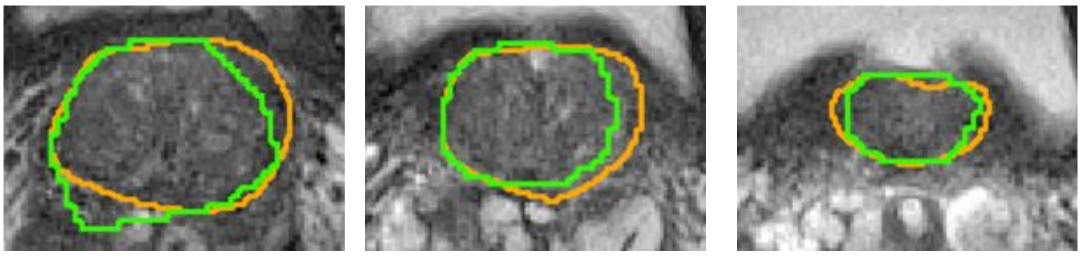
At the same time, the calculated error metrics may provide a wrong impression of the prediction outcomes. Considering prostate anatomy, the gland consists of the apex, midsection, and base (illustrated in Figure 2.1). The midsection thereby encompasses almost 90 percent of the prostate and shows sufficient edge and texture information. This characteristic enables good segmentation results. An exemplary prediction is illustrated in Figure 5.1.

However, the remaining portion of apex and base practically have no features or gradients to be seen in MRI [26]. Indeed, much of the discrepancy between manual and automatic segmentation arises in those areas, where the structure boundaries are unclear, and thus, the absolute nature of the ground truth is questionable [15]. Three representative image slices from the region of the base of the prostate with comparatively good label prediction are shown in Figure 5.2.

During various test runs it has been found that the DSC at midsection level almost always exceeds 0.90. On the other hand, scores for regions at the base or apex mostly amount only 0.40. The reason for this poor segmentation is that boundaries in these regions need to be estimated by experts as well as by the learning algorithm, because image features are insignificant. If the learning algorithm is able to produce these kinds of estimates, the training was successful.

Suppose the prediction accuracy of 11 slices is measured with a DSC of 0.95. However, 3 image slices that cap the gland from the bottom and top poorly visualize the prostate tissue, resulting in scores of 0.40. The error calculation that is being performed does not implement any weighting and therefore does not consider the amount of prostate pixels on a specific image slice. Error metrics are computed individually for every image slice and are then averaged over the amount of slices. For the respective values in this example, this means that the final DSC score would be calculated as $\frac{11\times0.95+3\times0.40}{14} \approx 0.83$. Despite the fact that the large majority of prostate regions are detected with almost maximum accuracy possible, the final score of 0.83 does not seem to reflect the quality of this result.

Vincent *et al.* also experienced the lack of accuracy at the top and bottom of the prostate and described the problem in [18]. They created a map of mean distance error projected onto a three-dimensional model of the prostate gland. Results of Vincent *et al.* are comparable to the research findings in this work, and thus, their error map is illustrated in Figure 5.3.



(a) (b) (c)

Figure 5.2: Successive MR image slices of the top of the prostate with representative segmentation boundaries. Orange color marks the ground truth reference and green boundaries indicate the prediction result. The prostate lacks boundary and texture information in this region, while surrounding tissues have stronger but variable edges.

Much research has been conducted in the field of prostate segmentation, for example presented in [31] and [34]. Recently, studies have been published that focus on individual zonal segmentation of the prostate gland. Litjens *et al.* in [56] present a pattern recognition approach and reach mean DSCs of 0.89 and 0.75 for the central gland and peripheral zone, respectively. Similar DSC scores of 0.87 (CG) and 0.76 (PZ) are achieved by Makni *et al.*, who demonstrate an atlas-based method in [57]. However, both approaches incorporate essential shape information and even manual segmentations of the whole prostate. As a result, the segmentation problem is reduced to a decision between either one of the respective anatomical zones. Thus, the nontrivial problem of localizing the prostate gland within an MRI volume is eliminated.

In 2012 the PROMISE12[1] challenge was hosted with the goal of comparing interactive and (semi)-automatic segmentation algorithms for MRI of the prostate. The automatic whole-prostate segmentation approach of Vincent *et al.* ranked first and is described in [18]. By utilizing active appearance models the team reached a median DSC of 0.89 for the whole prostate. No zonal differentiation was performed.

A study in [56] involving three observers showed that mean DSCs of 0.96 ± 0.06 for the CG and 0.86 ± 0.13 for the PZ were obtained through manual segmentations. An automatic approach should ideally be as accurate as a manual segmentation by an expert; hence, these scores are considered as ultimate objective in terms of DSC values.

In summary, it can be said that the results presented in Chapter 4 are satisfying and comparable to other approaches. In fact, especially the volume calculation demonstrates the overall accuracy of this automated segmentation technique. Even in cases with strong discrepancies between automatic and manual segmentation, the results are comparable with the variation in volume estimates using the prolate ellipsoid formula (see Table 4.2). The proposed technique uses no expert-based knowledge or shape information about the prostate in addition to ground truth training labels (with the exception of orthogonal measurements in I mode), and thus, this method is generalizable to other structures in the human body as well.

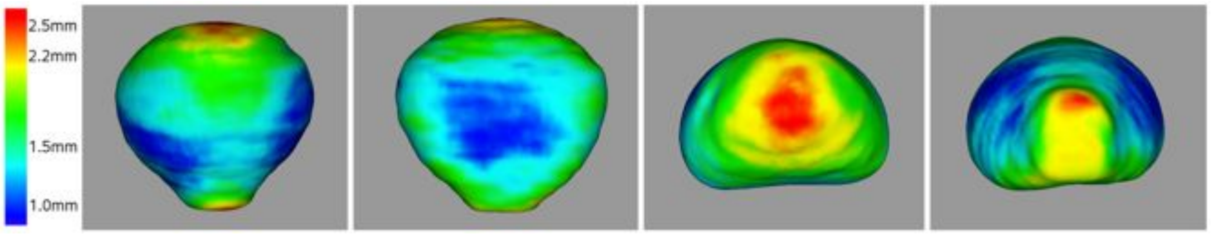


Figure 5.3: Map of projected mean distance error in prostate volume calculations. From left to right: anterior, posterior, base, apex. Blue and green indicate relatively small distance error, whereas yellow and red represent large error. [18]

To increase the robustness of the segmentation method for practical applicability in the future, additional features and shape knowledge need to be implemented. Structured texture detectors might help to improve the accuracy in regions lacking edge information. Moreover, including more structures such as the bladder and rectum in the model would help to significantly improve the segmentation results. In terms of the prostate gland, the main focus should be on improving features for the peripheral zone detection, as the central gland is estimated satisfactorily using the existing representation.

Improvements can also be achieved through an extension of the postprocessing routine. Currently, solely morphological operations are performed on raw pixel predictions. In the

[1]MICCAI Grand Challenge: Prostate MR Image Segmentation 2012, `http://promise12.grand-challenge.org`, date accessed: September 2013.

future, shape restrictions or methods based on level-sets might help to better model the prostate shape.

The goal for the development of this segmentation framework was to provide an automatic solution. However, for a practical application it might be acceptable to allow a certain level of user interaction (e.g. an initial localization) to improve algorithm speed and accuracy. It might also be beneficial to offer a possibility for minor boundary corrections, after the automatic segmentation routine has been executed. The ultimate goal of future work should be to maximize the robustness of this method. Once a reliable segmentation result is obtained, additional functionalities like a detection of tissue abnormalities can be implemented.

Up to this point, the purpose of the existing framework is to accelerate future research processes. Assorted tasks such as viewing and handling DICOM images and corresponding NRRD files have already been implemented. Moreover, a feature representation has been generated whereby various related properties can be specified by the user and supplementary characteristics can be added with minimal effort. Five classifiers are integrated, each one offering multiple configuration possibilities. In addition, evaluations of the current setup are straight-forward as five error metrics can be computed from the segmentation result for unbiased performance comparisons. In addition to feedback in terms of error scores, results can be visualized utilizing various plots. Another feature of the framework is the integrated volume calculation, supporting concave structures. Utilizing the existing image-processing chain allows future researchers to have their main focus on algorithm enhancements without having to worry about essential but not performance-enhancing processes.

List of Acronyms and Abbreviations

CG	central gland
CT	computed tomography
DBN	Deep Belief Network
DICOM	Digital Imaging and Communications in Medicine
DRE	digital rectal examination
DSC	Dice similarity coefficient
GUI	graphical user interface
ML	machine learning
MRI	magnetic resonance imaging
NRRD	Nearly Raw Raster Data
PSA	prostate-specific antigen
PV	prostate volume
PZ	peripheral zone
RAM	Random Access Memory
SVM	support vector machine
T	tesla
TRUS	transrectal ultrasound

List of Figures

List of Tables

Bibliography

[1] American Cancer Society, "Cancer Facts & Figures 2013," American Cancer Society, Atlanta, GA 30303, Tech. Rep., 2013.

[2] S. Verma and A. Rajesh, "A clinically relevant approach to imaging prostate cancer: Review," *American Journal of Roentgenology*, vol. 196, pp. 1–10, March 2011.

[3] F. G. Claus, H. Hricak, and R. R. Hattery, "Pretreatment evaluation of prostate cancer: Role of MR imaging and H MR spectroscopy," *RadioGraphics*, vol. 24, pp. 167–180, October 2004.

[4] American Cancer Society, "Prostate cancer," online, Atlanta, GA 30303, September 2012.

[5] H. Hricak, "MR imaging and MR spectroscopic imaging in the pre-treatment evaluation of prostate cancer," *The British Journal of Radiology*, vol. 78, pp. 103–111, 2005.

[6] C. Tempany and F. Franco, "Prostate MRI update and current roles," *Applied Radiology*, vol. 41, no. 3, pp. 17–22, March 2012.

[7] J. O. Barentsz, J. Richenberg, R. Clements, P. Choyke, S. Verma, G. Villeirs, O. Rouviere, V. Logager, and J. J. Fuetterer, "ESUR prostate MR guidelines 2012," *European Radiology*, vol. 22, pp. 746–757, 2012. [Online]. Available: http://link.springer.com/article/10.1007/s00330-011-2377-y

[8] D. G. Engehausen, K. Engelhard, S. A. Schwab, M. Uder, S. Wach, B. Wullich, and F. S. Krause, "Magnetic resonance image-guided biopsies with a high detection rate of prostate cancer," *Scientific World Journal*, vol. 2012, p. 975971, March 2012.

[9] W. J. Ellis and M. K. Brawer, "Repeat prostate needle biopsy: who needs it?" *Journal of Urology*, vol. 153, no. 5, pp. 1496–1498, May 1995.

[10] C. G. Roehrborn, G. J. Pickens, and J. S. Sanders, "Diagnostic yield of repeated transrectal ultrasound-guided biopsies stratified by specific histopathologic diagnoses and prostate specific antigen levels," *Urology*, vol. 47, no. 3, pp. 347–352, March 1996.

[11] I. M. Thompson, D. P. Ankerst, C. Chi, M. S. Lucia, P. J. Goodman, J. J. Crowley, H. L. Parnes, and C. A. Coltman, "Operating characteristics of prostate-specific antigen in men with an initial PSA level of 3.0 ng/ml or lower," *Journal of the American Medical Association*, vol. 294, no. 1, pp. 66–70, July 2005.

[12] I. M. Thompson, D. K. Pauler, P. J. Goodman, C. M. Tangen, M. S. Lucia, H. L. Parnes, L. M. Minasian, L. G. Ford, S. M. Lippman, E. D. Crawford, J. J. Crowley, and C. A. Coltman, "Prevalence of prostate cancer among men with a prostate-specific antigen level < or =4.0 ng per milliliter," *New England Journal of Medicine*, vol. 350, no. 22, pp. 2239–2246, May 2004.

[13] Y. Gao, R. Sandhu, G. Fichtinger, and A. R. Tannenbaum, "A coupled global registration and segmentation framework with application to magnetic resonance prostate imagery," *IEEE Transactions on Medical Imaging*, vol. 29, no. 10, pp. 1781–1794, 2010.

[14] M. Roethke, A. G. Anastasiadis, M. Lichy, M. Werner, P. Wagner, S. Kruck, C. D. Claussen, A. Stenzl, H. P. Schlemmer, and D. Schilling, "MRI-guided prostate biopsy detects clinically significant cancer: analysis of a cohort of 100 patients after previous negative TRUS biopsy," *World Journal of Urology*, vol. 30, no. 2, pp. 213–218, April 2012.

[15] P. D. Allen, J. Graham, D. Williamson, and C. Hutchinson, Eds., *Segmentation of Prostate MRI Volumes Using 3D Shape Model Constrained Tissue Classification*, vol. 14, Imaging Science and Biomedical Engineering, University of Manchester. Manchester, United Kingdom: International Society for Magnetic Resonance in Medicine, 2006.

[16] S. Martin, V. Daanen, and J. Troccaz, "Atlas-based prostate segmentation using an hybrid registration," *International Journal of Computer Assisted Radiology and Surgery*, vol. 3, pp. 485–492, June 2008.

[17] C. Reynier, J. Troccaz, P. Fourneret, A. Dusserre, C. Gay-Jeune, J.-L. Descotes, M. Bolla, and J.-Y. Giraud, "MRI/TRUS data fusion for prostate brachytherapy. preliminary results," *Medical Physics*, vol. 31, no. 6, pp. 1568–1575, 2004. [Online]. Available: http://link.aip.org/link/?MPH/31/1568/1

[18] G. Vincent, G. Guillard, and M. Bowes, "Fully automatic segmentation of the prostate using active appearance models," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2012)*. Kilburn House, Manchester Science Park, Manchester, M15 6SE, UK: Imorphics Ltd., 2012.

[19] M. J. Moghaddam and H. Soltanian-Zadeh, *Artificial Neural Networks - Methodological Advances and Biomedical Applications*. InTech, April 2011, ch. Medical Image Segmentation Using Artificial Neural Networks, pp. 121–138.

[20] S. Ghose, J. Mitra, A. Oliver, R. Marti, X. Llado, J. Freixenet, J. C. Vilanova, D. Sidibe, and F. Meriaudeau, "A random forest based classification approach to prostate segmentation in MRI," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2012)*. UMR CNRS 6306, Universite de Bourgogne, 12 Rue de la Fonderie, 71200 Le Creusot, France: Laboratoire Le2I, 2012.

[21] M. B. Mengel, W. L. Holleman, and S. A. Fields, *Fundamentals of Clinical Practice*, 2nd ed. Springer US, 2002.

[22] E. R. Davies, *Computer and Machine Vision: Theory, Algorithms, Practicalities*, 4th ed. Elsevier, March 2012.

[23] I. N. Bankman, *Handbook of Medical Image Processing and Analysis*, 2nd ed. Elsevier, December 2008.

[24] N. Sharma and L. M. Aggarwal, "Automated medical image segmentation techniques," *Journal of Medical Physics*, vol. 35, no. 1, pp. 3–14, 2010.

[25] R. Zwiggelaar, Y. Zhu, and S. Williams, "Semi-automatic segmentation of the prostate," in *Pattern Recognition and Image Analysis*, ser. Lecture Notes in Computer Science, F. J. Perales et al., Ed. Springer Berlin Heidelberg, 2003, vol. 2652, pp. 1108–1116. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-44871-6_128

[26] S. Vikal, S. Haker, and G. Fichtinger, "Prostate contouring in MRI guided biopsy," in *Medical Imaging 2009: Image Processing*, J.P.W. Pluim and B.M. Dawant, Ed., vol. 7259, 72594A. SPIE, March 2009. [Online]. Available: +http://dx.doi.org/10.1117/12.812433

[27] Y. Zhu, S. Williams, and R. Zwiggelaar, "A hybrid ASM approach for sparse volumetric data segmentation," *Pattern Recognition and Image Analysis*, vol. 17, no. 2, pp. 252–258, 2007. [Online]. Available: http://dx.doi.org/10.1134/S1054661807020125

[28] Y. Zhu, S. Williams and R. Zwiggelaar, "Segmentations of volumetric prostate MRI data using hybrid 2D + 3D shape modelling," in *Proceedings of Medical Image Understanding and Analysis*, 2004, pp. 61–64.

[29] P. D. Allen, J. Graham, D. C. Williamson, and C. E. Hutchinson, "Differential segmentation of the prostate in MR images using combined 3D shape modelling and voxel classification," in *Biomedical Imaging: Nano to Macro, 2006. 3rd IEEE International Symposium on*, 2006, pp. 410–413.

[30] N. Makni, P. Puech, R. Lopes, R. Viard, O. Colot, and N. Betrouni, "Automatic 3D segmentation of prostate in MRI combining a priori knowledge, markov fields and bayesian framework," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, 2008, pp. 2992–2995.

[31] R. Toth, P. Tiwari, M. Rosen, G. Reed, J. Kurhanewicz, A. Kalyanpur, S. Pungavkar, and A. Madabhushi, "A magnetic resonance spectroscopy driven initialization scheme for active shape model based prostate segmentation," *Medical Image Analysis*, vol. 15, no. 2, pp. 214–225, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S136184151000112X

[32] S. Martin, J. Troccaz, and V. Daanen, "Automated segmentation of the prostate in 3D MR images using a probabilistic atlas and a spatially constrained deformable model," *Medical Physics*, vol. 37, no. 4, pp. 1579–1590, 2010. [Online]. Available: http://link.aip.org/link/?MPH/37/1579/1

[33] J. A. Dowling, J. Fripp, S. Chandra, J. P. W. Pluim, J. Lambert, J. Parker, J. Denham, P. B. Greer, and O. Salvado, “Fast automatic multi-atlas segmentation of the prostate from 3D MR images,” in *Prostate Cancer Imaging. Image Analysis and Image-Guided Interventions*, ser. Lecture Notes in Computer Science, A. Madabhushi et al., Ed. Springer Berlin Heidelberg, 2011, vol. 6963, pp. 10–21. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23944-1_2

[34] S. Klein, U. A. van der Heide, I. M. Lips, M. van Vulpen, M. Staring, and J. P. Pluim, “Automatic segmentation of the prostate in 3D MR images by atlas matching using localized mutual information,” *Medical Physics*, vol. 35, no. 4, pp. 1407–1417, 2008. [Online]. Available: http://link.aip.org/link/?MPH/35/1407/1

[35] E. G. Amaro, M. A. Nuno-Maganda, and M. Morales-Sandoval, “Evaluation of machine learning techniques for face detection and recognition,” in *Electrical Communications and Computers (CONIELECOMP), 2012 22nd International Conference on*, 2012, pp. 213–218.

[36] L. Deng and X. Li, “Machine learning paradigms for speech recognition: An overview,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 21, no. 5, pp. 1060–1089, 2013.

[37] M. Gao, J. Huang, X. Huang, S. Zhang, and D. N. Metaxas, “Simplified labeling process for medical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2012)*, ser. Lecture Notes in Computer Science, N. Ayache at al., Ed. Springer Berlin Heidelberg, 2012, vol. 7511, pp. 387–394. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33418-4_48

[38] E. Geremia, O. Clatz, B. H. Menze, E. Konukoglu, A. Criminisi, and N. Ayache, “Spatial decision forests for MS lesion segmentation in multi-channel magnetic resonance images,” *NeuroImage*, vol. 57, no. 2, pp. 378–390, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1053811911003740

[39] S. Wang, W. Zhu, and Z.-P. Liang, “Shape deformation: SVM regression and application to medical image segmentation,” in *IEEE International Conference on Computer Vision*, 2001, pp. 209–216.

[40] L. C. Carbo, M. A. Haider, and I. S. Yetik, “Supervised prostate cancer segmentation with multispectral MRI incorporating location information,” in *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, 2011, pp. 1496–1499.

[41] J. Jiang, P. Trundle, and J. Ren, “Medical image analysis with artificial neural networks,” *Computerized Medical Imaging and Graphics*, vol. 34, no. 8, pp. 617–631, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895611110000741

[42] A. V. D’Amico, H. Chang, E. Holupka, A. Renshaw, A. Desjarden, M. Chen, K. R. Loughlin, and J. P. Richie, “Calculated prostate cancer volume: The optimal predictor of actual cancer volume and pathologic stage,” *Urology*, vol. 49, no. 3, pp. 385–391, May 1997.

[43] A. Heidenreich, P. J. Bastian, J. Bellmunt, M. Bolla, S. Joniau, M. D. Mason, V. Matveev, N. Mottet, T. H. van der Kwast, T. Wiegel, and F. Zattoni, "{EAU} guidelines on prostate cancer," *European Urology*, vol. 53, no. 1, pp. 68 – 80, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0302283807011451

[44] H. Hricak, G. C. Dooms, J. E. McNeal, A. S. Mark, M. Marotti, A. Avallone, M. Pelzer, E. C. Proctor, and E. A. Tanagho, "MR imaging of the prostate gland: normal anatomy," *American Journal of Roentgenology*, vol. 148, no. 1, pp. 51–58, January 1987.

[45] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[46] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[47] A. Karargyris and N. Bourbakis, "Detection of small bowel polyps and ulcers in wireless capsule endoscopy videos," *Biomedical Engineering, IEEE Transactions on*, vol. 58, no. 10, pp. 2777–2786, 2011.

[48] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, October 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1010933404324

[49] National Electrical Manufacturers Association, "The DICOM standard," online, 1300 N. 17th Street, Rosslyn, Virginia 22209, USA, 2011. [Online]. Available: http://medical.nema.org/standard.html

[50] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2012.

[51] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," Master's thesis, Technical University of Denmark, DTU Informatics, E-mail: reception@imm.dtu.dk, Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark, 2012, supervised by Associate Professor Ole Winther, owi@imm.dtu.dk, DTU Informatics, and Morten Mørup, mm@imm.dtu.dk, DTU Informatics. [Online]. Available: http://www.imm.dtu.dk/English.aspx

[52] M. Beauchemin, K. P. B. Thomson, and G. Edwards, "On the hausdorff distance used for the evaluation of segmentation results," *Canadian Journal of Remote Sensing*, vol. 24, no. 1, pp. 3–8, January 1998. [Online]. Available: http://geogratis.gc.ca/api/en/nrcan-rncan/ess-sst/e30cbdcf-b9b7-512b-a799-fcd3c0c163b2.html

[53] H.-P. Wieser, "Robust volume calculation of closed surface models," July 2011, bachelor's Thesis.

[54] L. M. Eri, H. Thomassen, B. Brennhovd, and L. L. Haheim, "Accuracy and repeatability of prostate volume measurements by transrectal ultrasound," *Prostate Cancer and Prostatic Diseases*, vol. 5, no. 4, pp. 273–278, December 2002.

[55] S. B. Park, J. K. Kim, H. N. Noh, E. K. Ji, and K. S. Cho, "Prostate volume measurement by TRUS using heights obtained by transaxial and midsagittal scanning: Comparison with specimen volume following radical prostatectomy," *Korean Journal of Radiology*, vol. 1, no. 2, pp. 110–113, June 2000.

[56] G. Litjens, O. Debats, W. Ven, N. Karssemeijer, and H. Huisman, "A pattern recognition approach to zonal segmentation of the prostate on MRI," in *Medical Image Computing and Computer-Assisted Intervention â€" MICCAI 2012*, ser. Lecture Notes in Computer Science, N. Ayache, H. Delingette, P. Golland, and K. Mori, Eds. Springer Berlin Heidelberg, 2012, vol. 7511, pp. 413–420. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33418-4_51

[57] N. Makni, A. Iancu, O. Colot, P. Puech, S. Mordon, and N. Betrouni, "Zonal segmentation of prostate using multispectral magnetic resonance images," *Medical Physics*, vol. 38, no. 11, pp. 6093–6105, 2011. [Online]. Available: http://link.aip.org/link/?MPH/38/6093/1

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre hiermit:

- dass ich die vorliegende Diplom/Masterarbeit selbstständig und ohne fremde Hilfe verfasst und noch nicht anderweitig zu Prüfungszwecken vorgelegt habe.
- dass ich keine anderen als die angegebenen Hilfsmittel benutzt, die den verwendeten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- dass die elektronisch abgegebene Arbeit mit der eingereichten Hardcopy übereinstimmt.
- dass ich einwillige, dass ein Belegexemplar der von mir erstellten Diplom/Masterarbeit in den Bestand der Fachhochschulbibliothek aufgenommen und benutzbar gemacht wird (= Veröffentlichung gem. § 8 UrhG).

(Ort, Datum) (Unterschrift Studierende/r)