

Hamburg University of Applied Sciences

Department of Life Sciences



Hochschule für Angewandte  
Wissenschaften Hamburg  
*Hamburg University of Applied Sciences*



Master's Thesis

# 3D Medical Image Segmentation with Vantage Point Forests and Binary Context Features

Author:

Evelin Vladislavova Hristova

April 27, 2017

Advisors:

Prof. Dr.-Ing. Thomas Schiemann

Department of Life Sciences

Dr. rer. nat. Hannes Nickisch

Philips GmbH Innovative Technologies

**Hristova, Evelin Vladislavova:**

*3D Medical Image Segmentation with Vantage Point Forests and Binary Context Features*

Master's Thesis, Hamburg University of Applied Sciences, 2017.

# Abstract

Automated segmentation of medical image data is an important, clinically relevant task as manual delineation of organs is time-consuming and subject to inter- and intraobserver fluctuations. This thesis builds upon a framework for segmentation of multiple organs in three-dimensional images. The approach employs a supervised recognition, where a training set with dense organs annotations is used to classify voxels in unseen images based on similarity of binary features extracted from the data. A combination of two different types of feature vectors is used to capture relevant structural and contextual information. The binary vectors are constructed by multiple pairwise intensity comparisons. Hence, the method is invariant to monotonic gray-level changes and can be applied to different imaging modalities or anatomies. The fast approximate nearest neighbor search, using Vantage Point Forests, does not require any explicit prior shape model knowledge and allows computationally efficient binary data classification. Training the algorithm takes several minutes, while segmenting a test image is in the order of a few seconds. The method is successfully applied to 68 CT abdominal and 42 MR pelvic images. With respect to ground truth, an average Dice overlap score of 0.74 for the CT segmentation of liver, spleen and kidneys is achieved. The mean score for the MR delineation of bladder, bones, prostate and rectum is 0.65. The results demonstrate surprisingly accurate segmentation, robustness and data-efficiency.





# Contents

<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>ix</b>
<b>List of Acronyms</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Structure of the Thesis . . . . .	2
<b>2 Background and Methods</b>	<b>3</b>
2.1 Machine Learning . . . . .	3
2.2 Binary Context Features . . . . .	4
2.2.1 Binary Robust Independent Elementary Features . . . . .	5
2.2.2 Local Binary Pattern . . . . .	5
2.3 Classification . . . . .	7
2.3.1 Vantage Point Forests . . . . .	9
2.4 Post Processing . . . . .	13
2.4.1 Random Walker Regularization . . . . .	13
2.4.2 Random Walker Regularization with Label Priors . . . . .	18
2.5 Evaluation Metrics . . . . .	18
2.5.1 Spatial Overlap-based Metrics . . . . .	19
<b>3 Experiments and Results</b>	<b>23</b>
3.1 Pipeline Overview . . . . .	23
3.2 Nearest Neighbor Visualization . . . . .	26
3.3 Parameters Settings . . . . .	30
3.4 Experiments . . . . .	33
3.4.1 CT Abdominal Data . . . . .	33
3.4.2 MR Male Pelvis Data . . . . .	39
3.5 Time and Storage Evaluation . . . . .	44
<b>4 Conclusion and Outlook</b>	<b>46</b>
<b>A Appendix</b>	<b>48</b>

**Bibliography**

**51**

# List of Figures

2.1	Supervised Learning Flow . . . . .	4
2.2	Sampling Patterns . . . . .	7
2.3	Finding Nearest Neighbors . . . . .	8
2.4	Vantage Point Tree Construction . . . . .	11
2.5	Vantage Point Tree Search . . . . .	12
2.6	Illustration of the Random Walker for Segmentation . . . . .	15
2.7	Random Walker Segmentation . . . . .	17
2.8	Classification Outcomes . . . . .	19
2.9	Confusion Matrix . . . . .	20
2.10	Confusion Matrix, Multiple Classes . . . . .	20
3.1	Training Phase . . . . .	24
3.2	Testing Phase . . . . .	25
3.3	NN Visualization by Patches (Test Data Included) . . . . .	27
3.4	NN Visualization by Patches (Test Data not Included) . . . . .	28
3.5	Distance to Nearest Neighbors . . . . .	29
3.6	Parameter Settings, Effects of Modifications . . . . .	31
3.7	CT Abdominal Segmentation (Confusion Matrix) . . . . .	34
3.8	CT Abdominal Segmentation (Dice Score) . . . . .	35
3.9	CT Abdominal Segmentation (CT Precision) . . . . .	35
3.10	CT Abdominal Segmentation (CT Sensitivity) . . . . .	36
3.11	CT Abdominal Segmentation (Inlier) . . . . .	37
3.12	CT Abdominal Segmentation (Outlier) . . . . .	38

---

3.13 MR Pelvis Segmentation (Confusion Matrix) . . . . .	39
3.14 MR Pelvis Segmentation (Dice Score) . . . . .	40
3.15 MR Pelvis Segmentation (Precision) . . . . .	41
3.16 MR Pelvis Segmentation (Sensitivity) . . . . .	41
3.17 MR Pelvis Segmentation (Inlier) . . . . .	42
3.18 MR Pelvis Segmentation (Outlier) . . . . .	43
A.1 Segmentation Code Structure . . . . .	49

# List of Tables

2.1	Feature Descriptors . . . . .	6
2.2	Search Tree Algorithms . . . . .	10
2.3	Laplacian Matrix L . . . . .	17
2.4	Overlap-based Evaluation Metrics . . . . .	22
3.1	Parameter Settings . . . . .	32
3.2	Duration and Storage Requirements . . . . .	45

# List of Acronyms

BF	Brute Force
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Invariant Scalable Keypoints
CT	Computed Tomography
FN	False Negative
FP	False Positive
FREAK	Fast Retina Keypoints
GT	Ground Truth
kNN	k Nearest Neighbor
LBP	Local Binary Pattern
MR	Magnetic Resonance
NN	Nearest Neighbor
OAR	Organ at Risk
RW	Random Walker
TN	True Negative
TP	True Positive
VP	Vantage Point
VPF	Vantage Point Forest
VPT	Vantage Point Tree







# 1. Introduction

## 1.1 Motivation

Image segmentation and object recognition are crucial procedures in areas such as autonomous driving, visual recognition, as well as various technical research and clinical settings. In general, segmentation refers to dividing an image into a set of semantically meaningful, homogeneous, and non-overlapping regions of similar attributes such as intensity, depth, color or texture. It can result in either image of labels identifying regions or a set of contours which describe the region boundaries [Despotović et al., 2015]. In 3D medical imaging, segmentation refers to delineating anatomical structures or abnormalities, which is an important preprocessing step for computer-assisted systems, image-guided interventions and radiation therapy planning. In traditional clinical practice, the delineation of organs is semi-automatic. It requires an expert clinician to perform the segmentation in an organ-by-organ and slice-by-slice manner in line with given constructs<sup>1</sup>. Manual delineation, however, is tedious, time-consuming and prone to inter- and intraobserver variability [Hu et al., 2016].

Several approaches have been proposed to complement and facilitate the work of the clinicians. Watershed [Grau et al., 2004], level sets [Chan and Vese, 2001] or thresholding methods, for instance, have been successfully applied to certain classes of images and rough segmentation results have been obtained. These approximate delineations can be then further refined by the intervention of human experts [Zhao and Xie, 2013]. Such methods provide valuable assistance as most of the manual work is cut down. However, despite the advantages these techniques have, they are only specific to given applications and require prior tuning of parameters.

Automatic medical image segmentation, on the contrary, aims to be more generic, accurate, fast and simple. Generality, is of high importance as algorithms shall be

---

<sup>1</sup>In radiotherapy scenarios, for instance, segmentation of organs has to be in line with the constructs introduced by the Radiation Therapy Oncology Group (RTOG).

independent from the modality used to acquire the images and applicable to any organ. High accuracy is essential in medical image segmentation. It is required both for the recognition of anatomical parts and the delineation of their boundaries. This, however, is often challenging due to image acquisition artifacts, anatomic variability of organs in shape, size and appearance among patients, fuzzy boundaries between tissues, etc. Speed is another major factor that has to be taken into account. Since large datasets often need to be segmented and certain medical procedures are time sensitive, fast segmentation algorithm is of interest. Moreover, the simpler such a method is, the easier it is to maintain and adapt to changes. Previous studies have shown that there is a trade-off between these four objectives when it comes to image segmentation. Highly accurate methods, for example, usually only apply to a single organ and take long to process and to develop.

The approach described in this thesis aims to find a balance between the four mentioned goals. The method is based on the work of [Heinrich and Blendowski \[2016\]](#), where they suggest a supervised machine learning segmentation, that employs a training set with label annotations to classify voxels in an unseen image. The classification is based on the similarity of features extracted within the voxels' spatial proximity. The implementation builds on the source code provided by them and is extended so that it can be used in a Python environment. The code organization and the extensions can be seen in [Chapter A](#). The generality of the method is tested on two different modalities with four organs each. Insights on the accuracy and speed of the algorithm are presented.

## 1.2 Structure of the Thesis

The thesis continues with a background chapter where the different concepts and methods required for segmentation are presented. The basics of supervised machine learning are discussed in [Section 2.1](#), two types of context features are presented in [Section 2.2](#) and a classification algorithm is discussed in [Section 2.3.1](#). Several segmentation evaluation metrics are presented in [Section 2.5](#). In [Chapter 3](#) the different components are merged and the segmentation framework is illustrated. The delineation algorithm is then applied to two different test scenarios ([Section 3.4.1](#) and [Section 3.4.2](#)) and the results are evaluated. Possible refinements of the method are derived from these tests, before a final conclusion is drawn in [Chapter 4](#) including the prospects of further development.

## 2. Background and Methods

### 2.1 Machine Learning

In the age of constantly improving technology, gathering and maintaining large collections of data is far less challenging as it used to be. Its analysis and handling, however, demands more care since it can hardly be manual and often high precision is desired. Therefore, machine learning techniques are used to build analytical models, helping computers "learn" from this data and operate with it. Unlike traditional algorithms that are explicitly programmed to perform certain tasks, machine learning methods are experience-based and can potentially evolve over time. They are usually built on knowledge from areas as statistics, control theory, machine vision, etc. and can be applied to various tasks such as object recognition, natural language processing or organ annotation in medical images. The techniques used for classification can be grouped into two main categories: supervised and unsupervised learning. Supervised learning requires data accompanied by labels, which is used as a reference for automatic classification of new unseen "test" data. Unsupervised learning, on the contrary, is expected to discover patterns and relationships in the data without any prior knowledge about labels.

The approach used in this thesis follows the supervised learning principle. It relies on features extracted from images with dense voxel annotations (labels) as a **Ground Truth (GT)** training data. Representing the raw data by feature vectors is a dimensionality reduction technique making the data more memory efficient, as well as, faster to process. The vectors are then fed into a classification algorithm that is used for building a model (classifier) to find similar features and predict the labels of the test data (Figure 2.1). The exact extraction and classification methods are discussed in detail in Section 2.2 and Section 2.3.

An additional step in the supervised learning is the validation of the classification algorithm. The k-fold cross-validation method, used here, repeatedly splits the training dataset into  $k$  subsets (folds) of roughly equal size where 1 fold is retained as test set,

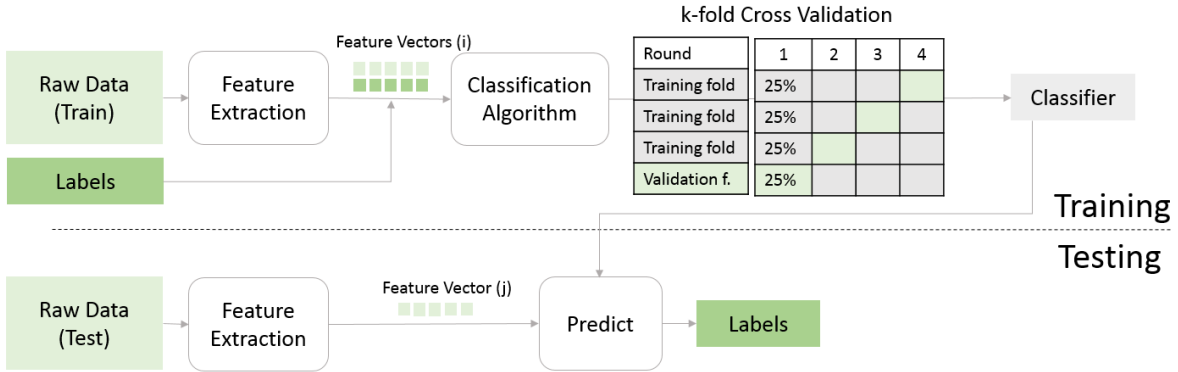


Figure 2.1: Supervised Learning Flow

During the training phase of a supervised learning, features and labels are extracted from images and fed into a classification algorithm. The classification algorithm is validated by a cross validation procedure that gives an insight how well would the model perform on new unseen data. The model itself is used to predict the labels of extracted features from the test data.

and the others  $k - 1$  are used to train the classifier. The operation is then executed  $k$  rounds and each time the performance of the classifier is evaluated. Averaging the validation results gives an estimate on how the model will generalize to an independent dataset and how successful the segmentation will be when deployed.

## 2.2 Binary Context Features

When observing medical images, often the intensity of the anatomical structures, as well as the contrast among them, differ, even when the same modality is used. Their appearance and relative position, however, are similar due to both the characteristics of the imaging systems and the human anatomy itself. Therefore, contextual information is exploited when pixel-wise features are extracted instead of using intensities alone [Deserno et al., 2014, p. 386]. In general, contextual feature descriptor algorithms encode the neighborhood information on a pixel basis by using sampling patterns. The binary descriptor methods, construct vectors by simply comparing the intensities of pixel pairs from the pattern. The main benefit of the binary descriptors (compared to vector-based descriptors) is the support of fast matching by calculating the Hamming distance between two feature vectors. For binary strings of the same length ( $h_i$  and  $h_j \in [0, 1]^n$ ) the distance is the number of bits equal to 1 when using the XOR operator between them:<sup>1</sup>

$$d_H(h_i, h_j) = \|h_i - h_j\|_1 = \#\{h_i \oplus h_j\} \quad (2.1)$$

Nowadays, the Hamming distance can be computed efficiently with the use of the POPCNT instruction in x86\_64 architectures [Persson and Loutfi, 2016].

<sup>1</sup>For instance, if  $h_i = 100110$  and  $h_j = 010011$ ,  $d_H(h_i, h_j) = \#\{100110 \oplus 010011\} = \#\{110101\} = 4$

Table 2.1 on the following page shows four common contextual descriptor algorithms and the different sampling patterns they use. There has been an increased use of these methods in literature lately [Kashif et al., 2016; Persson and Loutfi, 2016] and they are proven to perform well even in challenging cases such as viewpoint changes (Binary Robust Invariant Scalable Keypoints (BRISK)), blurring (Fast Retina Keypoints (FREAK)) or monotonic gray-level changes (Binary Robust Independent Elementary Features (BRIEF), Local Binary Pattern (LBP)). Most of the tests, however, are only applied on 2D images. In the context of this work, 3D descriptors are of interest and thus, the methods need to be adapted. Extending FREAK and BRISK is not a straight forward task, while doing this for BRIEF and LBP is almost trivial. Therefore a combination of the latter two is used in this work and described in the next sections.

### 2.2.1 Binary Robust Independent Elementary Features

The first method, BRIEF, uses information from single pixels to construct a feature vector and is thus, robust to monotonic intensity changes. Such a technique, however, is susceptible to noise artifacts as pixel values are compared. This obstacle can be avoided by prior smoothing of the images with a (Gaussian) filter and building the vectors based on patch intensity differences. Calonder et al. [2010] defines a BRIEF comparison test  $\tau$  on a patch  $\mathbf{p}$  of size  $S \times S$  as:

$$\tau(\mathbf{p}; \mathbf{x}, \mathbf{y}) := \begin{cases} 1 & \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

where  $\mathbf{p}(\mathbf{x})$  is the (smoothed) pixel intensity of  $\mathbf{p}$  at  $\mathbf{x} = (u, v)^T$ . In 3D, a third coordinate is added and  $\mathbf{x} = (u, v, w)^T$ . The same principle is then applied to a set of  $n_d$   $(\mathbf{x}, \mathbf{y})$  randomly selected location pairs, within a given displacement radius, to get a  $n_d$ -dimensional bitstring:

$$f_{n_d}(\mathbf{p}) := \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(p; x_i, y_i)$$

Figure 2.2 (a) shows a sample BRIEF sampling pattern applied to a patch (in blue). The green patches indicate the intensity comparisons points.

### 2.2.2 Local Binary Pattern

The second descriptor used here is constructed in a similar way to BRIEF. Instead of having all the sampling coordinates at random locations around the patch of interest, however, LBP [Ahonen et al., 2006] fixes one of the patches to the central as seen in Figure 2.2 (b).

Both BRIEF and LBP provide important contextual information. On the one hand, LBP focuses on relations around the central region while, on the other, BRIEF captures interactions among neighboring structures. Therefore, a combination of both is

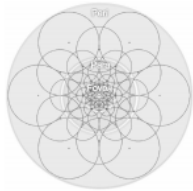
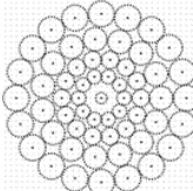
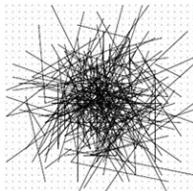
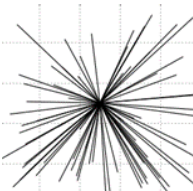
Algorithm	Sampling Pattern	Advantages	Disadvantages
FREAK (Fast Retina Keypoints)		Robust to blurring	Sensitive to illumination changes or compression Sensitive to viewpoint changes
BRISK (Binary Robust Invariant Scalable Keypoints)		Robust to viewpoint changes	Sensitive to photometric changes (blur, illumination changes)
BRIEF (Binary Robust Independent Elementary Features)		Easy to extend to 3D	Only tolerates small amounts of rotation (in most medical scans not of high importance)
LBP (Local Binary Pattern)		Easy to extend to 3D	

Table 2.1: Feature Descriptors

Source: [Kashif et al. \[2016\]](#); [Persson and Loutfi \[2016\]](#)

Different feature descriptors algorithms extract contextual information on a pixel basis with the help of sampling patterns. The methods construct binary vectors by comparing the intensities at pixel pairs from the patterns. Using the sign of the comparisons, instead of the actual differences, makes the algorithms robust to monotonic gray-level changes and also efficient for further processing. The sampling can also be performed on pre-smoothed patches in order to avoid any noise artifacts that may be sampled by the patterns.

believed to capture more information and is therefore implemented here (Figure 2.2 (c)). Extending the algorithms to match the 3D images is done by adding a third coordinate when selecting the random displacement of the comparison points. The choice of displacement distribution around the central point, the length of the vectors, as well as, the amount of BRIEF and LBP features are discussed in Chapter 3.

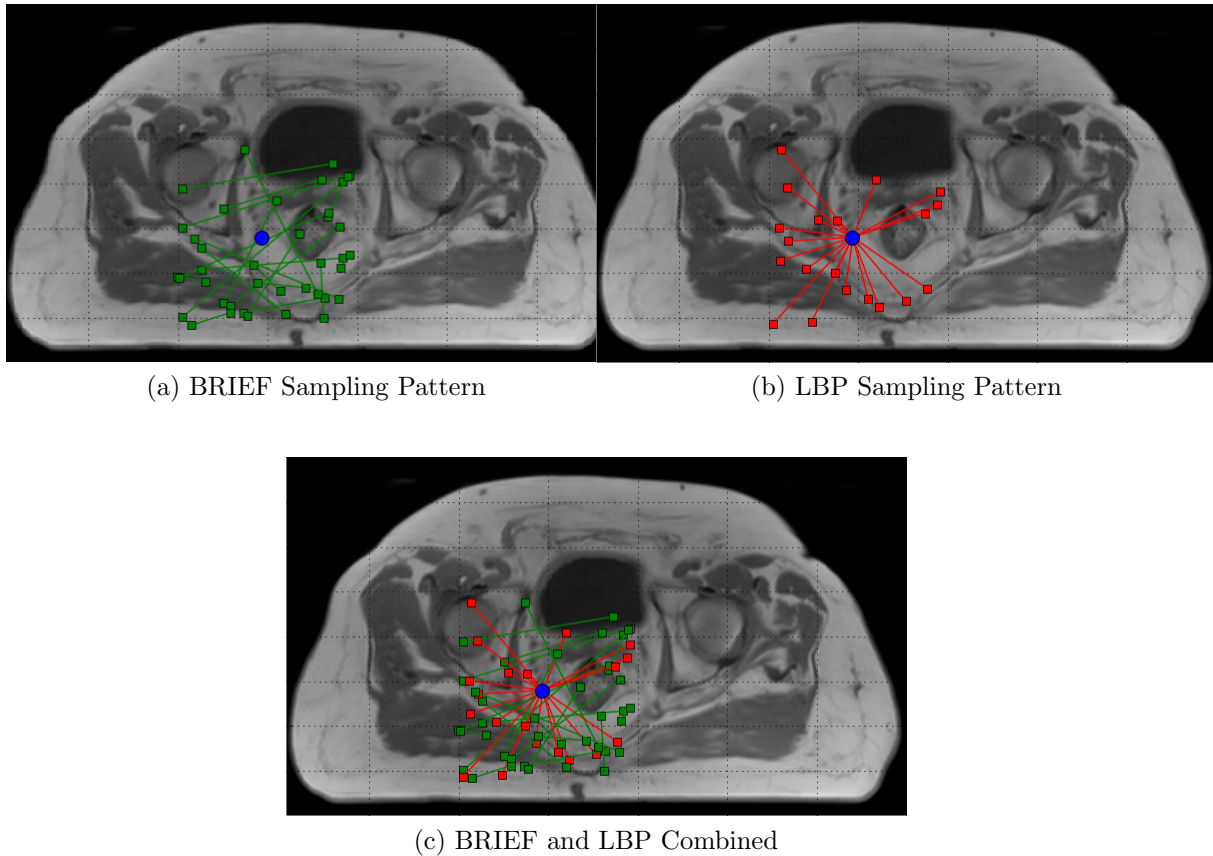


Figure 2.2: Sampling Patterns

Contextual information for a patch (in blue) is captured by comparing the mean intensities of offset patches around it (BRIEF) or by comparing the patch of interest to others (LBP). Such comparisons can be also applied to single pixels. However, these tests would be sensitive to noise artifacts.

## 2.3 Classification

In the context of machine learning there are various techniques used for classifying information, e.g. Neural Networks [Hall et al., 1992; Hu et al., 2016] or Support Vector Machines [Zhang et al., 2004]. As the type of data influences the performance of these algorithms, however, there is no generic solution that can be applied to all classification problems. Wolpert and Macready [2005] define this as the *No Free Lunch* theorem and state that there is no one model that works best for every scenario and that "any two



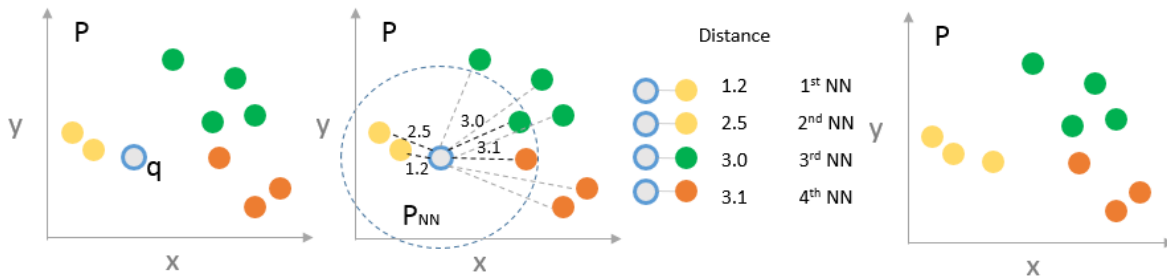


Figure 2.3: Finding Nearest Neighbors

The NN algorithm aims to find points  $P_{NN}$  in a dataset  $P$  'closest' to a query point  $q$ . The goal here is to find the color of a gray ball, based on the colors of the others. The distances to all balls are calculated and ranked. The final decision is based on the four nearest neighbors. By majority voting the color is predicted as yellow.

optimization algorithms are equivalent when their performance is averaged across all possible problems". In reality, there are trade-offs between speed, memory consumption and accuracy. In the framework of 3D image segmentation where large number of training features are processed, it is important to employ a non-parametric learning algorithm that does not make assumptions on the underlying data distribution. The classifier approach here is the **Nearest Neighbor (NN)** principle where new feature vectors are labeled based on *similarity* and *probability*. The method looks for a subset of points  $P_{NN}$  in the training dataset  $P$  "closest" to a query point  $q$  by a given distance  $d(p, q)^2$ . Then it makes a decision based on labels of the neighbors with highest probability.

Figure 2.3 illustrates the NN algorithm. The aim in this example is to predict the color of the gray ball ( $q$ ) based on the colors of the other balls in the dataset ( $P$ ). The NN search starts by calculating the distance to all available balls and ranking them by increasing distance to  $q$ : 1<sup>st</sup> NN, 2<sup>nd</sup> NN, and so on. In this case the decision is based on the information about the four nearest neighbors. By using majority voting, the gray point is predicted to be yellow. This procedure is known as exhaustive search or **Brute Force (BF)** nearest neighbor search. It explores all possible solutions and retains the best one(s). Although this search approach is straightforward and simple to construct, it is inefficient as it has to be conducted from scratch for each new query point.

Alternative solutions for finding nearest neighbors are *search trees* that organize data into hierarchical structures and thus, enable efficient search. In these trees, data points are stored in leaf nodes and internal nodes are used to prune the list of leaves returned by a query. Typically, tree construction begins by assigning all points to a root node. Then they are recursively partitioned into one or several children nodes until some termination criterion is met. The number of children, how they are chosen and how

<sup>2</sup>Common distances are Euclidean distance, Hamming distance, Manhattan distance, Minkowski distance.



the points are partitioned vary depending on the particular search tree method [Kumar et al., 2008].

Searching for nearest neighbors in trees can be performed in two different manners. One way, known as *range nearest neighbor search*, is to look for the set of points  $P_{NN} \subset P$  such that  $p_i \in P_{NN} \Leftrightarrow d(p_i, q) \leq \epsilon$ . This method guarantees that all results are within a preselected range  $\epsilon$ . Alternatively, one can search for a set of  $k$  points  $P_{NN} \subset P$  such that  $\forall p_i \in P_{NN}$  and  $\forall p_j \notin P_{NN}$ ,  $d(p_i, q) \leq d(p_j, q)$ . This approach, called **k Nearest Neighbors (kNNs)** search, does no promises on the distances to the most similar points, but instead fixes the number of retrieved neighbors. The method used here is the **kNN** as no assumptions on *closeness* of the training data need to be done.

Table 2.2 shows different search tree algorithms and a preview of how they would partition a 2D points set. The line thickness denotes partition order (thicker lines were performed first). As a direct consequence of their splitting rules, the algorithms result in different construction and search time. The vantage point tree at the bottom has both excellent search performance and fast construction time and is therefore selected as a classifier for the image segmentation here.

### 2.3.1 Vantage Point Forests

**Vantage Point Trees (VPTs)** as proposed by Yianilos [1993] are constructed by recursively splitting the data points using absolute distances from randomly chosen centers. These centers, called **Vantage Points (VPs)**, partition the data points at each iteration in such a way that approximately half of them are within a threshold  $\tau$ , and the other half are not. This results in a structure where tree neighbors also tend to be neighbors in space and thus, searching for such is efficient. Figure 2.4 visualizes a sample vantage point tree construction for a 2D set of points ( $P Q R S T U V W$ ). The procedure is as follows. First, all points are assigned to the root node. Then a random Vantage Point ( $R$ ) is selected (Figure 2.4 (a)). The threshold  $\tau$  is computed so that points are divided into two equal parts ( $\tau = 5$ ). All points that  $d(Point, VP) < \tau$  move to the left child node ( $P S Q$ ) and the rest move to the right ( $U T V W$ ) (Figure 2.4 (b)). First the right child node is processed. Again a random VP is selected ( $U$ ,  $\tau = 4$ ). Then all points that  $d(Point, VP) < \tau$  move to the left child node ( $T$ ) and the rest move to the right ( $V W$ ) (Figure 2.4 (b)). Then the left child node is processed. A random VP is selected ( $P$ ,  $\tau = 6$ ). Then all points that  $d(Point, VP) < \tau$  move to the left child node ( $S$ ) and the rest move to the right ( $Q$ ) (Figure 2.4 (d)). The construction is then terminated.

Once built, the tree can be queried for finding nearest neighbors. Figure 2.5 demonstrates a simplified version of how the nearest neighbor for a new point ( $X$ ) is found. The process starts at the root node ( $R$ ). Then the distance to the query point is measured (Figure 2.5 (b)). If  $d(QueryPoint, VP) < \tau$ , the process goes to the left child

---

<sup>3</sup>The big O notation ( $\mathcal{O}$ ) is often used in computer science to characterize runtime of algorithms by showing how quickly the duration would grow relative to the input size ( $n$ ), as the input gets arbitrarily large.




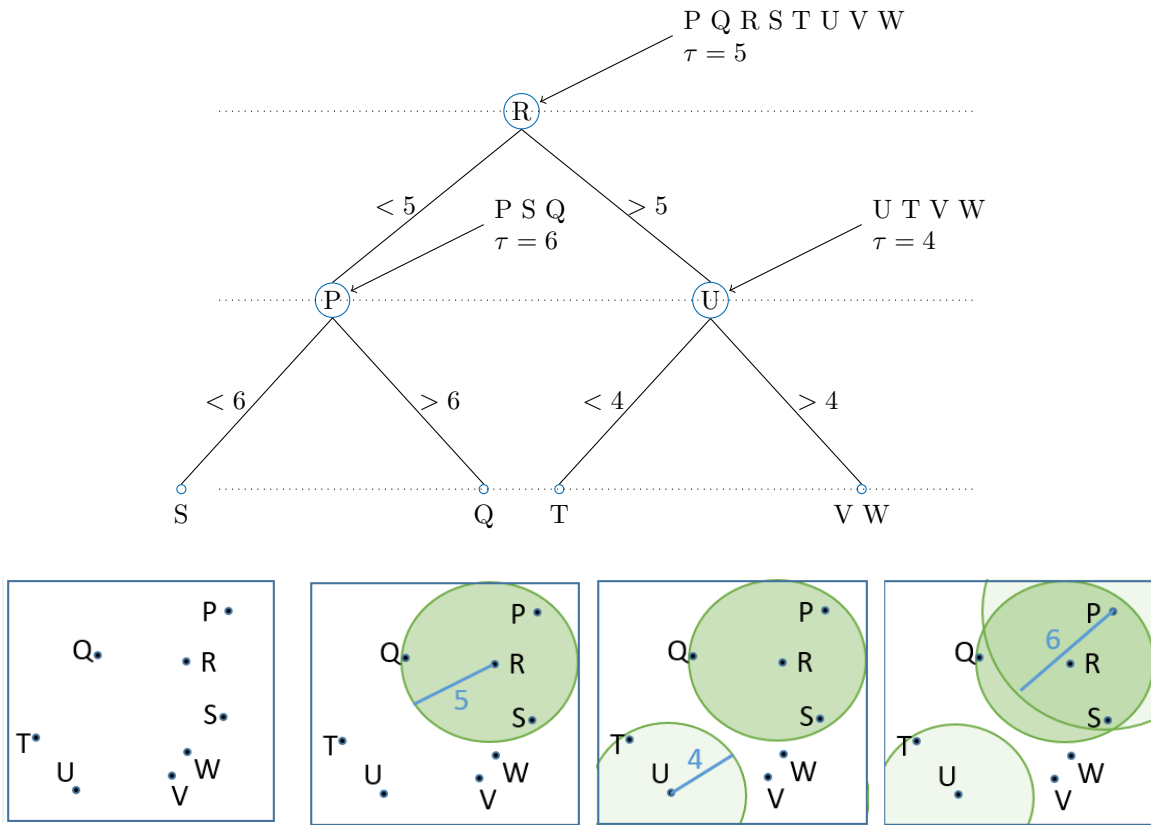
Structure	Data partitioning	Time Complexity <sup>3</sup> (Construction, Average Search)	Advantages	Disadvantages
kd-trees		$\mathcal{O}(n \log(n))$ $\mathcal{O}(\log(n))$ <small>True for low dimensions only</small>	Simple construction Efficient for low-dimensional data	Poor search performance for high dimensions
Ball Tree		$\mathcal{O}(n(\log(n))^2)$ $\mathcal{O}(n \log(n))$	Excellent search performance	Fair construction performance
Vantage Point Trees		$\mathcal{O}(n \log(n))$ $\mathcal{O}(\log(n))$	Excellent search performance Excellent construction time	-

Table 2.2: Search Tree Algorithms

Source: [Kumar et al. \[2008\]](#)

Efficient searching can be enabled by organizing data into hierarchical structures (trees). The table presents the partitioning of a 2D set of points using three different algorithms with a maximum leaf size of 1 and branching factor of 2. Line thickness denotes partition order (thicker lines were partitioned first). The different algorithms result in different construction and search time. As the Vantage Point tree is fast to construct and query, it is the one chosen for classification in this work.



a: Set of 2D points    b: R selected as VP    c: U selected as VP    d: P selected as VP

Figure 2.4: Vantage Point Tree Construction

The algorithm recursively chooses a random VP from a set of 2D points. Then divides the rest into two equal (or approximately equal) sets so that half of them are within a threshold  $\tau$  and the rest are outside it. All points that  $d(Point, VP) < \tau$  move to a left child node and the rest move to the right. The construction terminates when a predefined minimum leaf size is reached.

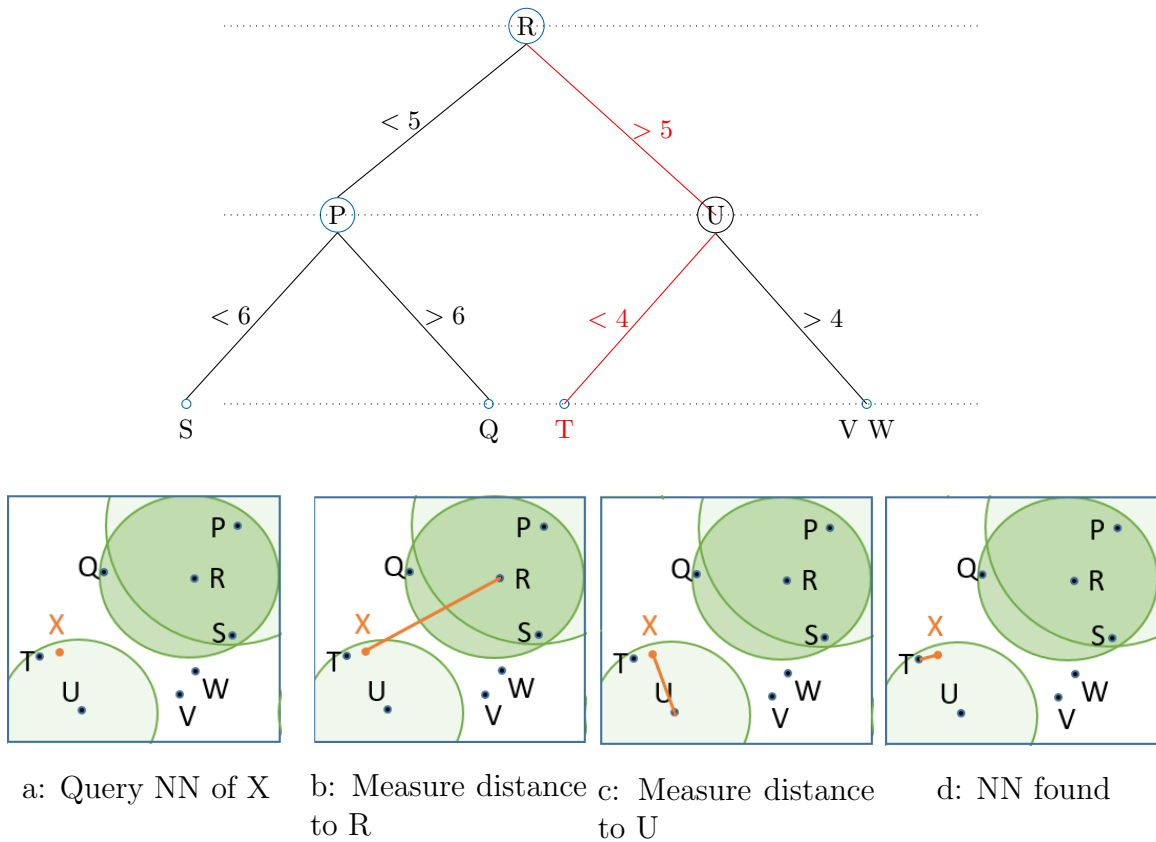


Figure 2.5: Vantage Point Tree Search

Searching for a NN starts at the root node. The distance the query point is measured and if  $d(\text{QueryPoint}, VP) < \tau$ , the search continues to the left child node, otherwise to the right. This method only finds approximate NNs and not necessarily the exact ones.

node, otherwise to the right. In this example,  $d(X, R) < \tau = 5$ , thus it moves to right. There again the distance to the VP ( $U$ ) is measured and a decision is made (Figure 2.5 (c)). As  $d(X, U) < \tau = 4$ , a move to left is done and the nearest neighbor ( $T$ ) is found (Figure 2.5 (d)).

Querying a VP tree for multiple  $k$  nearest neighbors does not require executing the described steps  $k$  times. Instead, a fixed size array for storing the NNs is used where elements go only if their distance to the query point is smaller than the distance to query point of any previously seen point. The threshold distance is initialized with infinity and is lowered every time a shorter distance is found. Before returning the kNNs array, the entries are sorted.

It must be noted that this procedure is capable of finding *approximate* NNs, but does not guarantee they are the absolute nearest ones. Finding the exact NNs requires more care and backtracking overhead that results in longer query time. Therefore, the approach here uses only approximate ones.

Algorithm 1 shows the pseudo code used for training the Vantage Point Forest (VPF) adopted from Heinrich and Blendowski [2016].

---

**Algorithm 1:** Vantage Point Forest Training

---

**Input** :  $|M|$  labeled training binary features  $h_j$  with labels  $k_j$ , parameters:  
number of trees  $T$ , minimum leaf size  $\mathcal{L}_{min}$

**Output:**  $T$  tree structures: indices of vantage points, thresholds  $\tau$  for every node, class probability distributions  $p_i^k$  and sample indices for leaf nodes

```

1 foreach  $t \in T$  do
2    $S = \text{Stack}(), S.\text{push}(M)$ 
3   while not  $S.\text{isEmpty}()$  do
4      $s = S.\text{pop}()$ , select vantage point  $j \in s$  (randomly)
5     if  $|s| > \mathcal{L}_{min}$  then
6       calculate  $d_H(i, j) = \|h_i - h_j\|_1 \forall i \in s$ , and median distance  $\tau = \tilde{d}_H$ ,
7       partition elements  $i$  of  $s$ :  $s_{Left} = \{i | d_H(i, j) < \tau\}$ ,
8        $s_{Right} = \{i | d_H(i, j) \geq \tau\}$  ( $s_{Right} \cup s_{Left} = s$ ,  $s_{Right} \cap s_{Left} = \emptyset$ )
9        $S.\text{push}(s_{Left}), S.\text{push}(s_{Right})$ 
10    else
11     store  $p_i^k$  and sample indices of  $S_l$  (leaf node)
12    end
13  end
14 end

```

---

## 2.4 Post Processing

With the help of the Vantage Point Forest, the indices of the nearest neighbors and their input labels can be retrieved and used to create probability maps. As done in the example with the circles, majority voting can be applied to predict final labels. This kind of classification might, however, not be spatially consistent as no prior shape knowledge is available. Therefore, a post processing step is required that takes into account the spatial nature of the images and *smoothens* any noisy or fragmented segmentations. Different algorithms such as graph cuts [Boykov and Jolly, 2001; Sinop and Grady, 2007], watersheds [Grau et al., 2004] or random walker [Grady and Funka-Lea, 2004] are commonly used as a post processing step and perform well in various scenarios. For the 3D image segmentation here, it is important that the smoothing incorporates both the computed label probabilities and the available voxel intensities. Therefore, the Random Walker Regularization algorithm is chosen.

### 2.4.1 Random Walker Regularization

The classical Random Walker (RW) algorithm, introduced by Grady and Funka-Lea [2004] and further refined in Grady [2005, 2006], can be used as an interactive tool for

image segmentation as seen in [Dong et al. \[2016\]](#); [Dukehart \[2009\]](#). It requires a set of user input label seeds  $V_S$  at a few pixels on the image and then "releases" random walkers from all the other unseeded pixels  $V_U$  ( $V_S \cap V_U = \emptyset$  and  $V_S \cup V_U = V$ , all pixels). By running this process multiple times, the likelihood for each walker to reach each label first is evaluated. Using this information, probability maps for all classes are obtained and the final pixel segmentation is based on the label with highest probability. [Figure 2.6](#) illustrates the classification obtained by this procedure for a  $4 \times 4$  image with three different labels ( $L_1$ ,  $L_2$  and  $L_3$ ).

This process, however, is relatively slow to converge, and therefore an alternative solution is of interest. For this purpose, the image segmentation task is treated as an optimization problem on a weighted graph. On such a graph, each vertex  $v \in V$  represents a pixel from the image and each edge  $e \in E$ , spanning two neighboring vertices<sup>4</sup>  $v_i$  and  $v_j$ , is assigned a weight  $w_{ij}$  corresponding to the probability that a random walker will cross that edge (e.g. a weight of zero means that the walker may not move along that edge) [[Grady and Funka-Lea, 2004](#)]. The weighting function

$$w_{ij} = e^{-\frac{(I(x_i) - I(x_j))^2}{2\sigma_w^2}} \quad (2.2)$$

is based on the intensities of neighboring pixels ( $I(x_i)$  and  $I(x_j)$ ) and has a maximum weight of 1 when they are equal.  $\sigma$  is a free tuning parameter acting as a length scale of similarity.

Finally, given the graph and set of classes  $K$  ( $k=1, 2, \dots, K$ ), the aim of the method is to assign a probability  $p_i^k$  to each node  $v \in V$  that it is assigned a label  $k$ .

In [Grady \[2006\]](#), it was established that these probabilities  $p_i^k = [p_1^k, p_2^k, \dots, p_K^k]$  can be obtained by minimizing the energy function

$$\begin{aligned} E_{RW}^k(p_i^k) &= (p_i^k)^T L p_i^k, \\ \text{subject to } p_i^k &= \hat{p}_i^k \forall v_i \in V \\ \forall v_i \in V_S, \hat{p}_i^k &= \begin{cases} 1 & \text{if pixel } i \text{ is marked with label } k \\ 0 & \text{if pixel } i \text{ is marked with another label} \end{cases} \end{aligned} \quad (2.3)$$

where  $L$  is the graph Laplacian matrix, defined as:

$$L_{ij} = \begin{cases} d_i = \sum_j w_{ij}, & i = j \\ -w_{ij}, & (v_i, v_j) \in E \\ 0, & \text{otherwise} \end{cases} \quad (2.4)$$

<sup>4</sup>Neighboring vertices in 4-connected neighborhood for 2D and 6-connected neighborhood for 3D images are assumed.

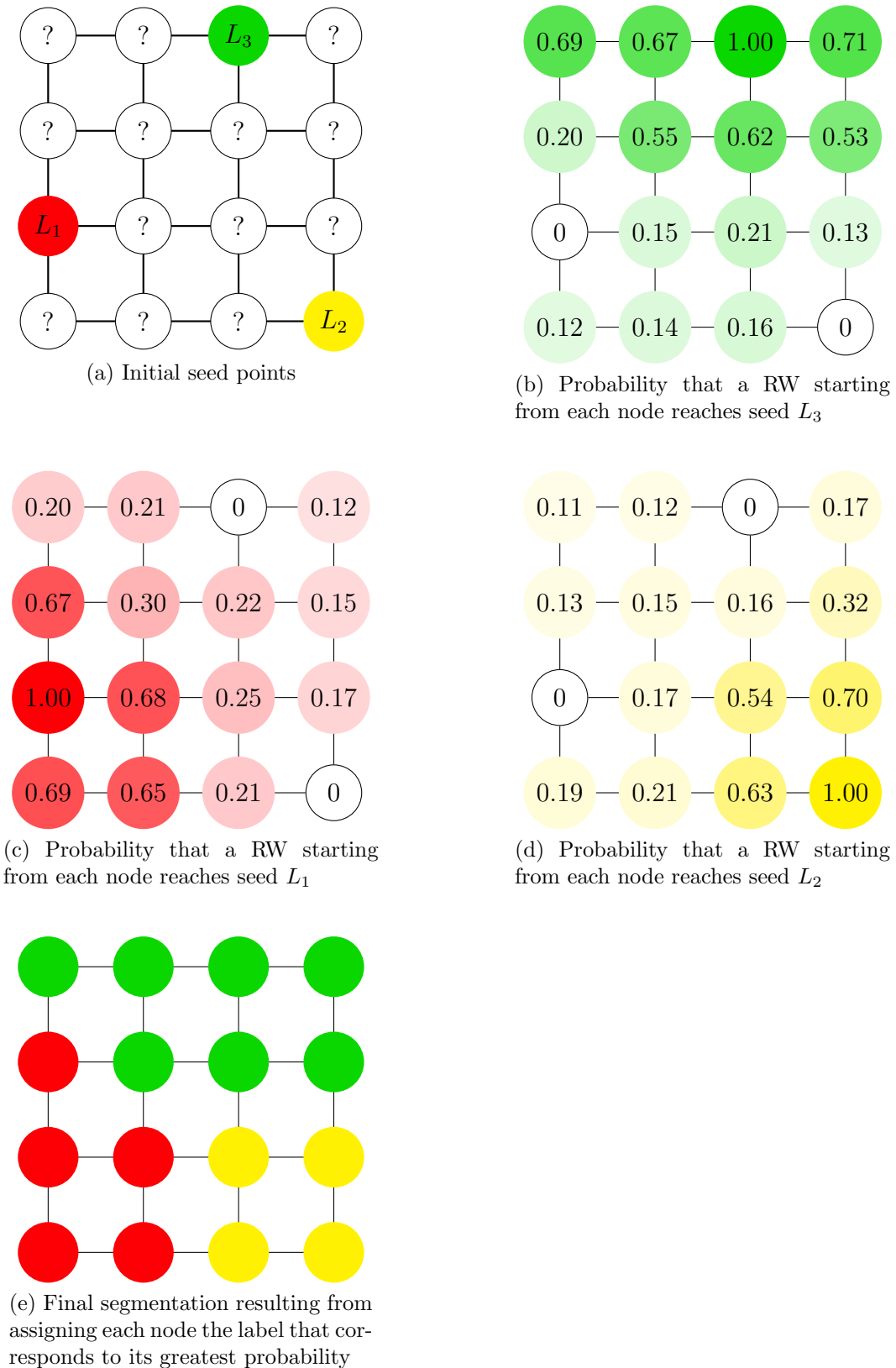


Figure 2.6: Illustration of the Random Walker for Segmentation

Classification obtained by the Random Walker algorithm for a  $4 \times 4$  image with 3 different label seeds ( $L_1$ ,  $L_2$ ,  $L_3$ ). From each unseeded node a Random Walker is released and the likelihood for each walker to reach each label first is evaluated. Final segmentation is based on the label with highest probability [Dong et al., 2016].

This matrix  $L$  can be also reordered to reflect the seeded and unseeded nodes partitioning:

$$L = \begin{bmatrix} L_S & B \\ B^T & L_U \end{bmatrix} \quad (2.5)$$

where  $L_S$  are the edges between seeded nodes,  $L_U$  are the edges between unseeded ones and  $B$  are the edges between seeded and unseeded nodes. Then, Equation 2.3 can be rewritten as

$$E_{RW}(p_U) = \frac{1}{2} [p_S^T p_U^T] \begin{bmatrix} L_S & B \\ B^T & L_U \end{bmatrix} \begin{bmatrix} p_S \\ p_U \end{bmatrix} = \frac{1}{2} (p_S^T L_S p_S + 2p_U^T B^T p_S + p_U^T L_U p_U) \quad (2.6)$$

where  $p_S$  and  $p_U$  are the probabilities of the seeded and unseeded nodes respectively.

As the labels of the seeded pixels are already known, only the probabilities of the unmarked portion of the Laplace matrix  $L_U$  need to be found. By definition,  $L$  is symmetric, positive semi-definite and therefore minima can be found at the critical points of  $E_{RW}^k$ . Thus, differentiating  $E_{RW}(p_U)$  with respect to  $p_U$

$$\frac{\partial E_{RW}(p_U)}{\partial p_U} = \frac{1}{2} (2B^T p_S + 2L_U p_U) \quad (2.7)$$

and finding the critical points yields

$$L_U p_U = -B^T p_S \quad (2.8)$$

which is a system of linear equations with  $|V_U|$  unknowns that the random walker needs to solve for each label.<sup>5</sup>

After minimizing  $E_{RW}^k$  for every class, the segmentation is obtained by retaining the label of maximum probability for each pixel  $k_i = \text{argmax } p_i^k$  [Baudin et al., 2012].

Figure 2.7 (a) shows a sample  $3 \times 3$  grid with some label seeds (foreground in red, background in green) and the assigned edge weights. Table 2.3 presents the Laplacian matrix computed according to Equation 2.4 for this graph.

<sup>5</sup>We use the Successive Over Relaxation (SOR) [Allahviranloo, 2005] method that iteratively solves the left hand side of an expression for an unknown parameter, using previous values for it on the right hand side.



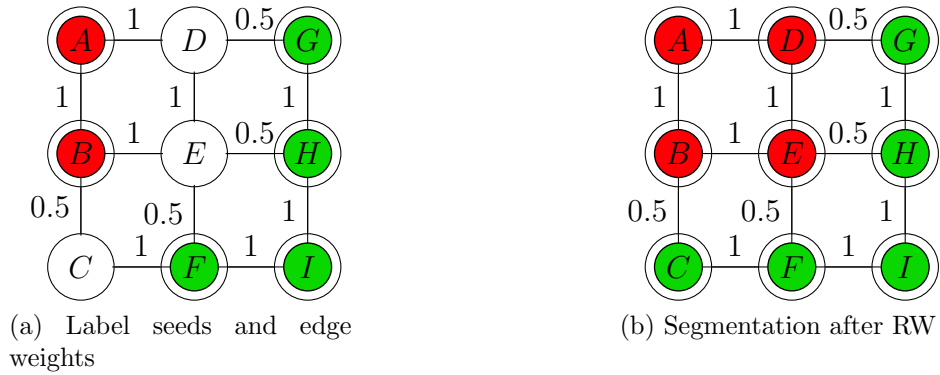


Figure 2.7: Random Walker Segmentation

Random Walker segmentation on a graph with seeded nodes (foreground: red/ background: green) and edges weights based on the image intensities.

	A	B	C	D	E	F	G	H	I
A	2	-1		-1					
B	-1	2.5	-0.5		-1				
C		-0.5	1.5			-1			
D	-1			2.5	-1		-0.5		
E		-1		-1	3	-0.5		-0.5	
F			-1		-0.5	2.5			-1
G				-0.5			1.5	-1	
H					-0.5		-1	2.5	-1
I						-1		-1	2

Table 2.3: Laplacian Matrix L

The matrix is computed for the graph in Figure 2.7(a) according to Equation 2.4. The main diagonal is colored in blue,  $L_U$  is in yellow and  $L_S$  - in green.

Solving Equation 2.8 for  $p_u$  :

$$\begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 2.5 & -1 \\ 0 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = - \begin{bmatrix} 0 & -0.5 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & -0.5 & 0 & 0 \\ 0 & -1 & -0.5 & 0 & -0.5 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 2.5 & -1 \\ 0 & -1 & 3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1 \\ 1 \end{bmatrix}$$

gives the foreground probabilities for the unseeded vertices:  $p_U = [0.33, 0.61, 0.54]$ .

Choosing the label with maximum probabilities (=thresholding at 0.5 when one foreground label is present), results in the final segmentation as seen in [Figure 2.7\(b\)](#).

## 2.4.2 Random Walker Regularization with Label Priors

Relying on user input for image segmentation is time consuming and often not feasible. Therefore, the Random Walker can be modified to regularize data when no label seeds are available, but instead parameter maps are given. Such can be obtained from machine learning algorithms, intensity-models or atlas-based registration.

Assuming a node-wise map  $\lambda_i^k$  representing the probability density that the intensity at node  $v_i$  belongs to  $g^k$ , the RW can be generalized to

$$(L + \gamma I)p^k = \gamma \lambda^k \tag{2.9}$$

where  $L$  is still derived from the original intensities,  $I$  is an identity matrix and  $\gamma$  is a balancing regularization parameter [[Grady, 2005](#)]. Note that for  $\gamma \rightarrow \infty$ , the previous RW definition ([Equation 2.8](#)) is recovered. Solving [Equation 2.9](#) for  $p^k$  results in the desired final smooth labels.

## 2.5 Evaluation Metrics

3D medical image segmentation is a challenging task and errors may occur. Such can be added regions (labels), added background, inside holes or border holes. Therefore, universal standard metrics are needed to reliably indicate the accuracy of the different classification algorithms. Even though, there are various assessment methods available, most of them can be grouped into overlap-based, volume-based, probabilistic-based, pair counting-based, information theoretic-based, and spatial distance-based ones [[Taha and Hanbury, 2015](#)].

The evaluation metrics used in this thesis are spatial overlap-based as they are the most common ones in 3D medical image segmentation and comparison to similar works is possible. The exact methods are discussed in detail in the following section.

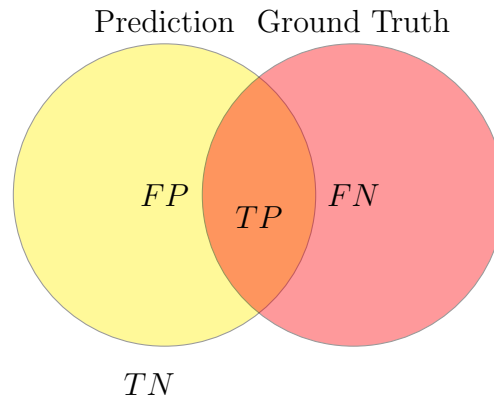


Figure 2.8: Classification Outcomes

The result of a classification procedure is rated against the ground truth. The four possible outcomes are: correctly positive segmented (**TP**), falsely positive segmented (**FP**), correctly negative segmented (**TN**) and falsely negative segmented (**FN**).

### 2.5.1 Spatial Overlap-based Metrics

The performance of the segmentation approach used in this thesis is evaluated during the validation phase, where the classification is rated against the ground truth segmentation. The four possible outcomes of such an assessment on a pixel (voxel) base are: True Positive (**TP**), False Positive (**FP**), True Negative (**TN**) and False Negative (**FN**) classification as shown in Figure 2.8.

The four basic classification results are also often presented in the form of a confusion matrix where the types of predictions are summarized with count values and provide an insight on the kind of errors being made. Figure 2.9 shows a sample binary confusion matrix for bladder segmentation where a total of 166 voxels are labeled as either bladder or background. It can be observed that 100 voxels were correctly classified as a bladder (**TP**) and 50 were labeled as background as they should be (**TN**). 6 voxels were falsely considered as bladder when in fact they were background (**FP**) and 10 were classified as background, although they were part of the bladder (**FN**).

When assessing multi-label segmentation, each pair of classes is evaluated separately and the results are presented in a complex confusion matrix. Besides the information about the four basic cardinalities, such a representation also gives an overview on whether (and if true, how often) certain classes get confused with others. The matrix in Figure 2.10 shows the outcomes of a three-class segmentation of 310 voxels. The counts of correctly classified bladder, prostate and background voxels are placed on the diagonal. The rest of the matrix indicates the incidence of confusion between the three labels.

Given the confusion matrix, several rates can be computed to evaluate the classification. Furthermore, depending on the purpose of the segmentation, the method can be adapted to maximize the results of these metrics.

		Predicted		total
		Bladder	Backgr.	
Actual	Bladder	True Positive 100	False Negative 6	106
	Backgr.	False Positive 10	True Negative 50	60
total		110	56	

Figure 2.9: Confusion Matrix

Sample confusion matrix for a binary bladder segmentation of 166 voxels. The table summarizes the occurrences of the four possible voxel-wise outcomes from a segmentation procedure (TP, TN, FP, FN).

		Predicted			total
		Bladder	Prostate	Backgr.	
Actual	Bladder	100	0	10	110
	Prostate	10	80	10	100
	Backgr.	30	0	70	100
total		140	80	90	

Figure 2.10: Confusion Matrix, Multiple Classes

Sample confusion matrix for multi-class segmentation of 310 voxels. The table summarizes the occurrences of correctly classified voxels (on the diagonal). Additionally it gives an overview how often classes get confused with others.

Sensitivity, or recall,  $(TP/(TP + FN))$ , for instance, is a statistic that refers to the portion of positive voxels in the GT that are identified as positive by the classification. It is defined in the range  $[0,1]$  and a maximum sensitivity of 1 means that no undersegmentation has occurred. It however, does not capture whether oversegmentation is present.

Precision  $(TP/(TP+FP))$ , on the other hand, indicates the portion of correctly labeled as positive samples from all positively labeled ones. Perfect precision (=1) implies that there is no oversegmentation, but does not report on undersegmentation. Then classifying every voxel as background will maximize the precision, but will result in zero sensitivity.

The Dice score  $(2TP/(2TP + FP + FN))$  incorporates both the precision and the sensitivity of an algorithm. It is their harmonic mean  $(2 \times Precision \times Sensitivity / (Precision + Sensitivity))$  and gives an insight on the spatial overlap between the ground truth and the segmentation result. The Dice score can take values in the range  $[0,1]$ , where 0 indicates no overlap and 1 - complete overlap.

Table 2.4 summarizes four common overlap based evaluation metrics and the formulas that define them.

For the values in the binary confusion table in Figure 2.9 the overlap metrics are computed as follows:

$$DICE = \frac{2TP}{2TP + FP + FN} = \frac{2 \times 100}{2 \times 100 + 10 + 6} = 0.93 \quad (2.10)$$

$$Recall = \frac{TP}{TP + FN} = \frac{100}{100 + 6} = 0.94 \quad (2.11)$$

$$Specificity = \frac{TN}{TN + FP} = \frac{50}{50 + 10} = 0.83 \quad (2.12)$$

$$Precision = \frac{TP}{TP + FP} = \frac{100}{100 + 10} = 0.91 \quad (2.13)$$

Since these four metrics are defined in the range 0-1, it can be seen that all results are closer to the maximum of 1. It can be concluded that although some under- and oversegmentation is present, the overall segmentation is *good*.

The spatial overlap rates for multiple classes are evaluated independently for each label with respect to the background. Although there are different approaches how to merge this into a single metric [Crum et al., 2006], the one adopted here is simply averaging the final results.

<b>Metric</b>	<b>Measure of interest</b>	<b>Definition</b>
Dice/F1-Measure	Overlap index	$DICE = \frac{2TP}{2TP+FP+FN}$
True Positive Rate Sensitivity/Recall	The portion of positive voxels in the <b>GT</b> that are identified as positive by the segmentation	$TPR = Recall = Sensitivity = \frac{TP}{TP+FN}$
True Negative Rate Specificity	The portion of negative voxels in the <b>GT</b> that are identified as negative by the segmentation	$TNR = Specificity = \frac{TN}{TN+FP}$
Positive Predictive Value Precision	The portion of correctly labeled as positive voxels from all positively labeled	$PPV = Precision = \frac{TP}{TP+FP}$

Table 2.4: Overlap-based Evaluation Metrics

Universal evaluation metrics are required as indicators of segmentation performance. Spatial overlap based metrics are the most common ones for assessing 3D medical image delineation. The methods shown in the table are based on the four possible outcomes (**TP**, **TN**, **FP**, **FN**) of a segmentation procedure on a pixel/ voxel basis.

## 3. Experiments and Results

In this chapter, the image segmentation framework is presented and the nearest neighbor search algorithm for binary features is described. Then the method is applied to two different test scenarios. Finally, the results of the experiments are presented and discussed.

### 3.1 Pipeline Overview

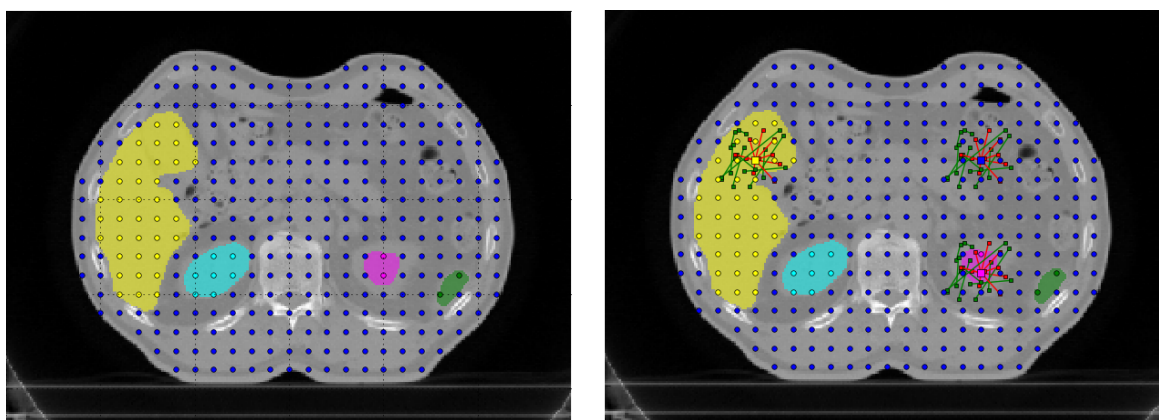
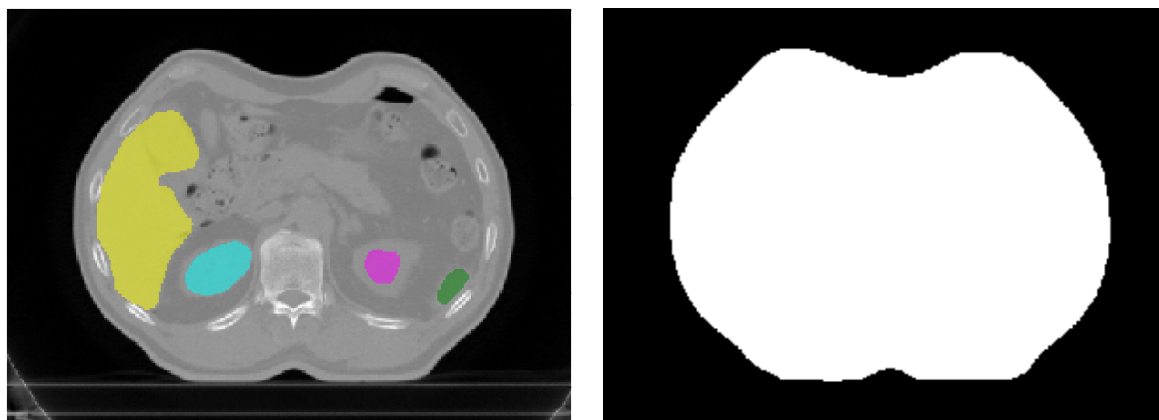
The suggested framework incorporates the concepts presented in Chapter 2. The pipeline can be viewed as two independent procedures: training and testing. Figure 3.1 illustrates the main steps of the training phase. The algorithm assumes images with dense voxel annotations (a), paired with binary body contour masks (b) that indicate which regions of the images are relevant for feature extraction. After pre-smoothing the images with a Gaussian filter, the feature extraction happens on regular grids (c) within the masks using the BRIEF and LBP sampling patterns (d). The binary feature vectors are then stored together with the labels<sup>1</sup> of the voxels they characterize (e).

Figure 3.2 presents the classification procedure for a new test image. Just as in the training phase, features are extracted from a regular grid with the BRIEF and LBP patterns (a, b). Although the grid does not have to be the same as the one used in the training phase<sup>2</sup>, the sampling pattern has to be identical. Then by using VPFs, the indices of the nearest neighbors (of the test features vectors) are queried and their labels are retrieved (c). This information is used to compute the probability for each label at each location of the grid (d). The label probabilities are then linearly interpolated across the whole image (e). Finally, Random Walker regularization is applied to ensure smooth labels (f).

---

<sup>1</sup>Different labels are indicated by different colors.

<sup>2</sup>Denser grids result in better segmentation, but take longer to process as more features are sampled.



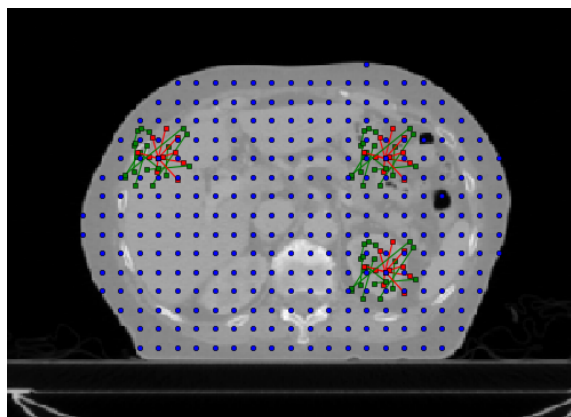
Feature	Index	0	1	2	3	4	5	6	7	...
	n=0		0	1	1	1	0	1	0	1
n=1		0	1	0	0	0	1	0	0	...
⋮		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	...
n=1280		0	0	1	1	0	0	1	1	...
Label		[Blue]		[Green]	[Pink]	[Blue]	[Yellow]	[Blue]		...

(e) Storing features and labels

Figure 3.1: Training Phase

During the training phase images with dense voxel annotations are loaded. If available, body contour masks (or any other binary masks) can be used so that the training only focuses on data from relevant regions. Then BRIEF and LBP features are extracted on regular grids within the masks and stored together with the labels.





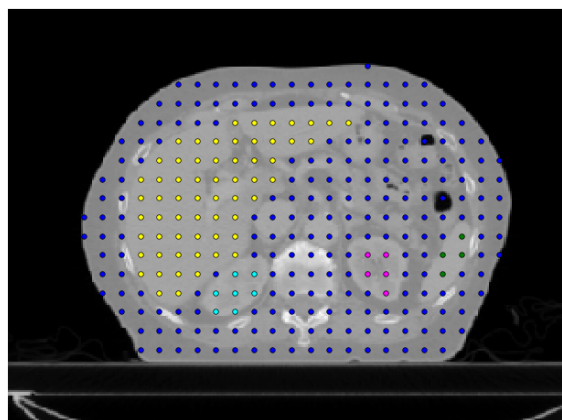
(a) Feature extraction on a test image (at 3 points)

Index	0	1	2	3	4	5	6	...
n=0	0	1	1	1	0	0	0	...
n=1	0	1	0	1	0	1	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n=1280	1	0	0	0	1	0	0	...

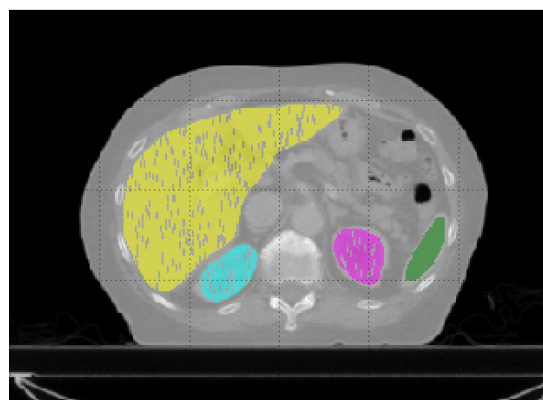
(b) Test feature vectors

Index	0	1	2	3	4	5	6	...
NN = 1	0	2	1	33	34	18	3	...
NN = 2	8	63	9	13	27	29	55	...
NN = 3	12	99	26	77	33	0	56	...
NN = 4	5	14	8	9	13	1	57	...
Labels								...

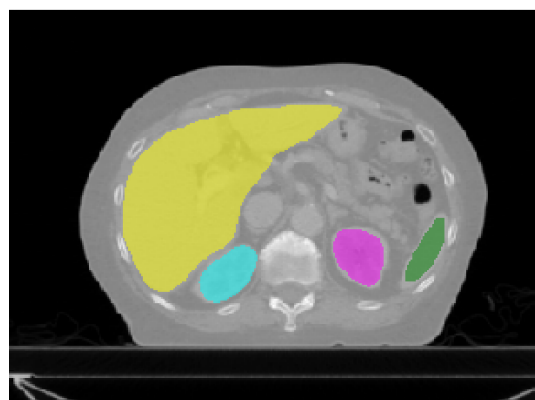
(c) Nearest neighbors querying and major voting



(d) Assigning labels to query points



(e) Label interpolation (inconsistent labels)



(f) After post processing (RW smoothing)

Figure 3.2: Testing Phase

When segmenting a test image, the same type of BRIEF and LBP features as in the training phase are extracted at points on a regular grid within mask. Querying VPFs containing the training feature vectors, gives the indices of the NNs of the test vectors. The labels of the NNs are retrieved and the probability for each label at each location of the grid is calculated. The probabilities are interpolated across the whole image and the final labels are regularized by a Random Walker.

## 3.2 Nearest Neighbor Visualization

The core of the segmentation algorithm proposed here is classifying voxels based on similarity of features extracted from context. To better illustrate this, a visual representation is shown in [Figure 3.3](#) and [Figure 3.4](#). They compare the results of querying 5 approximate nearest neighbors from VPF and Kd-trees trained on limited amount of features. Additionally the absolute 5 NNs are found by the BF method. The images show slices of the 3D patches around the voxels where the feature vectors were sampled. Although the features were constructed by patterns with greater displacement distribution around the central voxel, the slices capture only  $\pm 30$  voxels in x and y directions. The distances to the queried test vector are given below the patches and the different labels of the NNs are indicated by different colors. The vectors used for training and testing are 1280 bits long.

[Figure 3.3](#) presents a scenario where the test data is included in the training. It can be observed that all three methods succeed in finding the same feature as an absolute nearest neighbor with distance = 0. In all three approaches the Hamming distance increases with the rank of the neighbor, as expected. From the results it can be concluded that VPF finds neighbors with smaller distance compared to the Kd-trees, but they are not always the absolute nearest ones as the BF provides. It can be also observed that after the 4<sup>th</sup> NN for BF and after the 2<sup>nd</sup> for the other two methods, the labels do not match the test feature class. This (intended) "mistake" occurs due to the limited training data. On the contrary, if more features are trained, the more "closer" ones will be available and getting nearest neighbors with different label will be less likely to happen.

[Figure 3.4](#) is visualizing the same procedure as in [Figure 3.3](#), but this time the test features are not part of the training data. Again, VPF performs better than the Kd-trees, but not as good as the BF approach. The difference to the absolute nearest neighbors given by BF, however, is acceptable considering the much shorter construction and search duration.

Additionally, the distribution of the distances to the nearest neighbors for both trained and not trained test data are shown in [Figure 3.5](#). It can be again seen that the VPF results in nearer neighbors in every case compared to the Kd-trees. Additionally, the VPF implementation used here is also faster than the Kd-trees and is therefore preferred.

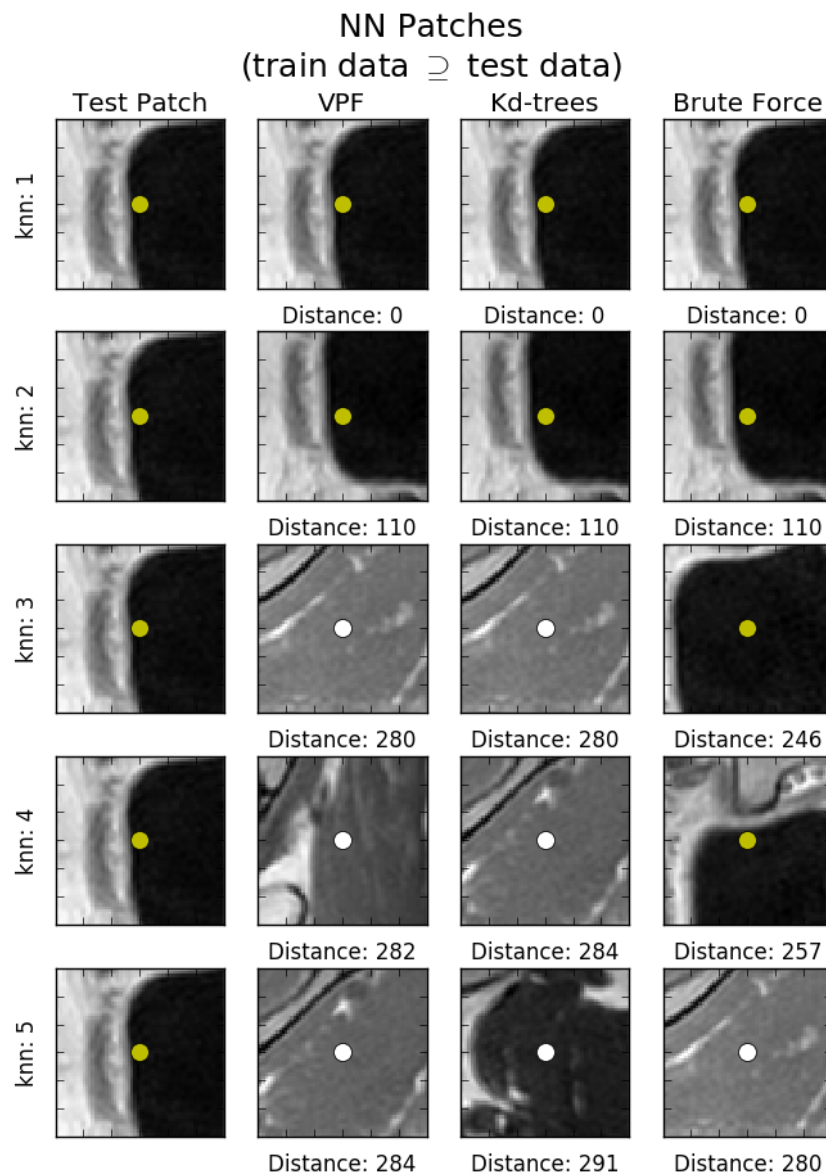


Figure 3.3: NN Visualization by Patches (Test Data Included)

The 5 approximate NNs for a test feature vector, included in the training, are queried from a VPF and Kd-trees trained on limited amount of data. Additionally, the absolute 5 NNs are found by the BF method. For each feature, a slice of the 3D patch around it ( $\pm 30$  pixels in  $x, y$ ) is shown and its label is given by the color of the central point. The Hamming distances to the test feature vector indicate that VPF finds closer neighbors than the Kd-trees, but not always the absolute ones as the BF. Not having sufficient training features similar to the query vector may result in NNs that have different labels.

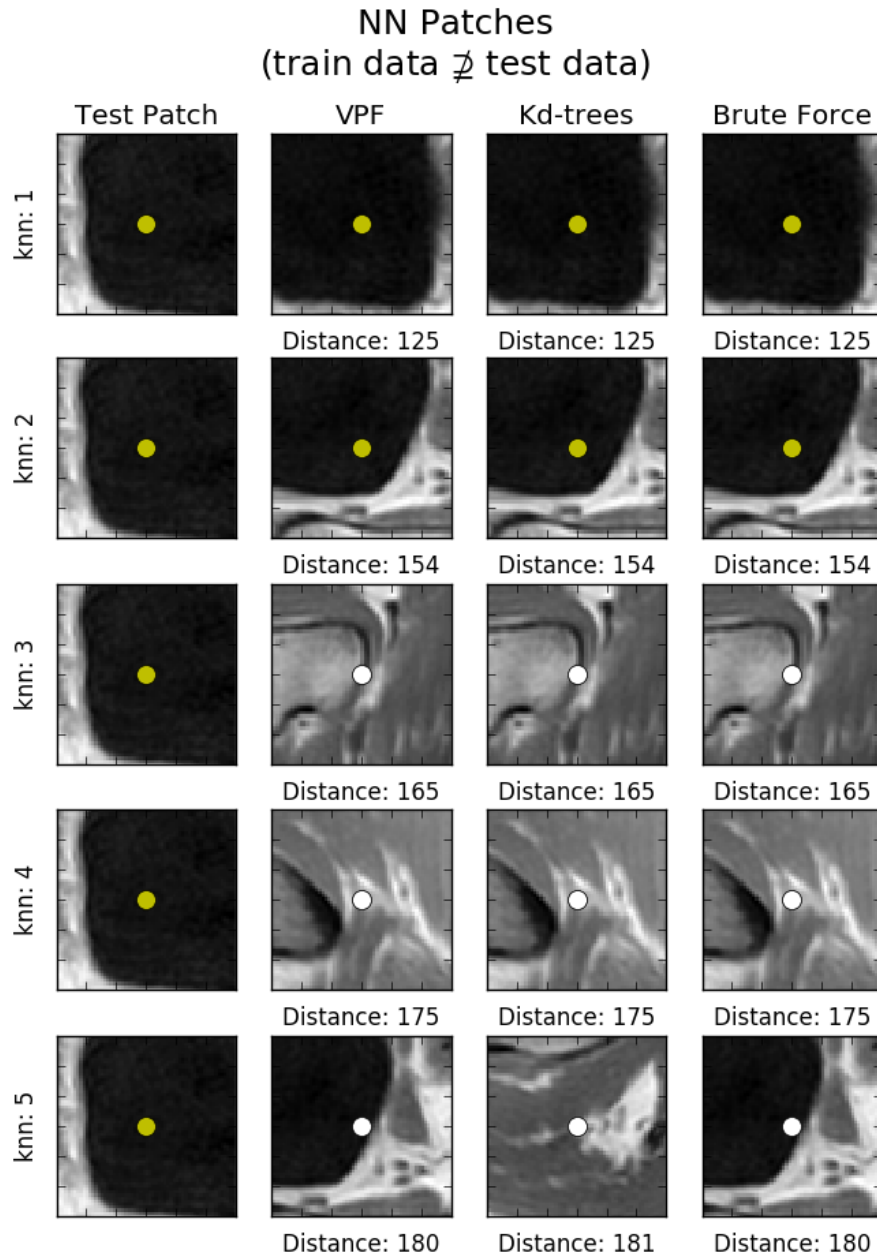
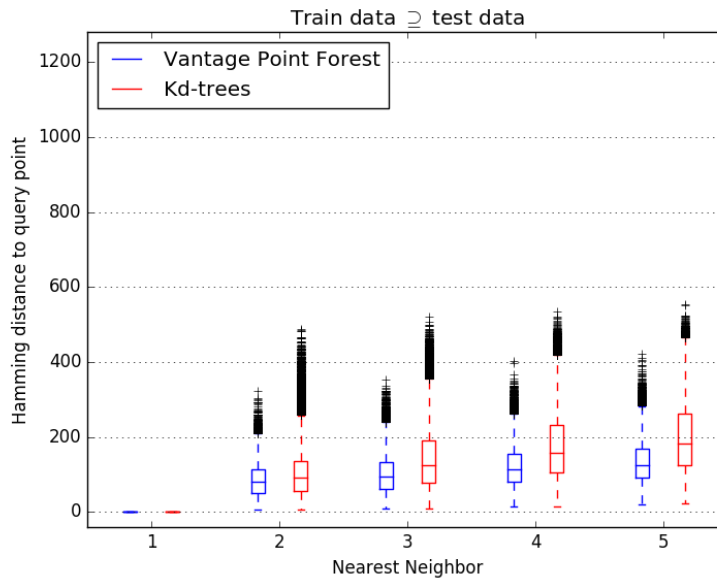
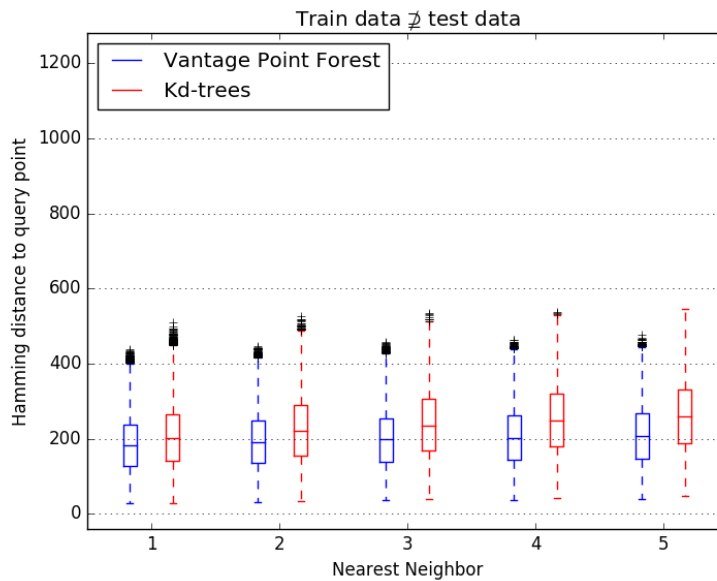


Figure 3.4: NN Visualization by Patches (Test Data not Included)

The 5 approximate NNs for a test feature vector, **not** included in the training, are queried from a VPF and Kd-trees. Additionally, the absolute 5 NNs are found by the BF method. For each feature, a slice of the 3D patch around it ( $\pm 30$  pixels in  $x, y$ ) is shown and its label is given by the color of the central point. The Hamming distances to the test feature vector indicate that VPF finds closer neighbors than the Kd-trees. In this example the VPF actually retrieves the absolute ones as the BF. The wrong labels of the retrieved NNs are due to limited training data.



(a) Test data included in training



(b) Test data not included in training

Figure 3.5: Distance to Nearest Neighbors

The distribution of the distances to the 5 NNs for 1280-bits long vectors is compared between VPF and Kd-trees. In (a) the test data is included in the training, and in (b) it is not. In (a) both methods find the exact test vectors (distance=0). Vantage Point Forest retrieves neighbors with shorter distances to the query points compared to the Kd-trees and is therefore chosen as a classifier.

### 3.3 Parameters Settings

Over the course of the segmentation procedure, there are several parameters that can be configured. Choosing optimal values for these, however, is not a straight forward task. Figure 3.6 illustrates the effects of modifying the number or queried NNs, the minimum leaf size limit  $\mathcal{L}_{min}$ , the number of VP trees T in a VPF, the regularization  $\gamma$  and  $\sigma_w$  on the Dice score, the precision and the recall metrics. It can be observed that lowering or incrementing the values does not improve or worsen the evaluation metrics in the same manner over all images. Higher regularisation  $\sigma_w$  values, for example, improve the Dice score of the forth image, but  $\sigma_w > 10$  has no effect on the other three images. When segmenting image 3, using less trees in a forest improves the recall, but lowers the precision values.

On the other hand side, there are parameters that affect the results in a consistent way. For instance, extracting features for each voxel of the training and testing data, improves the segmentation accuracy. However, this results in less efficient classification. On the contrary, if the feature vectors are only obtained for voxels far away from each other, the procedure is much faster, but the accuracy is negatively influenced. The exact BRIEF and LBP sampling patterns used to construct the binary strings directly influence the performance of the classification too. When comparing intensities really close to the the patch of interest to construct the features, the patterns "oversee" relevant neighborhood information. In contrast, by focusing on intensity information far away they *fail* to represent the patch itself. Table 3.1 lists the configurations of some settings so that a good balance between speed and accuracy is achieved. Moreover, it gives an insight on how alternating these values would affect the results.

On average, the following parameter values resulted in the highest scores and are therefore selected in such a way. After pre-smoothing the images with a Gaussian kernel with 3 voxels, 1280-bit long random features (BRIEF: 20-40%, LBP: 80-60%) are extracted. The displacement distribution of the sampling pattern points around the voxel of interest is normal with a standard deviation of  $[20/30; 80]^3$ . For each experiment 15 trees are either fully grown or terminated at a fixed leaf size of  $\mathcal{L}_{min} = 15$  with features extracted from every  $4^{th}/6^{th}$  voxel in x, y, z directions. Then, for the feature vectors extracted from each  $2^{nd}/4^{th}$  voxel from the test images, 20 nearest neighbors are queried. The labels of the neighbors are retrieved and the probability maps for each label are regularized with  $\gamma = 20$  and  $\sigma_w = 10$ .

---

<sup>3</sup>The minimum and maximum values correspond to the standard deviation of the displacement distribution of the sampling points measured by voxels. These values are adapted for images of size  $\sim 520 \times 520 \times 400$ .

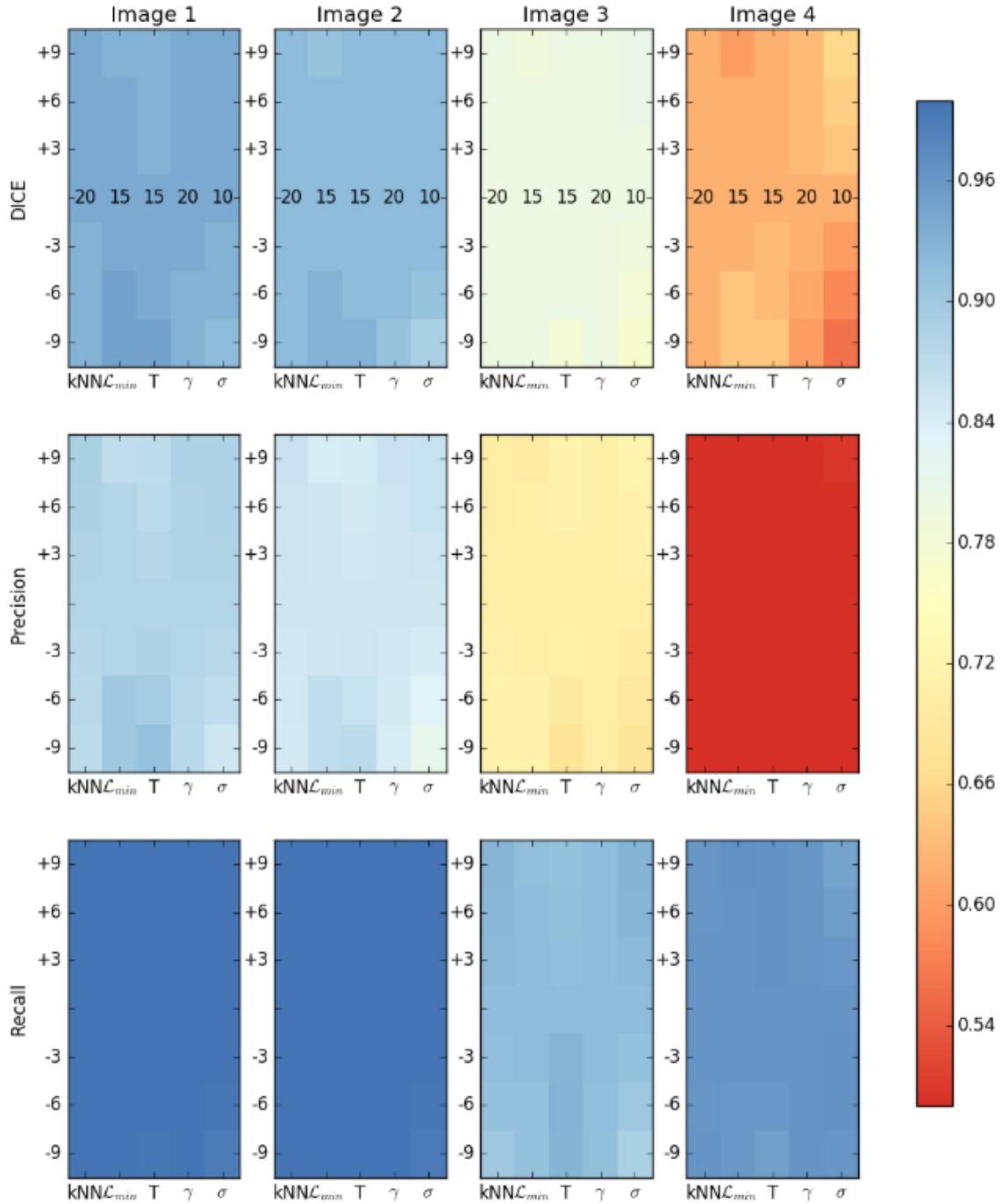


Figure 3.6: Parameter Settings, Effects of Modifications

The effects of modifying the number or queried NNs, the minimum leaf size limit  $\mathcal{L}_{min}$ , the number of VP trees  $T$ , the regularization  $\gamma$  and  $\sigma_w$  are evaluated on four different images. Even though the scores for some of the images benefit from higher (or lower) values for some of the parameters, others do not. Therefore, the final values chosen for the experiments are:  $kNN = 20$ ,  $\mathcal{L}_{min} = 15$ , number of trees  $T = 15$ ,  $\gamma = 20$ ,  $\sigma = 10$

Parameter	Smaller	Default setting	Larger
Sampling pattern std. deviation of displacement distribution [min, max]	Focus on too narrow area around the patch of interest, worse segmentation	[20/30; 80]	Capture information that is too far away from patch of interest, worse segmentation
BRIEF:LBP ratio	Features might oversee important relations between neighboring regions	$0.2 \leq \text{ratio} \leq 0.4$	Features might not characterize sufficiently the area of interest
Binary feature length	Not characteristic features, worse segmentation	1280 bits	Longer feature vector will not necessarily capture more contextual information, but will slow down the procedure
Training grid stride	Slightly better segmentation, but significantly slower	$4 \leq \text{stride} \leq 6$	Faster segmentation, lower accuracy
Testing grid stride	Slightly better segmentation, but significantly slower	$2 \leq \text{stride} \leq 4$	Faster segmentation, lower accuracy

Table 3.1: Parameter Settings

We present the values chosen for the experiments so that a good balance between accuracy and speed is achieved. Furthermore, the effects of incrementing or lowering these values are listed.



## 3.4 Experiments

### 3.4.1 CT Abdominal Data

Rapid and accurate segmentation of abdominal Computed Tomography (CT) images is important for computer-aided diagnosis, radiotherapy planning as well as cancer delineation and staging where the estimation of the anatomical boundaries needs to be accurate [Hu et al., 2016]. This task, however, is challenging as large variations exist in location, shape and size of abdominal organs among individuals. Moreover, these organs are mainly surrounded by soft tissues with similar appearance and intensities, resulting in fuzzy boundaries that are hard to delineate.

The CT data used here has dense segmentations of liver, spleen, left and right kidneys. 68 images of size  $512 \times 512 \times 394 = 103\,284\,736$  voxels, each voxel of size  $1.37 \times 1.37 \times 1.36$  mm, are used for a 4-fold cross validation. At each of the 4 rounds  $3/4 \times 68 = 51$  images are trained and  $1/4 \times 68 = 17$  are tested.

Initial tests showed that sampling training data from a regular grid results in imbalanced data with high prevalence towards background features. Therefore, during the training phase the size of background voxels within the body contour is downweighted by a factor of 5.

Figure 3.7 presents the average confusion matrix over all classified images. The high values on the diagonal indicate that voxels belonging to the organs are correctly classified in most cases. However, organs are occasionally labeled as background some of the time as background features still dominate in the training data ( $\sim 2000$  times more). Inter-organ confusion happens in very few cases.

Figure 3.8 shows the distribution of the Dice overlap measure. It can be observed that the results are relatively high and the spread is rather small. This means that algorithm is stable across different organs and results in a good delineation. The average Dice scores for the organs are: liver - 0.82, spleen - 0.71, left kidney - 0.72, right kidney - 0.71. The overall Dice score is 0.74. These values give an impression how well the algorithm would perform in general when new data with same parameters is tested.

Figure 3.9 illustrates the precision of the algorithm. As the precision presents the portion of correctly labeled as positive voxels from all positively labeled voxels, it can be seen that occasionally some voxels are classified as organs, although they belong to the background. Such kind of faulty classifications indicate oversegmentation.

The sensitivity of the algorithm is also evaluated. In Figure 3.10 the recall values are presented. They correspond to the portion of positive voxels in the ground truth that are positively labeled by the segmentation. The high results in the figure indicate that in most of the cases all voxel belonging to either the background or the organs are also discovered by the algorithm. It can be concluded that undersegmentation is rarely happening.

Figure 3.11 visualizes a sample "good" CT abdominal segmentation. Each of the four organs is presented separately and the achieved Dice scores are given. The figure gives

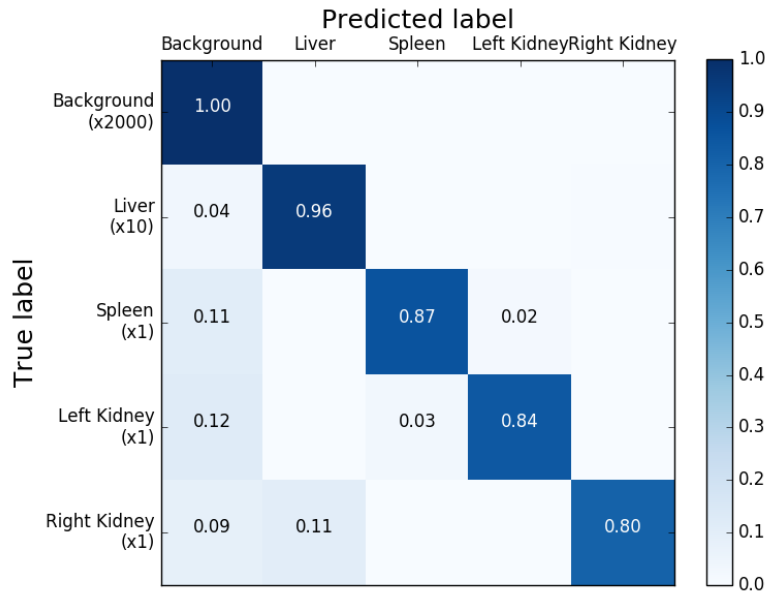


Figure 3.7: CT Abdominal Segmentation (Confusion Matrix)

The confusion matrix summarizes the average outcome of the cross validation. The approximate amount of features compared to the organ with the least samples (right kidney), is indicated below each class. The strong diagonal gives an insight that most of the times voxels are correctly classified. Rarely an organ is confused with another one. The biggest confusion is when organs are predicted to be background. This is mainly due to the imbalanced training data. Approximately 2000 times more background features are available compared to organ features.

an overview of the label probabilities after interpolating the VPF data, the probability maps after RW regularization and the final labels after majority voting. Additionally, in the last column the segmentation is compared to the ground truth delineating. The TP voxels are shown in green, FP in red and FN in blue. It can be seen that some under-segmentation occur, but usually the errors are due to oversegmentation. Nevertheless, most of the classification is correct.

Figure 3.12 illustrates one of the outlier cases where the segmentation has a lower Dice score for the left and right kidneys. It can be observed that the locations of both organs are correctly identified, but under- and over-segmentation occur. This example shows that even though the algorithm performs well for bigger structures, the performance for smaller ones has room for improvement.

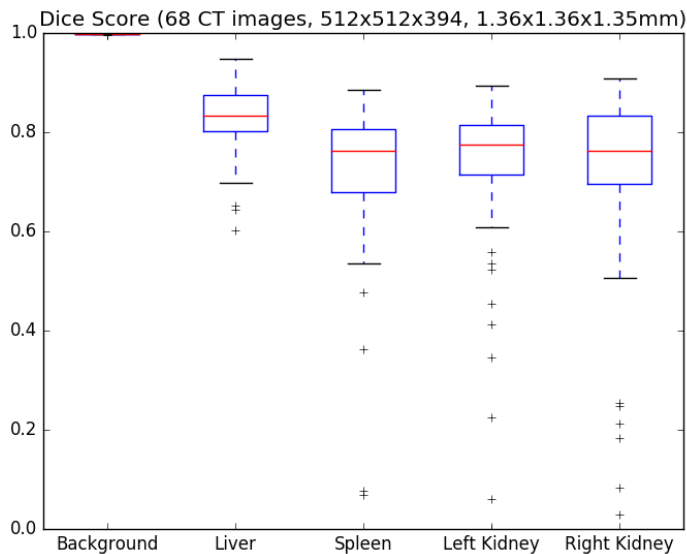


Figure 3.8: CT Abdominal Segmentation (Dice Score)

The box plot illustrates the distribution of the DICE score. The relatively small spread shows that the algorithm is stable across the different organs.

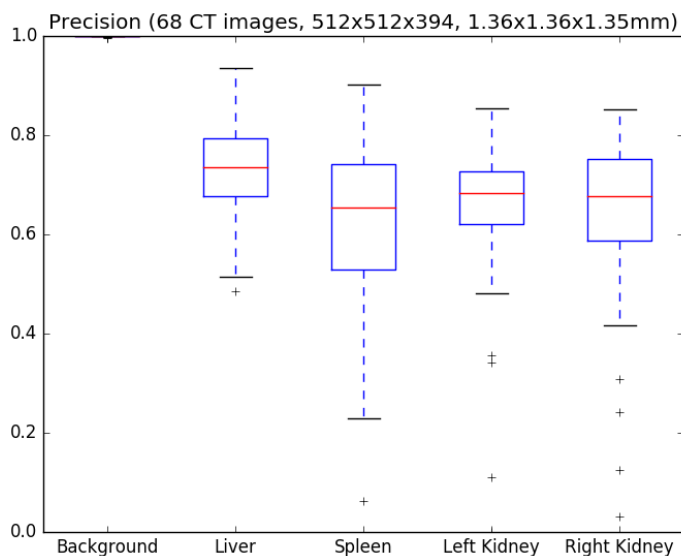


Figure 3.9: CT Abdominal Segmentation (CT Precision)

The precision metric indicates the portion of correctly labeled as positive voxels from all positively labeled voxels. The results show that the segmentation occasionally classifies voxels as organs when they in fact belong to the background (oversegmentation).

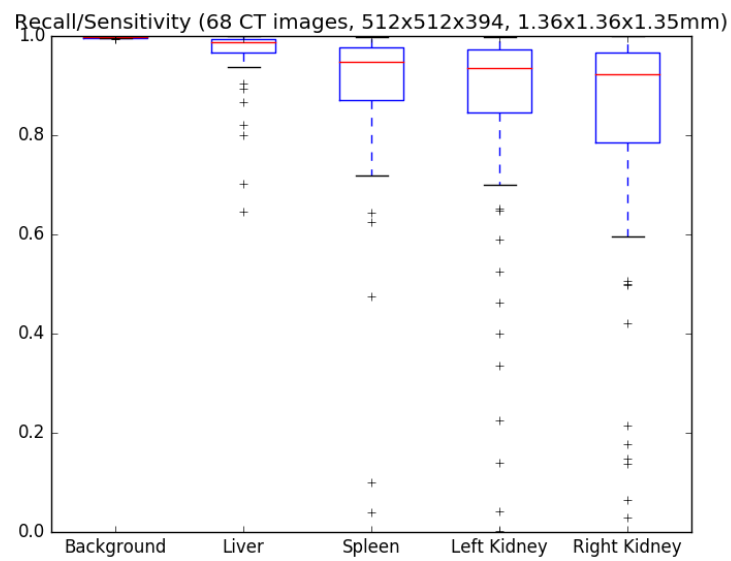


Figure 3.10: CT Abdominal Segmentation (CT Sensitivity)

Sensitivity or recall corresponds to the portion of positive voxels in the ground truth segmentation that are positively labeled by the algorithm. The distribution of the sensitivity measure indicates that in most cases all voxels belonging to organs are also discovered by the algorithm. This means that undersegmentation is a rather seldom event.

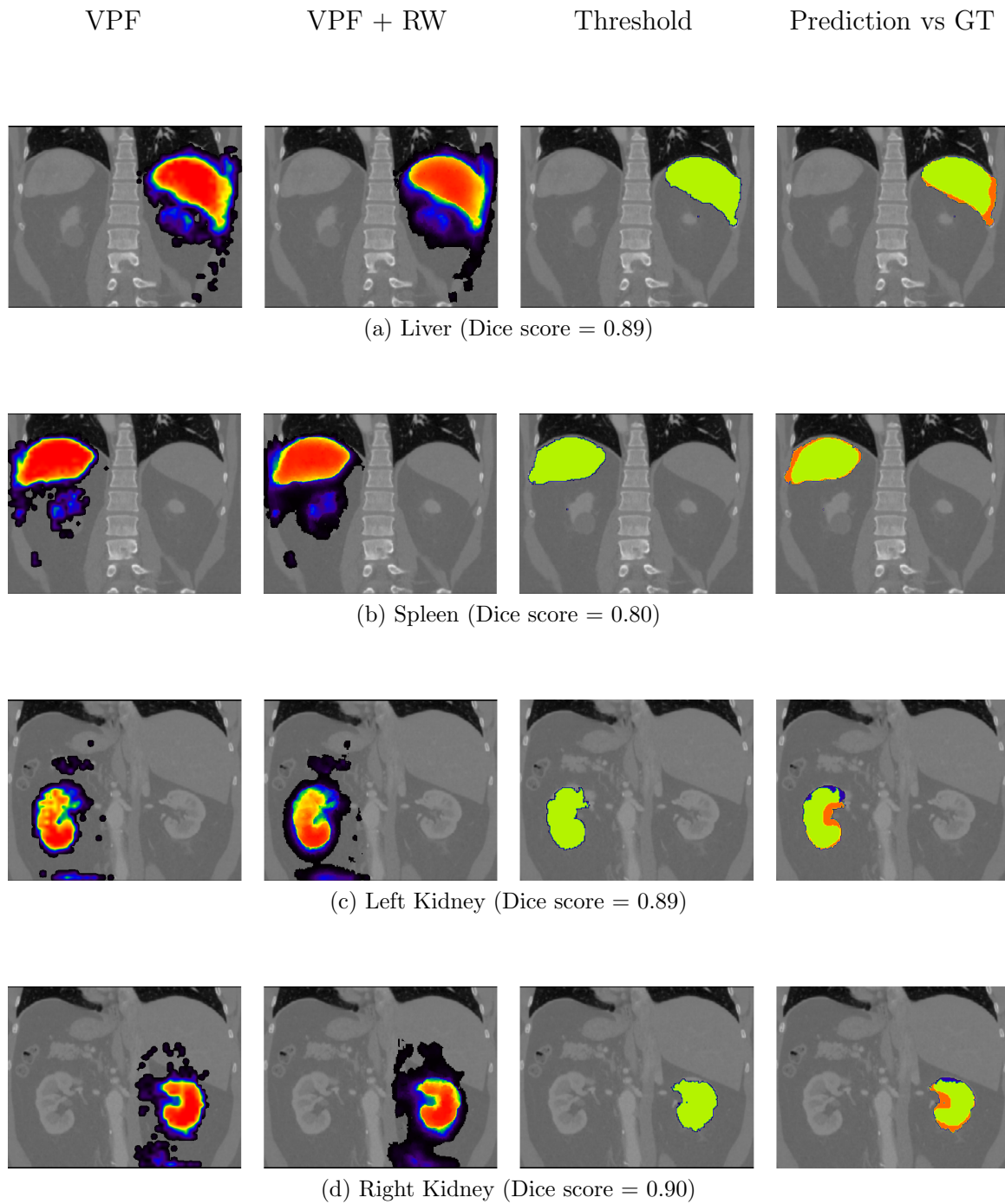


Figure 3.11: CT Abdominal Segmentation (Inlier)

Preview of a CT abdominal image segmentation. From left to right: heat map showing the label probability after interpolation of VPF results; heat map after RW regularization; final labels after label major voting; difference image with ground truth image: TP in green, FP in red, FN in blue.

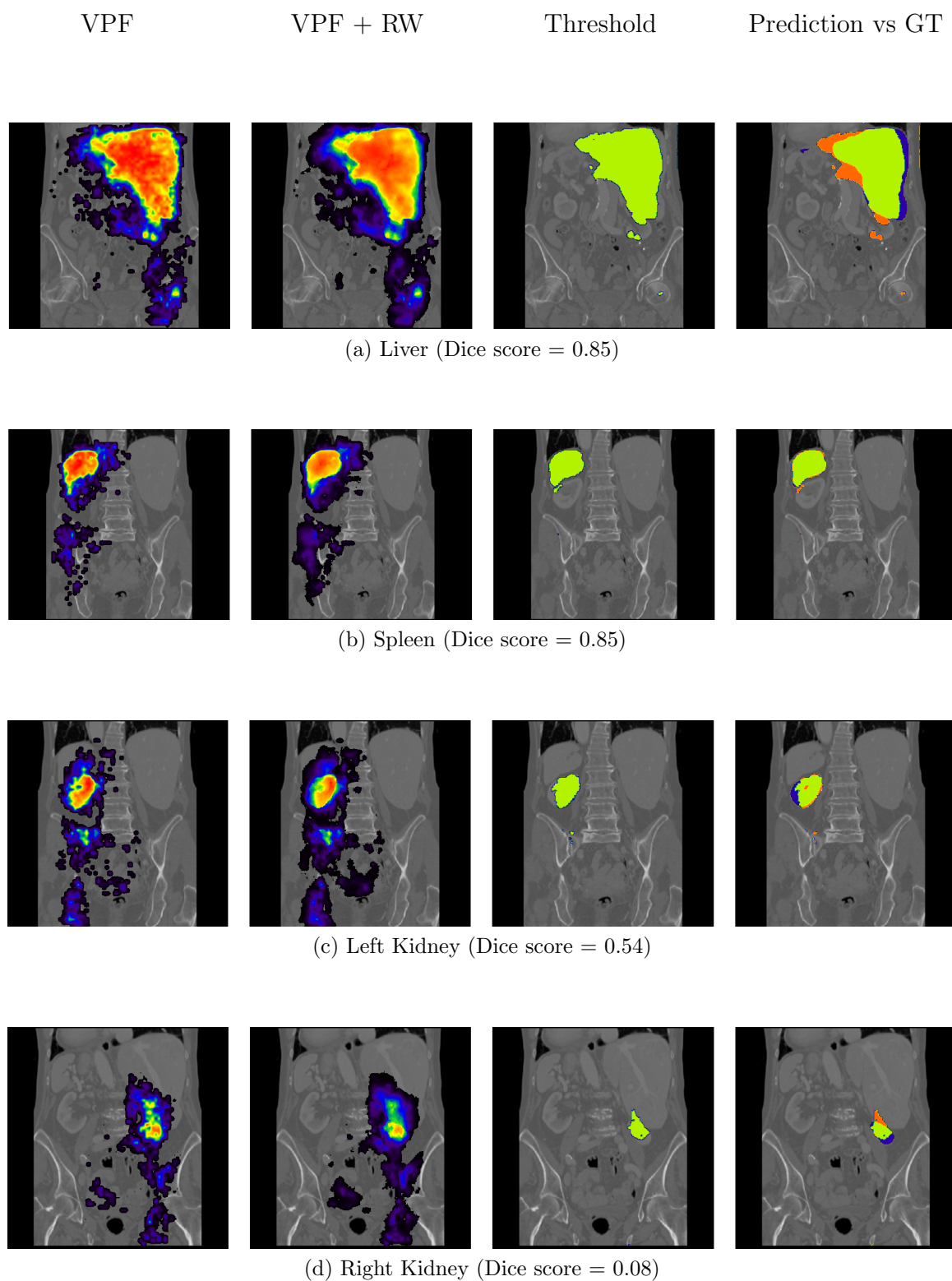


Figure 3.12: CT Abdominal Segmentation (Outlier)

Preview of a CT abdominal image segmentation. From left to right: heat map showing the label probability after interpolation of VPF results; heat map after RW regularization; final labels after label major voting; difference image with ground truth image: TP in green, FP in red, FN in blue.

### 3.4.2 MR Male Pelvis Data

Segmenting male pelvic organs is a prerequisite for prostate cancer radiotherapy planning. Delineation of the prostate, bladder and rectum in MR images is a difficult task as images often show several types of variabilities: field of view, variable clinical procedures, and inter-patient organ differences. Organ boundaries are often only partially visible and the organs vary greatly in size, shape and appearance, especially the bladder [Schadewaldt et al., 2013].

42 Magnetic Resonance (MR) inphase images with dense bladder, bone, prostate and rectum segmentation are used for a 7-fold cross validation. Each image consists of  $528 \times 528 \times 120 = 33\,454\,080$  voxels with a spacing of  $1.05 \times 1.05 \times 2.5$  mm. As the size of the voxels is not isotropic, the displacement distribution of the sampling points of the BRIEF/LBP patterns is adjusted and is  $\sim 2.5$  times smaller in z direction. The background training samples are again downweighted by a factor of 5 in order to have better balanced data.

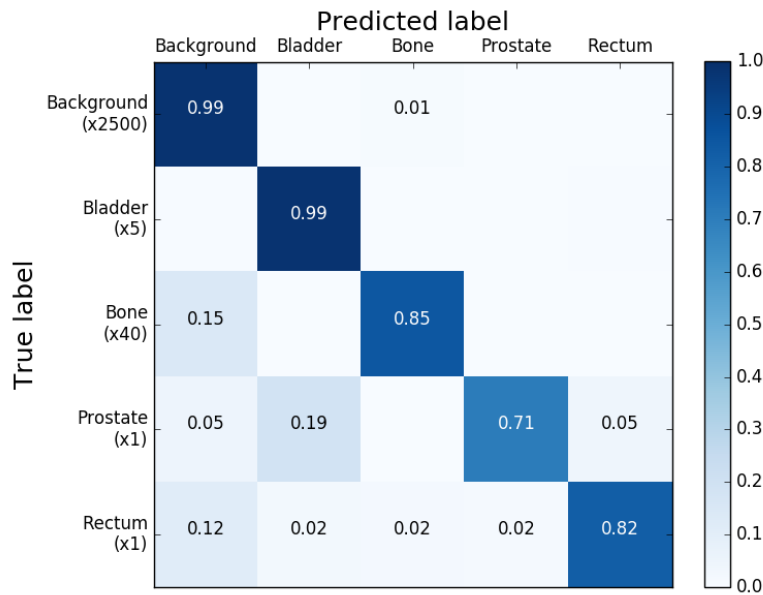


Figure 3.13: MR Pelvis Segmentation (Confusion Matrix)

Confusion matrix of the MR cross validation. The approximate amount of features compared to the organ with the least samples (prostate), is indicated below each class. The high values on the diagonal of the confusion matrix indicate that pelvic organs get correctly classified in most of the cases. The confusion with background can be explained by the prevalence of background training data ( $\sim 2500$  times more). It can also be seen that the algorithm has difficulties in recognizing the bladder and the prostate and occasionally confuses them with other organs.

Figure 3.13 presents the confusion matrix of the classification. The strong diagonal proves again that the algorithm is capable of correctly identifying organs. Occasionally,

however, it has difficulties in recognizing the prostate and the rectum, and is predicting wrong labels for these organs. Confusion with the background occurs as well due to the high prevalence of background training data.

Figure 3.14 summarizes the distribution of the Dice overlap measure for the different pelvic organs. Even though the segmentation of some images has a high Dice score, the delineation is not robust for all. The average achieved Dice scores are: bladder - 0.72, bones - 0.65, prostate - 0.59, rectum - 0.64. The overall dice score is 0.65.

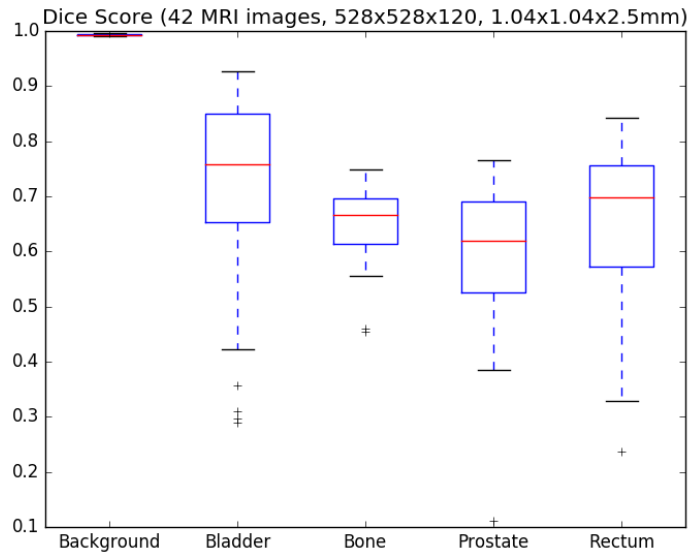


Figure 3.14: MR Pelvis Segmentation (Dice Score)

The wide spread of the Dice score distribution for bladder, prostate and rectum indicates that the algorithm achieves good segmentation for some of the images, but it is not robust.

Figure 3.15 and Figure 3.16 give an overview of the precision and the sensitivity achieved in the MR test scenario. The low precision values indicate frequent oversegmentation. The high sensitivity values with low spread show that undersegmentation is rarely happening. This can be also observed in Figure 3.17 where a preview of the segmentation of one pelvic image is shown. As in the CT example, for each organ the probabilities before and after random walker regularization are shown, the final segmentation and the difference to the ground truth image. The difference image adopts the same coloring scheme: TP voxels are colored in green, FP - in red and FN in blue.

An outlier case is shown in Figure 3.18. The segmentation method successfully identifies the locations of all organs. However, as the pelvic organs often vary in shape, size and appearance, the algorithm has difficulties in finding the exact boundaries. This scenario proves that the algorithm highly depends on the training data and would benefit from more diverse features.



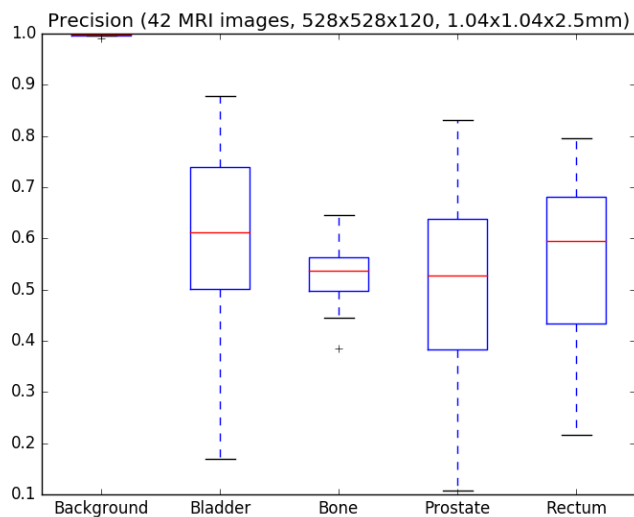


Figure 3.15: MR Pelvis Segmentation (Precision)

The low precision of the of the MR pelvis segmentation shows that the method often labels voxels positively, although they should be classified as background. Namely, oversegmentation occurs frequently.

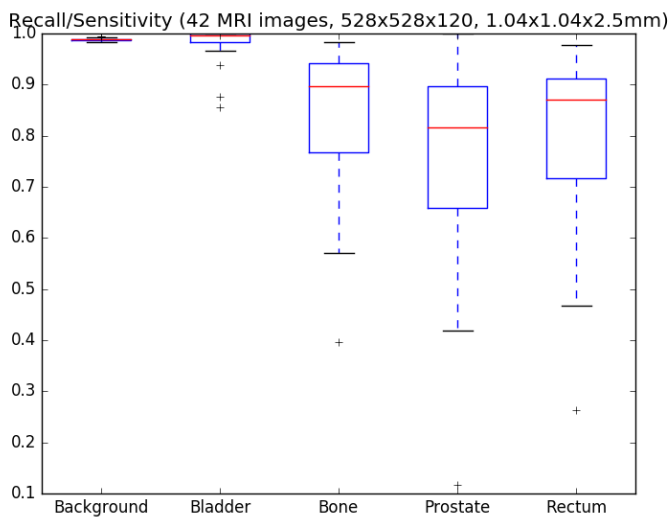


Figure 3.16: MR Pelvis Segmentation (Sensitivity)

The sensitivity distribution shows that portion of positive voxels in the *GT* that are positively labeled by the segmentation is high for all organs, especially the bladder. This indicates that almost no undersegmentation occurs.

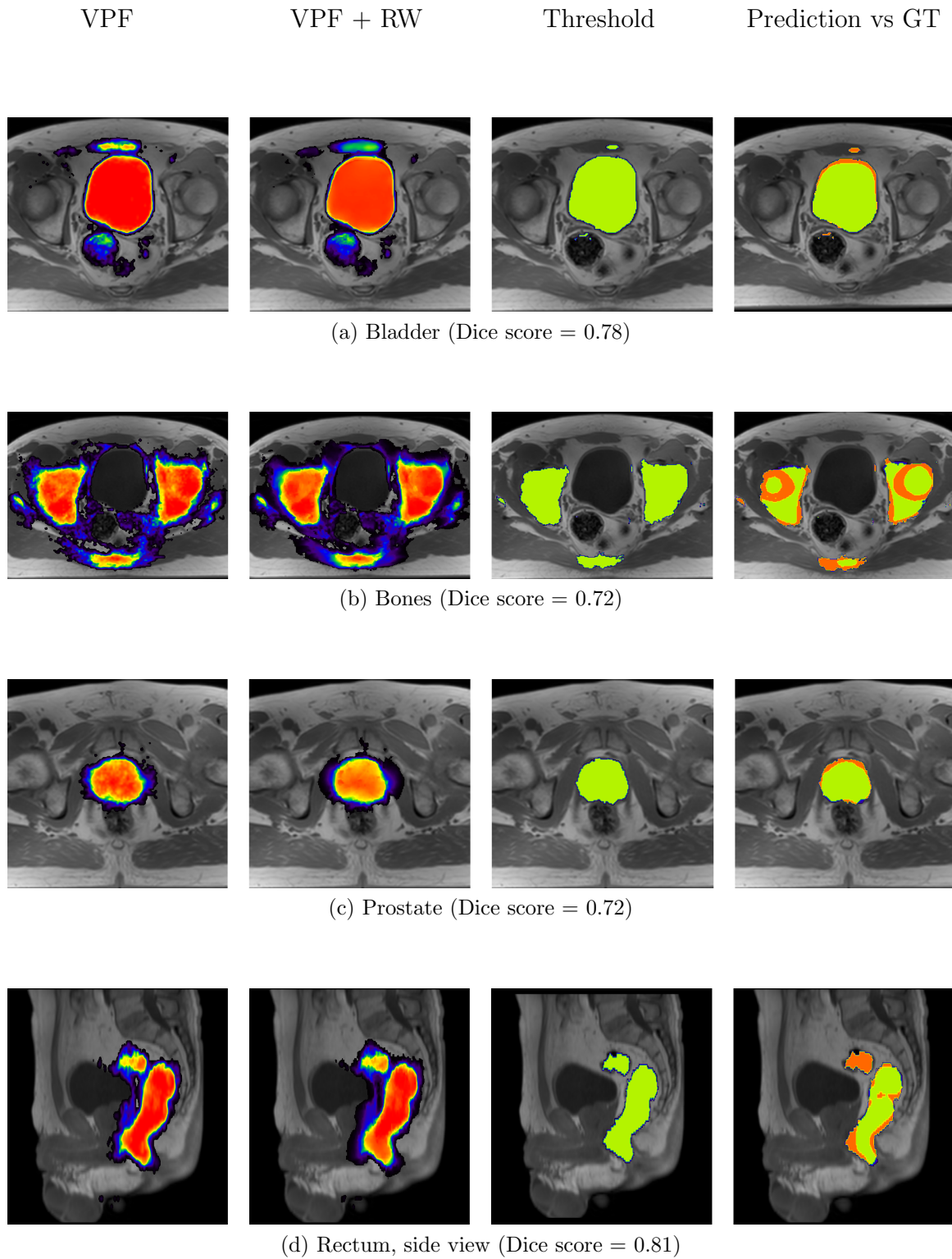


Figure 3.17: MR Pelvis Segmentation (Inlier)

Preview of a MR pelvis image segmentation. From left to right: heat map showing the label probability after interpolation of VPF results; heat map after RW regularization; final labels after label major voting; difference image with ground truth image: TP in green, FP in red, FN in blue.

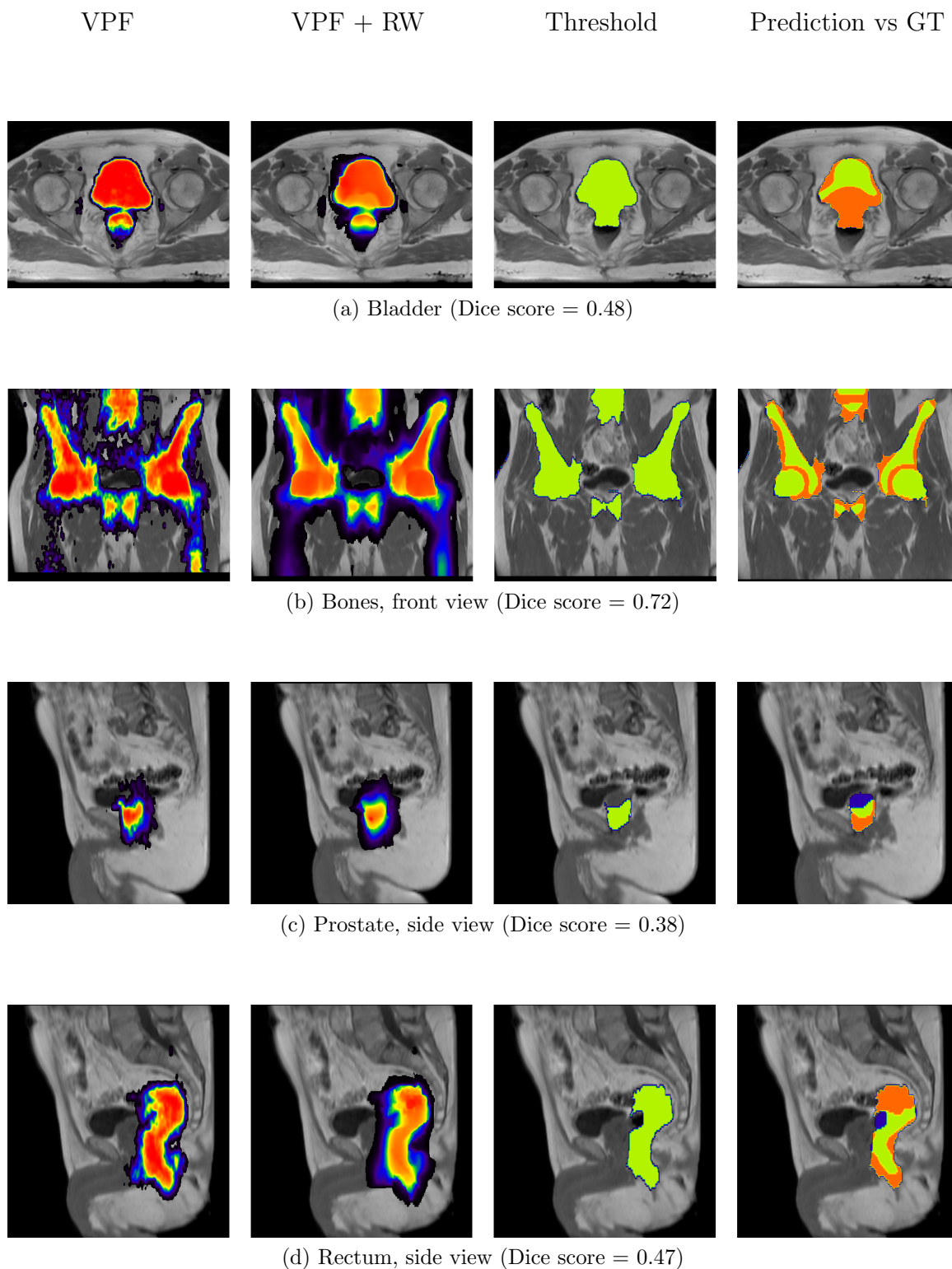


Figure 3.18: MR Pelvis Segmentation (Outlier)

Preview of a MR pelvis image segmentation. From left to right: heat map showing the label probability after interpolation of VPF results; heat map after RW regularization; final labels after label major voting; difference image with ground truth image: TP in green, FP in red, FN in blue.

### 3.5 Time and Storage Evaluation

The duration of the experiments differs with the amount and size of training and testing images. Furthermore, the length of each step can vary depending on the machine's parameters and computing power. To get a rough idea how long the steps take, the duration of the main operations and the required storage during 1 fold of the CT 4-fold cross-validation are listed in Table 3.2. It can be observed that extracting training data takes several minutes. The obtained features, however, can be stored and loaded whenever needed, instead of extracted for each test. Hence, when deployed less time would be needed. Then, the classification of test data takes only several seconds per file. The two steps that have the longest duration and highest memory consumption are the interpolation and the *RW* regularization of label probabilities. Approximately a minute for each of two steps is needed per test case.

From the overview of the time and storage required for a cross validation, it can be concluded that the segmentation procedure is feasible even by a standard computer. The segmentation, however, would benefit from machines with more CPUs and working memory. Furthermore, speed would also improve when extracted training features and labels are stored on disk and loaded whenever necessary, instead of extracted from images each time. Nevertheless, the current implementation meets the constraints of a typical clinical setting. Its duration and memory footprint are competitive with existing methods.

Procedure	Duration	Memory consumption
Loading $3/4 \times 68 = 51$ training images, extracting $\sim 540\,000$ training features, each 1280-bit long	18 min ( $\sim 21$ sec / file)	86 MB ( $\sim 1.68$ MB / file)
Loading $1/4 \times 68 = 17$ test images, extracting $\sim 3\,100\,000$ test features, each 1280-bit long (less images, but more features due to denser grids)	4.5 min ( $\sim 16$ sec / file)	$17 \times 3$ (test + mask + label) images $\times (512 \times 512 \times 394)$ voxels $\times 16$ bits (int) = 10.5 GB images ( $\sim 0.62$ GB per file) 496 MB features ( $\sim 30$ MB per file)
Construct a VPF with $\sim 540\,000$ training samples, query $\sim 3\,100\,000$	$\sim 2$ min	
Retrieve labels of NNs	2 min ( $\sim 7$ sec / file)	$20$ NNs $\times 3\,100\,000$ tr. samples $\times 8$ bits (char) = $\sim 62$ MB
Interpolation of label probabilities	17 min ( $\sim 1$ min/ file)	$17$ files $\times (512 \times 512 \times 394)$ voxels $\times 5$ labels $\times 32$ bits (float) = $\sim 35$ GB ( $\sim 2$ GB per / file)
RW Regularization	17 min ( $\sim 1$ min/ file)	$17$ files $\times (512 \times 512 \times 394)$ voxels $\times 5$ labels $\times 32$ bits (float) = $\sim 35$ GB ( $\sim 2$ GB per / file)

Table 3.2: Duration and Storage Requirements

The duration of the experiments is dependent on the computing power of the machine used. The approximate time of the main steps of one round of a 4-fold cross validation and the required storage per procedure are presented. Training the classifier takes several minutes, while testing a single image is in the order of seconds.

## 4. Conclusion and Outlook

A fully automatic method for multi-organ segmentation of 3D medical image data was developed. After being trained for several minutes, the classifier is able to segment test images within only a few seconds each. The classification itself, as presented, is *simply* based on the nearest neighbor search algorithm and the similarity of binary features extracted from the data.

The algorithm was successfully applied to images from two different modalities (MR and CT) and several organs, justifying its generality. The method can also be easily extended to other modalities and organs without further modifications. Satisfactory results were obtained for segmentation of liver, spleen, kidneys (CT) and bladder (MR). For the delineation of prostate, rectum and bones (MR), however, there is still room for improvement.

There are several concepts that can further enhance the proposed method.

First, the robustness of the extracted features could be improved. Instead of using the signs of the intensity comparisons, a minimum threshold value of difference can be set. Furthermore, the sampling patterns for feature extraction can be adapted in a way that the actual image parameters are considered. Alternative to providing displacement radius based on voxel numbers, the patterns shall be specific to the real size extent. Moreover, in the segmentation of MR data, the Dixon technique could be integrated so that features are extracted from 4 different contrasts (in phase, opposed phase, water and fat images) and the anatomical structures are better characterized.

Second, the classification could benefit from better balancing of the training data. Equal or comparative amount of features from the different classes shall lead to less confusion between organs.

Third, denser sampling of the training and testing data might improve the accuracy of the algorithm. This, however, would take significantly longer to process unless more powerful machines or parallel computing are available.

Forth, if the segmentation has a certain objective, other than a high dice score or no confusion between organs, a cost function specific to the task can be "invented". For instance, for a given procedure oversegmentation might not have any adverse effects, but undersegmentation should not happen at any time. Then the algorithm parameters can be tailored to minimize the value of that cost function.

Last, the method can be further improved by using cascaded classification as suggested by [Heinrich and Blendowski \[2016\]](#).

Nevertheless, the Vantage Point Forest algorithm with Binary Context Features was justified as a very simple, data-efficient generic approach the yield surprisingly accurate results and thus, has a potential in clinical usage.

## A. Appendix



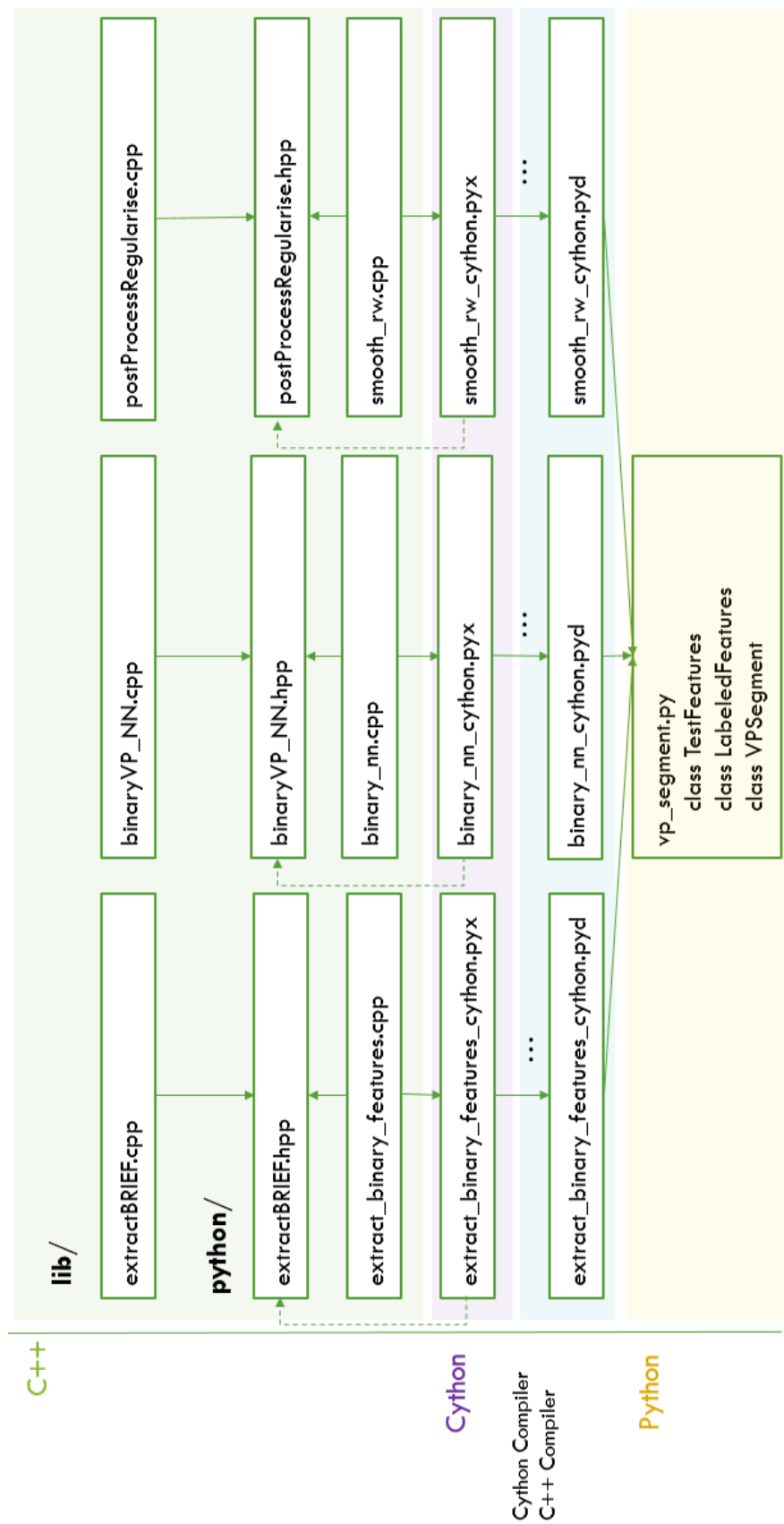


Figure A.1: Segmentation Code Structure

The diagram presents the segmentation code structure. It is organized into three main components: feature extraction, NN classification and post processing. The C++ code in the `lib/` directory is the source provided by Heinrich and Blendowski [2016]<sup>a</sup>. It is wrapped with the help of Cython<sup>b</sup> and used in Python environment.

<sup>a</sup>Code is publically available at [http://mphheinrich.de/code/vpForest\\_public.zip](http://mphheinrich.de/code/vpForest_public.zip)

<sup>b</sup>Cython is a superset of the Python language that supports calling C/C++ functions and declaring C types on variables and attributes. This approach results in a very efficient C/C++ code that can be accessed from Python environment.



# Bibliography

- Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041. (cited on Page 5)
- Allahviranloo, T. (2005). Successive over relaxation iterative method for fuzzy system of linear equations. *Applied Mathematics and Computation*, 162(1):189–196. (cited on Page 16)
- Baudin, P.-Y., Azzabou, N., Carlier, P. G., and Paragios, N. (2012). Prior knowledge, random walks and human skeletal muscle segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 569–576. Springer. (cited on Page 16)
- Boykov, Y. Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE. (cited on Page 13)
- Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer. (cited on Page 5)
- Campadelli, P., Casiraghi, E., and Esposito, A. (2009). Liver segmentation from computed tomography scans: A survey and a new algorithm. *Artificial intelligence in medicine*, 45(2):185–196. (cited on Page )
- Chan, T. F. and Vese, L. A. (2001). Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277. (cited on Page 1)
- Crum, W. R., Camara, O., and Hill, D. L. (2006). Generalized overlap measures for evaluation and validation in medical image analysis. *IEEE transactions on medical imaging*, 25(11):1451–1461. (cited on Page 21)
- Deserno, T. M., Handels, H., Meinzer, H.-P., and Tolxdorff, T. (2014). *Bildverarbeitung für die Medizin 2014: Algorithmen-Systeme-Anwendungen Proceedings des Workshops vom 16. bis 18. März 2014 in Aachen*. Springer-Verlag. (cited on Page 4)

- Despotović, I., Goossens, B., and Philips, W. (2015). Mri segmentation of the human brain: challenges, methods, and applications. *Computational and mathematical methods in medicine*, 2015. (cited on Page 1)
- Dong, C., Chen, Y.-W., Lin, L., Hu, H., Jin, C., Yu, H., Han, X.-H., and Tateyama, T. (2016). Simultaneous segmentation of multiple organs using random walks. *Journal of Information Processing*, 24(2):320–329. (cited on Page 14 and 15)
- Dukehart, S. P. (2009). *GPU Random Walkers for Iterative Image Segmentation*. ProQuest. (cited on Page 14)
- Forsyth, D., Torr, P., and Zisserman, A. (2008). *Computer Vision-ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings*, volume 5305. Springer. (cited on Page )
- Franz, A., Schadevaldt, N., Schulz, H., Vik, T., Bergtholdt, M., and Bystrov, D. (2016). Precise anatomy localization in ct data by an improved probabilistic tissue type atlas. In *SPIE Medical Imaging*, pages 978444–978444. International Society for Optics and Photonics. (cited on Page )
- Fu, A. W.-c., Chan, P. M.-s., Cheung, Y.-L., and Moon, Y. S. (2000). Dynamic vp-tree indexing for n-nearest neighbor search given pair-wise distances. *The VLDB Journal—The International Journal on Very Large Data Bases*, 9(2):154–173. (cited on Page )
- Grady, L. (2005). Multilabel random walker image segmentation using prior models. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 763–770. IEEE. (cited on Page 13 and 18)
- Grady, L. (2006). Random walks for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1768–1783. (cited on Page 13 and 14)
- Grady, L. and Funka-Lea, G. (2004). Multi-label image segmentation for medical applications based on graph-theoretic electrical potentials. In *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis*, pages 230–245. Springer. (cited on Page 13 and 14)
- Grau, V., Mewes, A., Alcaniz, M., Kikinis, R., and Warfield, S. K. (2004). Improved watershed transform for medical image segmentation using prior information. *IEEE transactions on medical imaging*, 23(4):447–458. (cited on Page 1 and 13)
- Hall, L. O., Bensaid, A. M., Clarke, L. P., Velthuizen, R. P., Silbiger, M. S., and Bezdek, J. C. (1992). A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain. *IEEE transactions on neural networks*, 3(5):672–682. (cited on Page 7)

- Heinrich, M. P. and Blendowski, M. (2016). Multi-organ segmentation using vantage point forests and binary context features. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 598–606. Springer. (cited on Page 2, 13, 47, and 49)
- Hu, P., Wu, F., Peng, J., Bao, Y., Chen, F., and Kong, D. (2016). Automatic abdominal multi-organ segmentation using deep convolutional neural network and time-implicit level sets. *International Journal of Computer Assisted Radiology and Surgery*, pages 1–13. (cited on Page 1, 7, and 33)
- Jabbour, S. K., Hashem, S. A., Bosch, W., Kim, T. K., Finkelstein, S. E., Anderson, B. M., Ben-Josef, E., Crane, C. H., Goodman, K. A., Haddock, M. G., et al. (2014). Upper abdominal normal organ contouring guidelines and atlas: a radiation therapy oncology group consensus. *Practical radiation oncology*, 4(2):82–89. (cited on Page )
- Kashif, M., Deserno, T. M., Haak, D., and Jonas, S. (2016). Feature description with sift, surf, brief, brisk, or freak? a general question answered for bone age assessment. *Computers in biology and medicine*, 68:67–75. (cited on Page 5 and 6)
- Khalifa, F., Soliman, A., Elmaghraby, A., Gimel'farb, G., and El-Baz, A. (2017). 3d kidney segmentation from abdominal images using spatial-appearance models. *Computational and Mathematical Methods in Medicine*. (cited on Page )
- Kumar, N., Zhang, L., and Nayar, S. (2008). What is a good nearest neighbors algorithm for finding similar patches in images? In *European conference on computer vision*, pages 364–378. Springer. (cited on Page 9 and 10)
- Muja, M. and Lowe, D. G. (2012). Fast matching of binary features. In *Computer and Robot Vision (CRV), 2012 Ninth Conference on*, pages 404–410. IEEE. (cited on Page )
- Muja, M. and Lowe, D. G. (2014). Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240. (cited on Page )
- Pattanayak, P., Turkbey, E. B., and Summers, R. M. (2017). Comparative evaluation of three software packages for liver and spleen segmentation and volumetry. *Academic Radiology*. (cited on Page )
- Pauly, O., Glocker, B., Criminisi, A., Mateus, D., Möller, A. M., Nekolla, S., and Navab, N. (2011). Fast multiple organ detection and localization in whole-body mr dixon sequences. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 239–247. Springer. (cited on Page )
- Persson, A. and Loutfi, A. (2016). Fast matching of binary descriptors for large-scale applications in robot vision. *International Journal of Advanced Robotic Systems*, 13(2):58. (cited on Page 4, 5, and 6)

- Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE. (cited on Page )
- Schadewaldt, N., Bystrov, D., Vik, T., Schulz, H., Peters, J., Franz, A., Buerger, C., and Bzdusek, K. (2013). Robust initialization of multi-organ shape models. In *MIC-CAI Challenge Workshop on Segmentation: Algorithms, Theory and Applications*, volume 1, page 3. (cited on Page 39)
- Sinop, A. K. and Grady, L. (2007). A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE. (cited on Page 13)
- Taha, A. A. and Hanbury, A. (2015). Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool. *BMC medical imaging*, 15(1):29. (cited on Page 18)
- Whitfield, G. A., Price, P., Price, G. J., and Moore, C. J. (2013). Automated delineation of radiotherapy volumes: are we going in the right direction? *The British journal of radiology*, 86(1021):20110718–20110718. (cited on Page )
- Wolpert, D. H. and Macready, W. G. (2005). Coevolutionary free lunches. *IEEE Transactions on Evolutionary Computation*, 9(6):721–735. (cited on Page 7)
- Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, volume 93, pages 311–21. (cited on Page 9)
- Zhang, J., Ma, K.-K., Er, M.-H., and Chong, V. (2004). Tumor segmentation from magnetic resonance imaging by learning via one-class support vector machine. In *International Workshop on Advanced Image Technology (IWAIT'04)*, pages 207–211. (cited on Page 7)
- Zhao, F. and Xie, X. (2013). An overview of interactive medical image segmentation. *Annals of the BMVA*, 2013(7):1–22. (cited on Page 1)
- Zikic, D., Glocker, B., and Criminisi, A. (2014). Encoding atlases by randomized classification forests for efficient multi-atlas label propagation. *Medical image analysis*, 18(8):1262–1273. (cited on Page )

---

I declare that this Master's thesis has been completed by myself independently without outside help and only the defined sources and study aids were used. Sections that reflect the thoughts or works of others are made known through the definition of sources.

Hamburg, 27.04.2017