FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Visual Feature Exploration for ssTEM Image Segmentation

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Ivan Maricic
Matrikelnummer 0130041

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Dipl.-Inf. Hendrik Schulze

Wien, 27. August 2013

(Unterschrift Verfasser/in) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Visual Feature Exploration for ssTEM Image Segmentation

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Visual Computing

by

Ivan Maricic
Registration Number 0130041

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Dipl.-Inf. Hendrik Schulze

Vienna, August 27, 2013 _____ _____
(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Ivan Maricic
Laxenburgerstr. 138/2/8, 2331 Vösendorf

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser/in)

Abstract

In order to (preferably) automatically derive the neuronal structures from brain tissue image stacks, the research field computational neuroanatomy relies on computer assisted techniques such as visualization, machine learning and analysis. The image acquisition is based on the so-called *transmission electron microscopy* (TEM) that allows resolution which is high enough to identify relevant structures in brain tissue images (less than 5 nm per pixel). In order to get to an image stack (or volume) the tissue samples are sliced (or sectioned) with a diamond knife in slices of 40 nm thickness. This approach is called *serial-section transmission electron microscopy* (ssTEM). The manual segmentation of these high-resolution, low-contrast and artifact afflicted images would be impracticable alone due to the high resolution of 200,000 images of size $2,000,000 \times 2,000,000$ pixel in a cubic centimeter tissue sample. But, the automatic segmentation is error-prone due to the small pixel value range (8 bits per pixel) and diverse artifacts resulting from mechanical sectioning of tissue samples. Additionally, the biological samples in general contain densely packed structures which leads to a non-uniform background that introduces artifacts as well. Therefore, it is important to quantify, visualize and reproduce the automatic segmentation results interactively with as few user interactions as possible.

This thesis is based on the *membrane* segmentation proposed by Kaynig-Fittkau [2011] which for ssTEM brain tissue images outputs two results: (a) a certainty value per pixel (with regard to the analytical model of the user selection of cell membrane pixels) which states how certain the underlying statistical model is that the pixel is belonging to the *membrane*, and (b) after an optimization step the resulting edges which represent the *membrane*. In this work we present a visualization-assisted method to explore the parameters of the segmentation. The aim is to interactively mark those regions where the segmentation fails in order to structure the post- or re-segmentation and to prove-read the segmentation results. This is achieved by weighting the membrane pixels by the uncertainty values resulting from the segmentation process.

We start here and employ user knowledge once more to decide *which* data and in *what form* should be introduced to the random forest classifier. The aim here is to improve the segmentation results by either improving the segmentation quality, by increasing the segmentation speed or by reducing the memory consumption for the segmentation. In this regard we focus our attention especially on the visualizations of the uncertainty, the errors and the multi-modal data. The interaction techniques are explicitly used in those cases where we expect the highest gain at the end of the exploration. We show the effectiveness of the proposed methods using the

freely available ssTEM brain tissue dataset of the drosophila fly. Because we lack the expert knowledge in the field of neuroanatomy we base our assumptions and methods on the underlying ground truth segmentations of the drosophila fly brain tissue dataset.

In this work we carry out five experiments with six feature sets and three training sets for the segmentation of membranes. The experiments indicate that creating new features with so-called aspect windows help improve the prediction performance through lower prediction error and higher precision. Furthermore, the visualization-assisted feature exploration presented in this work leads to a reduction of the original feature set of 42 % which results in slightly higher (0.07 %) prediction error. The reduction of the feature set also leads to shorter processing times and to lower memory space requirements which is necessary especially for large ssTEM brain tissue images.

Kurzfassung

Der Forschungsbereich Computational Neuroanatomy verwendet rechnergestützte Techniken wie Visualisierung, Modellierung und Analyse, um die neuronalen Strukturen von Gehirnproben aus Bild-Stapeln (semi-)automatisch abzuleiten. Für die Gewinnung der Bilder, in der relevante Strukturen hoch genug aufgelöst sind, verwendet man die sogenannte *Transmissionselektronenmikroskopie* (TEM), die eine Pixelauflösung von unter 5 nm ermöglicht. Um einen Bild-Stapel oder ein Volumen zu erhalten, wird die Probe mit einem Diamantenmesser in 40 nm dicke Slices geschnitten, dies bezeichnet man als *serial-section Transmissionselektronenmikrospie* (ssTEM). Die manuelle Segmentierung dieser hochaufgelösten aber kontrastarmen und mit Aufnahmefehlern behafteten Bilder wäre allein wegen der hohen Auflösung der 200.000 Bilder mit $2.000.000 \times 2.000.000$ Pixel pro Bild in einem Kubikzentimeter Gehirnprobendatensatz inpraktikabel. Da die Pixel einen geringen Wertebereich (8 Bits pro Pixel) haben und verschiedenste Aufnahmeartefakte beinhalten, ist eine vollautomatische Segmentierung fehleranfällig. Erschwerend kommt noch hinzu, dass biologische Proben dichtgepackt mit Strukturen sind, was in einem nicht-uniformen Hintergrund resultiert und zu weiteren Artefakten führt. Deshalb ist es wichtig, die Ergebnisse aufwendiger automatischer Segmentierungsverfahren zu quantifizieren, zu visualisieren und interaktiv mit möglichst wenig Benutzer-Interaktionen reproduzieren zu können.

Diese Arbeit basiert auf dem *Membran*-Segmentierungsverfahren von Kaynig-Fittkau [2011], welches für ssTEM Hirngewebe-Bilder unter anderem folgende Ausgaben liefert: (a) ein Zuversichtswert für jeden Pixel (bezüglich des analytischen Modelles der Benutzerwahl der Zellmembranen) der angibt, wie sicher sich das zugrundeliegende statistische Modell ist, dass das Pixel zur *Membran* gehört und (b) die nach der Optimierung erhaltenen Kanten, welche die Zellmembranen repräsentieren. In dieser Arbeit präsentieren wir eine visualisierungsgestützte Methode, die Parameter der Segmentierung zu explorieren. Wir möchten dem Benutzer, der eine Nachsegmentierung oder ein Korrekturlesen der Segmentierungsergebnisse vornehmen möchte, interaktiv Stellen der Unsicherheit anzeigen, indem wir jede Kante mit den Wahrscheinlichkeiten der zur Kante gehörenden Pixel gewichten und die Stellen der Unsicherheit visualisieren.

Wir setzen genau hier an und verwenden Benutzerwissen ein weiteres Mal, um entscheiden zu können, *welche* und in *welcher Form* man Daten dem Random-Forest-Klassifizierer zur Verfügung stellen muss. Das Ziel dabei ist es, die Segmentierungsergebnisse zu verbessern, in dem man die Qualität der Segmentierung verbessert, die Segmentierung beschleunigt oder den Speicheraufwand reduziert. In diesem Zusammenhang gehen wir insbesondere auf die Visualiserung von Un-

sicherheiten, Fehlern und multimodalen Daten ein. Die Interaktionstechniken werden gezielt dort eingesetzt, wo wir den höchsten Gewinn am Ende der Exploration erwarten. Wir überprüfen unsere Erkenntnisse dann mit einem frei zur Verfügung stehendem Beispiel, nämlich anhand eines Drosophila-Larven Datensatzes. Da uns das Expertenwissen aus dem Bereich der Neuroanatomie fehlt, stützen wir unsere Annahmen und Methoden auf den zugundeliegenden vorsegmentierten Beispielen.

In dieser Arbeit führen wir fünf Experimente mit sechs Merkmalsmengen und drei Trainingsmengen durch, die für die Segmentierung von Membranen verwendet werden. Die Experimente zeigen, dass das Erzeugen von neuen Merkmalen mithilfe von sogenannten Aspektfenstern die Vorhersage, durch geringere Vorhersagefehler und höhere Präzision, verbessert. Des Weiteren führt die in dieser Arbeit vorgestellte visualisierungsgestützte Exploration der Merkmalsmengen zu einer Reduktion der ursprünglichen Merkmalsmenge von 42 %, das in einem gering höheren (0.07 %) Vorhersagefehler resultiert. Die Reduktion der Merkmalsmenge führt auch zu kürzeren Rechenzeiten wie zur geringeren Speicherplatz Anforderungen, was insbesondere für große ssTEM Hirngewebe-Bilder notwendig ist.

Acknowledgements

I would not have had the ability to take the line of studying for this very labour-intensive and in any way demanding studies at the Vienna University of Technology if it were not for my parents Zdenka and Dr. med. Vladimir Maričić. I would like to thank them not only for the possibility to study. They are those kind of parents which were incredibly proud of any achievement of their children, my brother Tomislav Maričić MSc and I. At the same time they worried about any difficulties my brother and I had to master. They fled from our homeland to Austria where they regained formal recognitions of their scholastic diploma and academic titles even though not speaking a single word of German at the beginning. It is not only that they never gave up. They even gave up their short-term gains for the education of their children. In this regard I have not forgotten the words my father told me after we settled in Austria: *Son, in this country I can neither pass on any property, nor a barnyard. You have no other choice but to learn!*.

However, we would not have succeed in Austria if it where not for our friends. The most special one, especially with regard to my studies, is Aloisia Pötz which has been and is supporting me before, during and after the years of study at the Vienna University of Technology sometimes even over her own possibilities.

For these and many other reasons I say *Thank You*!

Danksagung

Ich hätte nicht die Möglichkeit gehabt, dieses sehr arbeitsintensive und in jeder Hinsicht sehr fordernde Studium an der Technischen Universität Wien abzuschließen, wären nicht meine Mutter Zdenka und mein Vater Dr. med. Vladimir Maričić gewesen. Ich möchte ihnen nicht nur für die Möglichkeit zum Studium danken. Sie gehören zu jenen Eltern, die überschwänglich stolz auf jeden Erfolg ihrer Kinder, Dipl.-Ing. Tomislav Maričić und ich, waren und sind. Gleichzeitig machten sie sich über jede Schwierigkeit, die mein Bruder und ich bewältigen mussten, Sorgen. Sie flohen aus unserem Heimatlad nach Österreich wo sie ihre Hochschulabschlüsse nostrifizierten, auch wenn sie anfangs kein Wort Deutsch sprachen. Es ist nicht nur, dass sie nie aufgaben. Sie ordneten ihre persönlichen Bedürfnisse stets der akademischen Ausbildung ihrer Kinder unter. In diesem Zusammenhang habe ich die Worte meines Vaters, als wir in Österreich aufgenommen wurden, nicht vergessen: *Sohn, ich kann dir in diesem Land weder Besitz noch einen Hof hinterlassen. Dir bleibt nichts anderes übrig als zu lernen!*.

Jedoch wären wir in Österreich nicht erfolgreich gewesen, hätten wir nicht auch Freunde gehabt. Die Außergewöhnlichste, insbesondere für mein Studium, ist Aloisia Pötz, die mich zuvor, während und auch nach den Studienjahren an der Technischen Universität Wien unterstützte und immer noch unterstützt, machmal sogar über ihre Verhältnisse.

Aus diesen und vielen anderen Gründen sage ich *Danke*!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Example is the school of mankind, and they will learn at no other."

Letters on a Regicide Peace
Edmund Burke

Image understanding is a relatively new research topic in the field of digital image processing and pattern recognition. The idea is to use computational methods (a) to pre-process images, (b) to recognize logical structures in images, (c) to extrapolate object movements, and (d) to summarize the contents of an image in order to understand the overall scene. The prerequisite for the most image understanding tasks is to identify *logically coherent* objects in an image, either automatically or semi-automatically. Logical coherent objects are objects which make "sense" to the domain expert but not necessarily to the software system. This process, also known as *image segmentation*, is guided by a human expert user who knows which pixels are belonging to a bigger structure and therefore to a logically coherent object. The problem here is that it is a non-trivial task to recognize image objects and to represent user knowledge of what an object might look like.

IMAGE SEGMENTATION

1.1 Short Introduction to Neuroanatomy

In the field of *neuroanatomy* the focus lies on understanding the central nervous system which basically consists of two main cell types: the *neurons* and the *neuroglia*. While the former cells receive impulses (through *dentrites*), conduct them and send them (through *axons*) to other cells (e.g., other neurons or muscle cells), the latter ensures the proper functionality of the neurons (Patestas and Gartner [2006]). The dentrites and the axons of a neuron are also called *neuronal processes*. Each neuron communicates with other neurons and other target cells through the so called *synapses* which are located at the *axon terminal*, as illustrated in Figure 1.1. There are two types of synapses: An *electrical synapse* exchanges ions via gap junctions, where the gap (also called the synaptic cleft) width is about 2-4 nm (Patestas and Gartner [2006]). A *chemical synapse* having a synaptic cleft of 20-30 nm exchanges messenger molecules which are contained in synaptic *vesicles*. These molecules are then released into the synaptic cleft and registered by the

NEURO-ANATOMY

NEURONAL PROCESSES

SYNAPSES

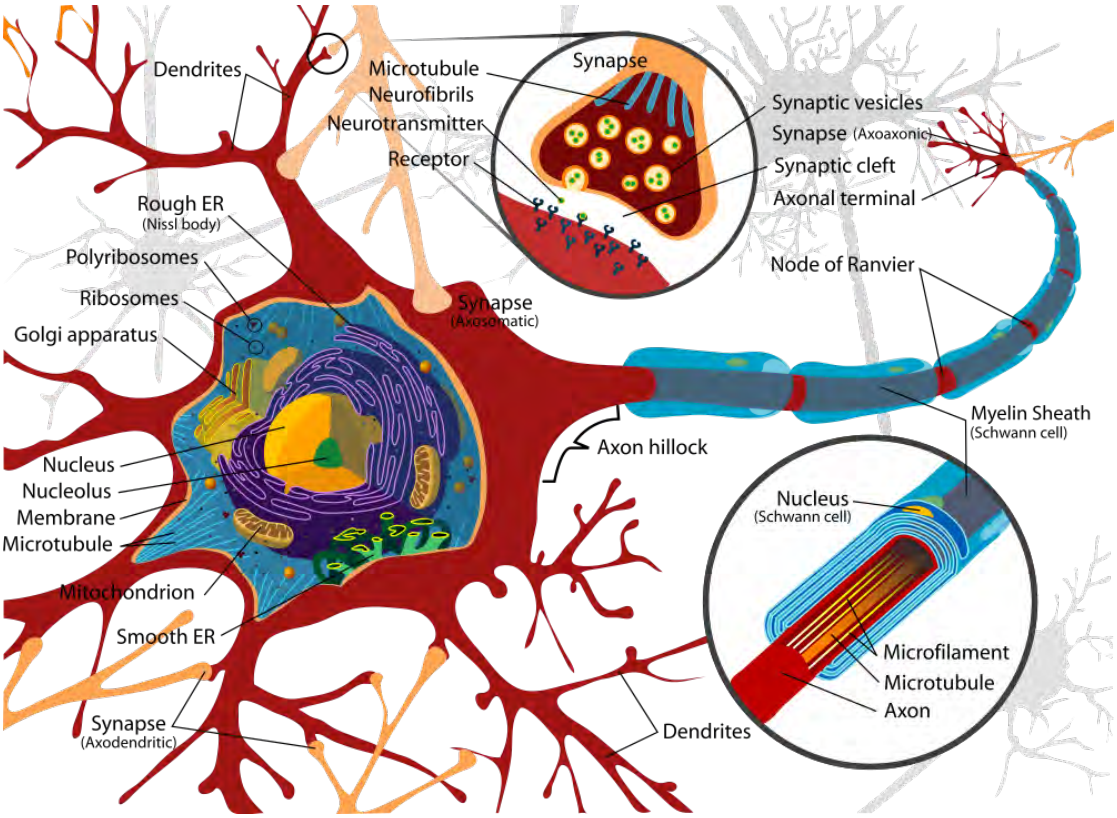


Figure 1.1: Illustration of a neuron cell with neuronal processes (© 2007 by Mariana Ruiz Villarreal http://commons.wikimedia.org).

receptors on the other side of the cleft. Furthermore, it is important to know that some axons have a fatty substance called *myelin sheath* which protects its internals from the surrounding tissue. All neuronal processes contain mitochondria and vesicles.

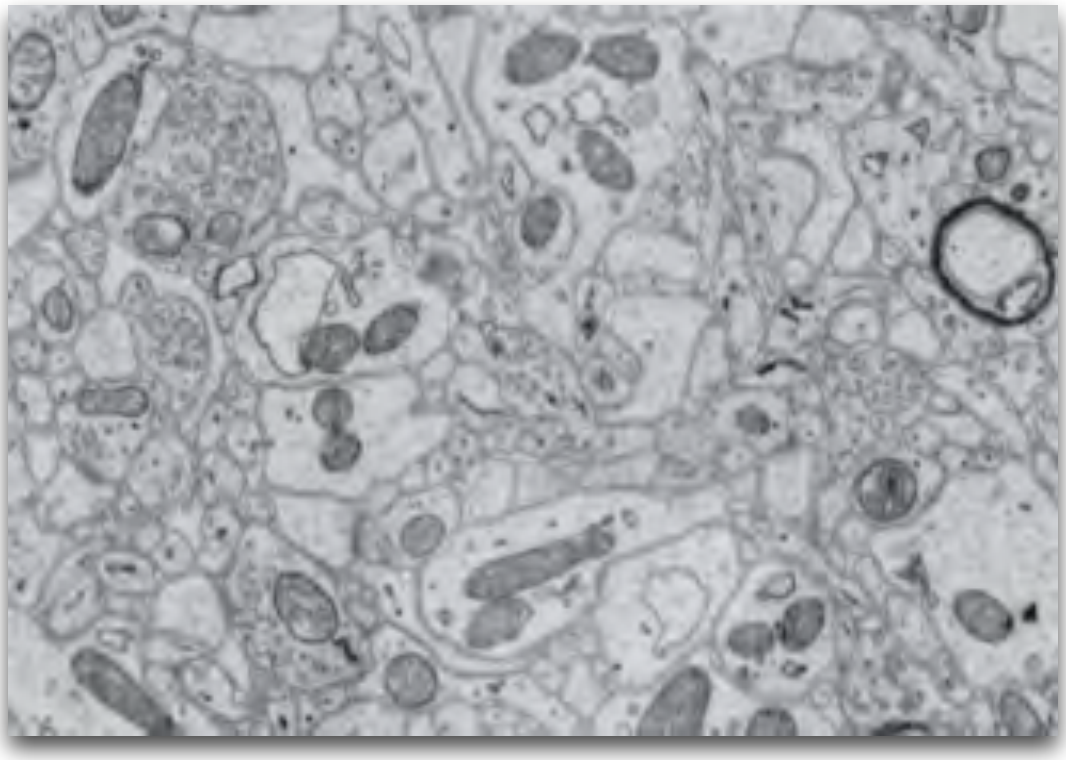
MYELIN SHEATH

1.2 Transmission Electron Microscopy

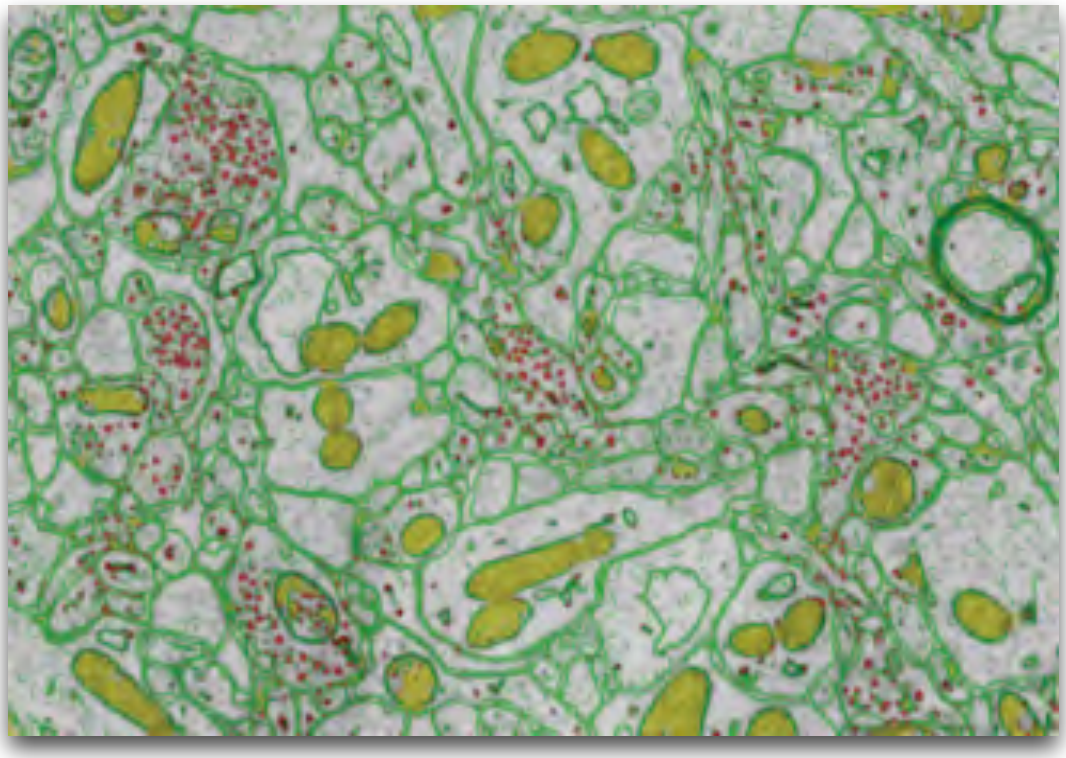
The images discussed in this work show neuroanatomical structures like membranes, vesicles and mitochondria, as illustrated in Figure 1.1 and illustrated in a real-world example in Figure 1.2. In order to capture such small objects an imaging technique is necessary which allows a resolution of below 5 nm per pixel (Kaynig-Fittkau [2011]). Such an apparatus is called *Transmission Electron Microscope* (TEM) with examples of the resulting images shown in Figure 1.3.

TEM

Williams and Carter [2009] state some drawbacks of TEM imaging. First, we are only able to view small parts of the specimen at a time. In other words, if we would like to capture a larger tissue area we need to take a worse sampling quality into account. Second, we are only able to produce two-dimensional images of non-uniform widths of three-dimensional specimens. In this setting, it is difficult to recognize structures for an expert user as well. Third, all pixels we see in a section image are averages through the thickness of the specimens and a section image

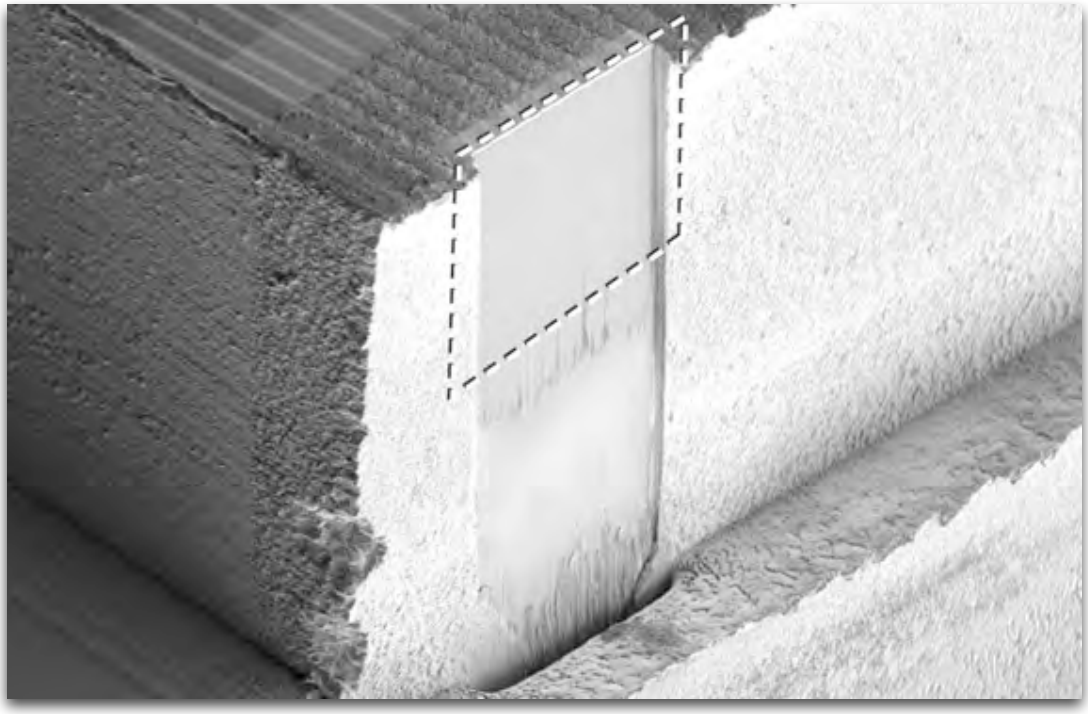


(a) ssTEM image

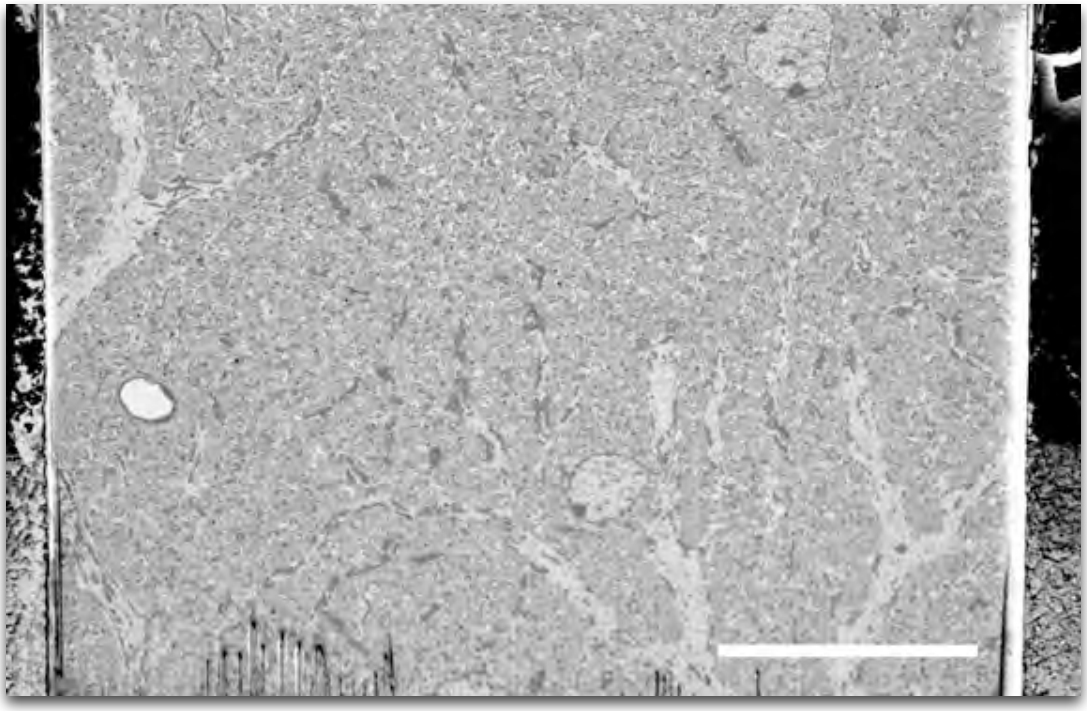


(b) Segmentation result

Figure 1.2: Sliced image of densely packed neuronal structures such as membranes of neuronal processes (dark green), mitochondria (light green) and vesicles (red) (from Vazquez-Reina et al. [2011]).



(a) Milled region



(b) Milled tissue sample

Figure 1.3: An example of a milled region (dashed rectangle) of a human male brain tissue using serial section TEM (scale bar: $20\,\mu m$; from Knott et al. [2008]).

does not have depth sensitivity (Williams and Carter [2009]).

In Figure 1.3b we see the trail of the diamond knife which mills sections from the brain tissue sample. The automatic process of tissue milling and capturing is called *serial section transmission electron microscopy* (ssTEM). For an introduction to electron microscopy we refer to Kaynig-Fittkau [2011] and Williams and Carter [2009].

ssTEM

1.3 Problem Statement

One focus of *computational neuroanatomy* is the automatic segmentation of neuronal structures in images of brain tissue samples. For this purpose, computing techniques such as the visualization, the modeling and the analysis of image and volumetric datasets are used (Kaynig-Fittkau [2011]). The images to be segmented are captured using *transmission electron microscopy* which allows to detect very small structures such as vesicles. Vesicles are shown as small dark circles which while being in a cluster are a hint for being part of a synapse.

COMPUTATIONAL NEUROANATOMY

On the one side, it is important to state that the manual segmentation is not practicable. Lets assume that each pixel represents a region area of $A = s_x \times s_y = 5 \times 5\,nm$ (for $s_x = s_y = 5$) and we only have a brain tissue sample of, lets say, $D = 1\,cm^3 = 10{,}000{,}000^3\,nm^3$. Furthermore, if the section (also called *slice*) thickness in z direction is $s_z = 20\,nm$ then we have a volume resolution of

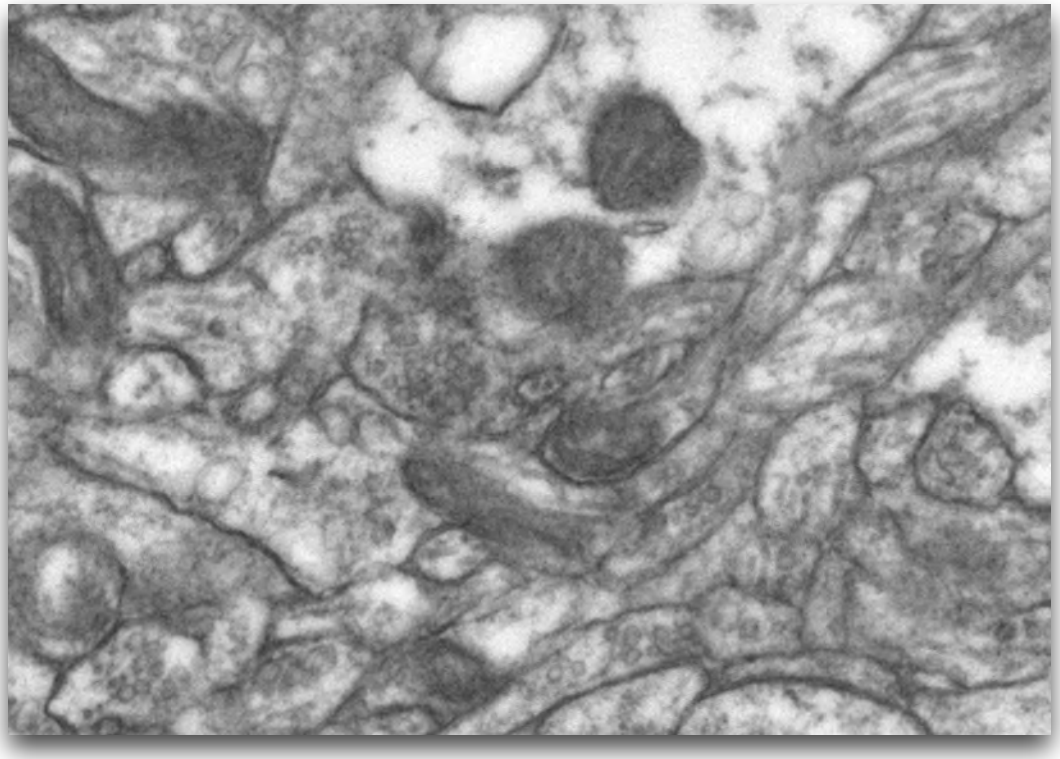
$$V = \left(\frac{10{,}000{,}000}{5}; \frac{10{,}000{,}000}{5}; \frac{10{,}000{,}000}{20}\right) = (2{,}000{,}000;\ 2{,}000{,}000;\ 500{,}000)$$

voxels. If we only use eight bits per voxel then we need a storage requirement of $2{,}000{,}000 \times 2{,}000{,}000 \times 500{,}000 \times 8 = 1.6e^{19}$ bits or 1776+ petabyte.

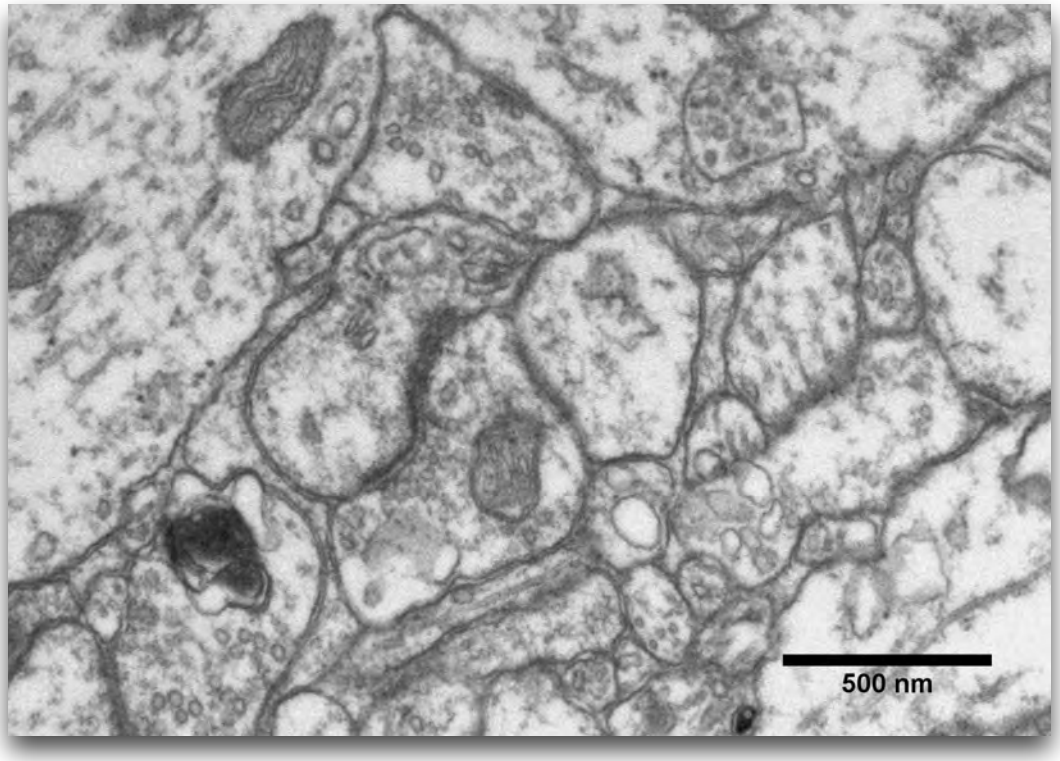
On the other side, the quality of automatic segmentation is greatly influenced by the quality of the captured images that contain densely packed structures which are not easily separable with simple density value thresholding. In general, ssTEM produces anisotropic volumes and results in two additional artifacts: The voxel size in z direction is larger than in x and y directions. Therefore, the quality of volume rendering and slice structure tracing across multiple slices of anisotropic volumes is greatly influenced by slice stitching. Another common artifact in ssTEM images is the problem of compressed zones, as illustrated in Figure 1.4, which lead to blurry membranes and make it difficult to achieve a satisfying segmentation.

The central issues in this work are the selection of *features* (short for *feature images*) and the training sample selection for the task of ssTEM segmentation of membranes. Domingos [2012] describes why the feature selection is important:

> "*Easily the most important factor is the features used. If you have many independent features that each correlate well with the class, learning is easy. On the other hand, if the class is a very complex function of the features, you may not be able to learn it. Often, the raw data is not in a form that is amenable to learning, but you can construct features from it that are.*" (p. 81).



(a) Slice thickness: 60 nm



(b) Slice thickness: 40 nm

Figure 1.4: Influence of slice thickness and compressed zones on image quality using the same brain tissue (from Kaynig-Fittkau [2011]).

1.4 Contributions of this Work

This work copes with the selection and construction of *features* (also called *predictor variables* in statistics) to improve the statistical learning task of finding the pixels belonging to the neuronal membrane structures in TEM images as shown in Figure 1.2. We build on the previous work of Kaynig-Fittkau [2011] and explore visualization and interaction techniques for the selection of existing features and the construction of new features from the old features. Furthermore, we discuss an incremental approach for the construction of the training sample. The overall goal is to improve the segmentation result through either a smaller prediction error rate or smaller computation time.

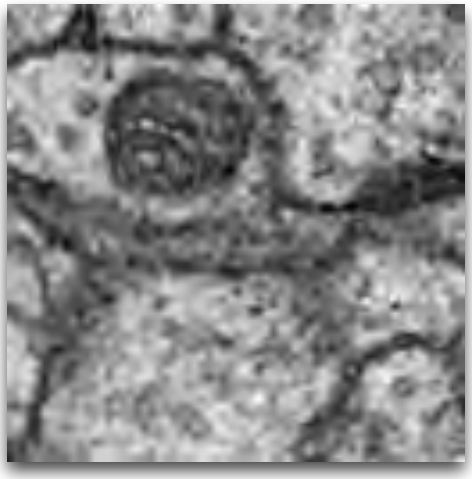
1.5 Pipeline Overview

In this work to explore a set of features that is more representative as the original set of 90 features proposed by Kaynig-Fittkau [2011]. Figure 1.5 shows five feature images derived from the density feature D. Figure 1.5b shows a template matching result R_0 using rotated template images (with rotation angle being $0°$). Figures 1.5c-1.5e are combinations of the rotated template images: variance, maximum and minimum. Figure 1.5f shows the per-pixel local histogram image h_2 of the bin having pixel intensity values in range $[25.5, 51]$ (this means that we discretize the 8 bits per pixel value range $[0, 255]$ into 10 bins and take the second bin). The feature construction is described in depth in Chapter 3. The overall target is to improve the segmentation process through either a smaller error rate, a lower computation time or a lower memory requirement. For this purpose, we combine visualization, user interaction and machine learning approaches into a coherent feature exploration framework for ssTEM image segmentation.

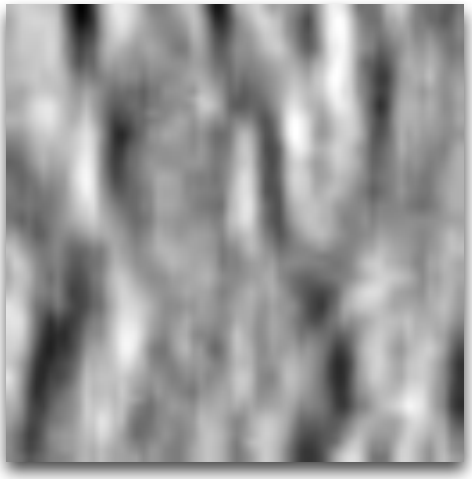
1.5.1 Membrane Segmentation

The machine learning methods for the segmentation of membranes in ssTEM images can be split into different stages as illustrated in Figure 1.6. At the beginning we have a *density volume* (or a stack of *intensity images*) which has been (a) correctly distorted because of the used lens system, (b) stitched and (c) aligned in the z direction (we refer to Kaynig-Fittkau [2011] for an in-depth discussion of the image processing workflow in connectomics). Because of the large data sets we must rely on automatic distortion correction, stitching and alignment. The aligned anisotropic density volume holds either 8 or 16 bits voxels.

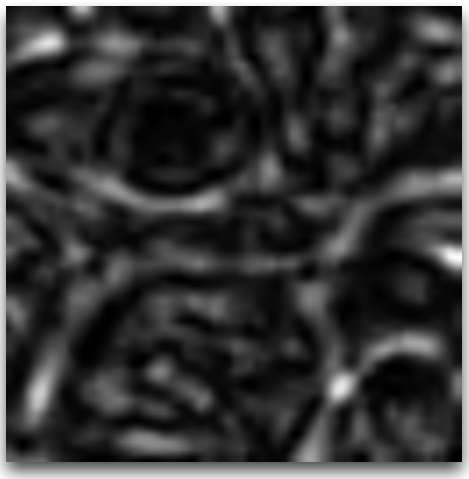
Through the manual user labeling of a small sub-set of all existing pixels in a slice image (couple of hundreds) we construct positive and negative examples of membrane pixels as the training data table. In other words, each table row contains an M-dimensional representation of a pixel. These M dimensions correspond to the number of features in the data table that need to be pre-calculated (see Chapter 3 for feature creation). After the feature pre-calculation step we now have the data base which we can use to train the random forest classifier (see



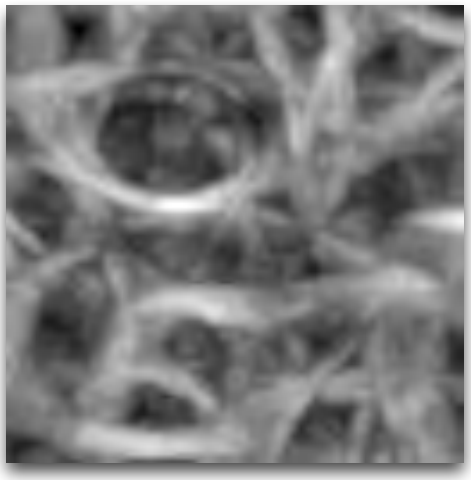
(a) D



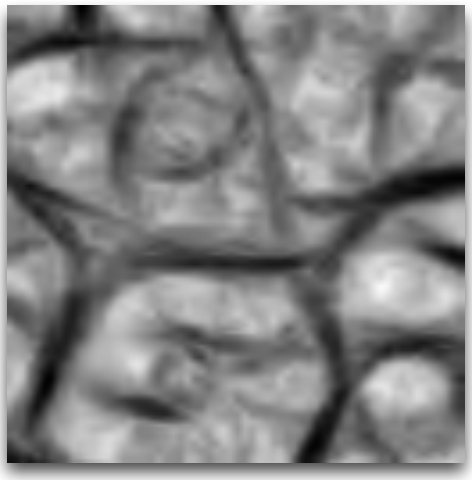
(b) R_0



(c) σ_R^2



(d) Max_R



(e) Min_R

(f) h_2

Figure 1.5: A sub-set of six feature images derived from density slice D illustrating value distributions of higher (brighter) and lower (darker) feature values.

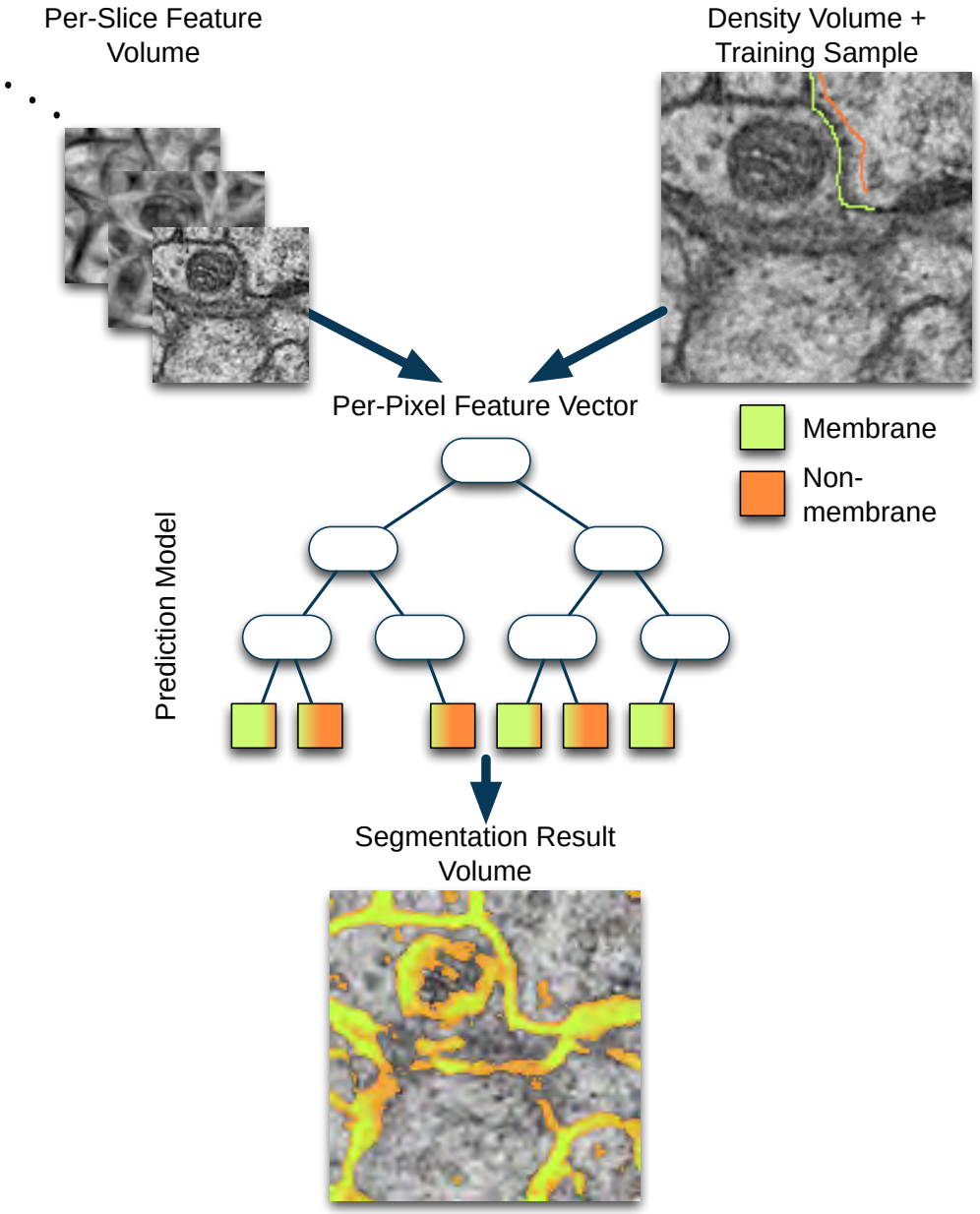


Figure 1.6: The segmentation process relies on a pre-computed set of features for each slice and training data. From this set we build a model (here illustrated as a decision tree) that has the ability to classify each new voxel in the class *membrane* (green) or the class *non-membrane* (orange).

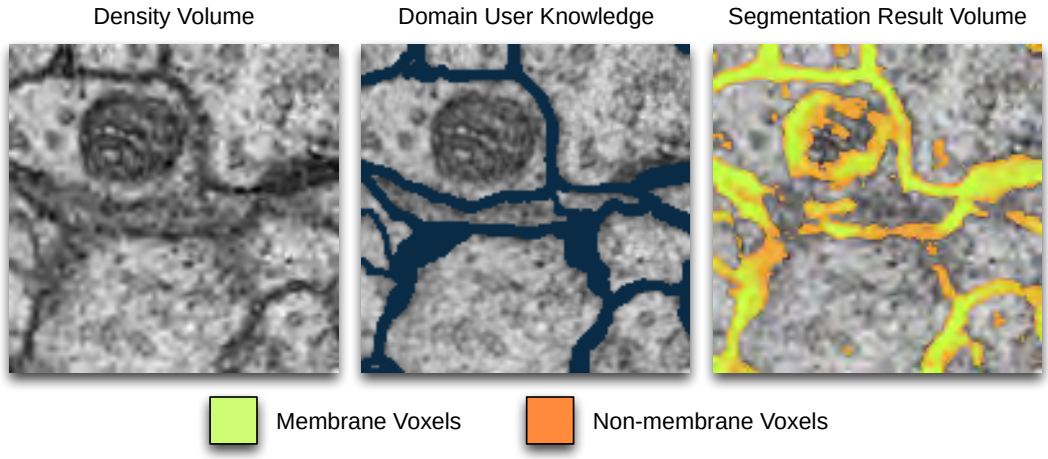
Section 2.3.5). The result of the classification/segmentation stage is a confidence value which describes for each pixel how confident the random forrest is that this pixel is belonging to a *membrane* or a *non-membrane*.

1.5.2 Feature Exploration

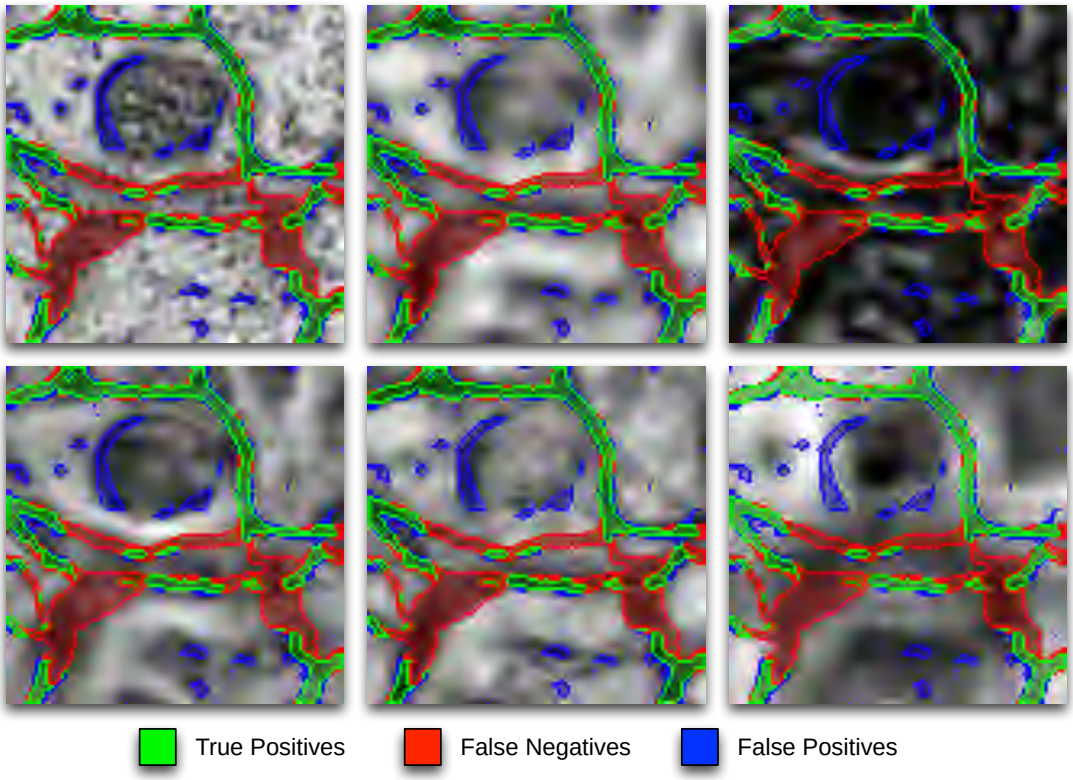
The overall feature exploration workflow is illustrated in Figure 1.7. We start with a density image and domain user knowledge or a ground truth image (Figure 1.7a). The ground truth volume contains pre-segmented *membrane* (blue) or *non-membrane* labels for each pixel and the resulting segmentation volume contains for each voxel a confidence value that states how confident the machine learning algorithm is that this voxel belongs to the class *membrane* (green) or *non-membrane* (orange). Figure 1.7b compares the features after the segmentation. The errors made are shown as false negative (red) and as false positive (blue) regions. The true positives are shown in green and the false positives are not shown at all.

1.5.3 Organization

In Chapter 2 we provide an introduction to a fundamental data structure that allows the storage and access of spatially related objects which is used to represent arrays, matrices and volumes. Furthermore, we will discuss tensor-based visualization methods of spatial data structures. Then, we give a short introduction in machine learning basics which are needed to understand the previous work as well as the contribution of this work. At the end, we discuss the previous work in the field of neuronal process segmentation as well as feature selection and creation using statistical and machine learning approaches. In Chapter 3 we learn how new features can be derived from the original intensity image and how they are used to improve the overall error rate during segmentation. Over 90 different features are derived from the intensity image through intensity transformation and filtering. In Chapter 4 we use the pre-computed features and explore their expressiveness by searching for redundancies and guide the search for relevant features. Furthermore, we combine the segmentation results with expert knowledge to explore the impact of the feature set for the purpose of random forrest classification. We will answer the following questions: *Which features are relevant to the task of membrane segmentation in ssTEM images? Which features can be omitted? Can we construct new features as a combination of existing ones that better discriminate between membrane and non-membrane voxels?* In Chapter 5 we will discuss the results from the exploration process by performing and analyzing the outcomes of six experiments. We will see the impact of newly created feature sets as a result of the feature exploration process. Finally, in Chapter 6 we will provide conclusive remarks and possible future research directions.



(a) Initial situation



(b) Feature comparison

Figure 1.7: Feature exploration workflow. (a) Initially, we only have a density volume, the expert user knowledge of what a *membrane* voxel is and the segmentation result volume. (b) The confusion matrix slice plot for six different feature images which originated from the density slice.

Chapter 2

Fundamentals & Previous Work

"Probability provides a way of summarizing the uncertainty that comes from our laziness and ignorance."

Artificial Intelligence: A Modern Approach, 2nd Edition
Stuart Russel, Peter Norvig (p. 464)

In the previous chapter we have mentioned that the density volume alone is not enough for a satisfactory segmentation of ssTEM brain tissue. To overcome this issue we derive new feature volumes from the original density volume and provide these feature volumes to a machine learning algorithm that classifies each pixel individually into *membrane* or *non-membrane*. But to get to this point we need an appropriate data structure – called *tensor data structure* – that stores different kinds of feature volumes in a unified way. In order to utilize the expressiveness of scientific and information visualization we need different visualization methods to visualize and compare both feature volumes as well as feature slices. Finally, we need to have a machine learning tool which allows the segmentation of single pixels by not only providing a classification result but also a confidence percentage or a probability.

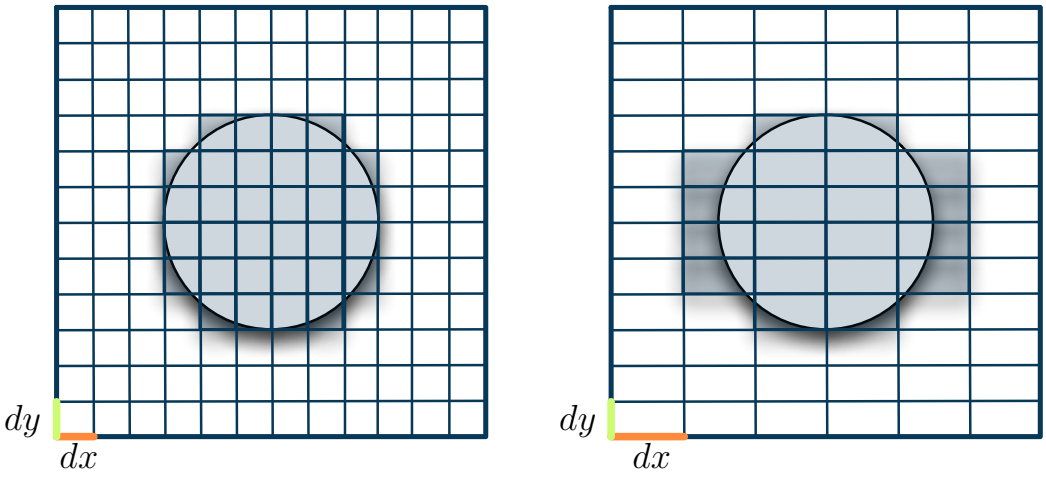
2.1 Tensor Data Structure

One of the most commonly used data structures in scientific visualization is the *equidistant grid* which stores scalar arrays, two-dimensional texture images and three-dimensional density volumes. Two examples of such grids are illustrated in Figure 2.1: Cartesian grid in case of $dx = dy$ and the rectangular grid for $dx \neq dy$. In order to handle different types of equidistant spatial data (such as intensity, three-channel and four-channel color images) as well as different dimensions (such as images and volumes) we organize its elements in a *tensor data structure*.

EQUIDISTANT GRID

TENSOR DATA STRUCTURE

Originally, a tensor is defined as a mathematical description of a physical object which can be viewed as multi-dimensional array and as the generalization of vectors and matrices (Kolda and Bader [2009]). For the purpose of processing data based on equidistant grids extend the tensor data structure in such a way that it can hold any kind of storable information that is organized spatially according to the order of the tensor which also determines its storage requirements:



(a) Cartesian grid $dx = dy$ (b) Rectangular grid $dx \neq dy$

Figure 2.1: Two types of equidistant grids used in scientific visualization: (a) Cartesian grid and (b) Rectangular grid.

SCALARS **0th-Order** Tensors of order 0 are called *scalars* (e.g., temperature value, density value, probability value, confidence value).

VECTORS **1st-Order** Tensors of order 1 are called *vectors* (e.g., directions)

MATRICES **2nd-Order** Tensors of order 2 are called *matrices* (e.g., two-dimensional equidistant grids)

VOLUMES **3rd-Order** Tensors of order 3 are called *volumes* (e.g., density volume, confidence volume).

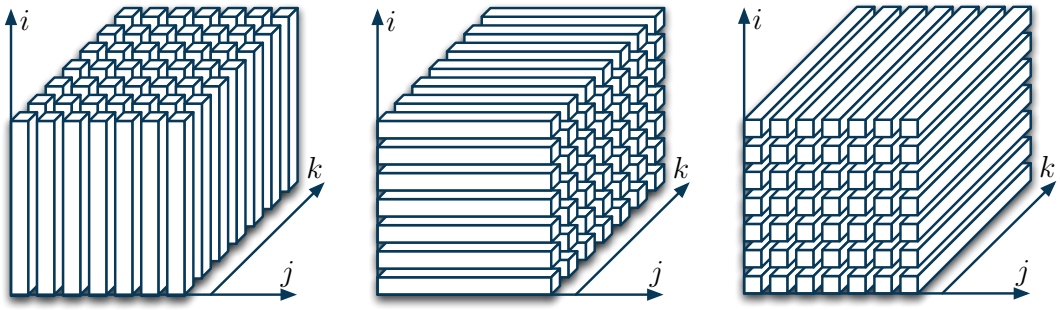
We use the same notation as Vasilescu and Terzopoulos [2004]: (a) scalars are denoted by lower case letters $(x_1, x_2, \ldots)$, (b) vectors are denoted by bold lower case letters $(\mathbf{x}, \mathbf{y}, \ldots)$, (c) matrices are denoted by bold upper-case letters $(\mathbf{A}, \mathbf{B}, \ldots)$, and (d) for higher-order tensors we use calligraphic upper-case letters $(\mathcal{A}, \mathcal{B}, \ldots)$.

More generally, we can say that the order of a tensor $\mathcal{A} = \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is N (Vasilescu and Terzopoulos [2004]), where I_l is the size of the tensor in the l-th dimension for $l = 1, \ldots, N$. For example, for the 3rd-order tensor displayed in Figure 2.2 the number of dimensions $N = 3$ and $l \in \{I_1, I_2, I_3\}$ with $i = 1, \ldots, I_1$, $j = 1, \ldots, I_2$ and $k = 1, \ldots, I_3$. We can extend the tensor data structure to not only to store scalars but to store a *tensor element* (TE) – in computer science we speak about a structure rather than a single scalar value. For example, this is necessary particularly in visualization where we would like to store Red-Greed-Blue-Alpha channels for each image pixel. In this case the TE holds four scalars instead of one.

TENSOR ELEMENT

Additionally, for the purpose of image slice and volume visualization we consider 1st-, 2nd- and 3rd-order tensors. A vector has a number of elements where the i-th element of a vector $\mathbf{a}$ is denoted by a_i with $i = 1, \ldots, I_1$ where I_1 is the

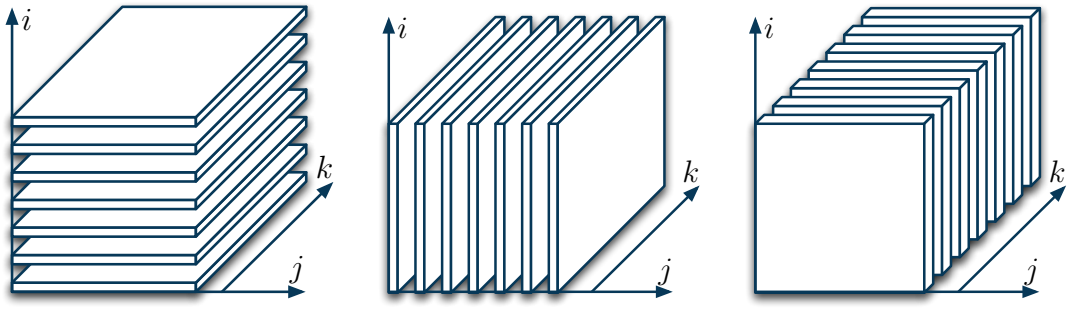
number of elements stored in the vector. A matrix has a number of rows I_1 and columns I_2 and its elements $a_{i,j}$ can be accessed using an index pair (i, j) with $i = 1, \ldots, I_1$ and $j = 1, \ldots, I_2$. A volume has a number of rows I_1, a number of columns I_2 and a number of tubes I_3 and its elements can be accessed using an index triple (i, j, k) with $i = 1, \ldots, I_1$, $j = 1, \ldots, I_2$ and $j = 1, \ldots, I_3$. Apart from that, we also use *fibers* as the higher-order analogue of matrix rows and columns (Kolda and Bader [2009]), as illustrated in Figure 2.2. FIBERS



(a) Mode-1 (column) fibers: $\mathbf{x}_{:,j,k}$ (b) Mode-2 (row) fibers: $\mathbf{x}_{i,:,k}$ (c) Mode-3 (tube) fibers: $\mathbf{x}_{i,j,:}$

Figure 2.2: 3rd-order tensor fibers (from Kolda and Bader [2009]).

Particularly for the purpose of volume visualization, we can cut the volume into coordinate-plane-parallel *slices* as illustrated in Figure 2.3. Kolda and Bader [2009] defines slices as two-dimensional sections of a tensor where two indices are varying and all others are fixed. The 3rd-order tensor can have horizontal (i fixed, j and k varying), lateral (j fixed, i and k varying) and frontal slices (k fixed, i and j varying). SLICES



(a) Horizontal slices: $\mathbf{x}_{i,:,:}$ (b) Lateral slices: $\mathbf{x}_{:,j,:}$ (c) Frontal slices: $\mathbf{x}_{:,:,k}$

Figure 2.3: 3rd-order tensor slices parallel to the (a) j-k, (b) i-k and (c) i-j coordinate-plane (from Kolda and Bader [2009]).

2.2 Visualization of Tensor-based Data Structures

As stated before, tensors provide an efficient way to organize, store and access spatial datasets. In this work we focus on the visualization of 0th, 2nd and 3rd order tensors. The main goal of *visualization* can be stated as follows:

VISUALIZATION

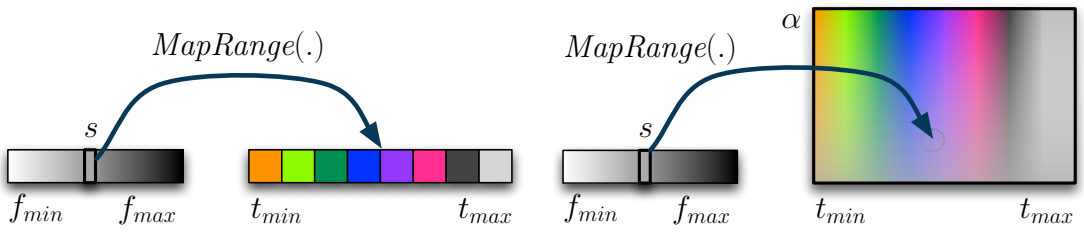
> *"The goal of visualization is to leverage existing scientific methods by providing new scientific insight through visual methods."* Johnson and Hansen [2004] (p. xiv).

In order to provide insight in this work we need to display tensor-based data structures by encoding not just the spatial relationships but also other properties as well.

2.2.1 Mapping Scalars to Colors

The 1st order tensor is a scalar value which is associated with a point or cell of a dataset (Johnson and Hansen [2004]). The most common scalar visualization algorithm is the *color mapping* which maps scalar data to colors, as illustrated in Figure 2.4a. First, the scalar value is mapped to an index in a color lookup table,

COLOR MAPPING



(a) Scalar-color mapping (b) Transfer function mapping

Figure 2.4: Two mappings of a single scalar value according to (a) color lookup table and (b) transfer function with transparency parameter α on the y axis.

holding a color array of length n. Because the scalar might not be an integer which is in the range of $[0, \ldots, n-1]$ we need to map its value $s \in [f_{min}, \ldots, f_{max}]$ to an index i into a target range $[t_{min}, \ldots, t_{max}]$. This can be achieved by applying the range mapping function on the scalar s.

DEFINITION 2.1 (Range Mapping)

The *range mapping function* maps a sclar value $s \in \mathbb{R}$ which has a source range $[f_{min}, f_{max}]$ to a target range $[t_{min}, t_{max}]$ with

$$MapRange(s) = t_{min} + \frac{s - f_{min}}{f_{max} - f_{min}} \cdot (t_{max} - t_{min})\,. \tag{2.1}$$

Second, the index $i = \lfloor MapRange(s) \rfloor$ is then used to retrieve a color from the color lookup table.

The color lookup table is a discrete data structure containing a fixed number of colors. A more general color mapping function is the *transfer function* which maps a scalar value not to a discrete index but to a color specification by interpolating between two colors, as illustrated in Figure 2.4b (Johnson and Hansen [2004]). Additionally, we can extend the standard RGB color definition by a transparency part α (A). All images produced in Chapter 5 are generated using a transfer function.

TRANSFER FUNCTION

2.2.2 Image Visualization

The visualization of a 2nd order tensor, also represented as an image, is straight forward: each pixel in a ssTEM image has an intensity value which is mapped using a transfer function into the color space. Each pixel value s is looked up using the transfer function which interpolates between two neighboring colors. The looked up color is saved in the image buffer.

In case of the visual comparison of multiple images (e.g., an intensity image with the corresponding segmentation result) it is important to visualize objects so that they can be compared easily. Gleicher et al. [2011] propose a taxonomy that divides the space of comparative designs into three categories for information visualization, as illustrated in Figure 2.5. The *juxtaposition* design shows images next to each other either in space or in time. This is shown in Figure 2.5a where the drosophila fly brain image is shown in comparison to the ground truth. Another comparative design is the *superposition* (or *overlay*) which shows two images on top of each other. See Figure 2.5b for blending the ground truth (green) and original density image. The third design is *explicit encoding* which encodes explicit relationships between the images (Gleicher et al. [2011]). This is shown in Figure 2.5c where differences of two segmentation results are shown. Yellow indicates highest differences, red indicates medium differences and the blue pixels indicate low differences. Comparative ssTEM image visualization is discussed in Chapter 4.

JUXTAPOSITION

SUPERPOSITION

EXPLICIT ENCODING

2.3 Machine Learning

One of the most cited and broad definitions of *machine learning* (ML) is the one by Tom M. Mitchell:

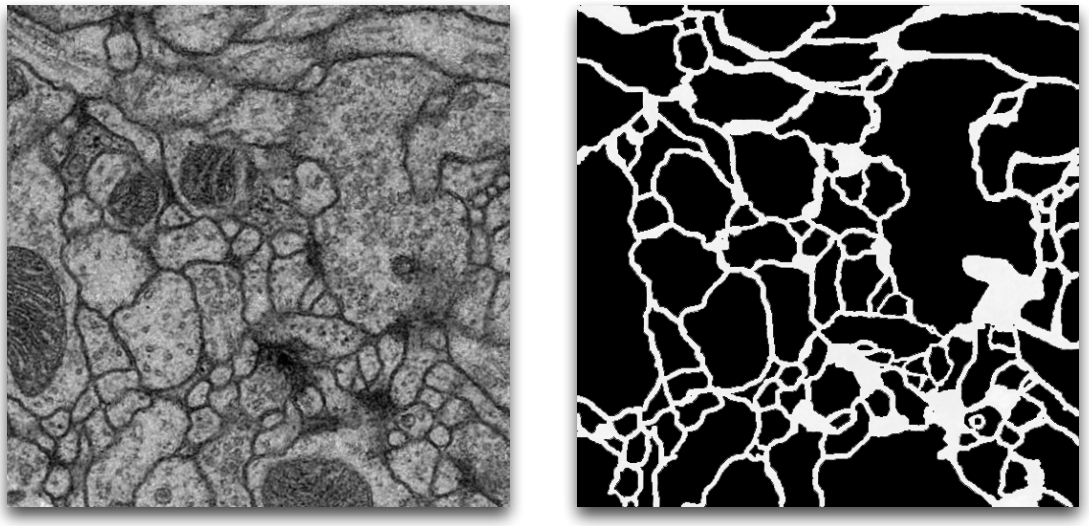
MACHINE LEARNING

> *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."* Mitchell [1997] (p. 2).

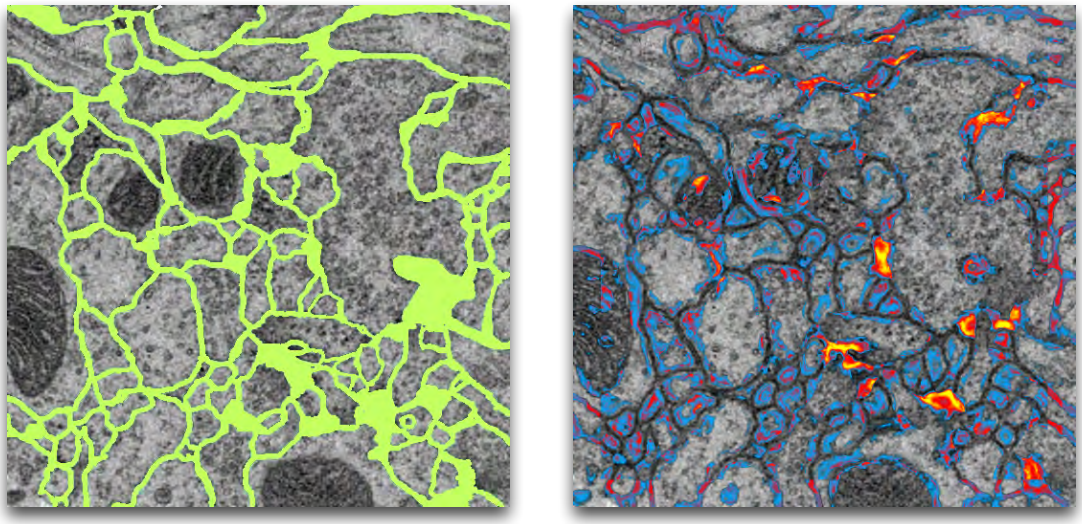
In the context of this work, the learning problem can be stated as follows:

Task T: Classify each pixel according to two classes *membrane* ($\oplus$) or *non-membrane* ($\ominus$) (i.e., do a *binary classification*). Therefore we use $\oplus$ and $\ominus$ as abbrevi-

BINARY CLASSIFICATION



(a) Juxtaposition



(b) Superposition (c) Explicit encoding

Figure 2.5: Three comparative designs used for the exploration of segmentation results of drosophila fly brain ssTEM images.

ations for the classes.

Performance Measure P**:** Measure the percentage ρ of incorrectly classified pixels in a slice image, where $\rho \in [0, 1]$ and $\rho \in \mathbb{R}$.

Training Experience E**:** Domain expert provides a small sub-set of *membrane* and *non-membrane* examples to the machine.

This means, that based on the training examples the machine should extract relevant information of how a membrane pixel is described and embed this information into a model. This model is then used for new pixels with the following objective:

> *"The main practical objectives of machine learning consist of generating accurate predictions for unseen items and of designing efficient and robust algorithms to produce these predictions, even for large-scale problems."* Mohri et al. [2012] (p. 2)

For this purpose, we first must create a data structure that holds the descriptions of the pixels. Such data structure can be represented by a *data table* (also known as *data sample*) S with $X = \{\mathbf{x}_i\}$ and $\mathbf{x}_i = (x_{i,j})$, $i = 1, \ldots, N$ and $j = 1, \ldots, M$, as illustrated in Figure 2.6. X is a data table with N rows or

DATA TABLE

	F_1	$\ldots$	F_j	$\ldots$	F_M	C
$\mathbf{x}_1$	$x_{1,1}$	$\ldots$	$x_{1,j}$	$\ldots$	$x_{1,M}$	c_1
$\vdots$	$\vdots$	$\ddots$	$\vdots$		$\vdots$	$\vdots$
$\mathbf{x}_i$	$x_{i,1}$	$\ldots$	$x_{i,j}$	$\ldots$	$x_{i,M}$	c_i
$\vdots$	$\vdots$		$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathbf{x}_N$	$x_{N,1}$	$\ldots$	$x_{N,j}$	$\ldots$	$x_{N,M}$	c_N

Figure 2.6: A schematic view on the underlying data table S found in many ML applications. Each observation $\mathbf{x}_i$ is represented by a vector of feature values $(x_{i,1}, \ldots, x_{i,j}, \ldots, x_{i,M})$ and has a class label c_i. Each feature F_j can hold nominal, ordinal or numerical values.

observations (also known as *examples* or *items*) and M columns or features (also known as *attributes* or *variables*) where each column can either hold *nominal* (e.g, strings such as gender or names), *ordinal* (e.g., week days, date, time) or *numeric* values (e.g., integer or real scalars). Additionally to the feature columns we also have a class column C in which a class c_i is assigned to each observation in case of classification. The number of distinct classes in C is denoted by N_C, i.e., for a binary classification we have $N_C = 2$ and $c_i \in \{\oplus, \ominus\}$ where $\oplus$ and $\ominus$ are the classes (e.g., *membrane* $= \oplus$ and *non-membrane* $= \ominus$). It is possible to encode the data table as well as a 2nd-order tensor by encoding nominal and ordinal features as numerical features and therefore storing only numerical values as TEs.

With that in mind, we can define the *target function* as follows:

TARGET FUNCTION

DEFINITION 2.2 (Target Function)

Mitchell [1997] defines the *target function* that needs to be represented and approximated by a machine learning algorithm as

$$f : \mathbf{X} \longrightarrow \mathbf{c} \tag{2.2}$$

which maps each possible input tuple – also known as a *feature vector* – $\mathbf{x}_i \in \mathbf{X}$ to a categorical output value – also known as the *class* or response vector $\mathbf{c}$ (Breiman [2001]) – $c_i \in \mathbf{c}$ in case of *Classification* (or a numerical value $c_i \in \mathbf{c}$ in case of *Regression*) with $i = 1, \ldots, N$.

In general, the target function is not known and needs to be *estimated*, *fitted* or *learned* from the training sample by either providing the corresponding class labels for each training feature vector (illustrated as column C in Figure 2.6) or by not providing it. The former case is called *Supervised Learning* (SL) which requires a training sample with the class label assigned to each example. The SL approaches include algorithms such as decision trees, naive Bayes classifiers, support vector machines and random forests. The second case is called *Un-supervised Learning* which does not require this example-to-class mapping for training. We only rely on the power of the algorithm to separate the feature vectors into a number of clusters which corresponds to the number of classes we assume are in the sample. Such methods include algorithms like artificial neural networks, self organizing maps, k-means clustering and mixture models.

SUPERVISED LEARNING

UN-SUPERVISED LEARNING

2.3.1 Machine Learning Issues

As Domingos [2012] puts it: "*The fundamental goal of machine learning is to generalize beyond the examples in the training set.*" (p. 79). In this context we need to take care of certain inherent problems that occur in any machine or statistical learning setting.

RANDOM GUESSING

First, ML should be used if it outperforms *random guessing* (Domingos [2012]). This requirement can be fulfilled if the sample is split into a *training* and a separate *test set* (see Section 2.3.2). The problem here is that most of the machine learning algorithms have parameters (e.g., number of nodes in a decision tree or the number of trees in a random forest) which need to be tuned. Therefore, a third validation data set is taken from the training data set. It is used to evaluate different parameter settings for the learned models but it leaves us with fewer training examples and possibly biased prediction models. To overcome this issue *cross-validation* can be used by randomly splitting the training data into $N_T \in \mathbb{N}$ sub-sets. While the models are trained with (say) $N_T - 1$ training sub-sets we use one sub-set to validate these models. Each of the $N_T - 1$ models are created with different parameter settings from which we only choose the one where the validation sub-set performs best (Domingos [2012]). The difference between the validation and test set is that the validation set is used during the construction of

CROSS-VALIDATION

the prediction model while the test set is a priori chosen to test the model after it has been constructed.

Second, it is not sufficient to provide the machine learning algorithm with just more data to improve its prediction accuracy. In order to improve the generalization, the predictive model needs some prior information, knowledge or assumptions about the problem domain (Domingos [2012]). This issue is discussed in Chapter 4.

OVER-FITTING

Third, the problem of *over-fitting* occurs when insufficient domain knowledge is provided or if insufficient data is provided in order to model the correct classifier. In such cases we are creating a classifier that is not grounded in the problem domain but it is incorporating noise in the data (Domingos [2012]). The problem of over-fitting can only be discovered if we have the ground truth to compare the classifier to.

GENERALIZATION ERROR

BIAS

VARIANCE

Fourth, as mentioned before, generalization is an essential goal for any ML algorithm. A measure of how well a ML algorithm generalizes to test data is the so-called *generalization error*. Over-fitting can also be described by splitting the generalization error into *bias* and *variance*. As Domingos [2012] describes it: "Bias is a learner's tendency to consistently learn the same wrong thing. Variance is the tendency to learn random things irrespective of the real signal." (p. 80).

CURSE OF DIMENSIONALITY

Fifth, the second biggest problem in machine learning is the *curse of dimensionality*: "*Generalizing correctly becomes exponentially harder as the dimensionality (number of features) of the examples grows, because a fixed-size training set covers a dwindling fraction of the input space.*" (Domingos [2012] (p. 81)). This issue is explored further in this work for the segmentation ssTEM images using over 90 features and an image size of $512 \times 512 = 262{,}114$ pixels. Even if we use all the image pixels as training set we still cover only about 262,114 examples of an input space which is actually $262{,}114 \cdot 90 = 23{,}590{,}260$ examples large. We refer to Domingos [2012] for an extensive introduction to machine learning issues and to Pereira et al. [2009] for an introduction to image segmentation using machine learning methods.

2.3.2 Sample Management

Even state-of-the-art machine learning algorithms can fail to predict the correct class if the algorithm is not provided with representative data samples. This is, because the algorithm can only rely on the data we provide and it has no prior knowledge about the data domain. It is therefore imperative to provide *clues* of what representative data samples are. The clues include the selection of data samples and weighting of data samples for training.

Moreover, prior to the task of learning we need to ensure that the ML classifier follows common principles in statistical testing by splitting training samples from test samples as described by Mohri et al. [2012] and illustrated in Figure 2.7. The *training sample* $\mathcal{L}$ (yellow) should contain representative examples, correct class labels assigned to each example and a feature set that is descriptive with regard to the classification problem. After the training the examples in the training sample are considered as *seen*. The *test sample* $\mathcal{E}$ (blue) should contain *unseen*

TRAINING SAMPLE

TEST SAMPLE

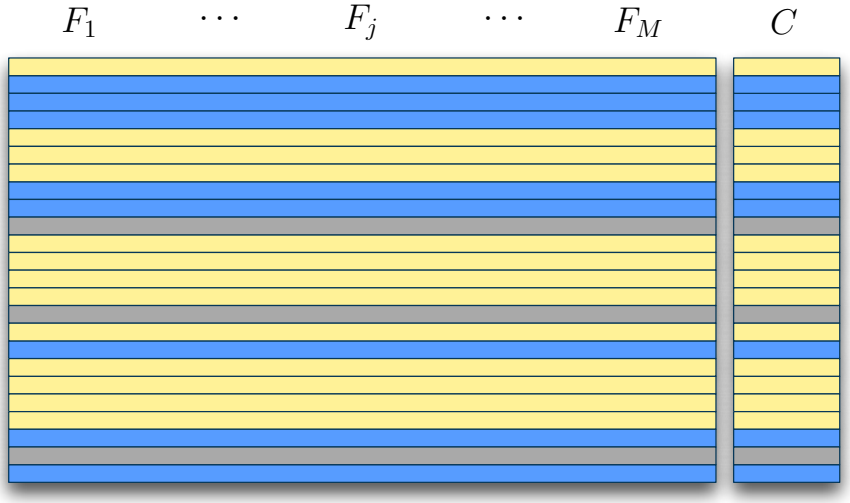


Figure 2.7: Illustrative example for splitting the data table into training (yellow), validation (gray) and test samples (blue).

examples that need to be predicted in order to get a robust accuracy estimate of the predictive model. The test sample must also contain the class label assigned to each example in order to determine the prediction accuracy of the learned model. Some learning algorithms provide a range of model parameters that can be tuned as well. For that purpose, we remove randomly examples from the training sample and put them into a *validation sample* (Mohri et al. [2012]). These examples are not used for training but for evaluation of the parameters while creating the prediction model.

VALIDATION SAMPLE

2.3.3 Evaluation of Classification Results

In order to distinguish good classifiers from bad classifiers we need to compare the results through an *evaluation function* (also known as *objective function* or *scoring function*) (Domingos [2012]). The evaluation is done using an unseen test sample $\mathcal{E}$, as described in the previous section. The standard evaluation function is the one which determines the *prediction accuracy*: $\nu = N_c/N_{\mathcal{E}}$ where N_c is the number of correctly classified examples in the test set and $N_{\mathcal{E}} = |\mathcal{E}|$ is the number of all test examples. It is also possible to rephrase the prediction accuracy as *prediction error* $\varepsilon = (N_{\mathcal{E}} - N_c)/N_{\mathcal{E}}$ (Witten and Frank [2005]).

EVALUATION FUNCTION

PREDICTION ACCURACY

PREDICTION ERROR

For a binary classifier, rather than considering only the prediction accuracy or the prediction error we could further separate the classification results into a 2×2 *confusion matrix* (Forman and Scholz [2010], Davis and Goadrich [2006]) as illustrated in Figure 2.8. True positives (TP) is the number of examples correctly classified as *positive*. False negatives (FN) is the number of examples classified as *negative* that are in fact *positive* and false positives (FP) is the number of examples classified as *positive* that are in fact *negative*. True negatives (TN) is the number of examples correctly classified as *negative*. We consider FP as type I error and FN as type II error. We can reformulate the test set prediction accuracy

CONFUSION MATRIX

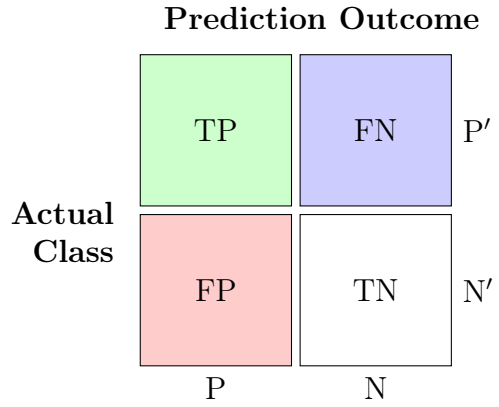


Figure 2.8: Confusion matrix representing the outcomes of a two-class prediction (Witten and Frank [2005]).

as $\nu = (TP+TN)/(TP+FN+FP+TN)$ and the prediction error as $\varepsilon = 1-\nu$ (Forman and Scholz [2010]). As we will see, the confusion matrix provides an intuitive way of visualizing and identifying errors in image segmentations.

The combinations of the four factors of a confusion matrix lead to different measures. The *sensitivity* (also known as *true positive rate* or *recall*) is defined as $\rho_{tp} = TP/P'$ where $P' = TP + FN$ is the total number of positives. It is a measure how well the model classifies *positive* examples as *positive* (Witten and Frank [2005]). Analogous to sensitivity, the false positive rate is defined as $\rho_{fp} = FP/N'$ with $N' = FP + TN$ being the total number of negatives. The *specificity* (also known as *true negative rate*) on the other hand measures the rate of correctly classified *negative* examples. It is defined as $\sigma = TN/N'$ with $N' = FP + TN$ (Forman and Scholz [2010]). The *precision* is defined as $\varpi = TP/P$ where $P = TP + FP$.

SENSITIVITY

SPECIFICITY

PRECISION

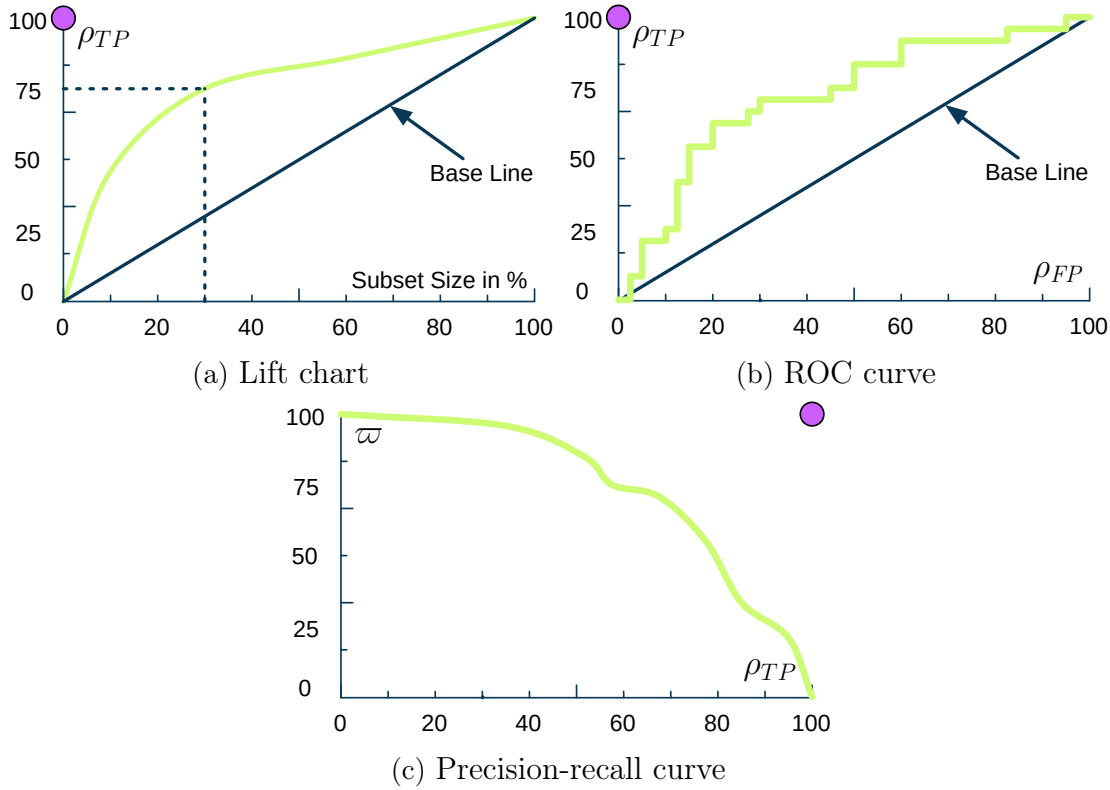
In standard machine learning literature there are numerous ways to illustrate the effectiveness of a classifier (Witten and Frank [2005]). We will discuss three of them: lift charts, ROC curves and recall-precision curves. Some classifiers (such as random forests) output confidence values rather than classes for each test example. To produce a binary decision we need to set a confidence value threshold. While for the lift chart the threshold is set, for the ROC and the recall-precision curves is not set but is varying. This variation leads to the curve points for ROC and recall-precision curves.

The *lift chart* as illustrated in Figure 2.9a provides a way to visually compare the prediction performance for true positives when adding more and more data to the test sample. In other words, it tells us, for example, that if the prediction model is correct we will get 75 percent of all positives (dotted line) by just using 30 percent (i.e., the cost) of the test data. The desirable position on the chart is the top left corner.

LIFT CHART

Another visual evaluation method is the so-called *receiver operating characteristic* (ROC) curve which plots the false positives rate ρ_{fp} against the true positive rate ρ_{tp}, as illustrated in Figure 2.9b (Witten and Frank [2005]). ρ_{fp} is a fraction

ROC



(a) Lift chart (b) ROC curve

(c) Precision-recall curve

Figure 2.9: Three ways to show the goodness of a classifier. The optimum classifier would have measure properties that bring it to the corners of the charts (purple spots).

of the negative examples which are misclassified as positive, while ρ_{tp} is a fraction of positive examples that are correctly classified as positive (Davis and Goadrich [2006]). The desirable position on the chart is the top left corner.

PRECISION-RECALL CURVE

The last kind of curve we would like to discuss is the *precision-recall curve* where we plot recall $\rho_{tp} = TP/P'$ against the precision $\varpi = TP/P$, as illustrated in Figure 2.9c. We see that ρ_{tp} is the true positive rate while ϖ is the fraction of examples classified as positive that are in fact positive (Davis and Goadrich [2006]). For more information on the different evaluation methods for classifiers we refer to Witten and Frank [2005], Davis and Goadrich [2006] and Forman and Scholz [2010].

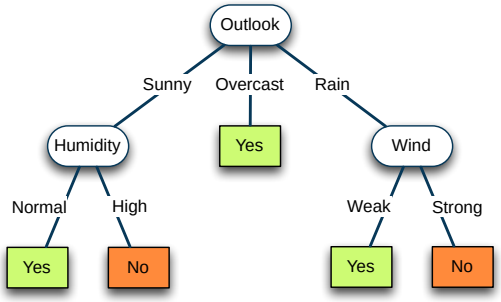
2.3.4 Decision Tree Learning

CONCEPT

Before discussing the random forest classifier let us consider what a decision tree is. A decision tree is a tree data structure which models a *concept*. For example, the decision tree shown in Figure 2.10b summarizes the concept of when a person does play tennis. The underlying data table is found in Figure 2.10a. The decision tree has a root node with possibly multiple successor nodes. Each successor node might either be an intermediate node (round corner rectangle) or a leaf node (rectangle). While the root and intermediate nodes represent feature variables

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

(a) Data table



(b) A corresponding decision tree

Figure 2.10: Example illustrating a decision tree for the concept *Play Tennis* (Mitchell [1997]). (a) The underlying data sample to be learned from. (b) The resulting decision tree for the concept *Play Tennis*.

and the links represent *feature values*, the leaf nodes represent the classes. The tree data structure can also be interpreted as a set of rules, i.e., disjunctions of conjunctions that represent the learned concept. Therefore, the concept *Play Tennis* can be represented as a data table (see Figure 2.10a), as a decision tree (see Figure 2.10b) or as a set of rules, deduced from the decision tree:

$$\begin{aligned} \textit{Play Tennis} = &(\textit{Outlook} = \textit{sunny} \wedge \textit{Humidity} = \textit{high}) \\ &\vee (\textit{Outlook} = \textit{overcast}) \\ &\vee (\textit{Outlook} = \textit{rain} \wedge \textit{Wind} = \textit{strong}) \,. \end{aligned}$$

The main issue during the construction of the tree is how to choose the next feature as an intermediate node or, as Mitchell [1997] puts it, we need to answer the question: "*Which attribute is the best classifier?*" (p. 55). An answer to this question provides a statistical property that measures how well an attribute separates the training sample according to the class (Mitchell [1997]). One such property is called the *information gain* which is defined using the *entropy* concept borrowed from information theory (MacKay [2002]):

INFORMATION GAIN

ENTROPY

DEFINITION 2.3 (Entropy Function)

Mitchell [1997] defines the *entropy* function with regard to the classification problem of a collection S of data examples as

$$\textit{Entropy}(S) = \sum_{i=1}^{N_c} -p_i \log_2 p_i \,, \tag{2.3}$$

where N_c is the number of classes and p_i is the proportion of S belonging to the class i. The base 2 logarithmic function is used because the entropy measures the encoding length of a message in bits (Mitchell [1997]).

In other words, the entropy function measures the (im)purity of an arbitrary col-

lection of examples S (Mitchell [1997]). Figure 2.11 shows the binary entropy function used in binary classification for the class $\oplus$ (e.g., person does play tennis). Equation 2.3 states also that in case that all examples in S do belong to

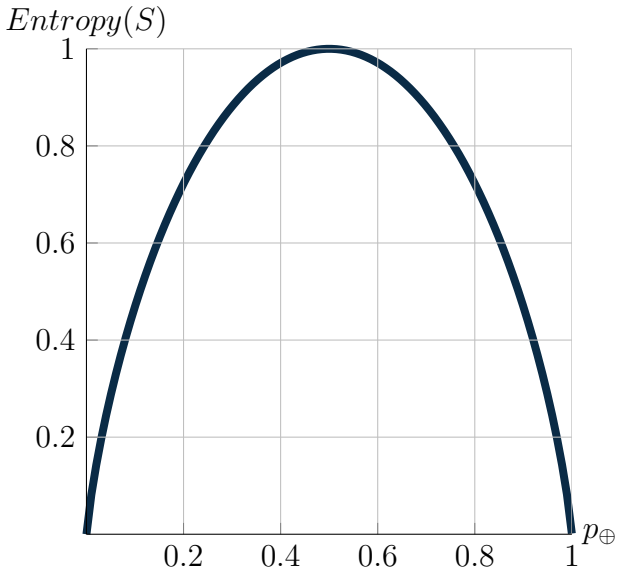


Figure 2.11: Entropy function relative to a boolean classification. $p_\oplus$ denotes the proportion of the class $\oplus$ in S.

the same class ($\oplus$ or $\ominus$) the entropy is 0. If we have the same number of class occurrences in S (so that $p_\oplus = 1/2$ and $p_\oplus = 1/2$) then the entropy is 1 (see $p_\oplus = 1/2$ in Figure 2.11).

With the definition of the entropy we now can specify the information gain function. Mitchell [1997] defines the information gain for an attribute as "*... the expected reduction in entropy caused by partitioning the examples according to this attribute*" (p. 57).

DEFINITION 2.4 (Information Gain)

Mitchell [1997] defines the *information gain* of an attribute F_j relative to a collection of examples S as

$$InfoGain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)\,, \qquad (2.4)$$

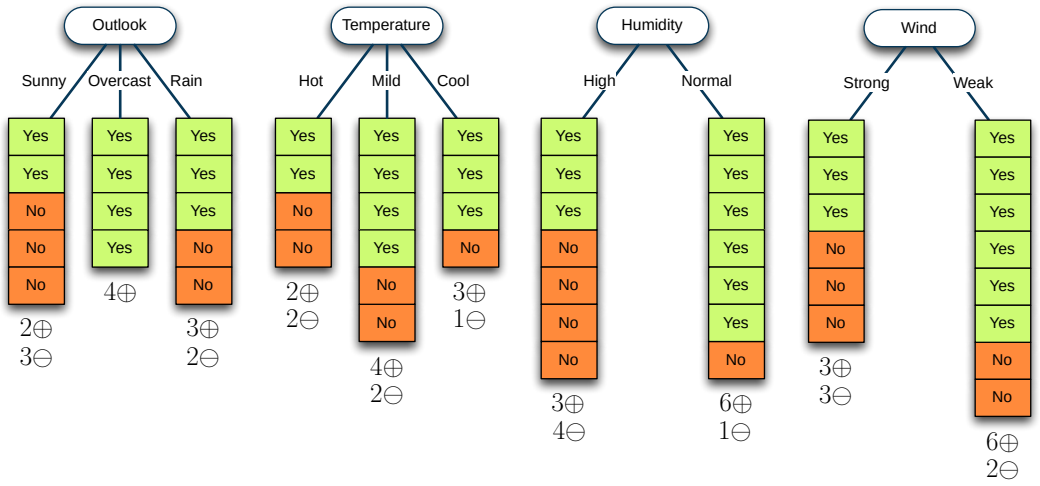
where $Entropy(S)$ is the entropy of the original collection S, $Values(A)$ is the set of all possible values for attribute A and S_v is the subset of S for which attribute A has value v, i.e., $S_v = \{s \in S | A(s) = v\}$ Mitchell [1997] (pp. 57).

The first part of the information gain is the entropy of the original collection while the second part is the sum of entropies of each subset S_v Mitchell [1997]. Let us determine the entropy of the data examples S in Figure 2.10a where we have 9 positive and 5 negative examples (denoted as a set $S = \{S_\oplus, S_\ominus\}$ with $S_\oplus =$

$\{D3, D4, D5, D7, D9, D10, D11, D12, D13\} = 9\oplus$ and $S_\ominus = \{D1, D2, D6, D8, D14\} = 5\ominus$ from table in Figure 2.10a):

$$Entropy(\{9\oplus, 5\ominus\}) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.9403$$

Using this we can calculate for each attribute its information gain and determine the one with the highest information gain as the root node. For the attribute *Outlook* we determine the second part of Equation 2.4 as illustrated in Figure 2.12a.



(a) *Outlook* (b) *Temperature* (c) *Humidity* (d) *Wind*

Figure 2.12: Illustration of proportions of positive and negative examples for the *Play Tennis* concept extracted from the data table in Figure 2.10a. These proportions are used to determine the information gain for each attribute.

$$\begin{aligned}
InfoGain(S, Outlook) &= Entropy(S) - \sum_{v \in \{Sunny, Overcast, Rain\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - \frac{5}{14} Entropy(S_{Sunny}) - \frac{4}{14} Entropy(S_{Overcast}) \\
&\quad - \frac{5}{14} Entropy(S_{Rain}) \\
&= Entropy(S) - \frac{5}{14} Entropy(\{2\oplus, 3\ominus\}) - \frac{4}{14} Entropy(\{4\oplus, 0\ominus\}) \\
&\quad - \frac{5}{14} Entropy(\{3\oplus, 2\ominus\}) \\
&= 0.9403 - \frac{5}{14} \cdot 0.9710 - \frac{4}{14} \cdot 0 - \frac{5}{14} \cdot 0.9710 \\
&= 0.2467 \text{ bits}
\end{aligned}$$

After having determined the information gains for each attribute we choose the

one with the *highest information gain* as the root node. The next step is to determine the intermediate node by calculating the information gain in case $Outlook = Sunny$. For example, if we choose the *Humidity* as the next intermediate node under $Outlook = Sunny$, we get:

$$\begin{aligned} InfoGain(S_{Sunny}, Humidity) &= Entropy(S_{Sunny}) - \frac{3}{5} Entropy(\{0\oplus, 3\ominus\}) \\ &= -\frac{2}{5} Entropy(\{2\oplus, 0\ominus\}) \\ &= 0.9710 \,\text{bits} \end{aligned}$$

Again, we choose the remaining attribute which has the highest information gain as the next intermediate node. This way, it is ensured that one attribute has been used at most once. This process is stopped if all examples can be classified correctly using the current decision tree. This algorithm is called ID3 (Quinlan [1986]) and we refer to Mitchell [1997] for a deeper discussion.

DISCRETIZATION

We have seen the construction process for nominal attributes. For numeric attributes we use *discretization* to split the value range of a numeric attribute (also called continuous-valued attribute) into discrete bins. For example, we could also use a numeric representation for the feature *Temperature* by discretizing it into three categories *hot*, *mild* and *cool* (Nguyen [2004]).

CART

There are also other well known decision tree approaches such as the *Classification and Regressoin Trees* (CART) by Breiman et al. [1984] and C4.5/C5.0 classifiers by Quinlan [1993]. Their differences and similarities are compared in Table 2.1. They appear to be similar: but while ID3 and C4.5/C5.0 store features in the intermediate nodes and its values in the links, the CART approach stores queries in the nodes and the left and right links are indicating whether these queries evaluate to `True` or `False`. During the decision tree induction (also known as the *construction* or the *growth*), the goodness of a feature is determined using statistical measures such as the information gain, the Gini index and normalized information gain. While Gini index measures the deviations between the probability distributions of the classes (Maimon and Rokach [2010]), normalized information gain measures the deviations in entropy values (Quinlan [1993]). Furthermore, *pruning* is used to avoid over-fitting by removing those sub-trees which have little influence on the classification result (Kohavi and Quinlan [1999]). Additionally, the ID3 algorithm does not cope with the problem of *over-fitting*, as discussed in Section 2.3.1. Another difference is that ID3 can not handle numeric features contrary to the others.

PRUNING

2.3.5 Random Forests

In the previous section we have seen what decision trees are and how they are induced. The *Random Forest* algorithm, presented by Breiman [2001], is a statistical classifier popular in many research fields especially in the field of computational biology, computational neuroanatomy and connectomics. As indicated in the name, the *forest* consists of one or multiple *decision trees* (created with CART) which are

RANDOM FOREST

	ID3	CART	C4.5/C5.0
Tree Type	Tree	Binary tree	Tree
Variable Types	Nominal	Nominal, numeric, ordinal	Nominal, numeric
Node Splitting Criterion	Information gain	Gini impurity	Normalized information gain
Pruning Behavior	No pruning	Post-prune tree by testing against un-seen examples	Post-prune tree by replacing branches irrelevant with leaf nodes
Root Node	Feature with highest information gain	Feature with minimum Gini index after splitting	Feature with highest normalized information gain
Over-fitting	Not solved	Solved through post-pruning	Solved through post-pruning

Table 2.1: Qualitative comparison of three decision tree approaches.

built in a specific way by introducing randomness in the creation process. This *set* of prediction models is also called an *ensemble* method. ENSEMBLE

Figure 2.13 illustrates the induction of the random forest on a simple example. From the original dataset of twelve examples we create a forest of six CART trees (see dark blue root nodes) each created using a randomly sampled, equally sized data set (dotted frames around data sets). In this context, *randomly sampled* means that a randomly chosen example might occur multiple times in the same dataset while others might not occur at all. Indeed, the left out examples, called *out-of-bag* (OOB) data, do serve a purpose: they are used to get an unbiased OUT-OF-BAG estimate of the classification error. This is achieved by cross-validating the trees during, rather than after induction, in order to obtain a model which is prone to variance (Breiman [2001]). The randomness is illustrated as differently distributed *membrane* (green) and *non-membrane* examples (orange) in each random dataset. The randomization in general is a way to reduce the impact of noise in the data. The resulting leaf nodes contain not only strict classification decision of *positive* and *negative* but the corresponding confidence values (see the shading between green and orange in leaf nodes). The randomness is also introduced during the selection of feature splits. For each splitting decision, we select randomly a predefined number of features $m \in M$ from the original feature pool of M.

After the random forest induction step we now have a number of decision tree

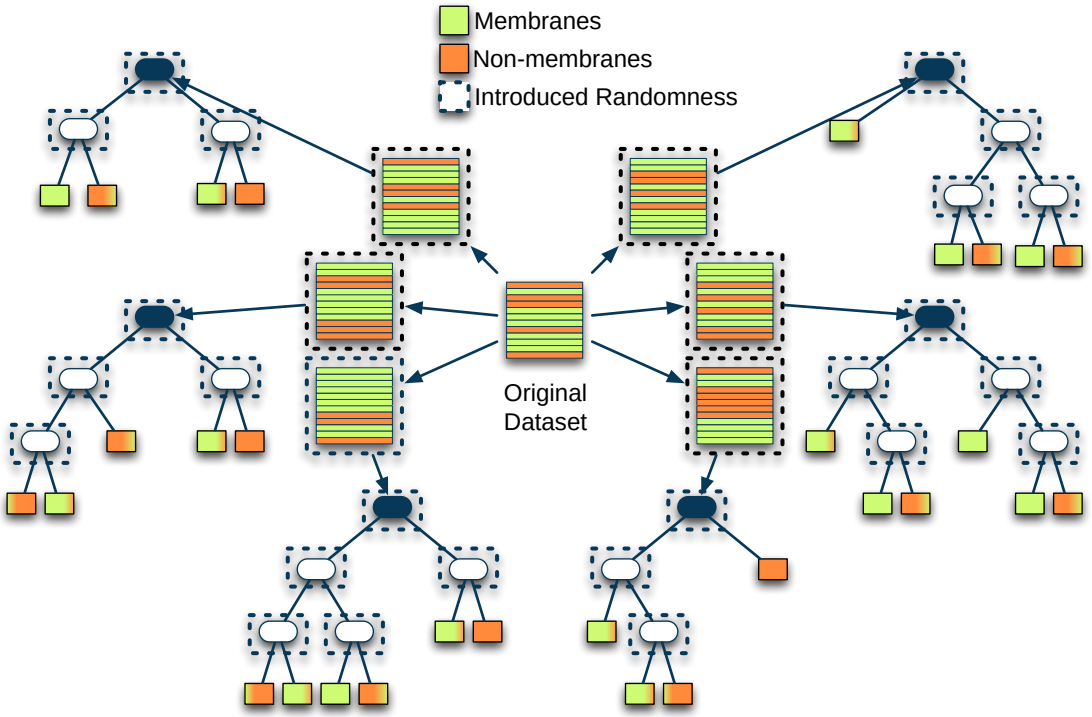


Figure 2.13: Illustration of random forest induction on a simple example. The six CART trees, created with six randomly sampled data sets, are forming the random forest.

CLASSIFICATION

models packed in an ensemble. The *classification* using the random forest ensemble is illustrated in Figure 2.14. A new (unseen) example is traced through each of the six decision trees and its outcome is noted. The prediction made by the random forest is the unweighted plurality (i.e., the relative majority) vote (Breiman [2001]) of all trees expressed as a confidence value. It is the class label with the highest total number of trees that predicted it. In the case of Figure 2.14 four trees are predicting the class *positive* and two are predicting the class *negative*. Therefore, the prediction of the whole random forest is *positive* with a certainty $p_{\oplus} = 0.6667$.

The random forest has many desirable properties (from Breiman [2001]). First, it provides a classification accuracy which is similar to other state-of-the-art approaches such as Support Vector Machines and C4.5/C5.0. Second, it can handle thousands of variables, missing values and unbalanced data sets (i.e., those where the class label frequencies differ greatly). Third, it avoids biased estimates through construction of full trees. Fourth, it avoids over-fitting by having multiple trees in the ensemble which are created using randomly sampled datasets and which decide in ensemble rather than using only one tree.

2.4 Previous Work on Feature Selection

Let us discuss previous work in feature selection, creation and weighting. A general overview on the topic of feature selection is provided by Guyon and Elisseeff

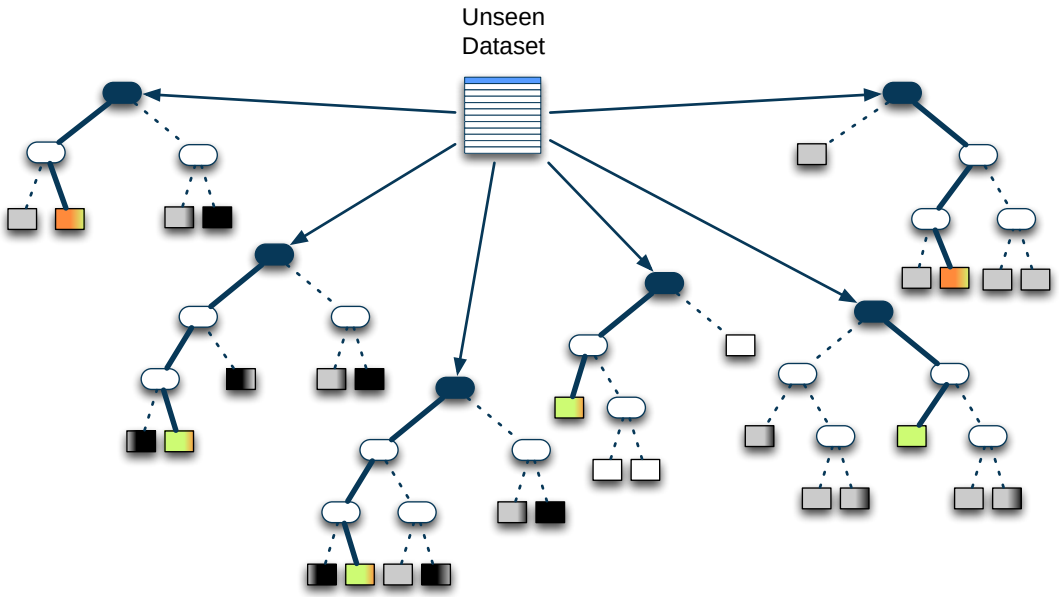


Figure 2.14: Classification of an unseen data example (light blue). The sample is traced through each tree and its outcome is noted (colored leaf nodes). This random forest predicts the class *positive* with a certainty of $p_{\oplus} = \frac{4}{6} = 0.6667$.

[2003]. The authors state that there are a couple of benefits to feature selection, such as: simpler data lead (a) to better data understanding and utilizes expressive visualization, (b) to reduced induction and execution times of machine learning algorithms, (c) to reduced storage requirements which is of special importance in the field of ssTEM image segmentation, and (d) to improved prediction performance due to the avoidance of the curse-of-dimensionality. It is important to note, that the feature selection algorithms proposed in the literature do not rely on domain-specific knowledge but only follow statistical and information theoretic guidelines. After all, the machine learning algorithms should be general purpose algorithms and not bound to a specific domain. The proposed feature selection methods differ in their output (Guyon and Elisseeff [2003]). While some lead to a sub-set of features relevant to the task, others output a ranking of all potentially relevant features. Guyon and Elisseeff [2003] state also that there might be a tradeoff between the relevance and the redundancy of features. The selection of the most relevant features is not preferred if they are redundant and just because a feature is redundant does not necessarily mean that they are not relevant. Yu and Liu [2004] propose a feature selection algorithm based on the analysis of the feature relevance and the feature redundancy.

There are of course statistical measures for redundancy and relevance of features as well as information theoretic ranking criteria. All these measures use the underlying feature values as a representation of the real feature value distribution which in general remains unknown. Auffarth et al. [2010] provide a discussion on measures for the selection based on the redundancy and relevance of features for

the segmentation of computer tomography images.

The aim of feature selection is to reduce the redundancies between features while at the same time increase the relevance of the feature set with regard to the classification task. Blum and Langley [1997] identifies the feature selection as a *heuristic search* which can be viewed from two different points of view: forward selection and backward elimination. While the *forward selection* starts with an empty feature set and adds features successively, the *backward elimination* starts with a full feature set and removes them from the consideration successively. We can evaluate the alternative feature sub-sets by some metric which shows how well a feature discriminates among classes in the training sample, as discussed in Section 2.3.3. The traversal of this search space of 2^M possible feature sub-sets of M features is not practicable. One solution to this problem is a greedy search which only considers the local changes when adding a feature to the feature set (Blum and Langley [1997]) and halts if no further improvement is possible.

HEURISTIC SEARCH
FORWARD SELECTION
BACKWARD ELIMINATION

2.4.1 Feature Relevance

RELEVANCE
STRONGLY RELEVANT
WEAKLY RELEVANT
IRRELEVANT

Peng et al. [2005] formally define three forms of feature *relevance*: strongly relevant, weakly relevant and irrelevant. A feature is called *strongly relevant* if the feature is always necessary for an optimal sub-set. A feature is called *weakly relevant* if the feature is not always necessary. By contrast, *irrelevant* features are those which are not necessary at all. Following this definitions, an optimal subset should include all strongly relevant features, a (not further declared) sub-set of the weakly relevant features and no irrelevant features. One of the central questions here is which weakly relevant features are to be removed and which ones to be selected. One answer to this question provides the concept of feature redundancy.

2.4.2 Feature Redundancy

REDUNDANCY

In machine learning literature, *redundancy* is defined formally by Peng et al. [2005]. However, in this work we concentrate on the intra-class covariance-based definition of redundancy (Guyon and Elisseeff [2003]). A feature F_i is called redundant if there exists a second feature F_j which correlates with F_i for a class label. The most commonly known correlation measure is the linear *correlation coefficient* for two variables (X, Y) defined as

CORRELATION COEFFICIENT

$$\rho = \frac{\sum_i (x_i - \overline{x}_i)(y_i - \overline{y}_i)}{\sqrt{\sum_i (x_i - \overline{x}_i)^2} \sqrt{\sum_i (y_i - \overline{y}_i)^2}}\,, \tag{2.5}$$

where x_i is the i-th value of feature X, $\overline{x}_i$ is the mean of variable X and y_i is the i-th value of feature Y, $\overline{y}_i$ is the mean of variable Y. In case that $\rho = 0$ the variables X and Y are called *uncorrelated* and for $\rho = \pm 1$ they are perfectly correlated. The denominator in Equation 2.5 is also called *covariance* (Guyon and Elisseeff [2003]). We will discuss feature redundancy extensively in Chapter 4.

UNCORRELATED

Chapter 3

Feature Pre-computation

> *"In general, a feature is* good *if it is* relevant *to the class concept but is not* redundant *to any of the other relevant features."*
>
> Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution
> Lei Yu and Huan Liu (p. 858)

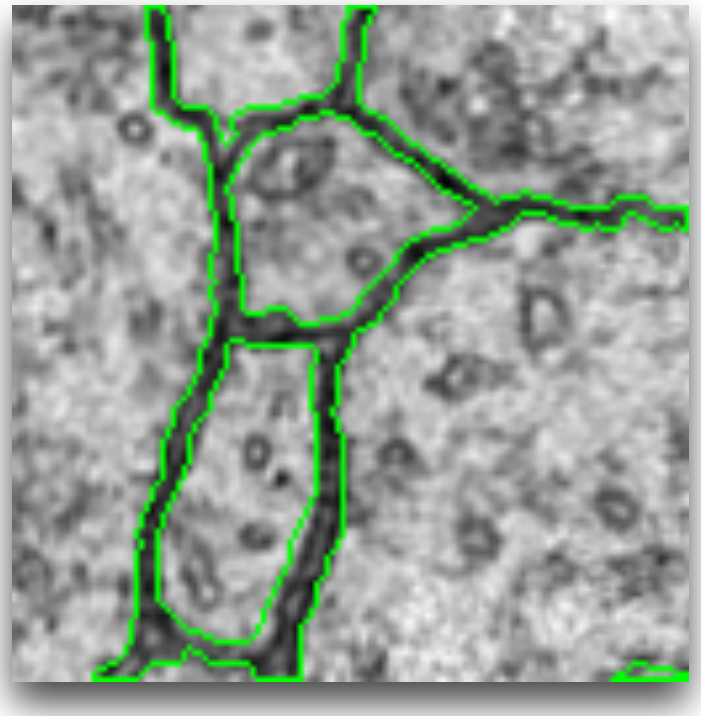
In Chapter 2 we have seen that for the ML-based segmentation it is necessary to transform the image in a tabular form where each row corresponds to a pixel and the set of columns to the feature set of that pixel. The obvious pixel feature is the pixel intensity. Any other feature needs to be derived from the intensity image. There is no spatial relationship between rows in the data table but there is one between neighboring pixels. This spatial relationship needs to be transformed from the image into the machine learning domain? In general, machine learning methods are unaware of the spatial relationships between pixels, the point spread function which determines the shape of each real-world object on an image and the nature of the image.

In Section 3.1 we discuss methods to create new features from the original density feature. The pre-computed features are split into four categories: (a) the original density feature discussed in Section 3.2, (b) the rotation-invariant membrane matching features discussed in Section 3.3, (c) the per-pixel local histogram features discussed in Section 3.4, and (d) the feature created using gaussian smoothing discussed in Section 3.5.

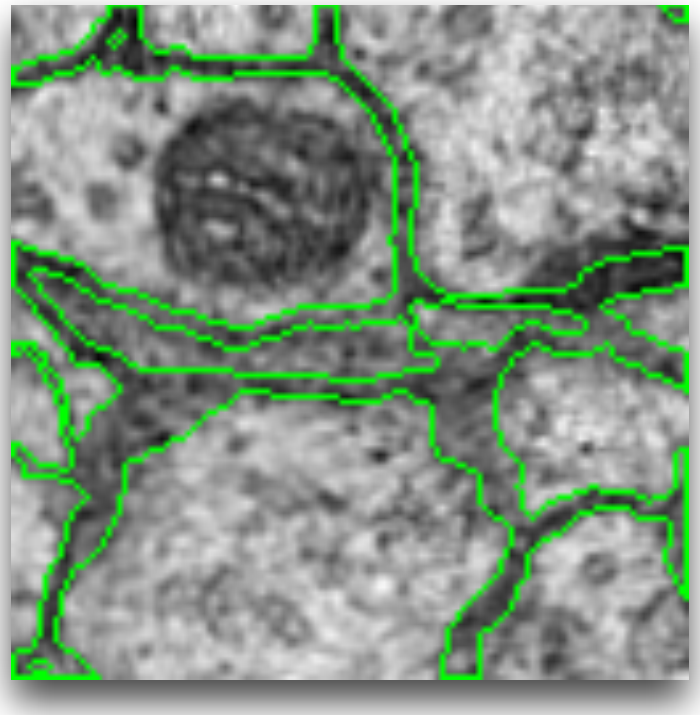
3.1 Introduction

The density (in volumes), or intensity feature (in images), shows the measured image of a real world object. In general, we know that image segmentation using only the intensity image leads to poor results. For example, a value range of $[20, \ldots, 60]$ does not only correspond to a membrane in ssTEM brain images but also to mitochondria (see dark blob in Figure 3.1b). A *membrane* in ssTEM brain images is indicated by low intensity pixels while *white matter* (also known as *cell background*) consists of high intensity pixels and it is surrounded by other

MEMBRANE



(a) Membranes vs. white matter



(b) Compression artifacts

Figure 3.1: Two examples of ssTEM brain tissue images with membrane ground truth shown as green perimeter lines. (a) A cleanly sliced section and (b) compression artifacts which lead to smeared membranes.

white matter objects (like synapses, axons or neurons). The regions between white matter objects are the membranes (see dark ribbons whose perimeter is highlighted in green in Figure 3.1a).

In general there are two domains in which we are able to process images: the spatial domain and the frequency domain. The *spatial domain* refers to the image plane itself in which the pixels are directly manipulated (Gonzalez and Woods [2006], see Figure 3.2). The spatial domain of a $S_x \times S_y$ image is the image

SPATIAL DOMAIN

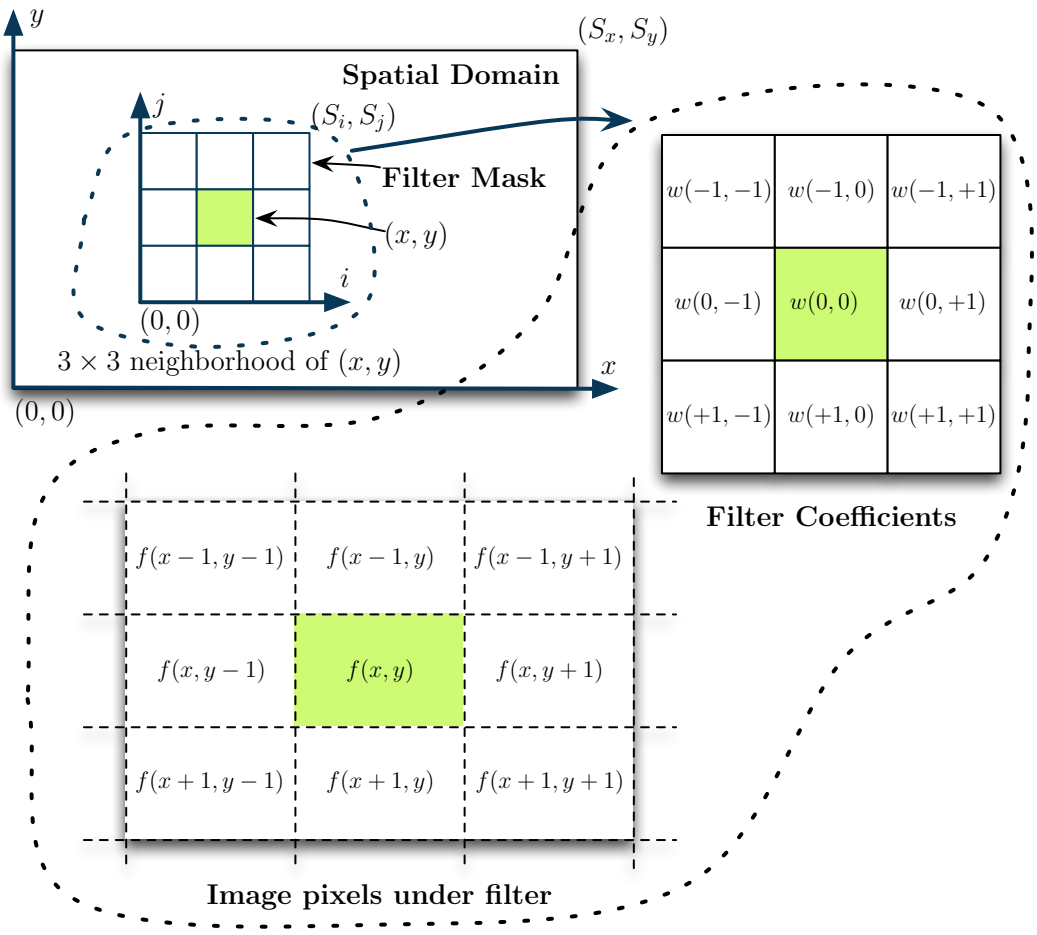


Figure 3.2: Common components in spatial domain (adopted from Gonzalez and Woods [2006]).

plane itself. A pixel (green) at the position (x, y) with $0 \leq x \leq S_x - 1$ and $0 \leq y \leq S_y - 1$ has a $S_i \times S_j$ neighborhood, in this illustration $S_i = S_j = 3$. For this neighborhood we can use a different coordinate system with the coordinates (i, j) whre $0 \leq i \leq S_i - 1$ and $0 \leq j \leq S_j - 1$, as shown in Figure 3.2.

The spatial domain algorithms can be further split into two categories: *intensity transformations* which manipulate single pixels (e.g., for contrast manipulation) and the *spatial filterings* which take the local neighborhood of each pixel into account (e.g., for blurring). Figure 3.2 also shows a filter mask (also known as *filter*

INTENSITY TRANSFORMATIONS

SPATIAL FILTERINGS

kernel) w around the pixel at (x, y). w has its own filter coefficients which are offsets in x and y-direction. A filter mask is laid over the image f in such a way that $w(0, 0)$ is right above f.

In the following we discuss general methods used for the creation of new features for the segmentation of membranes. Especially, local histograms and Gaussian filtering are used to create new features.

3.1.1 Intensity Histograms

Intensity histograms (or short *histograms*) are useful tools in digital image processing with which different intensity transformations become possible. The histogram can be defined as a function:

DEFINITION 3.5 (Histogram Function)

Gonzalez and Woods [2006] defines the *histogram function* of an $S_x \times S_y$ image with pixel intensity values in the range $[0, L-1]$ as a discrete function

$$h(v_k) = n_k \,, \tag{3.1}$$

where v_k is the k-th intensity value and n_k is the frequency of pixels with intensity v_k in the image. Furthermore, it is common to *normalize* the histogram by dividing each n_k by the number of all pixels $p(v_k) = \frac{n_k}{N \cdot M}$.

If, for example, the value range of pixel intensities is real we can discretize it into N_b histogram bins in such a way that v_k is not a value but a value range. If the number of bins N_b is less than the number of values we call it a *course scale* in contrast to *fine scale* histograms. The image histogram can be used to modify pixels based on the intensity distribution of the entire image (Gonzalez and Woods [2006]).

PER-PIXEL LOCAL HISTOGRAMS

As mentioned before, the membrane pixels in ssTEM brain tissue images have local properties such as (a) belonging to an elongated region of the same or similar pixel intensity and (b) being surrounded by white matter regions. Such knowledge can not be encoded by a global image histogram. Therefore, an adaption to histograms namely the *per-pixel local histograms* (also known as *per-pixel region histogram*) are more suitable to represent local neighborhood information, as illustrated in Figure 3.3. For each pixel (x, y) we consider its immediate neighborhood $f(i, j)$ with $i \in I$ and $j \in J$ where $I = J = \{-\frac{c_s}{2}, -\frac{c_s}{2}+1, -\frac{c_s}{2}+2, \ldots, -1, 0, 1, \ldots, +\frac{c_s}{2}-2, +\frac{c_s}{2}-1, +\frac{c_s}{2}\}$, with $c_s \in \mathbb{N}$. The immediate neighborhoods are shown as orange regions of size $c_s \times c_s$. For the immediate neighborhood we determine the histogram of that neighborhood for N_b bins (in Figure 3.3 $N_b = 6$ with $b_1, \ldots, b_6$). The histogram information can be transformed into N_b per-pixel local histogram images $h_1, \ldots, h_{N_b}$ illustrated on the right.

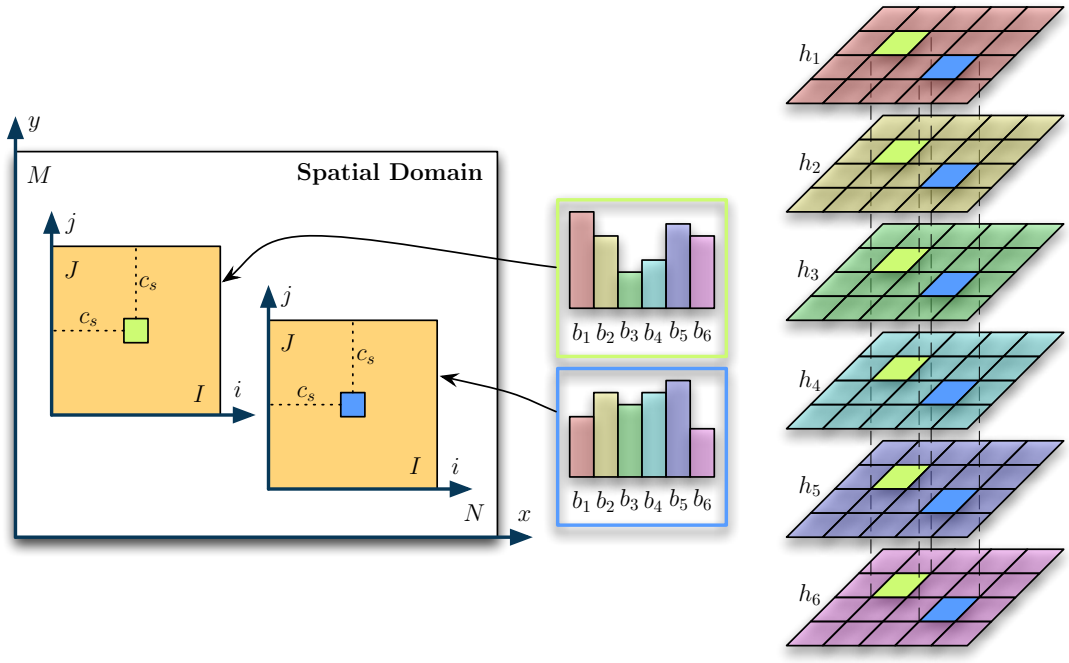


Figure 3.3: Per-pixel local histograms to capture neighborhood information (yellow region) of pixels (green and blue). As result we get for each pixel a histogram with six bins which are transformed into six per-pixel local histogram images $h_1, \ldots, h_6$.

3.1.2 Spatial Filtering

The second spatial domain transformation category is the filtering. Gonzalez and Woods [2006] describe the *filter* as follows: FILTER

> *"The name* filter *is borrowed from frequency domain processing, ... where "filtering" refers to accepting (passing) or rejecting certain frequency components."* (p. 167)

If we pass low frequencies (i.e., low-pass filter) we get a blurred image. This can be achieved either in frequency or spatial domain. The spatial domain offers in terms of filtering more benefits which is discussed by Gonzalez and Woods [2006].

Image filtering is a part of image pre-processing which indeed decreases the image information content but in many situations helps to rule out non-relevant information specific to a task (Sonka et al. [2007]). Image filtering is used for example to remove noise (e.g., median or averaging filter), to enhance the edges of image objects (e.g., Canny edge detector) or to match a template to an image. Especially, two concepts are of special interest for the spatial domain filtering: the correlation and the convolution. While *convolution* is used, for example, for smoothing of an image f by convolving the image with Gaussian filter, *correlation* is used to match an template image in f. CONVOLUTION CORRELATION

Correlation describes the movement of the filter mask over the image and the summation of products at each location (Gonzalez and Woods [2006]). Convolu-

tion, on the other hand, works the same except that the filter (or the image) needs to be rotated 180° first. This concept is illustrated for the case of image filtering in Figure 3.4. Figure 3.4a shows a simple example image f with a single pixel set to one. Such a function is also called a *discrete unit impulse function*. To avoid the handling of the image margins in the definition of convolution or correlation we extend the image by 0-padding and get $\hat{f}$ (see Figure 3.4b). $w_\star$ and w_* denote the filter masks used in case of correlation and convolution respectively and are shown in Figure 3.4c. Please note that w_* is a 180° rotated version of $w_\star$.

First, we start at the image origin $(0, 0)$ and apply the correlation and convolution (see Figures 3.4d and 3.4g). As a result we crop the final result images (see Figures 3.4f and 3.4i). Formally, the correlation of a padded image $\hat{f}$ with a filter mask $w_\star$ of size $n \times m$ is defined as

$$w_\star \star \hat{f} = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w_\star(i,j) \hat{f}(x+i, y+j)\,, \tag{3.2}$$

where $a = (n-1)/2$ and $b = (m-1)/2$ for a n and m being odd integer numbers. On the other hand, the convolution is defined as

$$w_* * \hat{f} = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w_\star(i,j) \hat{f}(x-i, y-j) \quad \text{or} \tag{3.3}$$

$$w_* * \hat{f} = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w_*(i,j) \hat{f}(x+i, y+j)\,. \tag{3.4}$$

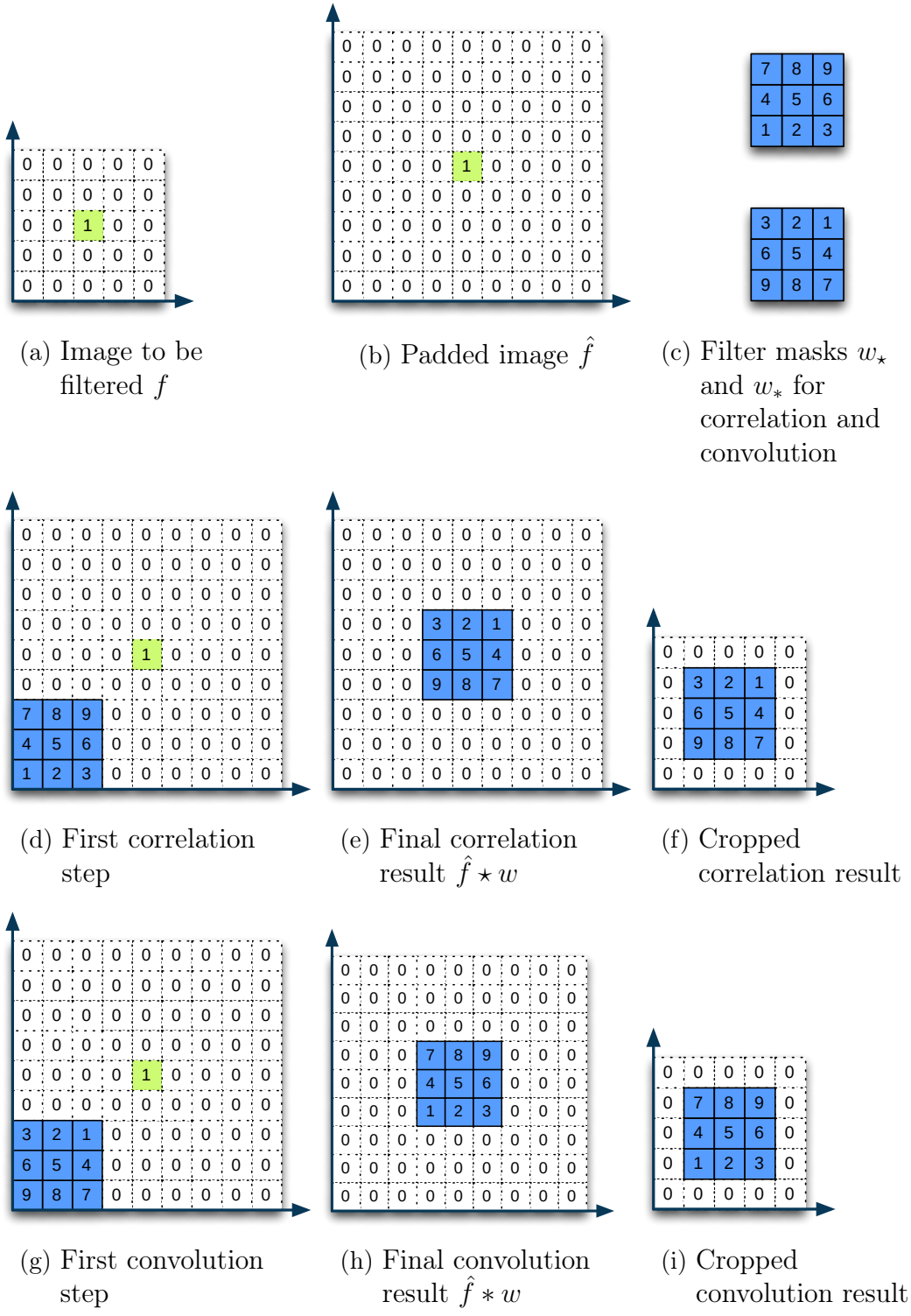
The minus signs correspond to $\hat{f}$ being rotated by 180° (see Equation 3.3; Gonzalez and Woods [2006]). We get the same result by rotating $w_\star$ and not $\hat{f}$ (see Equation 3.4). The main difference between correlation and convolution is that convolution is associative which means that for w_1 and w_2 being filter masks

$$w_1 * \left(w_2 * \hat{f}\right) = \left(w_1 * w_2\right) * \hat{f}\,. \tag{3.5}$$

Let us consider a low-pass filter in the spatial domain. A low-pass filter is a smoothing filter which averages the pixel intensities contained in the neighborhood of a pixel position (x, y) (Gonzalez and Woods [2006]). The filter mask defines the neighborhood and the coefficients which are the scalar values contained in the filter mask (for example, the coefficients in Figure 3.4c are $1, 2, \ldots, 9$). One low-pass filter is the *Gaussian smoothing filter*. The smoothing of an image f with Gaussian filter G is the same as convolving the image with the bi-variate Gaussian distribution function: $G * f$.

GAUSSIAN SMOOTHING FILTER

Figure 3.5 shows the bi-variate Gaussian distribution function that is applied to each pixel. The bi-variate Gaussian coefficients $G(x, y)$ can be considered as *weights* to be used on the image pixels. These weights are computed using



(a) Image to be filtered f

(b) Padded image $\hat{f}$

(c) Filter masks $w_\star$ and w_* for correlation and convolution

(d) First correlation step

(e) Final correlation result $\hat{f} \star w$

(f) Cropped correlation result

(g) First convolution step

(h) Final convolution result $\hat{f} * w$

(i) Cropped convolution result

Figure 3.4: Filtering toy example which uses a discrete unit impulse function f for correlation and convolution with filter masks $w_\star$ and w_* respectively (adopted from Gonzalez and Woods [2006]).

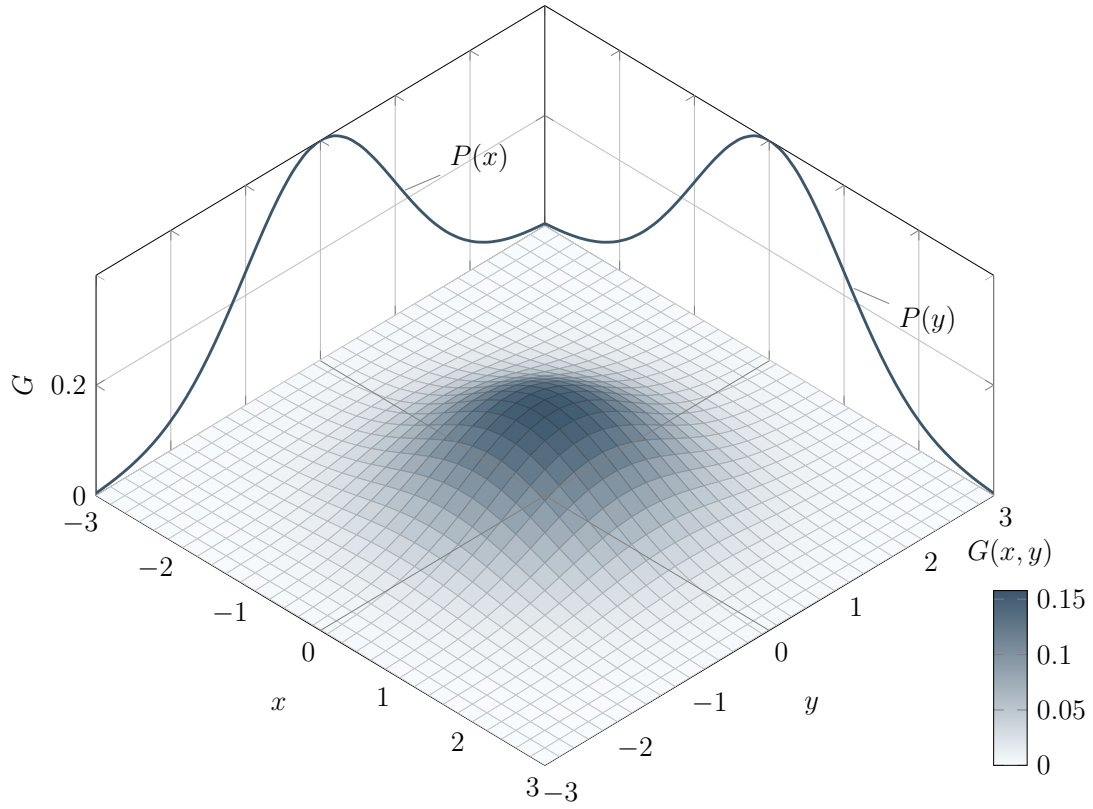


Figure 3.5: Bi-variate Gaussian distribution used as weighting function for the Gaussian smoothing filter.

$$G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left[\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - \frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y}\right]\right] \tag{3.6}$$

where ρ is the correlation between $x \in X$ with $x \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and $y \in Y$ with $y \sim \mathcal{N}(\mu_y, \sigma_y^2)$. In our case $\rho = 0$. μ_x and μ_y are the means and σ_x^2 and σ_y^2 are the variances of uni-variate Gaussian distributions $\mathcal{N}(\mu_x, \sigma_x^2)$ and $\mathcal{N}(\mu_y, \sigma_y^2)$ respectively. This means, that the highest weights are centered at $(0, 0)$ of the bi-variate Gaussian distribution while the lower weights are those farther away from the centre. Discrete Gaussian filter masks G_σ are shown in Figure 3.6 as textures of different sizes and different standard deviations σ where $\sigma = \sigma_x = \sigma_y = 1, 2, \ldots, 10$.

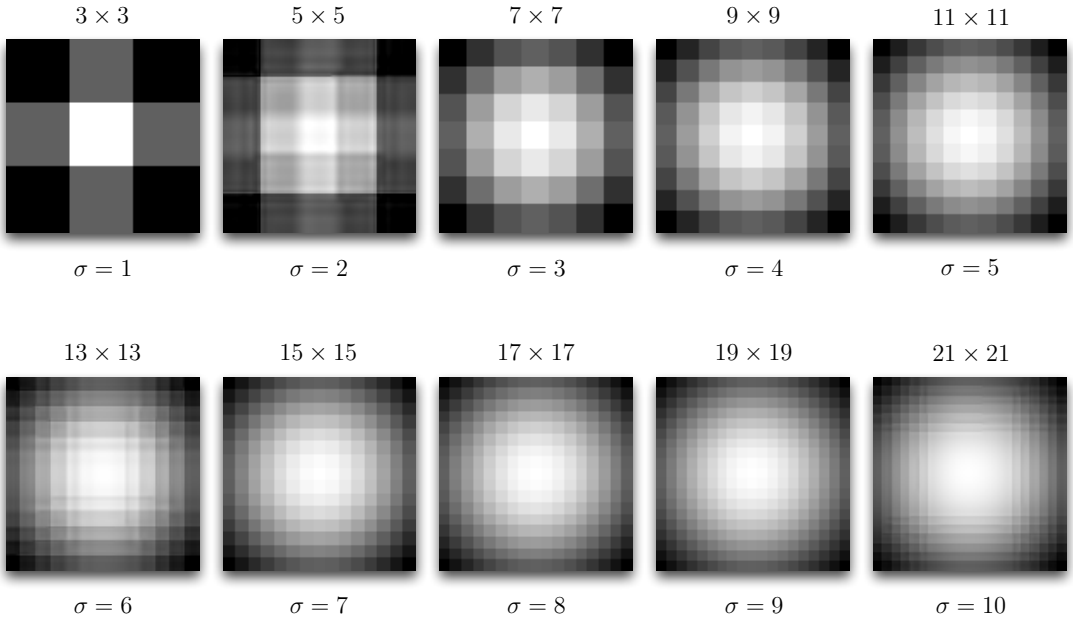


Figure 3.6: Gaussian filter masks G_σ for different standard deviations σ. The mask image size in pixels is determined by $(2\sigma+1)\times(2\sigma+1)$. All mask images are scaled to the same size.

3.1.3 Gradients

In image processing gradients are used to represent numerous informations. We can define the *gradient* as follows:

GRADIENT

DEFINITION 3.6 (Gradient Vector)

The *gradient vector* at position (x, y) of a function f is defined as

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{3.7}$$

where $\frac{\partial f}{\partial x}$ is the partial deviation of f in x, also denoted as f_x.

In image terms, the gradient vector points in the direction of the greatest rate of change at location (x, y) (Gonzalez and Woods [2006]). Furthermore, the length of the gradient vector is called the *gradient magnitude* and is defined as follows (Gonzalez and Woods [2006]):

GRADIENT MAGNITUDE

DEFINITION 3.7 (Gradient Magnitude)

The *gradient magnitude* of a gradient vector at position (x, y) is defined as

$$\text{GradMag}(x, y) = ||\nabla f|| = \sqrt{g_x^2 + g_y^2}\,. \tag{3.8}$$

The length $\text{GradMag}(x, y)$ of the gradient vector denotes the rate of change in the direction of that vector. Because $\text{GradMag}(x, y)$ is an image of real values we refer to this image shortly as the *gradient image* (Gonzalez and Woods [2006]).

GRADIENT IMAGE

3.1.4 Structure Tensors

The local neighborhood is of special interest in other image processing approaches as well. One example is the *structure tensor* which uses a local neighborhood around a point (x, y) to describe the intensity value distribution (i.e., the local structure information) at that point (Weickert and Hagen [2006]). Traditionally, we can estimate the local orientation with the help of gradient vectors, as discussed in Section 3.1.3. But, gradient vectors alone are not able to distinguish a corner from an edge. This leads to a more powerful tool, namely the structure tensors which Brox et al. [2006] describe as follows:

STRUCTURE TENSOR

> *"The structure tensor therefore extends the structure information of each pixel, which is described in a first order approximation by the gradient at that pixel, by the structure information of its surroundings weighted with a Gaussian window function. This comes down to the convolution of the structure data with a Gaussian kernel, i.e. Gaussian smoothing."* Brox et al. [2006] (p. 18).

In this sense, the structure tensor for an image f can be defined as follows:

DEFINITION 3.8 (Structure Tensor)

Brox et al. [2006] define the *structure tensor* $\mathcal{T}$ of a certain neighborhood of scale τ of an image f as the convolution of the Gaussian kernel G_τ with the so-called *initial matrix field*

$$\mathcal{T}_\tau = \mathcal{G}_\tau * \begin{bmatrix} f_x f_x & f_x f_y \\ f_x f_y & f_y f_y \end{bmatrix} = \begin{bmatrix} G_\tau * f_x^2 & G_\tau * f_x f_y \\ G_\tau * f_x f_y & G_\tau * f_y^2 \end{bmatrix}, \tag{3.9}$$

where the initial matrix field is the outer product of the gradient image of the image f

$$\begin{bmatrix} f_x f_x & f_x f_y \\ f_x f_y & f_y f_y \end{bmatrix} = \nabla f \nabla f^\top . \tag{3.10}$$

$f_x = \partial_x f$ and $f_y = \partial_y f$ are the partial derivatives of f.

From the definition we see that the structure tensor incorporates a smoothing operation which integrates the neighborhood information. Since $\mathcal{T}$ is a positive definite matrix it has two non-negative eigenvalues λ_1 and λ_2. The local gradient properties are estimated using the eigenvectors by determining the dominant orientation as the eigenvector of the largest eigenvalue. The eigenvalues are a measure of dominance in orientation. The direction with the smallest change is the eigenvector with the smallest eigenvalue of the structure tensor (Weickert and

Hagen [2006]). For further discussion on structure tensors we refer to Weickert and Hagen [2006] or Jähne [2005].

3.2 Density Feature

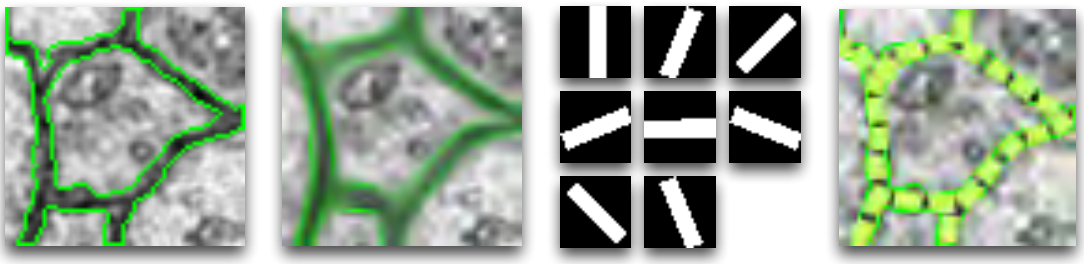
POINT-SPREAD FUNCTION

The quality of an intensity image f is influenced by the *point-spread function* which describes a two-dimensional distribution of light (or electrons in ssTEM images) of a light point (or a point object which scatters the electrons in ssTEM images) on the focal plane. In addition to the artifacts coming from the point spread function we also have sectioning artifacts which lead to washed out regions in ssTEM brain tissue images. Figure 3.1 on page 34 shows two sub-images of the same slice of the ssTEM volume of the drosophila fly brain. The green lines indicate the perimeter of the membrane ground truth. While Figure 3.1a shows a cleanly sliced section where the membranes have similar widths, Figure 3.1b shows washed out membranes where it is difficult to decide the real width of a membrane.

3.3 Rotation-invariant Membrane Matching

In ssTEM brain tissue images the most membranes can be characterized as connected elongated regions having similar thickness and intensity values. The idea behind membrane matching is to extract a template image which represents most parts of the membranes and find regions in the original ssTEM image which are similar to this template image.

In Figure 3.7, the density image f contains membranes (see Figure 3.7a) that can be described by ribbons (see Figure 3.7b). Each ribbon has a certain width and



(a) Original (b) Ribbons (c) Ribbon segments (d) Approximated ribbons

Figure 3.7: Ribbon analogy used to create features that contain information about membrane matching.

can be connected with other ribbons. This ribbon analogy can be simplified by an equally wide, straight ribbon segment which can be fitted onto the real membrane by duplication, translation and rotation. This simplified ribbon is described by a number of *template images* T_k which are illustrated in Figure 3.7c. The template images are created by rotating the top-left black-and-white ribbon segment image

TEMPLATE IMAGES

around the centre of the image with an angle $\varphi_k = \frac{k \cdot \pi}{8}$, for $k = 0, \ldots, 7$. The rotated template is then used as a template image to recognize similar ribbon segments in the density image. A popular method for object recognition is the *matching by correlation* which uses *normalized cross-correlation* (NCC) (Gonzalez and Woods [2006])

MATCHING BY CORRELATION

NCC

$$\gamma(x,y) = \frac{\sum_i \sum_j \left[T_k(i,j) - \overline{T}_k\right] \left[f(x+i, y+j) - \overline{f}_{xy}\right]}{\sqrt{\sum_i \sum_j \left[T_k(i,j) - \overline{T}_k\right]^2 \sum_i \sum_j \left[f(x+i, y+j) - \overline{f}_{xy}\right]^2}} \tag{3.11}$$

where T_k is the template image rotated by φ_k with $k = 0, \ldots, 7$, and i and j are the indices of the pixels in the template image. $\overline{T}_k$ is the mean of the template and $\overline{f}_{xy}$ is the mean of f in the region coincident with T_k (Gonzalez and Woods [2006]). NCC is also a two-dimensional version of the correlation coefficient discussed in Section 2.4.2 on page 32. As a result of NCC we get images $R_0, \ldots, R_7$, shown in Figure 3.8, containing the NCC coefficients $\gamma(x, y)$ for specific template T_k for all pixels. The darker the pixel the identical T and f are in that pixel. Furthermore, six additional feature images are computed from these NCC coefficient images where

$$\begin{aligned} Min_R &= min(R_0, \ldots, R_7) \\ Max_R &= max(R_0, \ldots, R_7) \\ \mu_R &= mean(R_0, \ldots, R_7) \\ \sigma_R^2 &= var(R_0, \ldots, R_7) \\ Median_R &= median(R_0, \ldots, R_7) \\ Diff_R &= Max_R - Min_R\,. \end{aligned}$$

3.4 Per-pixel Local Histograms

In Section 3.1.1 we discussed that global image histograms are not sufficient to capture local distributions of the pixel intensities in an image. We can capture the local properties of a membrane pixel in such a way that for N_b bins we create N_b different histogram images where the first h_1 holds frequencies of lowest intensity pixel values and the h_{N_b} histogram image contains local frequencies of the highest intensity pixel values. The resulting per-pixel local histogram features are shown in Figure 3.9.

3.5 Smoothing

Another possibility to integrate local neighborhood information is by smoothing (also known as *blurring*) the image with a corresponding filter. In Section 3.1.2 we discussed that applying a smoothing filter on an image is the same as convolving the image with the filter mask. Popular smoothing filters include the Gaussian filter, the normalized box filter, the median filter and the bilateral filter. For

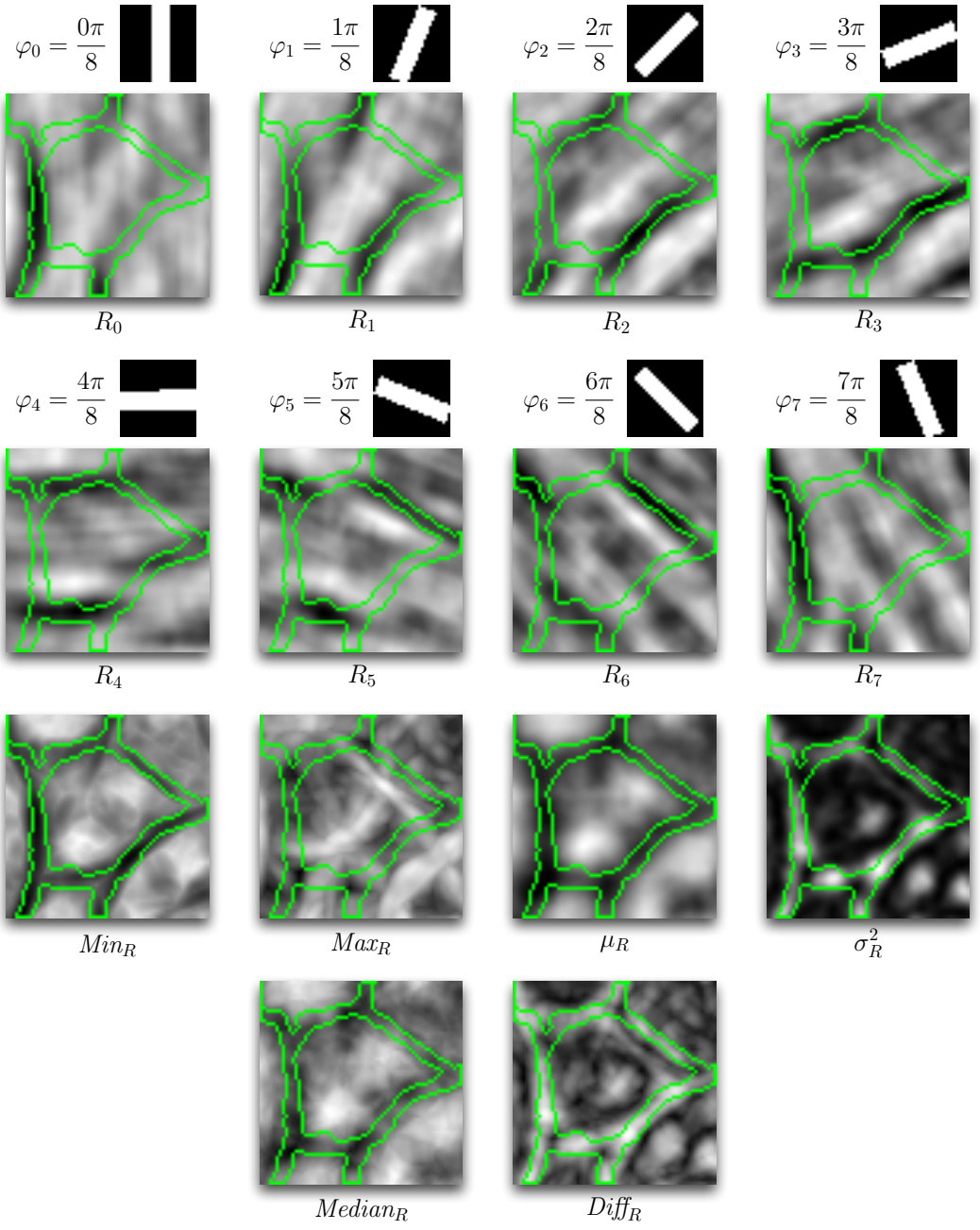
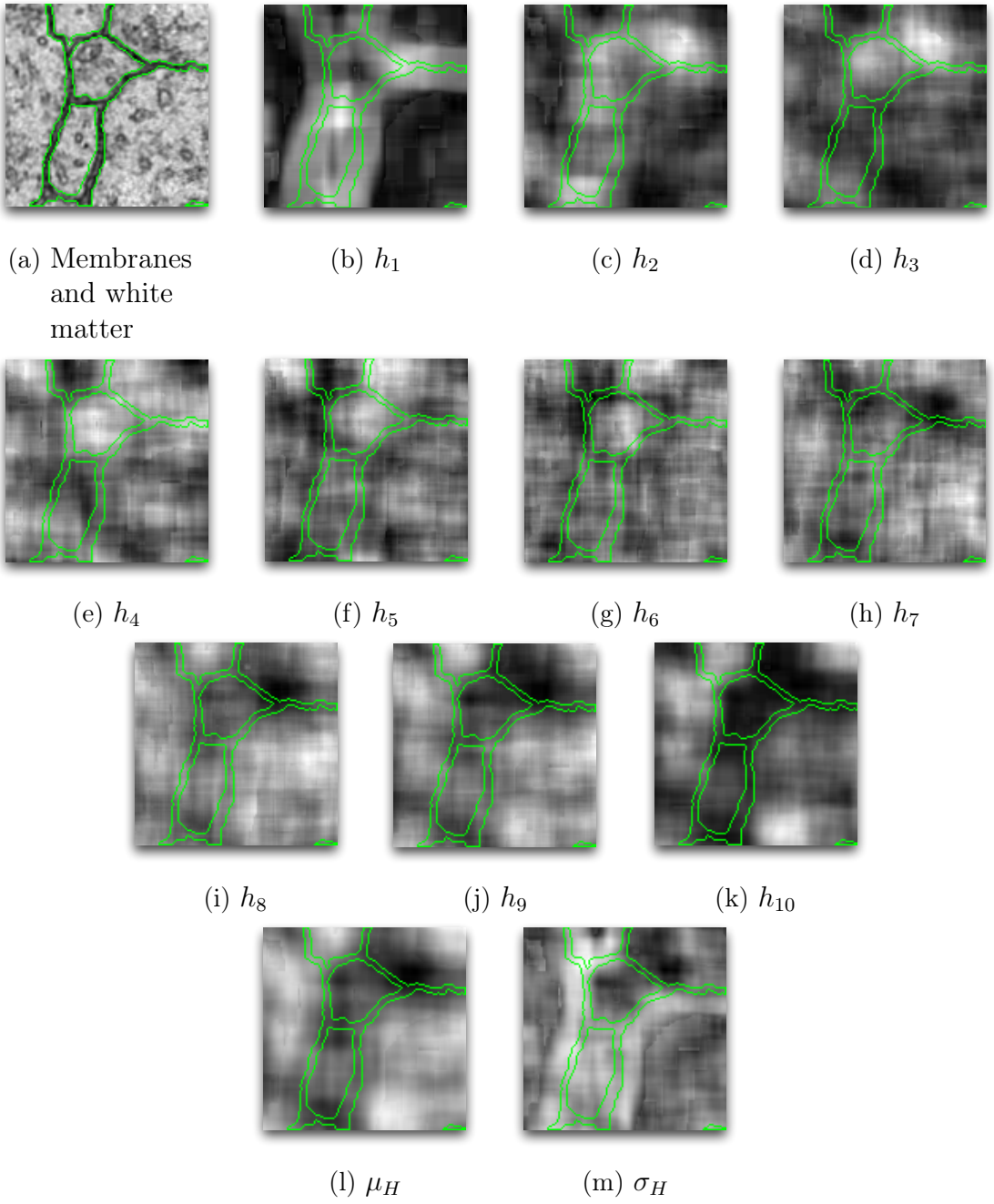


Figure 3.8: Rotation-invariant membrane filtering uses the original density image f to produce images $R_0, \ldots, R_7$. The last two rows Min_R, Max_R, μ_R, σ_R^2, $Median_R$ and $Diff_R$ show combined features of these images.



(a) Membranes and white matter (b) h_1 (c) h_2 (d) h_3

(e) h_4 (f) h_5 (g) h_6 (h) h_7

(i) h_8 (j) h_9 (k) h_{10}

(l) μ_H (m) σ_H

Figure 3.9: Per-pixel local histograms split into ten histogram features $h_1, \ldots, h_{10}$ with its combinations μ_H and σ_H. The subscript H denotes that μ_H and σ_H are determined using per-pixel local histogram features.

simplicity, we use simple Gaussian filtering to create new feature images.

In general, we can smooth any image we have produced so far. For this purpose, we use Gaussian filter masks G_σ shown in Figure 3.6 on page 41 with ten different standard deviations $\sigma = 1, \ldots, 10$ and sizes, as well as different images for the smoothing. The results are shown in Figure 3.10. The Gaussian smoothing filter (see Figure 3.10a) is applied to all four images with varying standard deviations and filter sizes. Figure 3.10b shows smoothing results of the original ssTEM brain tissue intensity image f. Figure 3.10c shows the smoothing results for the component-wise division of the two eigenvalue images λ_1/λ_2 determined by the structure tensor discussed in Section 3.1.4. Furthermore, Figure 3.10 contains the resulting images where the gradient magnitude image is determined for the smoothed intensity image $G_1(f)$ (G_1 is same as G_σ with $\sigma = 1$) and then smoothed again with G_σ (see Figure 3.10d), as well as the images where the gradient magnitude determined for each smoothed intensity image $G_\sigma(f)$ (see Figure 3.10e).

Additionally, Figure 3.11 shows some difference feature images with various standard deviations used for the Gaussian filtering. The set of difference feature images used in this work can be described by

$$\left\{ G_\sigma(f) - G_{\sigma'}(f) \,\middle|\, \sigma = 3, \ldots, 10 \wedge \sigma' = 2l + 1 \text{ for } l \in \mathbb{N} \wedge \sigma' \leq \sigma - 2 \right\}.$$

Furthermore, we also determine the variance of the eight features shown in Figure 3.12a. The result of the variance is shown in Figure 3.12b. In addition, we also determine the difference between the smoothed images which are smoothed with small and large standard deviations (see Figure 3.12c). Finally, we add the density feature f twice to the feature set which is used for the membrane segmentation. The reason is to increase the possibility of the density feature to be selected more often than other features during the construction of the random forrest.

3.6 Other Features

The set of features discussed in the previous sections is neither complete nor the optimal representation for the purpose of membrane segmentation in ssTEM brain tissue images, as we will see in Chapter 5. The discussed feature set offer diverse possibilities to incorporate the neighborhood information. Vazquez-Reina et al. [2011] propose a segmentation method that uses circular patches of brain tissue which are then transformed using the Zernike transform. Kumar et al. [2010] propose the so-called radon-like features which provide compact feature descriptors. An extension to the radon-like features provide Seyedhosseini et al. [2011]. In this work we will not deal with such features.

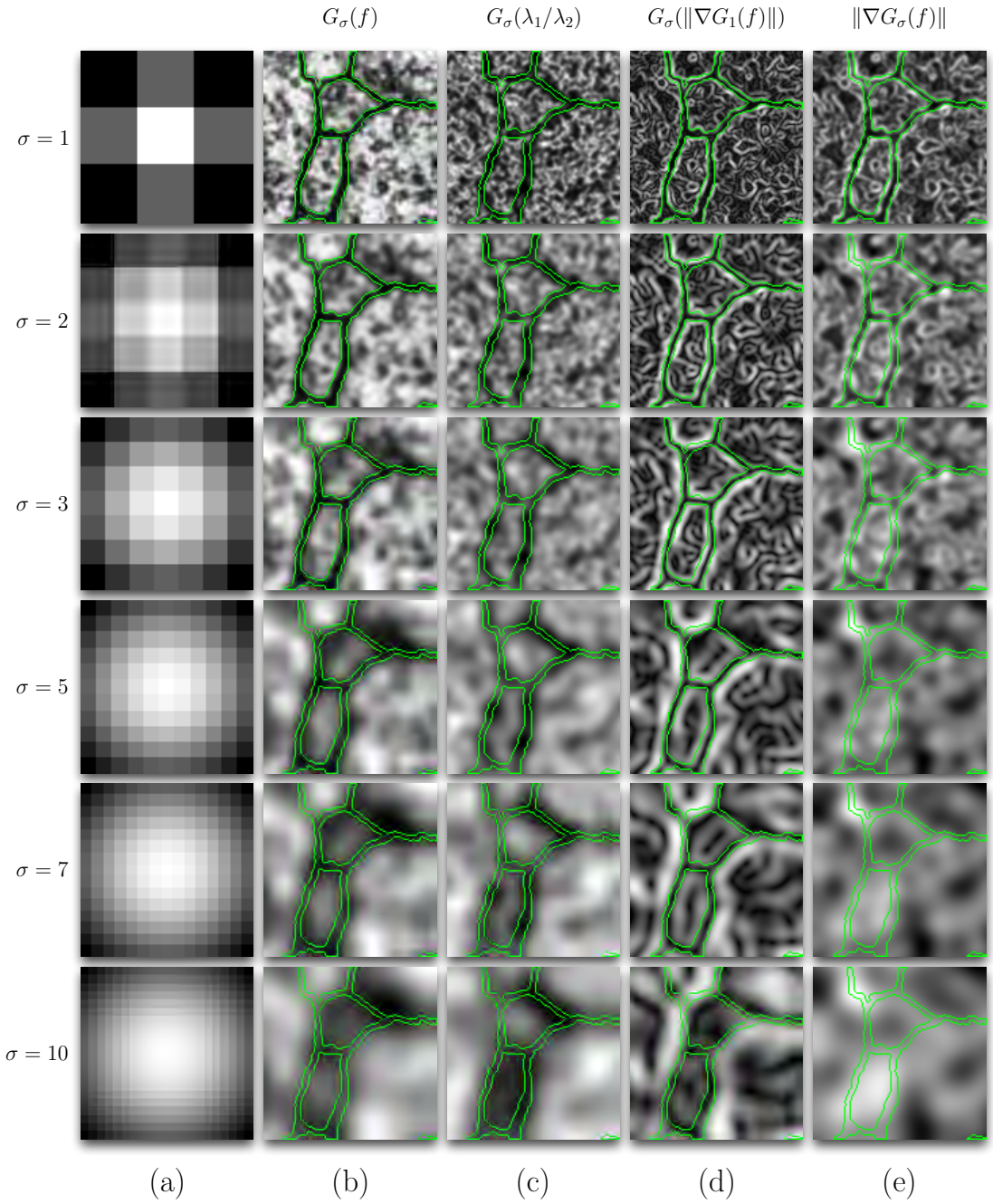


Figure 3.10: Image smoothing using Gaussian filter mask (a) with different standard deviations. (b) The smoothed intensity image f (c) the smoothed eigenvalue image λ_1/λ_2 (d) the smoothed gradient magnitude image of (smoothed once with G_1) , and (e) the gradient magnitude of the smoothed image.

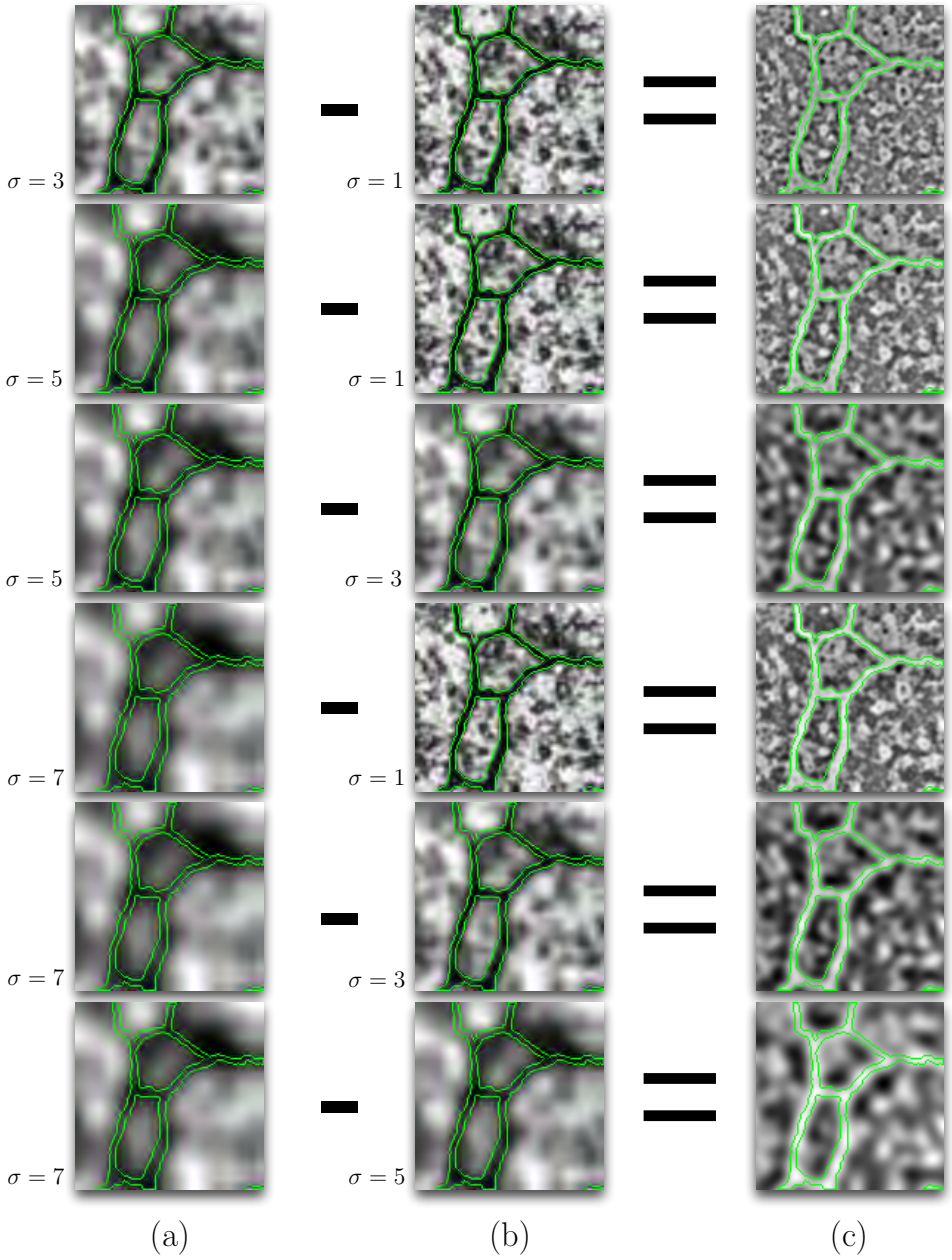
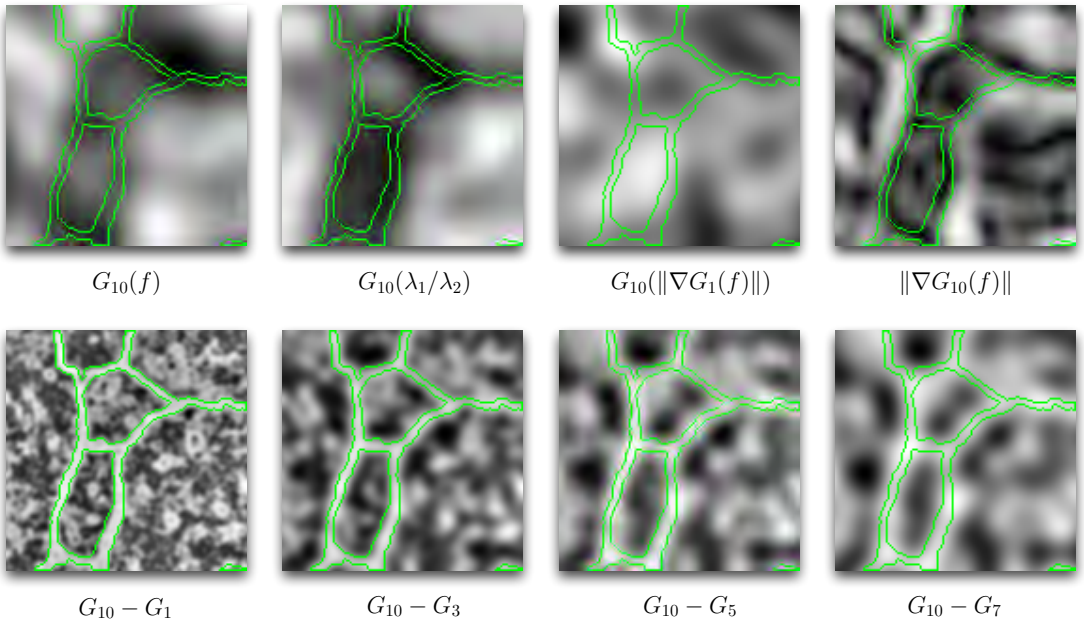
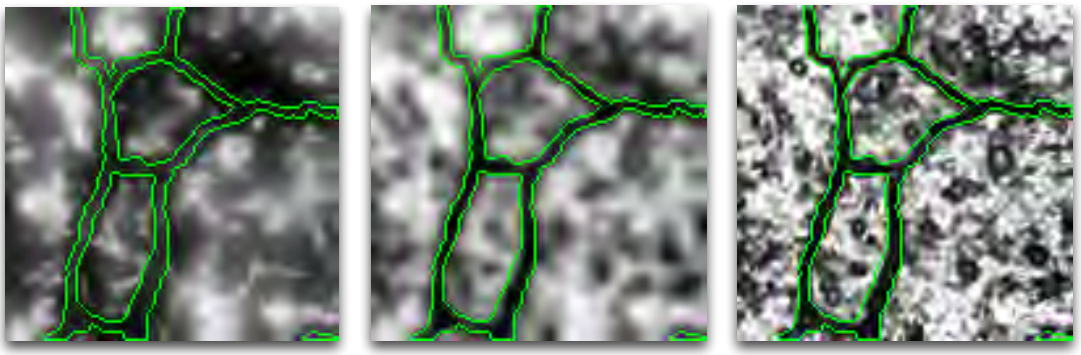


Figure 3.11: Variously smorrrothed original images using different standard deviations for the Gaussian filter mask. (c) Difference images of the smoothed images shown in (a) and (b).



(a) Set of features created using G_{10} (denoted as G^*_{10}).



(b) $\sigma^2(G^*_{10})$ (c) $G_2(f) - G_{50}(f)$ (d) f

Figure 3.12: Additional features used in this work. (a) G_10^* denotes a set of eight features which are created using the Gaussian filter with $\sigma = 10$. (b) Variance of G_10^*. (c) Difference in smoothing results when using small standard deviation ($\sigma = 2$) and large standard deviation ($\sigma = 50$). (d) Original density image.

Chapter 4

Sample Selection & Feature Exploration

> *"The traditional goal of the feature extractor is to characterize an object to be recognized by measurements whose values are very similar for objects in the same category and very different for objects in different categories."*
>
> Pattern Classification, 2nd Edition
> Richard O. Duda, Peter E. Hart, David G. Stork (p.11)

In Chapter 3 we learned that different transformations and filters integrate the local neighborhood information into a single pixel value. This value corresponds to a feature value $x_{i,j}$ in the underlying data table with i being the row and j the column in that table. In Chapter 2 we have seen that the more *expressive* a feature is, with regard to the classification task, the lower the prediction error is. Therefore, we would like to get answers to the following three questions: (a) How can we know whether a feature is expressive sufficient? (b) How can we know which feature set is enough to describe the concept of *membranes* in ssTEM brain tissue images? (c) How can we develop new features from existing ones that grasp this concept better?

4.1 Introduction

Before we start exploring the features created and discussed in Chapter 3, we need to organize them in a coherent way which is compatible with the *random forest* learning paradigm. In Section 2.3 we discussed that for the purpose of machine learning-based image segmentation the *data table* is the preferred data structure for holding pixel descriptions in the form of *feature images*. Because the feature images are scalar-valued, normalized images we can organize them individually in terms of tensors. The feature image of every slice can be described as a 2nd-order tensor while the set of features can be stacked together into a 3rd-order tensor of the corresponding slice. All feature images of a slice have the same size and are derived from the intensity image as illustrated in Figure 2.3 on page 15. This way we have a coherent representation which is easily transferrable between different memory domains – through tensor flattening – and which is also easily accessible

– through slicing of the 3rd-order tensor. Tensor flattening is of special concern when porting the algorithms on data-parallel computing architectures.

Since, we are not domain experts we need a manually segmented volume of the ssTEM brain tissue data sets which are produced by a domain expert that knows which pixels belong to *membranes* and which do not. Such a manually segmented image is called a *ground truth* image. If no *explicit* ground truth image is available in the form of a fully segmented image, it would suffice if the *domain expert* knows which pixels belong to which class. We call this case as *implicit* ground truth knowledge. Furthermore, the expert user also has additional knowledge which the software system does not incorporate. This means that even if we have the ground truth segmentation available we either do not understand all the decision processes and mechanisms that produced that result (e.g., the brushing tools used or the decision making process for manual segmentation of compressed regions as shown in Figure 1.4a on page 6), or we are not able to map all expert knowledge into existing machine learning frameworks. Essentially, this leads to segmentation errors, as discussed in Section 2.3.1, that need to be dealt with.

GROUND TRUTH

The classification of the image pixels differs from the prediction in general purpose domains usually discussed in machine learning literature. Such domains include: credit rating, functional gene classification, medical diagnosis and so on. The image segmentation does not have to suffer from the possibly complex high-dimensional separation hyperplanes as in general data mining. We can always simplify the feature set by combining the strengths of multiple features for different image regions into a single feature through filtering and transformation, as discussed in Chapter 3. Additionally, we have always an idea of what representative pixels are with regard to the classification task. We are aware of the imaging artifacts like the point spread function or the artifacts coming from the sectioning of brain tissue with a diamond knife. We need to take care of two fundamental concepts: (a) the selection of representative positive and negative pixels as training sample and (b) the selection of a feature set that describes the *membrane-non-membrane* concept well.

An exhaustive search with M features would require to test 2^M different feature sub-sets which need to be evaluated separately in order to determine the most relevant feature set. Blum and Langley [1997] provide five definitions of feature relevance from which we only focus on the *incremental usefulness* of a feature which is defined as follows:

INCREMENTAL USEFULNESS

DEFINITION 4.9 (Incremental Usefulness)

"Given a sample of data S, a learning algorithm L, and a feature set $\mathcal{A}$, feature x_i, is incrementally useful to L with respect to $\mathcal{A}$ if the accuracy of the hypothesis that L produces using the feature set $\{x_i\} \cup \mathcal{A}$ is better than the accuracy achieved using just the feature set $\mathcal{A}$." Blum and Langley [1997] (p. 248)

The feature selection based on incremental usefulness can be achieved through the heuristic search for a feature sub-set which is the most relevant to the target concept. In order for the heuristic search to be effective we need to consider the following points (Blum and Langley [1997]):

Starting Point There are two possibilities to start the search. While the *forward selection* starts with an empty feature set and adds new features successively, the *backward elimination* starts with all features and removes the irrelevant ones.

FORWARD SELECTION

BACKWARD ELIMINATION

Organization of Search An exhaustive search is not practicable. Therefore, the filtering of the search space and the use of greedy methods are necessary when choosing a search direction.

Evaluation Alternative feature subsets can be evaluated by the ability of a feature to discriminate between the classes. Alternatively, we can evaluate feature sets based on their prediction accuracies or in case of certainty output ROC and prediction/recall curves.

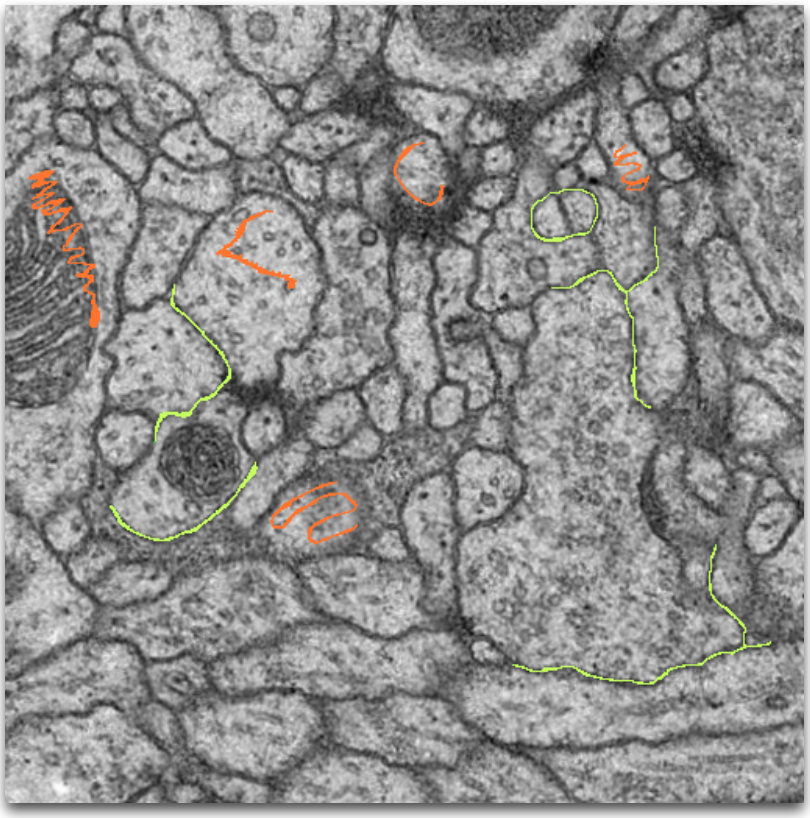
Stopping Criterion There are different ways to stop the search. For example, we can stop the search as soon as the prediction accuracy degrades or we can continue the search and possibly find a better alternative feature subset.

4.2 Sample Selection

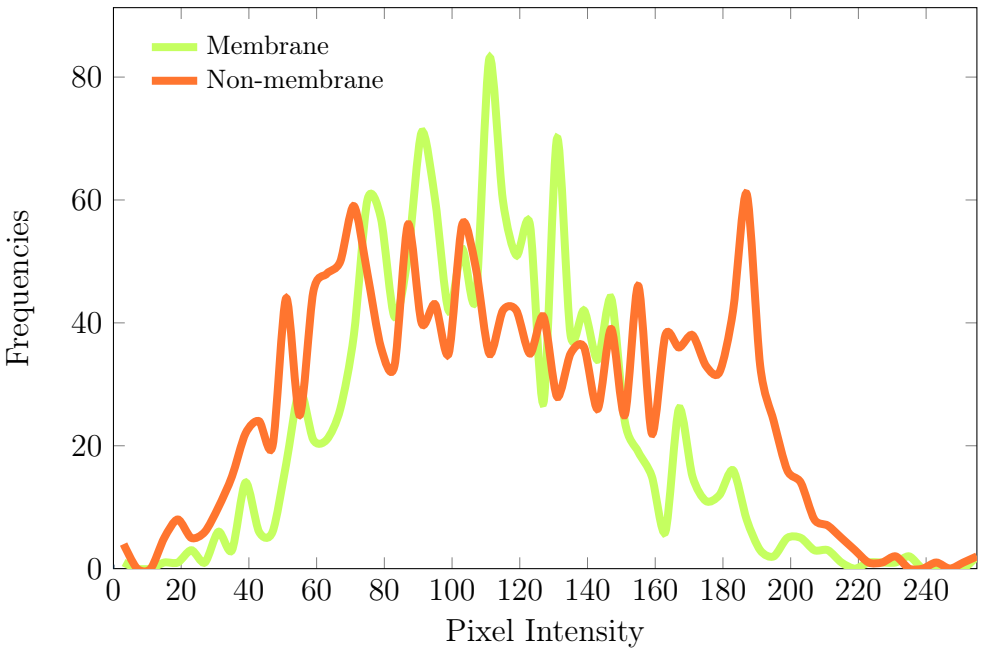
In Section 2.3 machine learning is defined as a computer program that learns from *experience* (i.e., the training sample). In this work, experience is provided through the labeling of a small sub-set of all pixels in ssTEM brain tissue images as *membranes* or *positives* and another pixel sub-set as *non-membranes* or *negatives*. The *labeling* is a visual representation of the *selection* process as discussed by Yi et al. [2007]. The selection itself is realized by the brushing of the pixels with two distinct colors, as illustrated in Figure 4.1, Figure 4.2 and Figure 4.3, and creating three training sets T_i, T_{GT} and T_{i+1}. Figure 4.1a, Figure 4.2a and Figure 4.3a show the selected pixels used for training where the training sets are denoted as T_i, T_{i+1} and T_{GT} respectively. The corresponding intensity value histograms for the training sets are shown in Figure 4.1b, Figure 4.2b and Figure 4.3b respectively. While the green histograms show the intensity value frequencies for the *membrane* pixels the orange histograms show the intensity value frequencies for the *non-membrane* pixels. We see that the ground-truth histograms differ in shape from the histograms for T_i and T_{i+1}. This means that, the random forest classifier which should approximate the real intensity distribution (shown in Figure 4.3b) might not be able to achieve this using only the training sample shown in Figure 4.1a and the underlying feature set which is derived from the intensity feature. Please note: the *intensity value distribution* corresponds to a histogram with 64 bins where instead of plotting the histogram bars we plot the frequencies as data points which are connected. This allows to identify the overlapping regions between the membrane and non-membrane histograms in the histogram plot.

LABELING

SELECTION

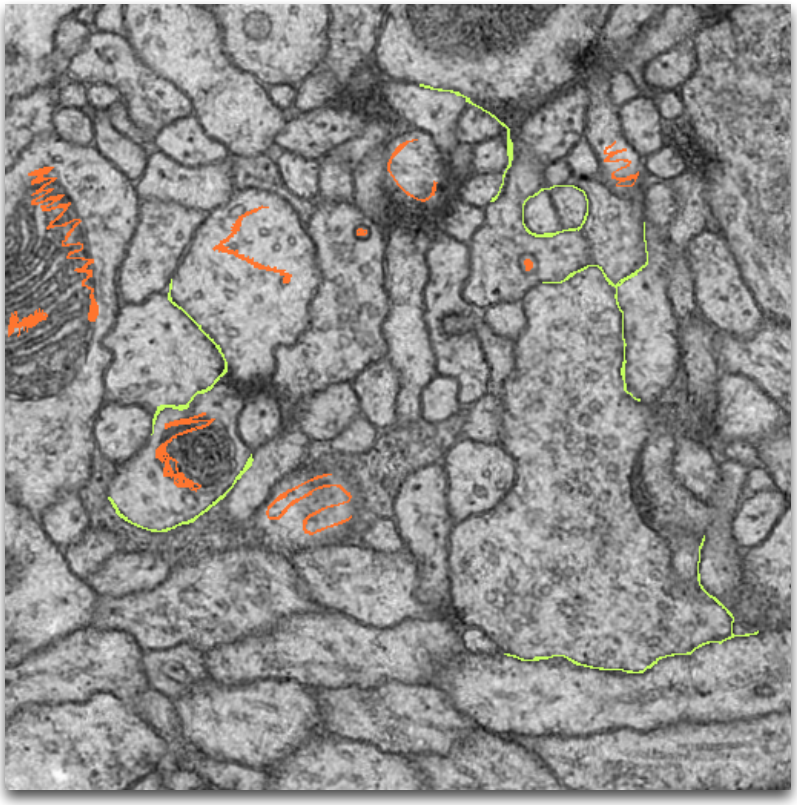


(a) Initial training sets T_i with prediction error $\varepsilon = 0.1315$

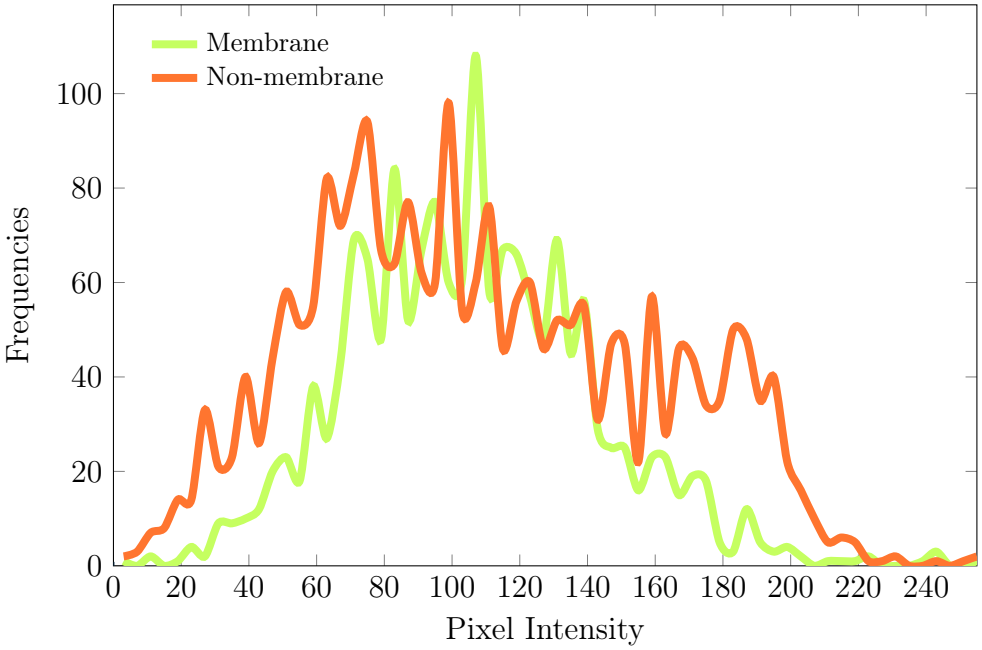


(b) Intensity value histogram for the training sample T_i

Figure 4.1: (a) training sets T_i with (c) the corresponding intensity value histogram for two classes (*memrane*=green, *non-membrane*=orange).

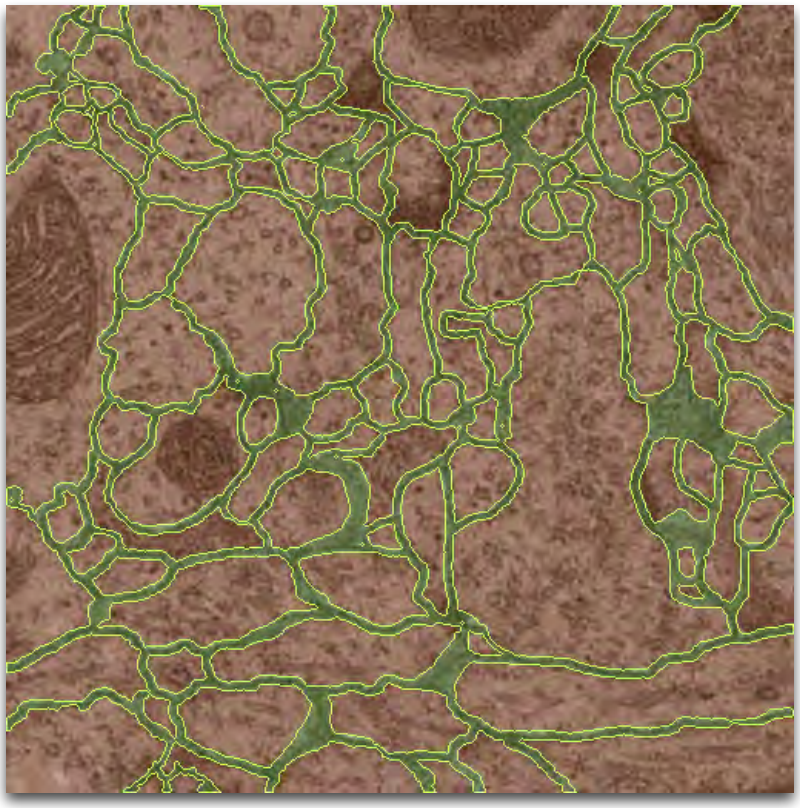


(a) Incremental training sample T_{i+1} with prediction error $\varepsilon = 0.1169$

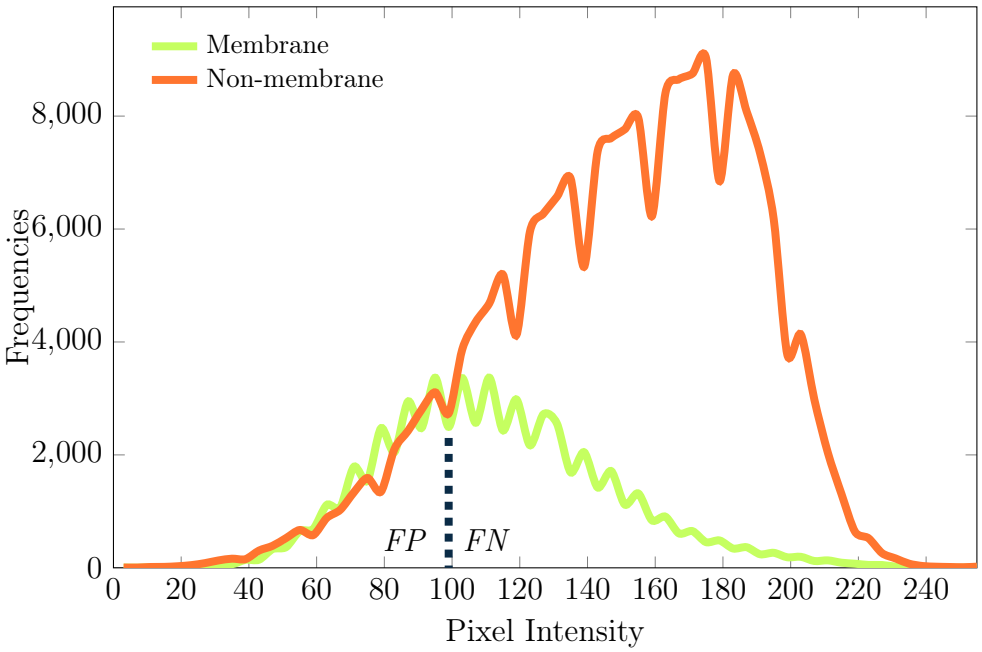


(b) Intensity value histogram for the ground truth

Figure 4.2: (a) Incremental training sample T_{i+1} with (b) the corresponding intensity value histograms for the two classes (*memrane*=green, *non-membrane*=orange).



(a) Ground truth as training sets T_{GT} with prediction error $\varepsilon = 0.1296$



(b) Intensity value histogram for the training sets T_{GT}

Figure 4.3: (a) Actual ground truth image with (b) the corresponding intensity value histograms for the two classes (*memrane*=green, *non-membrane*=orange).

4.2.1 Selection Histogram

Even though the selection made by a domain expert does not represent the real intensity value distribution, we assume that it must suffice for the (semi-)automatic segmentation. The reason for this requirement is that the user interaction should be small due to the large density volumes to be segmented. However, there is not only an *automatic* machine learning process. Additionally, there exists also an *incremental* learning process of the domain expert about the underlying machine learning method. The incremental learning process is not only desired but necessary due to the fact that a random forest with hundreds of trees acts like a black box and opposes the intuition of the domain expert especially if many features are necessary. This problem is illustrated in Figure 4.1, Figure 4.2 and Figure 4.3. An intuitive assumption would be that if we label every pixel in a ssTEM image correctly, by using the ground truth image as the training sample, we should get the best segmentation results (see Figure 4.3a). This is because the more training examples we use the more precise the prediction should become. However, the comparison of the prediction errors in Figures 4.1a, 4.2a and 4.3a shows that the prediction error ε for the bigger training set T_{GT} ($\varepsilon = 0.1296$) is higher than for the smaller (incremental) training set T_{i+1} ($\varepsilon = 0.1169$) for this image. This means that we do not need to label all pixels to perform well. A smaller, more expressive training sample may lead to better results.

4.2.2 Summary

CONSERVATIVE SAMPLE SELECTION

Blum and Langley [1997] describe a *conservative sample selection* algorithm as the one which only learns new examples if the current model misclassifies them. The pixels used for training in Figure 4.2a have been selected using this approach. The selection of *positive* and *negative* examples for the induction process of the random forest classifier can be outlined into an iterative algorithm shown in Algorithm 4.1. We extend the training sample T by *membrane* and *non-membrane* samples, build the random forest classifier and execute it on the entire image. The segmentation results are visualized in two ways. First, the segmentation results are blended over (a) the original intensity image and (b) the ground truth image. This way we can identify where the type I and type II errors are. These regions are then added to the set of *membranes* and *non-membranes* respectively and the training sample is iteratively extended. This is repeated as long as there is further improvement. The loop invariant in Line 14 can be specified in different ways: (a) a fixed number of iterations or (b) an iteration halt in case of no improvement (i.e., greedy search). Moreover, the selection of additional samples (i.e., corrections, in Line 13) can be done randomly from the set of all erroneous pixels. This knowledge-guided selection of FP and FN pixels is discussed in Section 4.4.1.

In our experiments (discussed in Chapter 5) we used training sets of $\sim 3,000$ and $\sim 4,000$ *membrane* and *nom-membrane* pixels for training. The labeling of the training examples takes 20 to 60 seconds. The training of $\sim 4,000$ examples using the random forest classifier takes three seconds and the testing on one slice of size

Input: Intensity image $f(x, y)$,
expert user knowledge as ground truth image $f_{GT}(x, y)$,
transfer functions TF_f, TF_{Seg} and TF_{GT} for intensity, segmentation and ground truth
Output: A representative training sample T

$T \leftarrow \{T_\oplus \cup T_\ominus\}$ ▷ Label some pixels as *positives* $\oplus$ and some as *negatives* $\ominus$
$b_C \leftarrow$ NewColorImage(Width(f),Height(f),(1, 1, 1)) ▷ Initialize background image to white
repeat
  $RF \leftarrow$TrainRF(T)
  $Seg \leftarrow$TestRF(RF,f) ▷ Segment f with learned RF
  $f_C \leftarrow$VisualizeTF(f,b,TF_f)
  $Seg_C \leftarrow$VisualizeTF(Seg,b_C,TF_{Seg})
  $G_C \leftarrow$VisualizeTF(f_{GT},b_C,TF_{GT})
  $S_C \leftarrow$AlphaBlend(Seg_C,f_C,Seg)
  Identify type I $\oplus$ and type II errors $\ominus$ in visualizations S_C and G_C
  $T_\oplus \leftarrow$ Label some type I errors as $\oplus$
  $T_\ominus \leftarrow$ Label some type II errors as $\ominus$
  $T \leftarrow \{T_\oplus \cup T_\ominus\}$ ▷ Extend training sample with some errors corrected
until No improvement possible

function VisualizeTF($f(x, y)$,b(x,y),TF) ▷ Visualize image f using transfer function TF and background image b
  $v_C(x, y) \leftarrow$ NewColorImage(Width(f),Height(f),(0,0,0))
  for each $p \in f(x, y)$ **do**
    $\mathbf{c}_1 \leftarrow$Sample(TF,p)
    $\mathbf{c}_2 \leftarrow$Sample(TF,$b(x, y)$)
    $v_C(x, y) \leftarrow$ InterpolateColors($\mathbf{c}_1$,$\mathbf{c}_2$,p) ▷ Alpha-blending
  return $v_C(x, y)$

function AlphaBlend($f_C(x, y)$,$g_C(x, y)$,$\alpha(x, y)$) ▷ Alpha-blend two images f_C and g_C using an alpha value image $\alpha(x, y)$
  $v_C(x, y) \leftarrow$ NewColorImage(Width(f_C),Height(f_C),(0,0,0))
  for each $p \in f(x, y)$ **do**
    $\mathbf{c}_1 \leftarrow$Sample(f_C,x,y) ▷ Sample color from $f_C(x, y)$
    $\mathbf{c}_2 \leftarrow$Sample(g_C,x,y) ▷ Sample color from $g_C(x, y)$
    $v_C(x, y) \leftarrow$ InterpolateColors($\mathbf{c}_1$,$\mathbf{c}_2$,$\alpha(x, y)$) ▷ Alpha-blending
  return $v_C(x, y)$

function InterpolateColors($\mathbf{c}_1$,$\mathbf{c}_2$,α) ▷ Linear color interpolation
  return $\mathbf{c}_1 \cdot \alpha + \mathbf{c}_2 \cdot (1 - \alpha)$

function NewColorImage(w,h,$\mathbf{p}$) ▷ RGB color image creation and initialization
  $v_C \leftarrow$New2D(w,h)
  for $y \leftarrow 1, h$ **do**
    for $x \leftarrow 1, w$ **do**
      $v_C(x, y) \leftarrow \mathbf{p}$ ▷ Initialize with pixel
  return $v_C(x, y)$

Algorithm 4.1: Iterative training sample construction.

512×512 takes five seconds using the random forrest classifier. The visualization of the segmentation results is realized under one second.

4.3 Feature Redundancy

The feature *redundancy* can be viewed as a measure of similarity between two features. In other words, it is a measure of *"... how much adding a feature to a given set of features contributes to prediction"* (Auffarth et al. [2010] (p. 250)). This contribution can be described by the *correlation* between two features, as discussed in Section 2.4.2 (see Equation 2.5 on page 32). From a statistical point of view, if two variables are uncorrelated they are also statistically independent and therefore not redundant (Duda et al. [2000]).

REDUNDANCY

CORRELATION

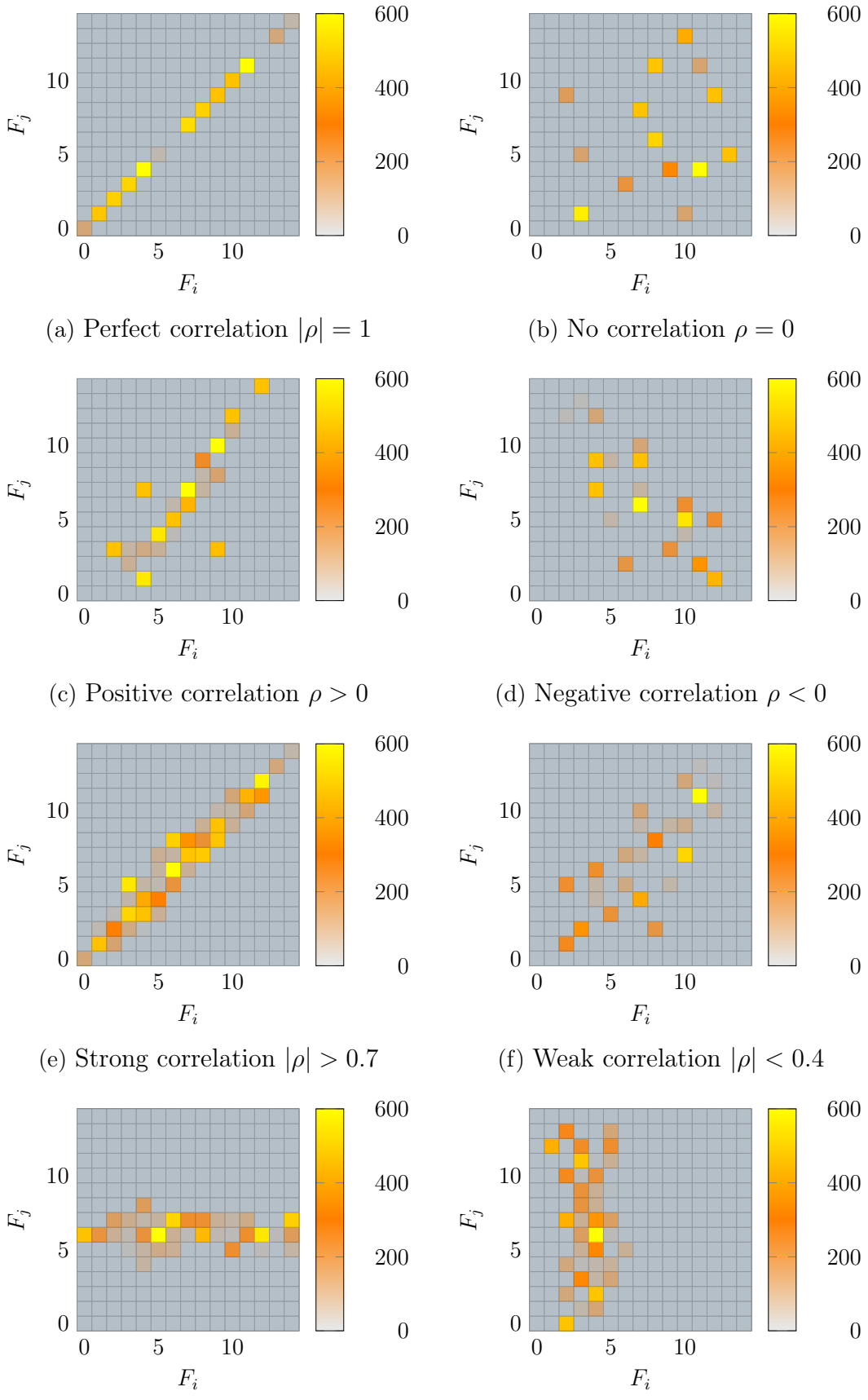
Figure 4.4 illustrates different types of correlation which can occur when comparing two features F_i (x-axis) and F_j (y-axis). All sub-figures show a two-dimensional histogram with 15×15 bins where red means high, orange means average and gray means no feature value occurrences. A *perfect correlation* (defined as $|\rho| = 1$ in Equation 2.5 on page 32) would mean that there is a linear relationship between the features (see Figure 4.4a) while in case of *no correlation* we have independent variables (see Figure 4.4b). *Positive correlation* implies that as the values of F_i increase, the values for F_j also increase but that are not perfectly correlated (see Figure 4.4c), while the *negative correlation* implies that as the values of F_i increase, the values for F_j decrease (see Figure 4.4d). A *strong correlation* means that the feature values are scattered in the vicinity of the diagonal (see Figure 4.4e) while a *weak correlation* means that they are scattered around the diagonal but are further away (see Figure 4.4f). If for a little change in one variable the other variable changes greatly we call this situation *low correlation* (see Figures 4.4g and 4.4h).

4.3.1 Visualization-assisted Feature Redundancy Framework

In general, those two features which correlate perfectly can be considered as *redundant* (see Figure 4.4a) because both have a strong linear relationship and therefore contain the same information. In Section 2.4.2 on page 32 we discussed the correlation coefficient ρ which is one tool to measure redundant features. We can select features based on correlation coefficients by setting a threshold to (say) $|\rho| \geq 0.7$ and then testing the feature set on a test set in a trial-and-error manner. The problem with this approach is that the manual comparison of correlation coefficients of over $M = 90$ features is cumbersome. Therefore, we propose a structured visualization-assisted tool for feature redundancy framework which incorporates interaction techniques such as *selection*, *encoding*, *elaboration*, *abstraction* and *connection* (as proposed by Yi et al. [2007]).

According to Gauss summation we would need to visualize and compare $N_C = \sum_{k=1}^{M} k = \frac{M(M+1)}{2} = 8{,}190$ different scatter plots. The visualization of that many scatterplots leads to an *information overload problem* which occurs when (a) having irrelevant data with respect to the task, (b) having processed the data inappropri-

INFORMATION OVERLOAD PROBLEM



(a) Perfect correlation $|\rho| = 1$

(b) No correlation $\rho = 0$

(c) Positive correlation $\rho > 0$

(d) Negative correlation $\rho < 0$

(e) Strong correlation $|\rho| > 0.7$

(f) Weak correlation $|\rho| < 0.4$

(g) Low horizontal correlation

(h) Low vertical correlation

Figure 4.4: Six special cases of feature correlation visualized as two-dimensional histograms with 15×15 bins. ρ is the correlation coefficient discussed in Section 2.4.2 on page 32.

ately, and (c) having presented the data inappropriately (Keim et al. [2008]).

In order to overcome this problem for the task of feature redundancy exploration we propose the correlation exploration view shown in Figure 4.5. This view

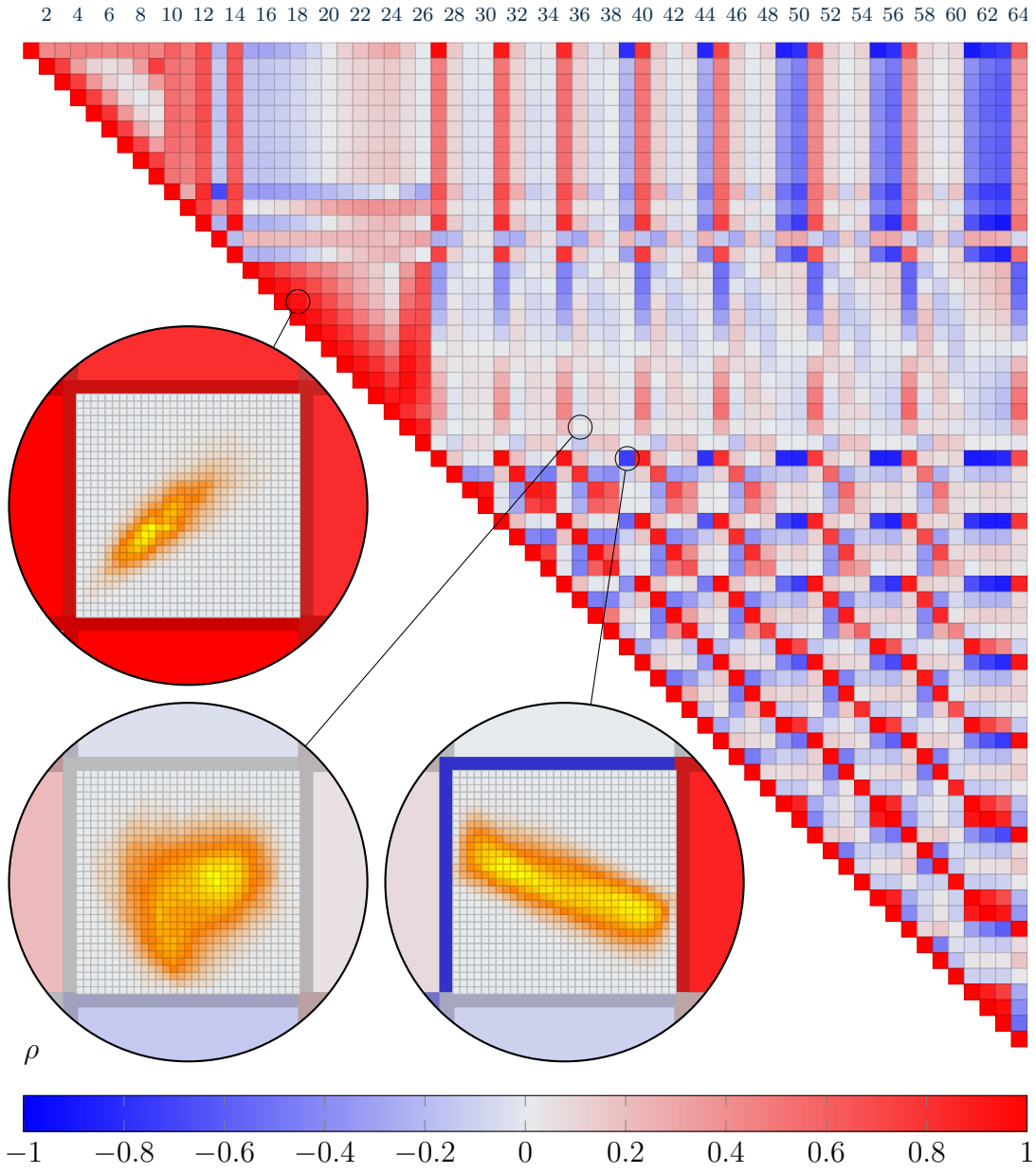


Figure 4.5: Feature redundancy exploration by visual elaboration and abstraction of correlation scatterplots. The numbers at the top indicate the indices of the features. The magnifying glasses show two-dimensional histograms with more detail.

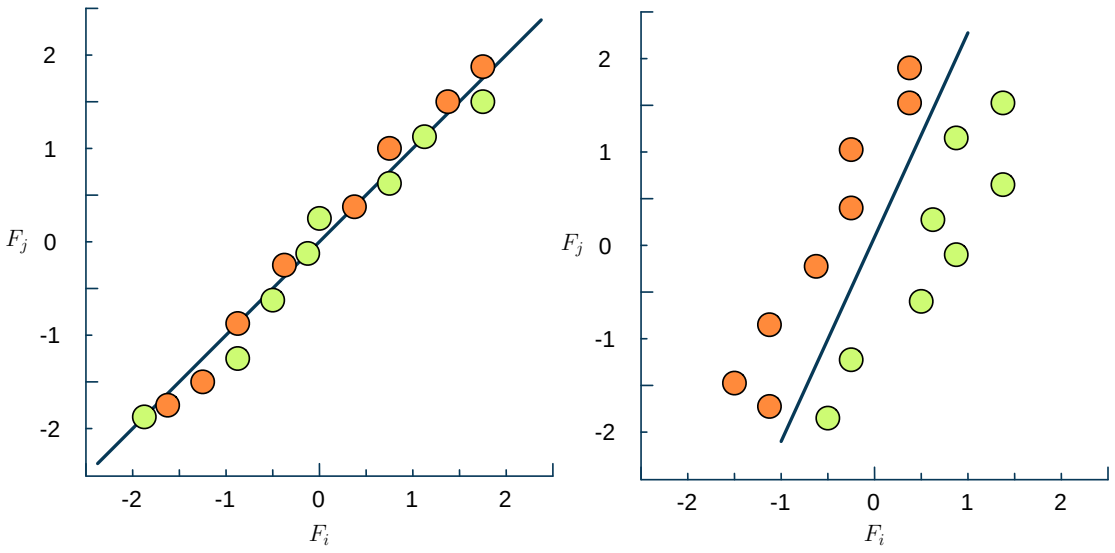
consists of two components based on the *elaboration-abstraction* concept (discussed by Yi et al. [2007]). The *correlation coefficient matrix view* (CCM view) encodes the correlation coefficients ρ of features by color (blue, gray, red). The correspond-

ing feature indices are displayed at the top of the view. Because a printed page is limited we show 64 out of 90 features in the CCM view. The correlation coefficient $\rho \in \{-1, \ldots, 0, \ldots, +1\}$ *abstracts* the result of the correlation of two features by a colored glyph (in our case: blue, gray and red squares). We allow the *elaboration* of each selected glyph by zooming into the CCM and visualizing the corresponding *correlation plot*. Instead of showing the scatter plots with hundreds of thousands of data points (in our case 262,144 data points per image) we use two-dimensional histogram plots (with 30×30 bins) from which the overall shape of the scattered points can be identified. The colors in these plots correspond to the number of data points within the corresponding bin: yellow corresponds to a high number of data points in that bin, orange corresponds to a medium and gray to a low number of data points. Because the correlation matrix is symmetric we only need to show half of the matrix (in this case: the upper triangular matrix). In Figure 4.5 we see 64 out of 90 feature correlation coefficients visualized as strong positive correlation (red), strong negative correlation (blue) and no correlation (gray).

Because the CCM view contains many objects that we can interact with we use a thresholding filter (discussed by Yi et al. [2007]) on the correlation coefficients. This allows interactive search and comparison of strongly correlated features ($|\rho| \geq 0.7$). However, a strong correlation does not necessarily mean that there is no gain by adding a feature to the feature set (Guyon and Elisseeff [2003]).

INTRA-CLASS COVARIANCE

This exceptional circumstance requires the calculation of the so-called *intra-class covariance* which is illustrated in Figure 4.6. Both cases show high correlation



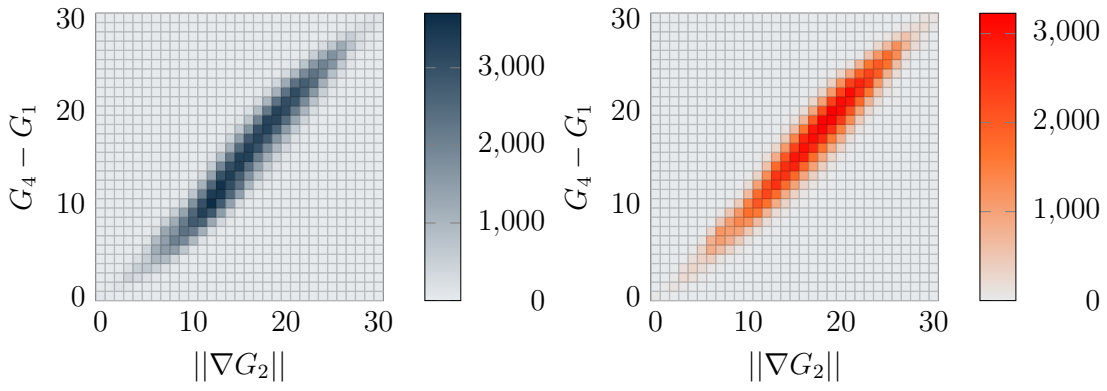
(a) No gain in using both features (b) Gain in using both features

Figure 4.6: Effect of the intra-class covariance shown on an simple example. (a) Adding both features to the feature set do not lead to a gain in separation betwwen classes (green, orange). (b) Adding both features lead to a gain in separation (adopted from Guyon and Elisseeff [2003]).

in the directions of the corresponding lines. The adding features F_i and F_j in

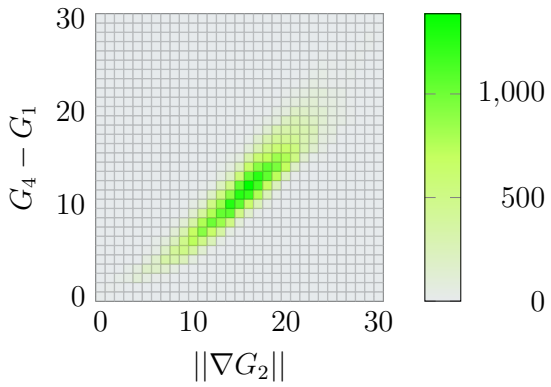
Figure 4.6a does not lead to a significant gain in the separation between the two classes. While adding F_i and F_j in Figure 4.6b to the feature set does lead to an important separation gain because F_i as well as F_j is necessary to identify the separation line shown in Figure 4.6b.

The intra-class covariance indicates where strongly correlating features do not automatically lead to omitting a feature. Instead of using only one of these features we use the second one as well if it helps separating the positives from the negatives, as shown in Figure 4.7. It shows that two highly correlated features $||\nabla G_2||$ and

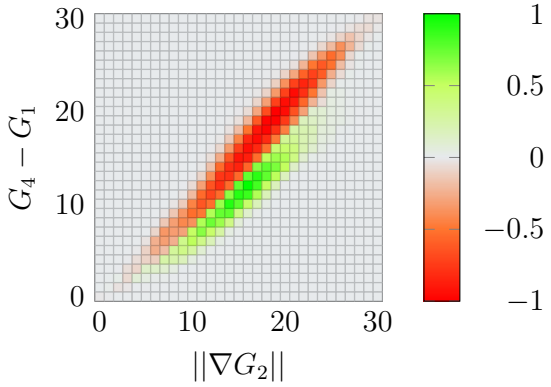


(a) High correlation $\rho = 0.96$

(b) Spatial distribution of negatives



(c) Spatial distribution of positives



(d) Per-class scaled histogram plot

Figure 4.7: Intra-class covariance for $||\nabla G_2||$ and $G_4 - G_1$ features. The separation boundary is shown in (d) between the positive (b) and the negative (c) histograms.

$G_4 - G_1$ with $\rho = 0.96$ represent basically the same information (Figure 4.7a). Furthermore, we can show the histograms of feature values for membranes and non membranes, which is illustrated Figure 4.7b and Figure 4.7c respectively. For the purpose of intra-class covariance we can merge the three histograms shown in Figure 4.7a, Figure 4.7b and in Figure 4.7c into a single histogram shown in

Figure 4.7d. However, the number of *membrane* pixels is much lower than the number of *non-membrane* pixels (compare the color bar ranges in Figure 4.7b and Figure 4.7c).

From a visualization point of view, each two-dimensional histogram bin contains two values: the number of *positives* and the number of *negatives*. One possibility of resolving this multi-modal situation is blending. But blending does not lead to expressive histograms where the bi-variate distribution of *positive* and *negative* bins is clearly visible because of the large histogram bin height discrepancies between the two classes. Therefore, we propose the *per-class scaled histogram plot* (PCSH plot), as shown in Figure 4.7d, where the positive histograms (see Figure 4.7b) and negative histograms (see Figure 4.7c) are scaled between $[0, 1]$ and [-1,0] respectively. This is achieved as follows: let $b_{\oplus}(i, j)$ denote the frequency of positives in the bin at position (i, j) in the positive histogram with $i = 1, \ldots, 30$ and $j = 1, \ldots, 30$ and let $b_{\ominus}(i, j)$ denote the frequency of negatives in the bin at the same position (i, j) in the negative histogram. Then we can scale the bin values in each histogram (positive and negative histograms separately) by finding the maximum frequency and dividing all frequencies by the maximum frequency. The results are scaled histograms which have bins denoted as $\hat{b}_{\oplus}(i, j)$ and $\hat{b}_{\oplus}(i, j)$ respectively. Then the corresponding bin in the PCSH plot has the value

PER-CLASS SCALED HISTOGRAM PLOT

$$b_{PCSH}(i, j) = \begin{cases} b_{\oplus}(i, j) & \text{if} \quad b_{\oplus}(i, j) \geq b_{\ominus}(i, j) \\ -b_{\ominus}(i, j) & \text{if} \quad b_{\oplus}(i, j) < b_{\ominus}(i, j) \end{cases} \tag{4.1}$$

The negation of the scaled bins of the negative histogram is done in order to separate the value range of the negative histogram bins from the value range of the positive histogram bins. This leads to a clear visual separation between two classes in a two-dimensional histogram but it does not show type I and type II errors made when separating both distributions. The PCSH plot falls into the interaction category of an *encoding view* (as used by Yi et al. [2007]).

Even though the framework provides an overview and structures the search for redundant features a lot user interaction is necessary. First, the user gets an overview of the correlation coefficients (CCM view). Second, because we have 8,190 different correlation coefficients the user can apply a threshold (say) $|\rho| \geq 0.7$ and rule out most correlation coefficients. For the remaining 222 correlation coefficients we need to check the intra-class covariance to decide whether the highly correlated features are though relevant.

4.3.2 Summary

We can summarize the visual exploration process of redundant features in terms of the procedure shown in Algorithm 4.2. The CCM view is used as an abstraction view with the purpose to identify and mark redundant features. The threshold-based filtering is an interaction technique which allows to exclude weakly correlated features from consideration. This means that out of 2080 possible feature correlations we filter those which have $|\rho| \geq 0.7$ and get 199 feature correlations to check. The histogram plots are used to identify the intra-class covariance which is an in-

Input: Feature image stack $\mathcal{F}_i(x, y; z)$ for slice $S_i(x, y)$ ($z = 1, \ldots, M$ is feature index),
ground truth image $f_{GT}(x, y)$,
transfer function TF_{Corr} for correlation coefficient matrix view
Output: Flag sequence to indicate redundant features $\mathbf{r} \in \mathbb{R}^M$

$Idx_{\oplus} \leftarrow f_{GT}(x, y) == 1$ ▷ Get positive pixel indices
$Idx_{\ominus} \leftarrow f_{GT}(x, y) == 0$ ▷ Get negative pixel indices
$n_b \leftarrow 32$ ▷ Number of 2D histogram bins is $n_b \times n_b$
$\mathbf{r}(.) \leftarrow 0$ ▷ Mark all features as non-redundant
$m \leftarrow$ CreateCCM($\mathcal{F}(x, y; z)$)
$v \leftarrow$ CreateCCMView(m, TF_{Corr})
$v_{Strong} \leftarrow$ ThresholdFilter(v,0.7) ▷ Filter strongly correlated features

for each $g \in v_{Strong}$ **do**
  $\{i, j\} \leftarrow$ Index(g)
  $\{V_{i,\oplus}, V_{j,\oplus}\} \leftarrow \{\mathcal{F}(x, y; i)(Idx_{\oplus}), \mathcal{F}(x, y; j)(Idx_{\oplus})\}$ ▷ Get value arrays for $\oplus$
  $\{V_{i,\ominus}, V_{j,\ominus}\} \leftarrow \{\mathcal{F}(x, y; i)(Idx_{\ominus}), \mathcal{F}(x, y; j)(Idx_{\ominus})\}$ ▷ Get value arrays for $\ominus$
  $h_{\oplus} \leftarrow$Histogram2D($V_{i,\oplus}$,$V_{j,\oplus}$,n_b) ▷ Get 2D histogram for class $\oplus$
  $h_{\ominus} \leftarrow$Histogram2D($V_{i,\ominus}$,$V_{j,\ominus}$,n_b) ▷ Get 2D histogram for class $\ominus$
  $h_{ICC} \leftarrow$ Histogram2D2Class($h_{\oplus}$,$h_{\ominus}$)
  if h_{ICC} indicates weak intra-class covariance **then**
    $\mathbf{r}(i) \leftarrow 1$

function CreateCCM($\mathcal{F}(x, y; z)$) ▷ Create correlation coefficient matrix
  $M \leftarrow$ Size($\mathcal{F}(x, y; z)$,3) ▷ Number of features
  $\rho \leftarrow$ New2D(M,M)
  for $j \leftarrow 1, M$ **do**
    for $i \leftarrow 1, M$ **do**
      $\rho(i, j) \leftarrow$Corr2($\mathcal{F}(x, y; i)$,$\mathcal{F}(x, y; j)$) ▷ Equation 2.5 on p. 32
  return $\rho(i, j)$ ▷ $\rho(i, j) \in \{-1, \ldots, 0, \ldots, +1\}$

function ThresholdFilter($v(i, j)$,t) ▷ Filter correlation coefficient $|\rho| \geq t$
  $M \leftarrow$ Size($\mathcal{F}(x, y; z)$,3) ▷ Number of features
  for $j \leftarrow 1, M$ **do**
    for $i \leftarrow 1, M$ **do**
      if $|v(i, j)| \geq t$ **then**
        $v(i, j) \leftarrow$Sample(TF_{Corr},0) ▷ Mark as uncorrelated
  return $v(i, j)$

Algorithm 4.2: Feature redundancy checking according to correlation coefficients and intra-class covariances.

function CREATECCMVIEW($m(i,j), TF_{Corr}$) ▷ Create correlation coefficient matrix view
  $M \leftarrow$ SIZE(m(i,j),1)
  $v \leftarrow$ NEWCOLORIMAGE(n,n,(0,0,0))
  $m \leftarrow$ CREATECCM($\mathcal{F}(x,y;z)$) ▷ Determine correlation coefficient matrix
  for $j \leftarrow 1, M$ **do**
    for $i \leftarrow 1, M$ **do**
      $v(i,j) \leftarrow$SAMPLE(TF_{Corr},$m(i,j)$)
  return $v(i,j)$

function HISTOGRAM2D($f_1(x,y)$,$f_2(x,y)$,n_b) ▷ Create 2D histogram having $n \times n$ bins
  $\{w,h\} \leftarrow \{$WIDTH(f_1),HEIGHT(f_1)$\}$
  $\{x_{min}, x_{max}\} \leftarrow \{0,1\}$
  $\{y_{min}, y_{max}\} \leftarrow \{0,1\}$
  $\{w_x, h_y\} \leftarrow \{(x_{max}-x_{min})/n_b, (y_{max}-y_{min})/n_b\}$
  $h \leftarrow$ NEW2D(n,n)
  for $j \leftarrow 1, h$ **do**
    for $i \leftarrow 1, w$ **do**
      $s_x \leftarrow$SAMPLE(f_1,i,j) ▷ Sample value from f_1
      $s_y \leftarrow$SAMPLE(f_2,i,j) ▷ Sample value from f_2
      if INRANGE(s_x,x_{min},x_{max}) **and** INRANGE(s_y,y_{min},y_{max}) **then**
        $\{idx_x, idx_y\} \leftarrow \{[s_x - x_{min})/w_x], [(s_y - y_{min})/h_y]\}$ ▷ [.] round to integer
        $h(idx_x, idx_y) \leftarrow h(idx_x, idx_y) + 1$ ▷ Count frequencies
  return $h(x,y)$

function HISTOGRAM2D2CLASS($h_\oplus(x,y)$,$h_\ominus(x,y)$) ▷ Create 2D histogram containing clearly separated classes
  $\{w,h\} \leftarrow \{$WIDTH($h_\oplus$),HEIGHT($h_\oplus$)$\}$
  $h_\oplus \leftarrow$ NORM01($h_\oplus$) ▷ Normalize frequencies in range $[0,1]$
  $h_\ominus \leftarrow$ NORM01($h_\ominus$) $\circ -1$ ▷ $\circ$ element-wise multiplication
  $h \leftarrow$ NEW2D(w,h)
  for $j \leftarrow 1, h$ **do**
    for $i \leftarrow 1, w$ **do**
      if $|h_\oplus(i,j)| \geq |h_\ominus(i,j)|$ **then**
        $h(i,j) \leftarrow h_\oplus(i,j)$
      else
        $h(i,j) \leftarrow h_\ominus(i,j)$
  return $h(i,j)$

function INRANGE(v,s_{min},s_{max}) ▷ Check v for being in $[s_{min}, s_{max}]$
  return $v \geq s_{min}$ **and** $v \leq s_{max}$

Algorithm 4.2: (continued)

dicator of redundant but yet relevant features. The Line 15 in Algorithm 4.2 states that it is up to the user to decide which feature he/she should mark as redundant. As a heuristic one can say that the intra-class covariance is considered strong if positive and negative distributions – shown as green and red bins in Figure 4.7d – are far away and do not overlap. The overlap can be determined visually by comparing Figure 4.7b and 4.7c.

4.4 Feature Relevance

RELEVANT

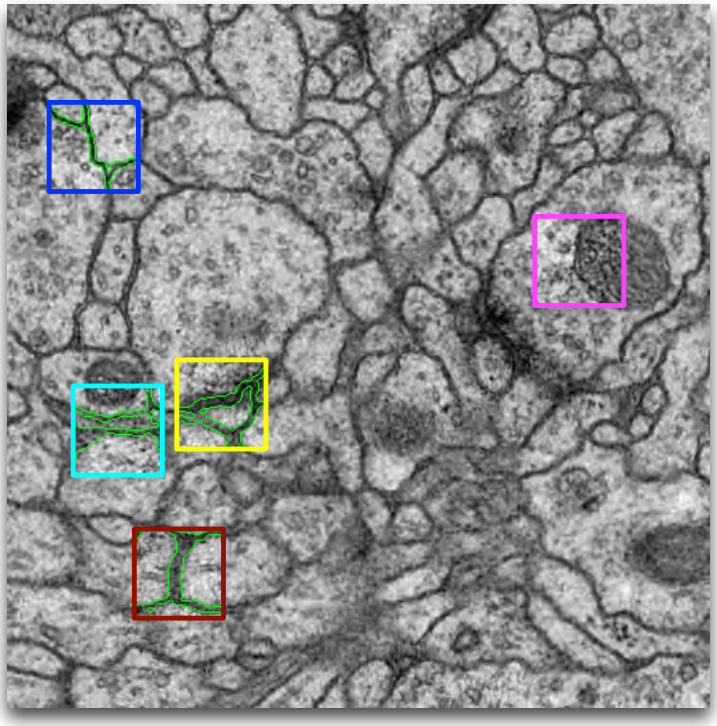
In this work, a feature is called *relevant* or expressive if it captures the *membrane* concept that discriminates between *membrane* and *non-membrane* pixels. In this regard, the most expressive image would be the ground truth image as it provides for each pixel always the correct label and makes no error. Or as Yu and Liu [2003] put it: *"... a feature is good if it is highly correlated with the class but not highly correlated with any of the other features."* (p. 858). Because the ground truth can only be produced manually by an expert user, it is of course not practicable to generate it for every ssTEM image.

4.4.1 Aspect-oriented Feature Exploration

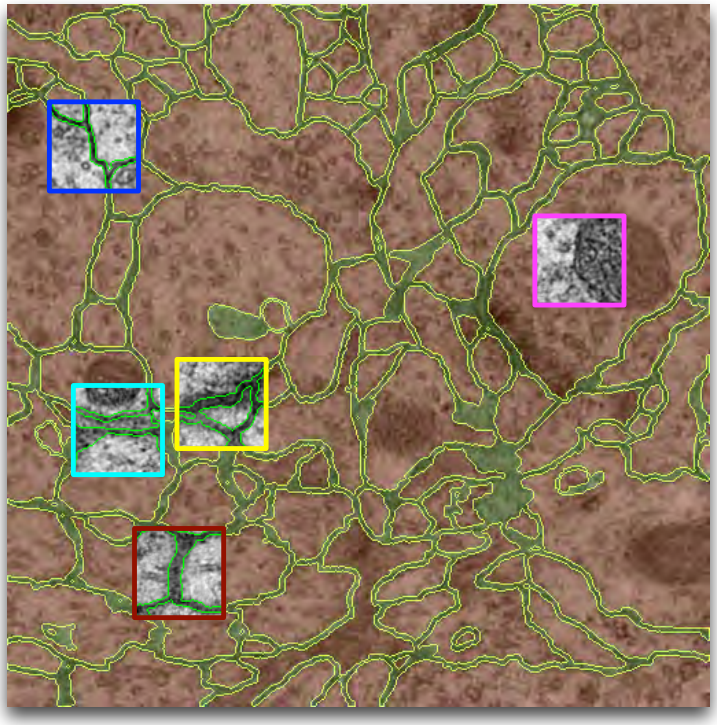
ASPECT WINDOW

If it is not possible to achieve complete separation using one feature, then multiple features can be combined into a set that captures different parts of the same concept. For this purpose, we define an *aspect window* as an image patch (i.e., a small image region) of interest which captures a part of the *membrane-non-membrane* concept. The identification of these parts can be structured in the following way: (a) Segment the intensity image with a training set, (b) visualize the corresponding confusion matrix (discussed in Section 4.5.3) to identify TP, TN, FN and FP, (c) select a number of aspect windows (say five) as representatives of TP, TN, FN and FP, and (d) connect each aspect window in the intensity image to the corresponding aspect window in all other feature images. This procedure consists of the selection and connection interaction techniques as proposed by Yi et al. [2007]. The five selected aspect windows are shown in Figure 4.8 while the corresponding connection view is shown in Figure 4.9. From a slice of the ssTEM brain tissue data set we select five regions of interest (i.e., aspect windows). Each aspect window marks a region in the slice image with potential problems as well as that can occur when segmenting the image. The overall aim is to findWith aspect visualization it becomes clear which features are best suited to represent which aspects. The visualization of the segmentation result is discussed in Section 4.5.1.

The five aspects shown in Figure 4.9 lead to a set of rules shown in Table 4.1 that are derived by comparing the intensity value distributions for both *membrane* and *non-membrane* regions. The blue image patch shows a *membrane* region without compression zones where in f the intensity value distribution is centered around 88 (mean) for the membrane and 161 (mean) for the non-membranes. We also see that other features capture the differences in pixel value distributions between *membrane* and *non-membrane* regions. The cyan and red patches show



(a) Density image with five aspect windows.



(b) Corresponding ground truth image with the same aspect windows.

Figure 4.8: Aspects as image patches of interest with regard to the *membrane-non-membrane* concept. (a) The density image with aspect windows. (b) The ground truth image with the corresponding aspect windows.

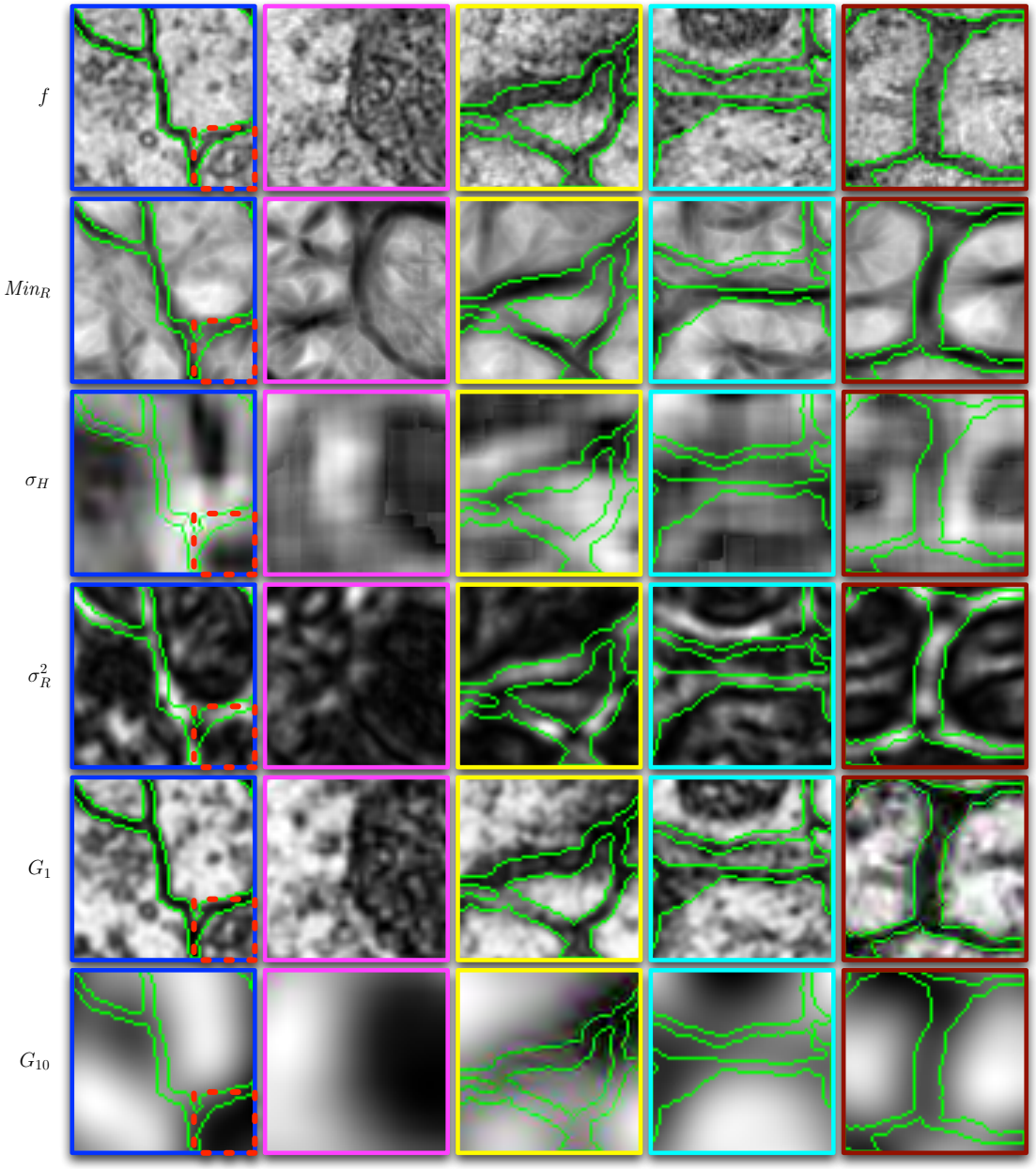


Figure 4.9: The corresponding feature patches for each aspect window defined in Figure 4.8.

Aspect	Membrane	Non-membrane
Blue	Membrane pixels are part of ribbon-like structures of equal thickness and have density values agglomerated around intensity value 88 as shown in Figure 4.3b on page 56.	Non-membrane pixels are blobs having density values agglomerated around intensity value 161 as shown in Figure 4.3b on page 56.
Magenta	No membrane pixels are shown, only white matter and a mitochondrion which has similar pixel intensities as the membrane shown in blue aspect window.	Both, the white matter as well as the mitochondrion (round dark region) are part of the non-membrane.
Yellow	Membrane pixels are part of ribbon structures with varying thicknesses as well as part of a synapse (darker pixels of the membrane).	Non-membrane regions contain pixels being part of the synapse.
Cyan	Compressed zones show membrane pixels as part of ribbon structures with varying thicknesses and pixel intensity distributions.	Non-membrane regions contain white matter as well as a part of a mitrochondrion.
Red	Membrane pixels are part of both the compressed image zones (thick vertical ribbon) as well as non-compressed image zones (horizontal ribbons).	Non-membrane regions contain white matter.

Table 4.1: Membrane observations lead to a high-level set of rules which are extracted and illustrated by aspect windows shown in Figure 4.8 and Figure 4.9.

different variants of compression artifacts. The magenta image patch shows the boundary of a mitochondrion and the white matter surrounding it. The yellow image patch shows a synaptic cleft. We also see that in the blue patches the differences in pixel value distributions between the *membrane* and *non-membrane* regions are represented by the density feature f, the minimum and variance of the rotation-invariant membrane matched images Min_R and σ_R^2, the Gaussian

smoothed images G_1. On the other hand we see that the *non-membrane* region in the magenta image patch is best approximated by σ_R^2 under the six selected features.

The visual exploration of features based on the selected aspect windows works as follows: (a) visualize the aspect windows for the original intensity image f (i.e., the first row in Figure 4.9), (b) for each aspect window visualize the the full set of features (columns in Figure 4.9), and (c) select those features which represent the *membranes* and the *non-membranes* the best. In the last step the *membranes* are best represented by a feature if the pixel values along the *membranes* are clustered around a pixel value (i.e., are similar along the *membrane*) and if those similar pixel values are drawn through the *membranes* (i.e., are represented by ribbons with uniform pixel values). f and G_1 show drawn through *membranes* (i.e., those where the variation of the pixel values is small) while σ_R^2, G_{10} and σ_H show not drawn through *membranes* in Figure 4.9 (i.e., it shows non-uniform snakes). On the other hand we select features which represent *non-membranes* as well. *Non-membranes* are represented as regions instead of ribbons. Therefore, we select at least one feature for each aspect window where at least one *non-membrane* region (in the blue aspect window, f shows four *non-membrane* regions) has uniform pixel values. For example, the blue aspect window in Figure 4.9 contains a *non-membrane* region (indicated by red dotted square in the lower-right corner) which is best represented by uniform dark regions shown in σ_R^2 and G_{10}.

4.4.2 Aspect-oriented Feature Creation

Aspect windows can also be used to identify those features which represent specific regions in an image better than the original intensity image. This can be achieved by comparing the pixel value distributions within the *membrane* regions as well as *non-membrane* regions. From Figure 4.9 we see that different features represent different parts of the same concept. In our case: some features represent different parts of the *membrane* ribbons better than the intensity image and some other features represent different parts of the *non-membrane* regions better than the intensity image. Therefore, we can use aspect windows to guide the construction of new features which are more expressive than the existing ones by combining these a feature sub-set using means of filtering and weighting operators.

For example, let us consider only the features shown in Figure 4.9 which evident problem for the segmentation of the intensity image. In the magenta aspect window, f shows only *non-membrane* pixels. The round dark region on the right is a mitochondrion which has similar pixel value distribution as the *membrane* ribbon shown in f in the blue aspect window. Therefore, the idea is to create a new feature that displays mitochondria differently from *membranes*. The construction of this feature is described in Algorithm 4.3. Because white matter pixels have higher pixel intensities than membranes we select pixels which are bright by splitting the value range of the Gaussian smoothed intensity image equally ($G_1 < 0.5$, $G_1 \geq 0.5$). Then, in aspect windows we search for a feature that represents the mitochondria uniformly. In our case σ_R^2 displays the mitochondria mostly as dark

Input: Feature images f, Min_R, σ_H, σ_R^2, G_1, G_{10}
Output: A new feature capturing the mitochondrion sub-concept

```
1:  {w, h} ← {WIDTH(f),HEIGHT(f)}
2:  v_n ← NEW2D(w,h)
3:  for y ← 1, h do
4:      for x ← 1, w do
                                          ▷ Handle white matter with mitochondria
5:          if G_1(x, y) ≥ 0.5 and σ_R^2(x, y) < 0.2 then
6:              v_n(x, y) ← G_10(x, y)
                                          ▷ Handle membrane-like pixel regions
7:          else if G_1(x, y) < 0.5 and σ_R^2(x, y) ≥ 0.5 then
8:              v_n(x, y) ← Min_R(x, y) ∘ G_1(x, y)   ▷ Element-wise feature weighting
9:          else
10:             if σ_H(x, y) ≤ 0.2 then          ▷ Reverse mitochondria pixel values
11:                 v_n(x, y) ← 1 − G_1(x, y)
12:             else
13:                 v_n(x, y) ← G_1(x, y)
14: return v_n(x, y)
```

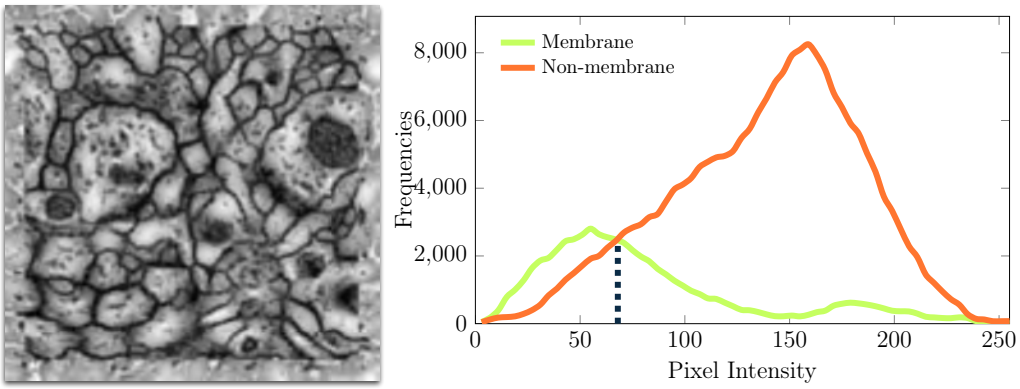
Algorithm 4.3: Creation of the new feature shown in Figure 4.10 based on the observations of the pixel value distributions in the aspect windows show illustrated in Figure 4.8 on page 68.

uniform regions. The next step is to find a proper threshold for σ_R^2 which can be achieved by providing a threshold slider for selecting pixels whose position is propagated to all aspect windows for σ_R^2 (i.e., row four). We choose a threshold with which most of the mitochondrion region, shown in σ_R^2 in the magenta aspect window, is captured ($\sigma_R^2 < 0.2$). Then we replace the selected pixel by a uniform region such as $G_1 0$ (see Line 6 in Algorithm 4.3. The same principle is applied also for *membrane* ribbons and other pixels which do not fall into the specified value ranges in Line 5 and Line 7.

The result of Algorithm 4.3 is shown in Figure 4.10. Adding the new feature to the original feature set of 90 features lead to an improvement of the prediction error from 12.70 % to 12.48 %.

4.5 Visual Comparison of Segmentation Results

The output of the random forest-based image segmentation is a confidence function $c(x, y) \in [0, 1]$ which maps a confidence value $c(x, y)$ to each image pixel (x, y). This confidence map (also called *confidence image.*) can be visualized using transfer functions, as discussed in Section 2.2 on page 16. Furthermore, if N_{Seg} distinct segmentation results are produced with different configurations (i.e., either with different training or feature sets) we can combine the confidence image tensor $\mathcal{C}(x, y; z)$. $\mathcal{C}(x, y; z)$ consists of a stack of N_{Seg} confidence images with $z = 1, \ldots, N_{Seg}$ using operators such as the *mean* and *variance* of N_{Seg} images



(a) New feature (b) Value histogram for new feature

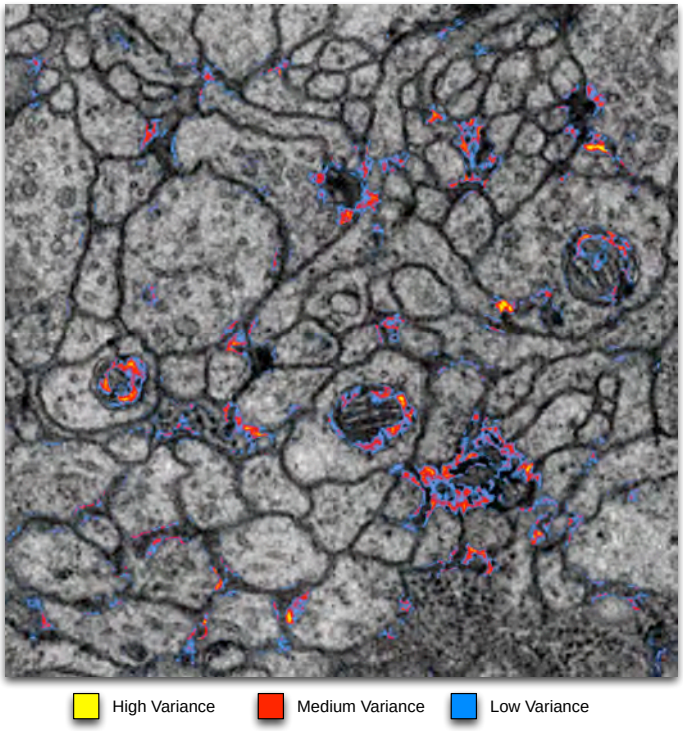
Figure 4.10: Aspect-oriented creation of a new feature from f, G_1, G_{10}, Min_R and σ_R^2 and corresponding pixel value histogram.

and in case of $N_{Seg} = 2$ also the *difference* between two confidence images. The function that determines these operators is shown in Algorithm 4.4.

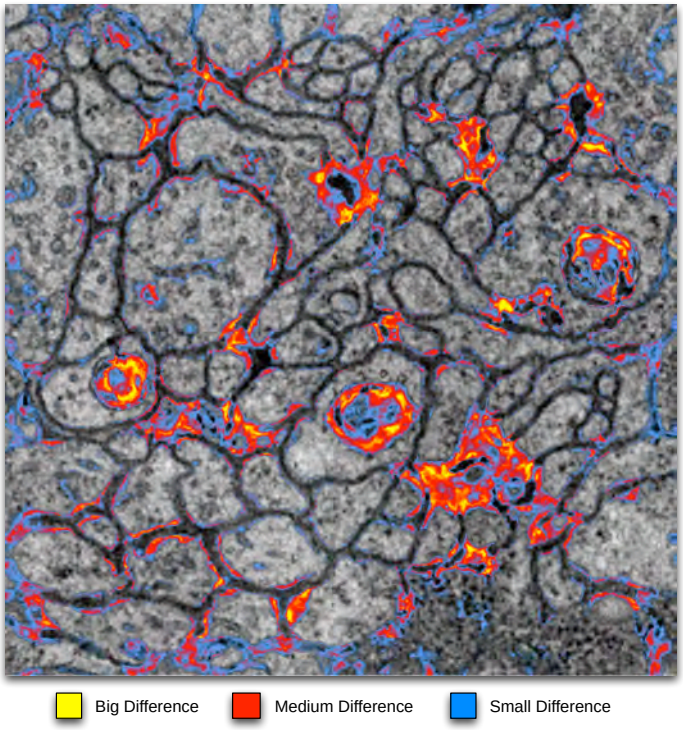
▷ Create encoded image of $\mathcal{C}(x, y; z) = \{c_z(x, y)|z = 1, \ldots, N_{Seg}\}$ confidence images
1: **function** ExplicitModelEncodingVisualization($\mathcal{C}(x, y; z)$,`Op`)
2: $\{w, h\} \leftarrow \{$Width(c_1),Height(c_1)$\}$
3: $e_c \leftarrow$ New2D(w,h)
4: **for** $y \leftarrow 1, h$ **do**
5: **for** $x \leftarrow 1, w$ **do**
6: $\mathbf{v} \leftarrow \mathcal{C}(i, j; :)$ ▷ Sample all values from $\mathcal{C}$ at position (i, j) with $\mathbf{v} \in \mathbb{R}^{N_{Seg}}$
7: **if** `Op` == `Mean` **then**
8: $e(x, y) \leftarrow$Mean($\mathbf{v}$) ▷ Determine mean of value array
9: **else if** `Op` == `Variance` **then**
10: $e(x, y) \leftarrow$Variance($\mathbf{v}$) ▷ Determine variance of value array
11: **else**
12: **if** $N_{Seg} == 2$ **then**
13: $e(x, y) \leftarrow |v_1 - v_2|$ ▷ Determine absolute of difference from $\mathbf{v} = \{v_1, v_2\}$
14: **return** $e(x, y)$

Algorithm 4.4: Explicit encoding of the mean, the variance or the difference between different segmentation results which are provided as confidence images.

The difference in segmentation result images shown in Figure 4.11a can be used to determine the impact of the incremental training sets (as discussed in Section 4.2) on page 53 on the outcome of the segmentation. Yellow regions indicate big differences between two models which have been created using training set T_{i+1} and the (final) incremental training set T_{GT}. Red indicate medium and blue low differences between the two models. We see that the additional selection of



(a) Difference in segmentation results using T_{i+1} and T_{GT}



(b) Variance of segmentation results using T_i, T_{i+1} and T_{GT}

Figure 4.11: Explicit encodings to summarize models using training sets T_i, T_{i+1} and T_{GT} from Figure 4.1, Figure 4.2 on page 56.

compressed, white matter as well as mitochondrion regions (compare Figure 4.1a to Figure 4.2a on page 54 and 55 respectively) lead to a degradation of the prediction performance in other regions such as mitochondria or *membrane* regions (shown in red and yellow). In Figure 4.11b the variance image shows those regions where the variance between the segmentation results using training sets T_i, T_{i+1} and T_{GT} is the greatest. This shows that a careful selection of the training examples (i.e., through conservative sample selection discussed in Section 4.2) can handle mitochondrial regions. The explicit encodings created using Algorithm 4.4 are used as a hint for the next iteration step in Line 13 of the Algorithm 4.1 on page 58.

4.5.1 Uncertainty Visualization

The segmentation output of the random forest classifier is a certainty image which states how confident the machine learning model is that a pixel is belonging to the class *membrane*. Therefore, the visualization of the model uncertainty is straightforward. We use a transfer function to generate the uncertainty image and blend this image over the original intensity image as shown in Figure 4.12 where green means high confidence ($c \geq 0.5$).

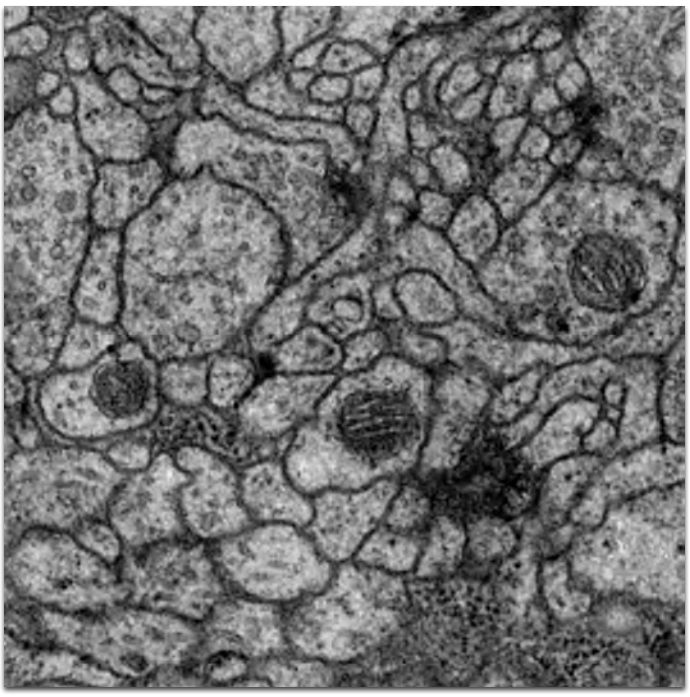
4.5.2 Final Segmentation Result

In the end, the segmentation of ssTEM images should result in a binary volume that indicates which voxels belong to the *membrane* class. For this purpose, the real-valued certainty images $\mathcal{C}(x, y; z)$ need to be converted into a binary image stack $\mathcal{C}_{0,1}(x, y; z) \mapsto \{0, 1\}$ which is a stack of binary indicator functions. The standard approach is to assign an indicator value of 1 to a pixel (x, y) if its certainty value $c(x, y)$ is greater or equal than the *certainty threshold* t_c. Because the certainty value corresponds to the highest number predictions of a class in the random forest relative to the total number of pixels in the image, choosing $t_c = 0.5$ would assign those pixels to the *membrane* class where the majority of the decision trees vote for *membrane*.

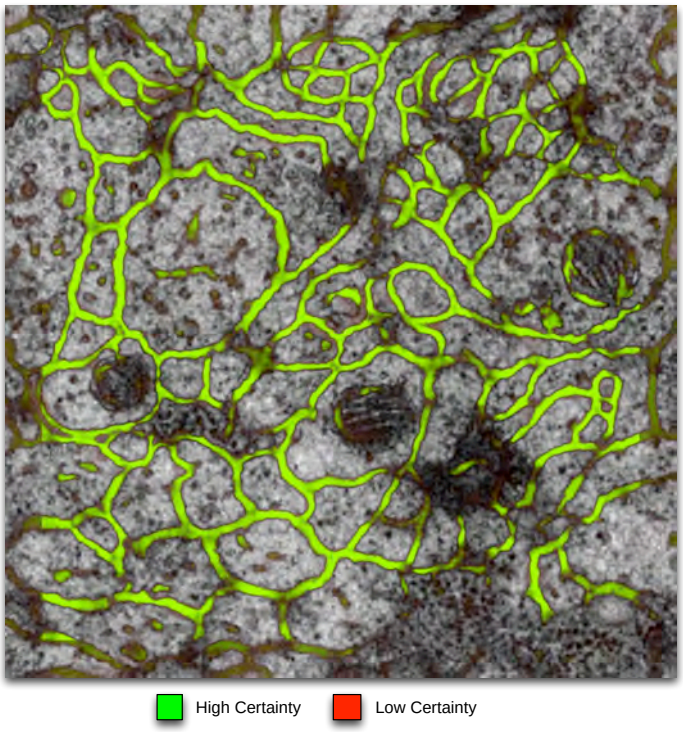
CERTAINTY THRESHOLD

4.5.3 Confusion Matrix Visualization

In this thesis we assumed that the domain knowledge is explicitly provided in terms of a ground truth segmentation. On new data sets where no ground truth is provided, the domain expert who wants to optimize the sample selection and the used feature sets knows which regions in a segmentation result image correspond to errors and which are successfully segmented. Therefore, the domain expert can select manually the errors and the successes made by the random forest segmentation either through brushing (i.e., implicit generation of a ground truth image) or just select the aspect windows without brushing. Either way the domain expert can apply the sample selection and feature exploration methods discussed in this chapter. The aim of this visualization is provide means of identifying the types



(a) Original slice image



(b) Confidence image

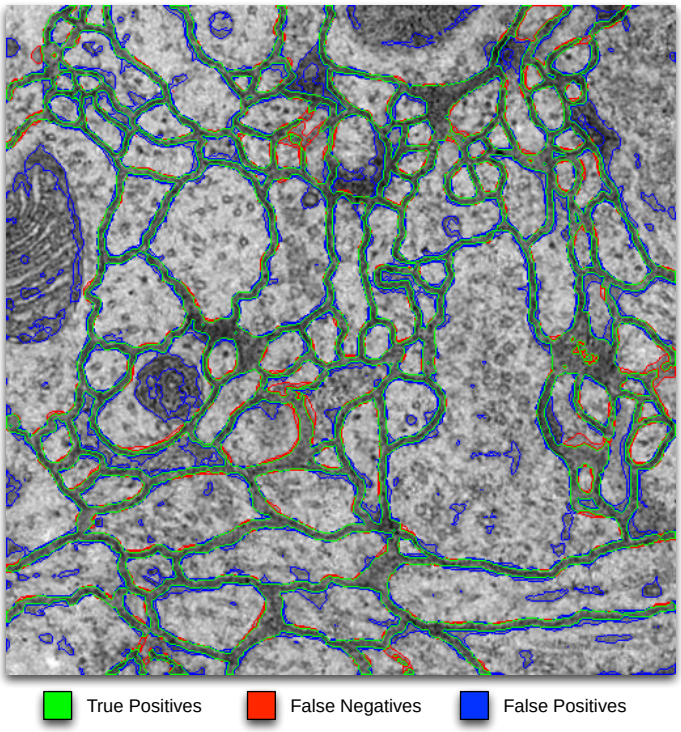
Figure 4.12: Uncertainty visualization for (a) intensity image by (b) blending the uncertainty image over the intensity image.

of errors made by the segmentation procedure and use this information during sample selection and feature exploration.

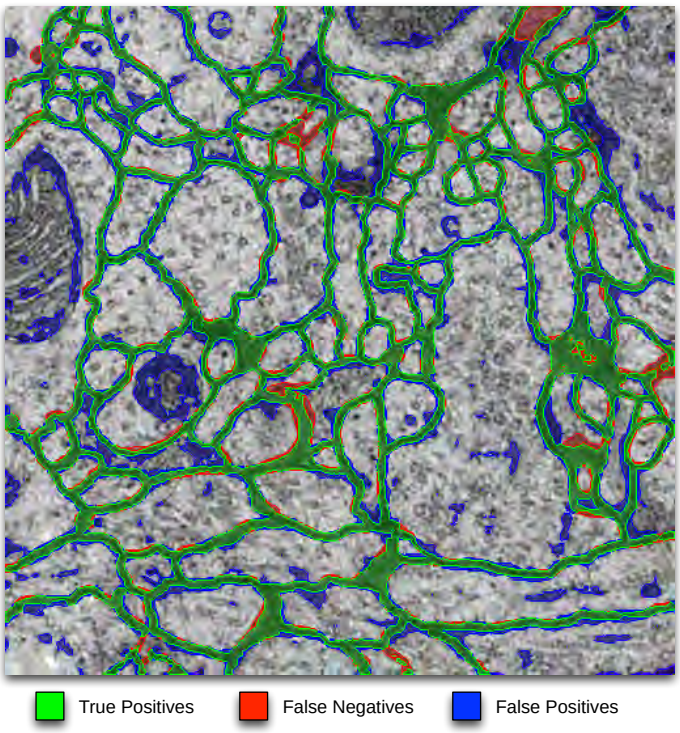
In Section 2.3.3 on page 22 we learned that instead of displaying the certainty values using a transfer function, as shown in Figure 4.12b, we get more information about the type I (FP; blue) and type II errors (FN; red) when visualizing the *confusion matrix image* for the segmentation result. As illustrated in Figure 4.13, the two coloring modes guide the analysis of aspect windows. Figure 4.13a shows transparent ($\alpha = 0$) TP, FP and FN regions which allow the identification of the pixel value distributions within these regions. Figure 4.13b shows an overview image where big error regions are identified more quickly by coloring the TP, FP and FN regions.

CONFUSION MATRIX IMAGE

The synthesis of the confusion matrix image is straightforward and it is shown in Algorithm 4.5. From the ground truth and certainty image we determine four confusion mask images: true positive (TP), true negative (TN), false positive (FP) and false negative (FN) mask images. Each of these indicator images are blended over the previous one using the corresponding alpha-weighted color.



(a) Transparent confusion matrix image
$\alpha = 0.0$



(b) Colored confusion matrix image
$\alpha = 0.5$

Figure 4.13: Two visualization modes of confusion matrix image.

▷ Create TP, TN, FP and FN images and blend them over f

function CreateConfusionMatrixImage($f(x,y)$,$c(x,y)$,$f_{GT}(x,y)$,α)
  $v \leftarrow$ NewColorImage(Width(f),Height(f),$(1,1,1)$) ▷ Initialize to white
  $m_{\oplus}^{GT} \leftarrow f_{GT}(x,y) == 1$ ▷ Determine TP
  $m_{\ominus}^{GT} \leftarrow f_{GT}(x,y) == 0$ ▷ Determine TN
  $m_{\oplus}^{c} \leftarrow c(x,y) \geq 0.5$ ▷ Determine predicted positives
  $m_{\ominus}^{c} \leftarrow c(x,y) < 0.5$ ▷ Determine predicted negatives
  ▷ Determine indicator images for all four categories
  $m_{TP} \leftarrow m_{\oplus}^{GT} \circ m_{\oplus}^{c}$
  $m_{TN} \leftarrow m_{\ominus}^{GT} \circ m_{\ominus}^{c}$
  $m_{FP} \leftarrow m_{\ominus}^{GT} \circ m_{\oplus}^{c}$
  $m_{FN} \leftarrow m_{\oplus}^{GT} \circ m_{\ominus}^{c}$
  ▷ Color each category and blend the results
  $\{m_{RGB},\ m_I\}$ ColorRegion(m_{TP}, (0,1,0,1), α,true)
  $v(x,y) \leftarrow$ BlendImages(m_{RGB},$f(x,y)$,m_I)
  $\{m_{RGB},\ m_I\}$ ColorRegion(m_{TN}, (0,0,0,0), α,false) ▷ Skip TN
  $v(x,y) \leftarrow$ BlendImages(m_{RGB},$v(x,y)$,m_I)
  $\{m_{RGB},\ m_I\}$ ColorRegion(m_{FP}, (0,0,1,1), α,true)
  $v(x,y) \leftarrow$ BlendImages(m_{RGB},$v(x,y)$,m_I)
  $\{m_{RGB},\ m_I\}$ ColorRegion(m_{FN}, (1,0,0,1), α,true)
  $v(x,y) \leftarrow$ BlendImages(m_{RGB},$v(x,y)$,m_I)
  return $v(x,y)$

▷ Color a region indicated by *Idx* using color **p** and enhance the perimeter of the region if *usePerimeter* is true
function ColorRegion(*Idx*(x,y),**p**,α,*usePerimeter*)
  $v_C(x,y) \leftarrow$ NewColorImage(Width(*Idx*),Height(*Idx*),$(1,1,1)$)
  $m_p(x,y) \leftarrow$ Perimeter(*Idx*) ▷ Get perimeter image
  if *usePerimeter* **then**
    $m_\alpha(m_p) \leftarrow 1$ ▷ Assign for perimeter pixels m_p alpha value 1
  $m(x,y) \leftarrow Idx \cdot \alpha$ ▷ Create alpha-weighted mask image
  $v_C(x,y) \leftarrow (m_\alpha(x,y), m_\alpha(x,y), m_\alpha(x,y))$ ▷ Convert to RGB image
  return $v_C(x,y) \leftarrow m_\alpha(x,y) \circ \mathbf{p}$ ▷ Color region by incorporating α into channels

▷ Blend images using α for f_1 and $(1-\alpha)$ for f_2
function BlendImages($f_1(x,y)$,$f_2(x,y)$,α)
  $\{w,h\} \leftarrow$ {Width(f_1),Height(f_1)}
  $v_C(x,y) \leftarrow$ NewColorImage(w,h,$(1,1,1)$)
  return $v_C(x,y) \leftarrow f_1(x,y) \cdot \alpha + f_2(x,y) \cdot (1-\alpha)$

Algorithm 4.5: Creation of confusion matrix image.

Chapter 5

Results & Discussion

In Section 2.3.1 on page 20 we discussed that multiple classification *models* can be learned from experience by configuring either (a) the parameters of the machine learning method (as discussed in Section 2.3.5), (b) the feature space (as discussed in Chapter 4) or (c) by using different training samples (as discussed in Section 4.2). In Section 2.3.3 we learned that the *evaluation function* is used to compare the segmentation results of different models. Depending on the chosen function we get the *prediction accuracy/prediction error* which is the standard way to describe the effectiveness of a model. Because the random forest classifier is applied on the image pixels where the expert user needs to select the training sample manually it is advantageous to categorize the errors into type I and type II errors (as discussed in Section 4.5.3). This is achieved by comparing models using the *confusion matrix*. For this purpose we also use (a) the *sensitivity* to measure the proportion of the *membrane* pixels which are classified as *membranes*, and (b) the *specificity* to measure the rate of correctly classified *non-membrane* pixels. That the size of the entire ssTEM drosophila fly brain volume is $512 \times 512 \times 30 = 7,864,320$ voxels.

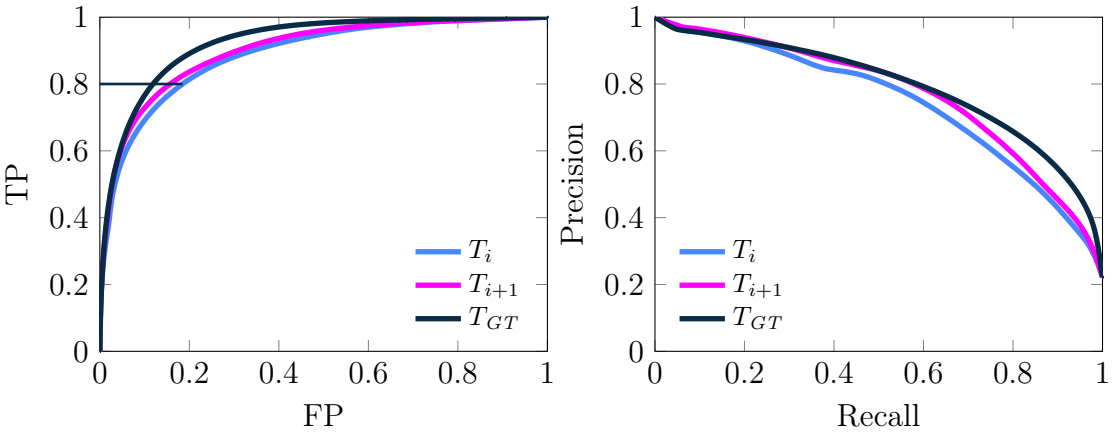
5.1 Evaluation of Sample Selection

In Section 4.2 we have seen that selecting the training examples iteratively lead to a better prediction performance than using the entire ground truth image as the training sample. This result is consistent with the machine learning literature which explains this phenomenon as being caused by model over-fitting. In the machine learning literature, the selection of the same examples for training and testing is *not* a valid method for model evaluation. Instead cross-validation is used to get more representative model estimates. In our case the aim is not to test the prediction capabilities of the model but instead we evaluate different training sample selection strategies for the same classifier using different feature sets.

In Figure 4.1, Figure 4.2 and Figure 4.3 we have seen the prediction accuracies for the corresponding training samples T_i, T_{i+1} and T_{GT} for a single slice of the drosophila fly brain dataset discussed by Cardona et al. [2012]. The subscripts i and $i+1$ denote that the incremental training set construction result in two consecutive selections of training sets. The training set with subscript $i+1$ has is created by adding new training examples to the training set with subscript i. While for the initial training sample T_i the prediction accuracy on that specific

slice is $\varepsilon = 0.1315$, for the incremental training sample T_{i+1} the lower error rate is $\varepsilon = 0.1169$ ($\varepsilon = 0.1169$ means that 11.69, % of the pixels have been misclassified). When using the ground truth image as the training sample we get an error rate of $\varepsilon = 0.1296$. The drosophila dataset is a stack of 30 512×512 images. Segmenting the entire volume with T_i, T_{i+1} and T_{GT} from only one slice leads to error rates of $\varepsilon_{T_i} = 0.1366$, $\varepsilon_{T_{i+1}} = 0.1255$ and $\varepsilon_{T_{GT}} = 0.1461$ respectively.

Figure 5.1 shows the ROC plot and the precision-recall plot for the entire drosophila brain volume. For example, from the ROC plot (as shown in Fig-



(a) ROC curves (b) Precision-recall curves

Figure 5.1: Evaluation curves for confidence-based segmentation results for three training samples T_i, T_{i+1} and T_{GT} shown in Figure 4.1 on page 54, Figure 4.2 on page 55 and Figure 4.3 on page 56 respectively.

ure 5.1a) we see that in order to get 80 % of the true *membrane* voxels the best choice between these three training samples is to use the ground truth sample because for 80 % true *membranes* we only get ~ 12 % of false *membranes*. We also see that the incremental training sample T_{i+1} is a better choice for training than the initial training sample T_i for every true positive rate.

We also plot the corresponding precision-recall (PR) curves in Figure 5.1b which takes the skewness in the dataset into account (Davis and Goadrich [2006]). *Skewed* means: huge difference between the number of *positive* and *negative* examples. In our case: 1,745,293 positives to 6,119,027 negatives. It captures the skewness because the precision $\varpi = TP/(TP + FP)$ compares the false positives to the true positives rather than to the true negatives (Davis and Goadrich [2006]). While the difference between the training samples T_i and T_{i+1} is small in the ROC curve, in the PR curve we see a clearer evidence for emphasizing the choice of T_{i+1} over T_i. This is because the PR curve for T_{1+1} is closer to the optimum classifier (top-right corner) than T_i. But selecting an entire slice T_{GT} of 262,144 pixels lead to better performance in the PR curve on the entire volume of 30 slices.

5.2 Feature Selection

In Chapter 3 we have discussed the original feature set of 90 features which can be divided into four categories: (a) the original density feature f from which all other features are derived, (b) the eight rotation-based template matching features $(R_0, \ldots, R_7)$ with six derivations (Min_R, Max_R, μ_R, σ_R^2, $Median_R$ and $Diff_R$), (c) the per-pixel local histogram features split into ten bins $(h_1, \ldots, h_{10})$ with two derivations (μ_H and σ_H), and (d) different images which have been smoothed using the Gaussian filters G_σ with varying standard deviations σ (see Figure 3.10, Figure 3.11 and Figure 3.12). Because each category serves a specific purpose we organize the feature selection process in such a way that each category is represented by a category–sub-set. Formally, let $\mathcal{F} = \mathcal{F}_f \cup \mathcal{F}_R \cup \mathcal{F}_H \cup \mathcal{F}_G$ be the entire feature set which consists of four categories with $|\mathcal{F}_f| = 1$, $|\mathcal{F}_R| = 14$, $|\mathcal{F}_H| = 12$ and $|\mathcal{F}_G| = 63$. The aim is to find for each feature category a feature sub-set $\mathcal{F}'_f \subseteq \mathcal{F}_f$, $\mathcal{F}'_R \subseteq \mathcal{F}_R$, $\mathcal{F}'_H \subseteq \mathcal{F}_H$, $\mathcal{F}'_G \subseteq \mathcal{F}_G$ that is relevant to the purpose of learning.

As discussed in Section 4.3.1 on page 59, let us first explore the feature redundancy for each category. For this purpose, we introduce a sub-view shown at the top of Figure 5.2. The sub-view encodes the column-wise summation of all absolute values of correlation coefficients in the re-configured CCM view. The CCM view differs from the one shown in Figure 4.5 on page 61. Instead of displaying only the upper triangular matrix we show the entire correlation coefficient matrix. Furthermore, instead of distinguishing between positive and negative correlation coefficients we take the absolute value of those and encode them as dark blue for high correlation and white as no correlation in the CCM view. The color legend of the sub-view at the top shows the sum of absolute value of the feature correlations. The higher the value (yellow) for a feature the higher the correlation of that feature when compared to all other features. This additional view allows us to threshold those features which in sum are redundant to the others. In other words: the more high correlation coefficients a feature has (in its column of the CCM) the yellower it is in the sub-view.

Let us start with the smallest category $\mathcal{F}_f$ which consists of only one feature namely the density image f (highlighted in red in Figure 5.2). Because all other features are derived from this feature by either reducing noise or introducing new artifacts it makes sense to leave it in the feature set. Noise is reduced through the Gaussian smoothing of the images. Artifacts are introduced for example through the determination of the per-pixel local histograms which results in hard edges shown in row σ_H in Figure 4.9 on page 69.

The second feature category $\mathcal{F}_R$ introduces the results of the rotation-invariant membrane matching R_i for rotation angle $\varphi = i \cdot \pi/8$ for $i = 0, \ldots, 7$ and are summarized by the Min_R, Max_R, μ_R, σ_R^2, $Median_R$ and $Diff_R$ features. Even though the features R_i are in sum having smaller correlation coefficients than its derivations we found out that adding derivations to the smaller feature set is more suitable as adding the R_i images to $\mathcal{F}_R$. We found out about this by comparing the prediction results by using 90 features and 82 features. Furthermore, we prefer

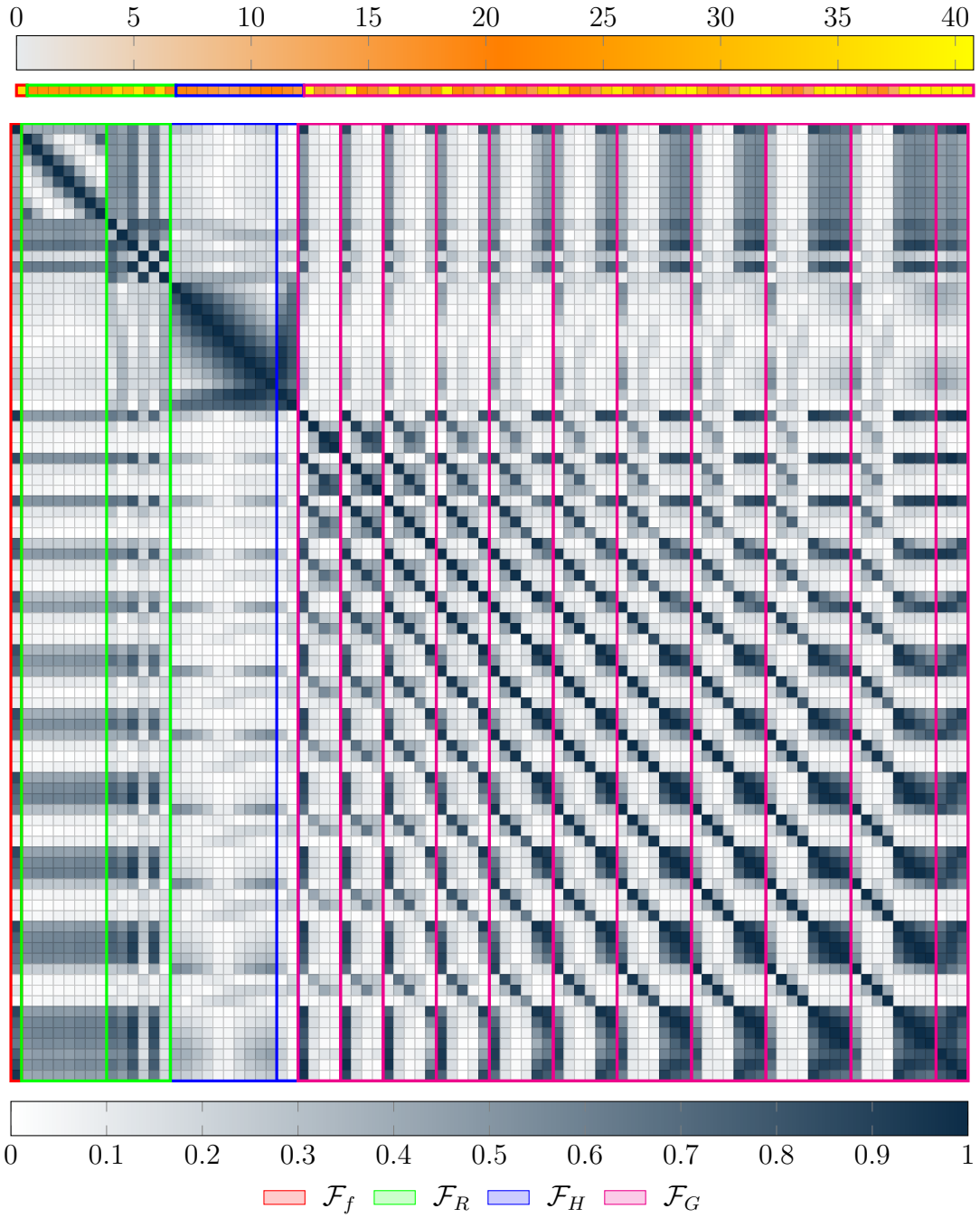


Figure 5.2: CCM view of original 90 features extended by column-wise summation sub-view at the top. Instead of showing the upper triangular matrix we show the full matrix.

the choice of the features derived from rotation-invariant matching because we use those features later in the construction of new features as discussed in Section 4.4.2. We also visually split the set of rotation-invariant membrane matching features into two sub-categories the left one (with eight columns in CCM view) indicates the correlation coefficients for the R_i features while the right one (with six columns) indicates the derivations of R_i.

For the features created with per-pixel local histograms h_b with $b = 1, \ldots, 10$ we select the derivations μ_H and σ_H over h_b because the validation of the corresponding feature sets lead to better results. Another question that arises is whether we can omit one of those derived features. The answer is shown in Figure 5.3 which indicates that using both features is more helpful in separating the positive from the negative examples because of the intra-class covariance.

The Gaussian-smoothed feature sub-set $\mathcal{F}_G = \{\mathcal{F}_{G_1}, \ldots, \mathcal{F}_{G_{10}}, \mathcal{F}_{G'}\}$ consists of 63 features (highlighted in magenta in Figure 5.2). In Section 3.5 we have seen that beside the original intensity image f, we also smooth the eigenvalue images $G_\sigma(\lambda_1/\lambda_2)$ as well as the gradient magnitude images $G_\sigma(\|\nabla G_1(f)\|)$, and we determine the gradient magnitude images from the smoothed original intensity image $\|\nabla G_\sigma(f)\|$. These four feature images are part of feature sub-set

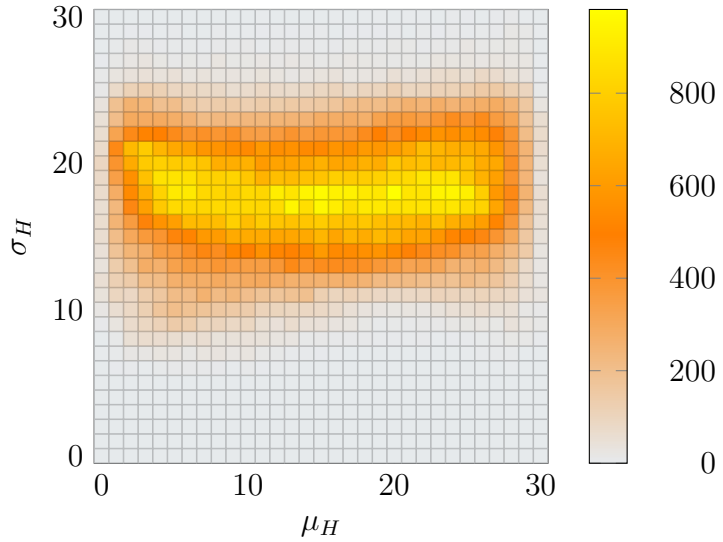
$$\mathcal{F}_{G_\sigma} = \left\{ G_\sigma(f), G_\sigma(\lambda_1/\lambda_2), G_\sigma(\|\nabla G_1(f)\|), \|\nabla G_\sigma(f)\| \right\} \cup \mathcal{F'}_{G_\sigma} . \tag{5.1}$$

$\mathcal{F}_{G_\sigma}$ is created for each $\sigma = 1, \ldots, 10$. In addition, $\mathcal{F'}_{G_\sigma}$ is the corresponding set of difference images $G_\sigma - G_{\sigma'-2}$ for $\sigma = 3, \ldots, 10$ and $\sigma' = 2l + 1$ where $\in \mathbb{N}$ and $\sigma' \leq \sigma - 2$ as discussed in Section 3.5 on page 44. In Figure 5.2 the magenta vertical lines mark the $\mathcal{F}_{G_\sigma}$. The correlations are visualized as dark clusters shown in sub-set $\mathcal{F}_G$ in Figure 5.2. Although, smoothed images produce correlations they also provide a way to introduce neighborhood information which is essential for the modeling of *membranes*. Therefore, from each sub-set of $\mathcal{F}_{G_\sigma}$ we select the $G_\sigma(f)$ features and those features where the correlation (in sum) is lower than $t_c = 23$ which is determined through trial-and-error feature set validation.

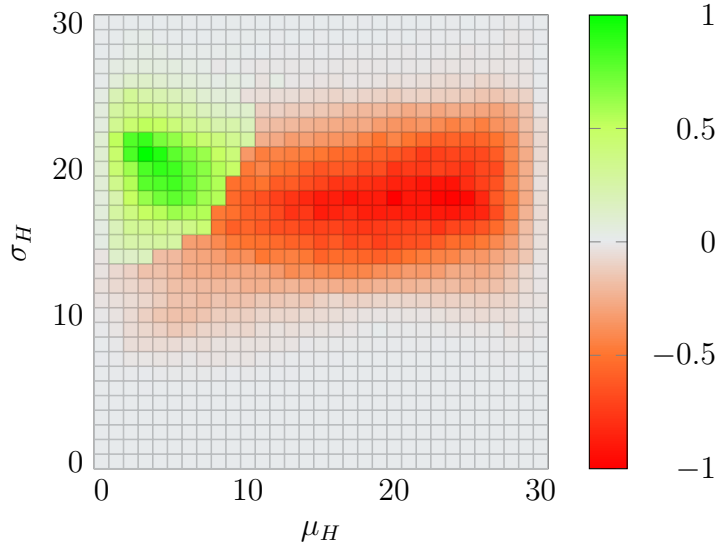
The results of this correlation-based feature filtering is shown in Figure 5.4 where white vertical stripes are marking those features which are excluded from $\mathcal{F}$. Out of 90 features only 52 are used for the segmentation which is a reduction of $\sim 42\,\%$. The reduction of features implies also a reduction of memory space as well as of computation time. Furthermore, the features belonging to the four categories are highlighted in the corresponding colors as well.

5.3 Segmentation Performance

In order to evaluate the effectiveness and efficiency of the proposed methods we present five experiments that measure (a) the mean error rate, (b) the mean precision and (c) the mean (this means: *mean* with regard to all 30 slices of the ssTEM drosophila brain data set) recall, as well as the time needed to (d) extract the features, (e) to train and (f) to test the random forest model on the entire volume. The measurements are presented in Table 5.1 for the three training samples T_i,



(a) Two-dimensional histogram



(b) Per-class scaled histogram plot

Figure 5.3: Intra-class covariance for μ_H and σ_H features.

T_{i+1} and T_{GT} shown in Figure 5.1.

5.3.1 Experiment 1: Full Feature Set

This experiment is intended to show the prediction performances using the entire feature set $\mathcal{F}_{90} = \mathcal{F}_{\{} \cup \mathcal{F}_R \cup \mathcal{F}_H \cup \mathcal{F}_G$ as shown in the CCM view in Figure 5.2. This is the reference feature set to which all other (smaller) feature sets are compared to. We see that the prediction error rate for the training set T_{i+1} (12.70,%) is the lowest among T_i (15.13, %) and T_{GT} (14.64, %), and because of being the largest

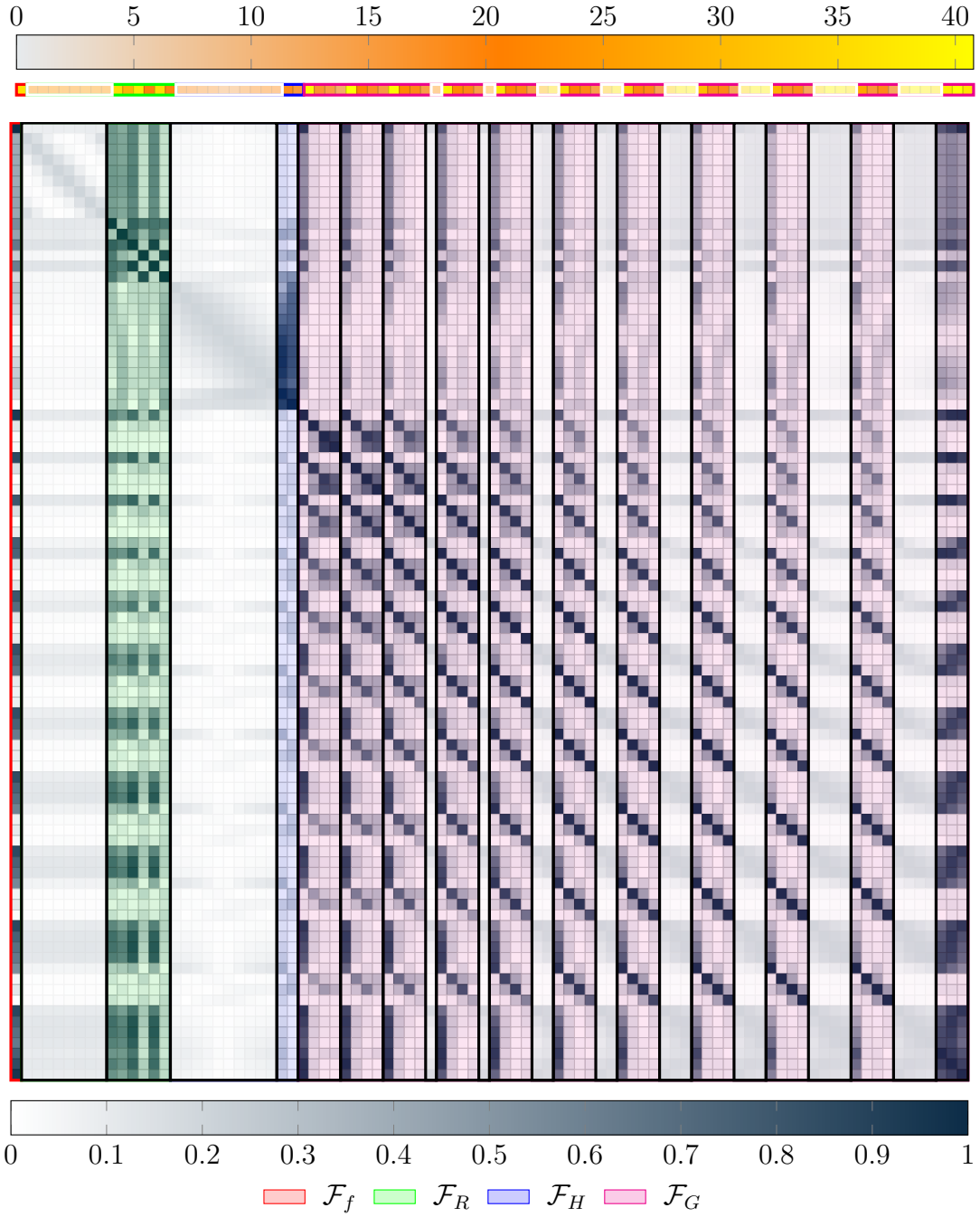


Figure 5.4: Results of feature selection showing removed features from each category (highlighted as white vertical stripes).

		Segmentation Results				Timing Results (sec)		
		T_i	T_{i+1}	T_{GT}		T_i	T_{i+1}	T_{GT}
	# Positives	1,358	1,613	61,451				
	# Negatives	1,636	2,350	200,691				
	Size in %	0.04	0.05	3.33				
	Error	0.1513	*0.1270*	0.1464	Extraction		975	
$\mathcal{F}_{90}$	Precision	0.6731	*0.7914*	0.6294	Training	2	3	94
	Recall	0.6528	0.5910	*0.8300*	Testing	151	154	251
	Error	0.1505	*0.1277*	0.1448	Extraction		891	
$\mathcal{F}_{52}$	Precision	0.6784	*0.7893*	0.6349	Training	2	3	60
	Recall	0.6457	0.5910	*0.8302*	Testing	147	154	182
	Error	0.1560	*0.1305*	0.1472	Extraction		**877**	
$\mathcal{F}_{49}$	Precision	0.6601	*0.7822*	0.6297	Training	**2**	**3**	**57**
	Recall	0.6510	0.5831	*0.8308*	Testing	**142**	**150**	**181**
	Error	0.1353	**0.1248**	0.1464	Extraction		994	
$\mathcal{F}_{90+1}$	Precision	0.7333	**0.8053**	0.6290	Training	4	5	96
	Recall	0.6307	0.5834	**0.8316**	Testing	198	212	254
	Error	0.1497	*0.1298*	0.1473	Extraction		892	
$\mathcal{F}_{52+1}$	Precision	0.6792	*0.7918*	0.6285	Training	2	3	62
	Recall	0.6451	0.5736	*0.8282*	Testing	150	157	185
	Error	0.1558	*0.1336*	0.1479	Extraction		884	
$\mathcal{F}_{49+1}$	Precision	0.6607	*0.7778*	0.6277	Training	2	3	58
	Recall	0.6472	0.5690	*0.8287*	Testing	144	152	178

Table 5.1: Segmentation results of feature exploration and creation for six different feature sets as well as timing results for building the feature set, training and testing. The corresponding minimum and maximum values for each feature set is highlighted in *italic*. The corresponding overall minimum and maximum values are highlighted in **bold**.

feature set, the feature extraction takes the longest. The question is, wether we can reach the same (or better) prediction error rates by using a smaller feature set which has the benefit of requiring less processing time. Furthermore, we can see that the mean rate of positively identified membrane pixels (79.14,%) for all 30 slices is in the middle of all experiments. In Figure 5.1 we see the ROC and the precision-recall curves for the entire data set using all three training samples T_i, T_{i+1} and T_{GT} and all 90 features. Using more data T_{GT} leads to better mean precision-recall rates.

5.3.2 Experiments 2, 3: Reduced Feature Sets

The second and the third experiment measure the effectiveness of the reduced feature sets $\mathcal{F}_{52}$ and $\mathcal{F}_{49}$ after the feature selection described in Section 5.2. The reason for the evaluation of both feature sets instead of just one is to evaluate the relevance of highly correlating but possibly relevant features which are encoded in the last three columns ($\mathcal{F}_{G'}$) in the 90×90 CCM view, as shown in Figure 5.4.

From Table 5.1 we see that the reduced feature set $\mathcal{F}_{52}$ has a similar error rate ($12.77, \%$) compared to the full feature set $\mathcal{F}_{90}$ ($12.70, \%$). The most important benefit in this regard is the speedup in segmentation: the feature extraction step is speeded up by $8.62, \%$. Although the speed up effect in case of the training is negligible for small training sets ($\sim 0\,\%$), it is noticeable ($36.17, \%$) for large training sets (such as T_{GT}) when using the entire feature set $\mathcal{F}_{90}$. The same effect can also be observed for the testing of the entire data set ($27.49, \%$).

In order to answer the question of whether the three highly correlating features $\mathcal{F}_{G'}$ are still relevant to the prediction task we compare the error rates for the training samples $\mathcal{F}_{52}$ (with $\mathcal{F}_{G'}$) and $\mathcal{F}_{49}$ (without $\mathcal{F}_{G'}$). We see that using a smaller feature set $\mathcal{F}_{49}$ we need to take a higher error rate into account when compared to $\mathcal{F}_{52}$.

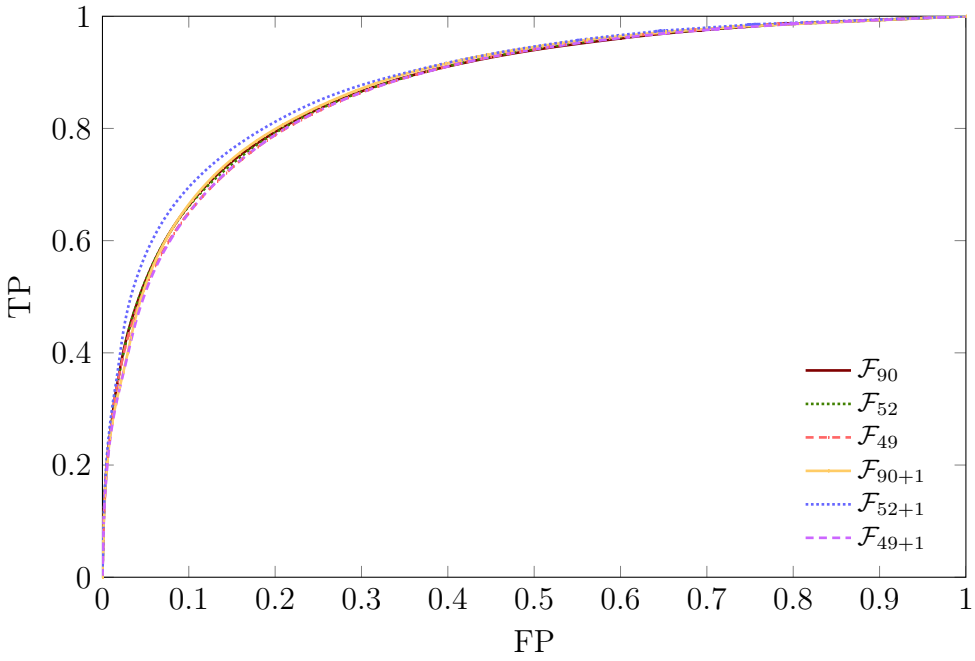
5.3.3 Experiments 4, 5, 6: Effect of Feature Construction

In Section 4.4.2 we learned that using aspects and the information displayed in the aspects (i.e., as ground truth and feature images) we can construct algorithms that produce new features. An expert user either knows implicitly (i.e., through user knowledge) or explicitly (i.e., as ground truth image) where the prediction model succeeds (true positive and true negatives) and where it fails (type I and type II errors). This information is used to guide the development of a feature creation algorithm as illustrated in an example in Algorithm 4.3 on page 72. The new feature is constructed with the aim to increase the number of true positives and true negatives.

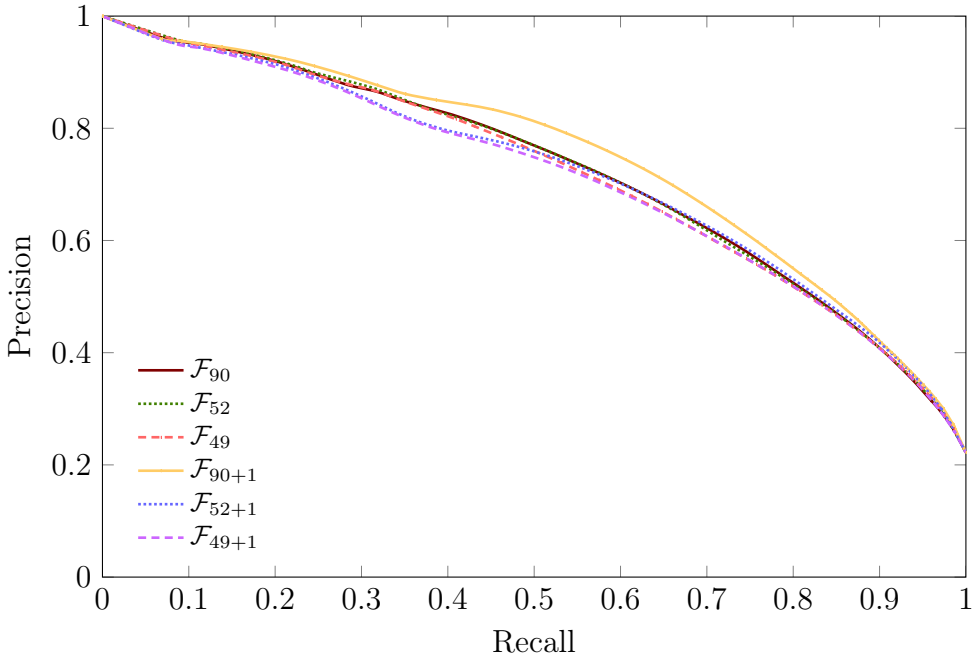
The results of these experiments are shown in Table 5.1 where in case of $\mathcal{F}_{90+1}$ we have an increase in precision of 1.39 percentage points when compared to $\mathcal{F}_{90}$. But this increase in precision does not necessarily lead to overall lower error rates as shown for feature sets $\mathcal{F}_{49}$ and $\mathcal{F}_{49+1}$ ($13.05, \%$ compared to $13.36\,\%$). Yet constructing new features can lead to overall better prediction results $12.48, \%$.

5.3.4 ROC & Precision-Recall Curves

Figure 5.5, Figure 5.6 and Figure 5.7 compares the performances of all evaluated feature sets. From the precision-recall plot of the initial training sample T_i (see Figure 5.5b) we see that by adding a new constructed feature to the original feature set $\mathcal{F}_{90}$ the recall rate increases from $\sim 44\,\%$ to $\sim 52\,\%$ for the fixed precision of $80\,\%$. But this effect diminishes for larger data sets. Furthermore, we also see that the the larger the training sample the more similar the outcomes of the prediction models are for different feature sets. This implies that the real benefit of using feature exploration is to reduce the number of features which results in smaller processing times as well as memory requirement.

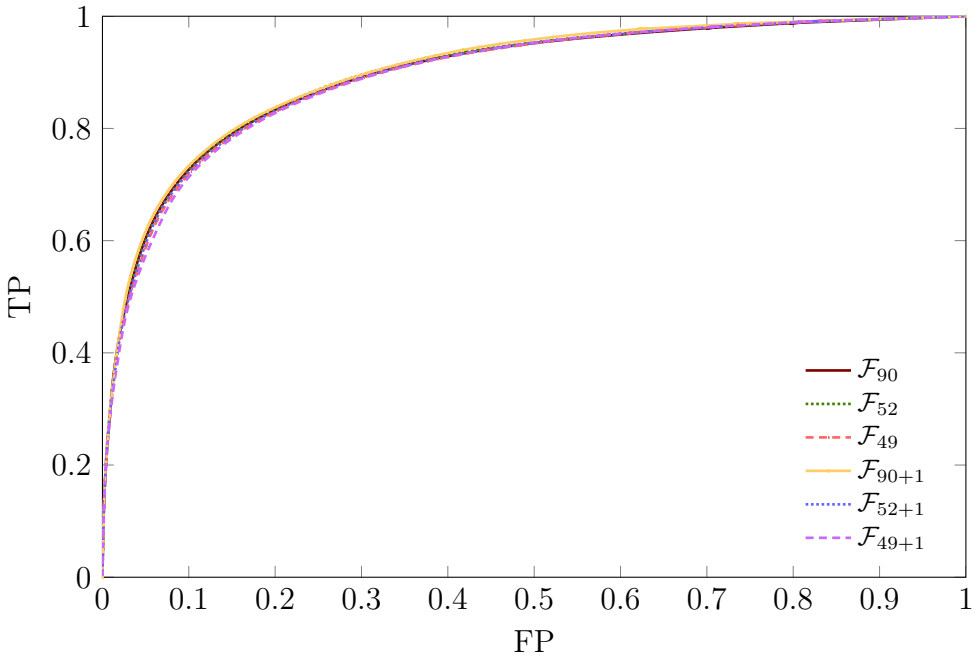


(a) ROC curves for T_i

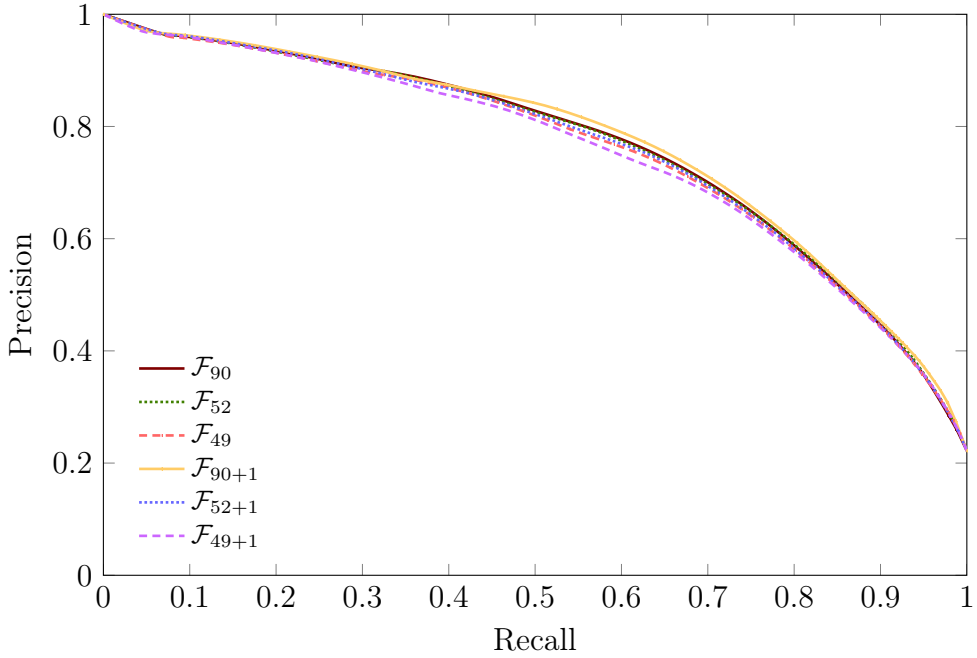


(b) Precision-recall curves for T_i

Figure 5.5: Evaluation curves for all evaluated feature sets and training set T_i.

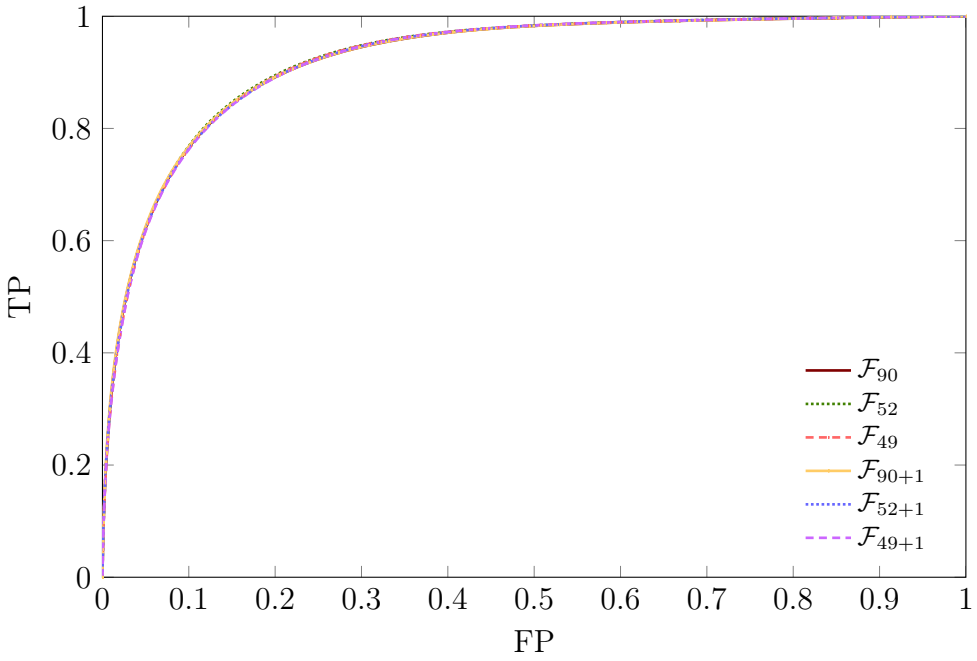


(a) ROC curve for T_{i+1}

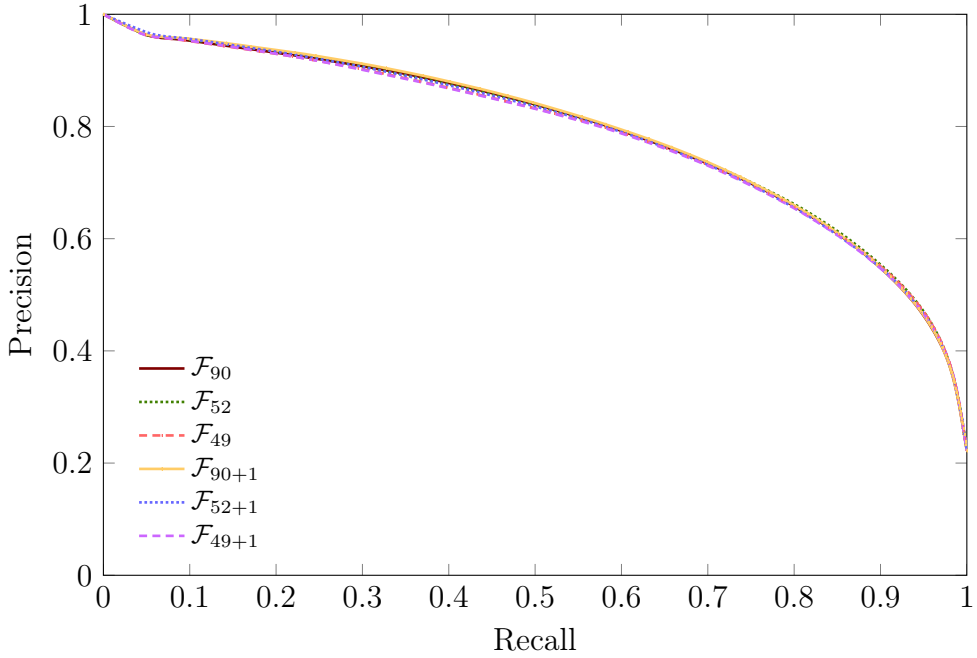


(b) Precision-recall curves for T_{i+1}

Figure 5.6: Evaluation curves for all evaluated feature sets and training samples T_{i+1}.



(a) ROC curve for T_{GT}



(b) Precision-recall curves for T_{GT}

Figure 5.7: Evaluation curves for all evaluated feature sets and training samples T_{GT}.

Chapter 6

Conclustion & Future Work

In this chapter we discuss the contributions of this work and conclude with the observations made through the experiments. Furthermore we also suggest future directions on the topic of visualization-assisted, machine learning-based ssTEM image segmentation.

6.1 Conclusion

We have learned that segmentation is a crucial part in image understanding of ssTEM images. The segmentation performance – either through prediction accuracy or processing times – is even more crucial when dealing with large images having a lot of artifacts due to the image capturing technique. Moreover statistical segmentation requires a careful selection of both the training sample as well as the feature set that represents the concept well.

This thesis presents an interactive exploration approach to feature image selection and feature image creation for ssTEM brain tissue volumetric data sets. It combines the expressiveness of scientific visualization through transfer functions with *abstract* and *elaborate* interaction techniques. It further helps bridging the gap between computational neuroanatomy, visualization and machine learning for the purpose of the segmentation of neuroanatomical brain tissue data sets.

The field of image processing provides a great number of image transformation techniques. Therefore, finding an appropriate feature image set that is transformed from the original intensity image is of special concern, especially for large ssTEM data sets which are used for machine learning-based segmentation. In this work we show how the search for a proper feature set can be organized. Furthermore we also compare the effects of different training samples on the outcome of the prediction. We demonstrated the construction of a training set using a conservative sample selection algorithm.

The filtering of features based on correlation coefficients lead to a smaller but as representative feature set as the original one. For this purpose we introduce a correlation coefficient matrix view that instantly shows the correlation values and helps to narrow down the search for redundant and relevant features. This search is further improved through two-dimensional histograms that indicate intra-class covariance which categorize redundant features into *truly redundant* and *indeed redundant but not-redundant in combination with others*. Furthermore we show

that simple histograms are a useful tool for estimating the effectiveness of single features as well as training samples.

The evaluation of the visual feature exploration approach solely on prediction outcome suggests that adding features to the feature set can help to improve the prediction accuracy. However, visual feature exploration for image segmentation is a subject for further research. Moreover adding new features to the feature set penalizes the pre-processing, training and testing of the volumes in terms of the execution times and memory requirement. This effect is further enhanced for larger volumetric data sets present in computational neuroanatomy.

From this work we conclude that careful consideration of the training samples as well as the feature sets lead to improvements in the prediction accuracy and/or to decreased computational costs in terms of processing times as well as in terms of storage requirements. We also conclude that areas of interest can be visualized effectively by using *aspects* as image patches in combination with *confusion matrix views* which show were the prediction model does make mistakes and which features can be used to avoid these mistakes.

6.2 Directions for Future Work

New feature categories may require other statistical evaluation methods than correlation coefficients as well as other visualization techniques in order to speed up the search for relevant features. We have observed the limits of the correlation for two the feature categories *rotation-invariant object matching* and *per-pixel local histograms*. Here, correlation coefficients were not helpful in deciding feature relevance. We solved this issue through trial-and-error but a more organized search would be preferable.

The state-of-the art segmentation methods of ssTEM images are using machine learning as a tool of handling large datasets automatically. The question remains of whether machine learning alone is suitable for the segmentation of artifact afflicted ssTEM images. In work of Kaynig-Fittkau [2011], which this thesis is based on, the author states that the result of the machine learning is used as a starting point for the membrane segmentation which closes spatial gaps that occur because the segmentation results are statistics-based and not spatial-based. Therefore, finding new features through the transformation of the density volume that capture the spatial relationships better than image smoothing is necessary.

Another interesting future direction is to improve the aspect-oriented feature exploration. In this work aspects are only showing a single feature together with either the ground truth or the confusion matrix image. In order to speed up the search for an adequate feature which is then used to construct new features a *connection* view might be helpful (from terminology introduced by Yi et al. [2007]). This view would show the related features, whose values in TP, TN, FP and FN regions fit better than the considered aspect feature.

Finally, with regard to the feature exploration, interactive steering can be achieved by segmenting the aspect window images (e.g., 128×128) instantly through the parallelization the random forest classifier. In this way it would be

possible to keep track of the local changes in the segmentation result which are influenced by both the training sample as well as feature set selection.

Bibliography

B. Auffarth, M. López, and J. Cerquides. Comparison of redundancy and relevance measures for feature selection in tissue classification of CT images. In *Proceedings of the 10th Industrial Conference on Advances in Data Mining: Applications and Theoretical Aspects*, ICDM'10, pages 248–262, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-14399-7, 978-3-642-14399-1. URL http://dl.acm.org/citation.cfm?id=1880672.1880695.

A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, Dec. 1997. ISSN 0004-3702. doi: 10.1016/S0004-3702(97)00063-5. URL http://dx.doi.org/10.1016/S0004-3702(97)00063-5.

L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL http://dx.doi.org/10.1023/A:1010933404324.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

T. Brox, R. Boomgaard, F. Lauze, J. Weijer, J. Weickert, P. Mrazek, and P. Kornprobst. Adaptive structure tensors and their applications. In J. Weickert and H. Hagen, editors, *Visualization and Processing of Tensor Fields*, Mathematics and Visualization, pages 17–47. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-25032-6. doi: 10.1007/3-540-31272-2_2. URL http://dx.doi.org/10.1007/3-540-31272-2_2.

A. Cardona, S. Saalfeld, J. Schindelin, I. A. Carreras, S. Preibisch, M. Longair, P. Tomancak, V. Hartenstein, and R. Douglas. Trakem2 software for neural circuit reconstruction. *PLoS ONE*, 7(6):e38011–e38011, 2012. ISSN 1932-6203. doi: 10.1371/journal.pone.0038011. URL http://tinyurl.sfx.mpg.de/t2js.

J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143874. URL http://doi.acm.org/10.1145/1143844.1143874.

P. Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, Oct. 2012. ISSN 0001-0782. doi: 10.1145/2347736.2347755. URL http://doi.acm.org/10.1145/2347736.2347755.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000. ISBN 0471056693.

G. Forman and M. Scholz. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *SIGKDD Explor. Newsl.*, 12(1):49–57, Nov. 2010. ISSN 1931-0145. doi: 10.1145/1882471.1882479. URL http://doi.acm.org/10.1145/1882471.1882479.

M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10 (4):289–309, Oct. 2011. ISSN 1473-8716. doi: 10.1177/1473871611416549. URL http://dx.doi.org/10.1177/1473871611416549.

R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006. ISBN 013168728X.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, Mar. 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944968.

B. Jähne. *Digital Image Processing 6th Edition*. Springer, Berlin, 2005. ISBN 3540240357 9783540240358.

C. Johnson and C. Hansen. *Visualization Handbook*. Academic Press, Inc., Orlando, FL, USA, 2004. ISBN 012387582X.

V. S. Kaynig-Fittkau. *Machine learning approaches for neuron geometry extraction and synapse detection in electron microscopy images*. PhD thesis, ETH Zürich, Zürich, 2011.

D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Information visualization. In A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, editors, *Information Visualization: Human-Centered Issues and Perspectives*, chapter Visual Analytics: Definition, Process, and Challenges, pages 154–175. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-70955-8. doi: 10.1007/978-3-540-70956-5_7. URL http://dx.doi.org/10.1007/978-3-540-70956-5_7.

G. Knott, H. Marchman, D. Wall, and B. Lich. Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling. *The Journal of Neuroscience*, 28(12):2959–2964, 2008. doi: 10.1523/JNEUROSCI.3189-07.2008. URL http://www.jneurosci.org/content/28/12/2959.short.

R. Kohavi and R. Quinlan. Decision tree discovery. In *Handbook of Data Mining and Knowledge Discovery*, pages 267–276. University Press, 1999.

T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, Aug. 2009. ISSN 0036-1445. doi: 10.1137/07070111X. URL http://dx.doi.org/10.1137/07070111X.

R. Kumar, A. Vazquez-Reina, and H. Pfister. Radon-Like Features and Their Application to Connectomics. In *Workshop on Mathematical Methods in Biomedical Image Analysis*, pages 186–193, 2010.

D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002. ISBN 0521642981.

O. Maimon and L. Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer Publishing Company, 2nd edition, 2010. ISBN 0387098224, 9780387098227.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1st edition, 1997. ISBN 0070428077, 9780070428072.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X, 9780262018258.

H. Nguyen. On exploring soft discretization of continuous attributes. In S. Pal, L. Polkowski, and A. Skowron, editors, *Rough-Neural Computing*, Cognitive Technologies, pages 333–350. Springer Berlin Heidelberg, 2004. ISBN 978-3-642-62328-8. doi: 10.1007/978-3-642-18859-6_13. URL http://dx.doi.org/10.1007/978-3-642-18859-6_13.

M. A. Patestas and L. P. Gartner. *A Textbook of Neuroanatomy*. Blackwell Science Ltd, 2006. ISBN 1-4051-0340-X.

H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.159.

F. Pereira, T. Mitchell, and M. Botvinick. Machine learning classifiers and fmri: A tutorial overview. *NeuroImage*, 45(1, Supplement 1):199 – 209, 2009. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2008.11.007. URL http://www.sciencedirect.com/science/article/pii/S1053811908012263.

J. R. Quinlan. Induction of decision trees. *Mach. Learn*, pages 81–106, 1986.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

M. Seyedhosseini, R. Kumar, E. Jurrus, R. Giuly, M. Ellisman, H. Pfister, and T. Tasdizen. Detection of neuron membranes in electron microscopy images using multi-scale context and radon-like features. In G. Fichtinger, A. Martel, and T. Peters, editors, *Medical Image Computing and Computer-Assisted Intervention, MICCAI 2011*, volume 6891 of *Lecture Notes in Computer Science*, pages 670–677. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-23622-8. doi: 10.1007/978-3-642-23623-5_84. URL http://dx.doi.org/10.1007/978-3-642-23623-5_84.

M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, 2nd edition, 2007. ISBN 049508252X.

M. A. O. Vasilescu and D. Terzopoulos. Tensortextures: multilinear image-based rendering. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 336–342, New York, NY, USA, 2004. ACM. doi: 10.1145/1186562.1015725. URL http://doi.acm.org/10.1145/1186562.1015725.

A. Vazquez-Reina, M. Gelbart, D. Huang, J. Lichtman, E. Miller, and H. Pfister. Segmentation fusion for connectomics. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 177–184, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-1101-5. doi: 10.1109/ICCV.2011.6126240. URL http://dx.doi.org/10.1109/ICCV.2011.6126240.

J. Weickert and H. Hagen. *Visualization and Processing of Tensor Fields*. Mathematics and Visualization. Springer, 2006. ISBN 9783540250326.

D. Williams and C. Carter. *Transmission Electron Microscopy: A Textbook for Materials Science*. Number Bd. 1 in Cambridge library collection. Springer London, Limited, 2009. ISBN 9780387765006.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 0120884070.

J. S. Yi, Y. Kang, J. Stasko, and J. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, Nov. 2007. ISSN 1077-2626. doi: 10.1109/TVCG.2007.70515. URL http://dx.doi.org/10.1109/TVCG.2007.70515.

L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 856–863. AAAI Press, 2003.

L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *J. Mach. Learn. Res.*, 5:1205–1224, Dec. 2004. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1005332.1044700.