ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG
INSTITUT FÜR INFORMATIK

Lehrstuhl für Mustererkennung und Bildverarbeitung
Prof. Dr. Hans Burkhardt

# Segmentation and Classification of Cell Nuclei in Tissue Slices using Voxel-Wise Gray-Scale Invariants

## Diplomarbeit

Janis Fehr

Mai - November 2004

# Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit von mir selbständig und nur unter Verwendung der aufgeführten Hilfsmittel erstellt wurde.

Freiburg i.Br. den,

## Aufgabenstellung für die Dimplomarbeit von Herrn Janis Fehr

## Lokalisierung und Erkennung von verschiedenen Zellkernen in Gewebeschnitten

Im Rahmen dieser Diplomarbeit soll untersucht werden, ob sich verschiedene Zellen in einem Gewebeschnitt anhand der 3D Struktur ihrer Zellkerne automatisch lokalisieren und identifizieren lassen. Vor allem sollen YoPro-gefärbte Endothelzellen, Epithelzellen, Erythrozyten und Perizyten in konfokalen 3D Mehrkanal-Aufnahmen segmentiert und erkannt werden.

Dabei sollen folgende Untersuchungen angestellt werden:

- Literaturrecherche zu dem Thema
- Verfahren/Tools zum einfachen Erstellen der benötigten Trainingsdatenbanken: Halbautomatische Segmentierung.
- Versuch der Klassifikation (z.B. mit Support Vektor Maschinen) jedes einzelnen Voxels in eine oder mehrere der oben genannten Klassen. Dafür sollen verschiedene Features (z.B. grauwertbasierte Invarianten) die aus der Umgebung des Voxels extrahiert wurden, zu Hilfe genommen werden.
- Intelligente Gruppierung (z.B. region-growing Ansätze) der klassifizierten Voxel, so dass die genaue Form des jeweiligen Objektes bestimmt werden kann.
- Beschleunigung der Klassifikation der einzelnen Voxel z.B. mit Hash-Tabellen, o.ä.

Literatur:

[1] P.H. Cossmann, P.S. Eggl, H. Kurz: Three-dimensional analysis of DNA replication foci: a comparative study on species and cell type in situ. Histochem Cell Bio (2000) 113:195-205

| | |
|---|---|
| *Referent:* | Prof. Dr.-Ing. H. Burkhardt |
| *Betreuer:* | Dipl.-Phys. O. Ronneberger |
| *Ausgabedatum:* | 22.5.2004 |
| *Abgabedatum:* | 22.11.2004 |
| *Bearbeitungszeit:* | 6 Monate (ganztags) |

# Abstract

This thesis introduces and discusses an algorithm for segmentation and classification of cells from 3D tissue probes. The 3D volume data of fluorescent marked cell nuclei has been recorded by a laser scanning microscope. Gray scale features for every single voxel, invariant towards rotation group transformation of its neighborhood, were extracted from the original data by integrating over the 3D Euclidean transformation group with non-linear kernels.
In an interactive process, support-vector machine models were trained for each cell type. With this reference database at hand, segmentation and classification can be achieved in one step, simply by classifying each voxel and performing a connected component labelling automatically without further human interaction. This general approach easily allows adoption of other cell types or tissue structures just by adding new training samples and re-training the model.

Experiments with datasets from chicken embryo chorioallantioic membrane and human intestinal wall showed very encouraging results.



# Zusammenfassung

In dieser Diplomarbeit wird ein Algorithmus zur Segmentierung und Klassifikation von Zellen in 3D Gewebsschnitten vorgestellt und diskutiert. Die 3D Aufnahmen von mit fluoreszierenden Fabstoffen markierten Zellkernen wurden mit einem Laser Scanning Mikroskop erstellt. Durch Integration über die Euklidische Transformationsgruppe mit nichtlinearen Abbildungen werden für jeden Voxel grauwertbasierte Merkmale extrahiert, welche invariant gegenüber Rotation der Voxel-Umgebung sind.
In einem interaktiven Prozess trainiert man dann ein Modell für alle Zelltypen mit einer Support-Vektor Maschine. Gegen dieses Modell lassen sich die einzelnen Voxel eines Testvolumens klassifizieren und mittels Connected-Component-Labeling die Zellen segmentieren. Dieser allgemeine Ansatz lässt sich durch erneutes Training leicht auf weitere Zelltypen erweitern.

Experimente mit Gewebsschnitten der Chorioallantois-Membran des Hühnereis sowie von der menschlichen Darmwand zeigen gute Ergebnisse.

# Contents


**0 Introduction** **6**

**1 Biomedical Background** **8**
1.1 Angiogenesis . . . 8
1.1.1 Motivation . . . 8
1.1.2 The Chicken Embryo Model . . . 9
1.1.3 Datasets . . . 10
1.1.4 Blood vessels . . . 10
1.1.5 Cell types . . . 11
1.2 Colon Wall Histology . . . 17

**2 Data Retrieval** **19**
2.1 Fluorescent Microscopy . . . 19
2.2 Confocal Laser Scanning Microscopy . . . 20
2.3 Data formats . . . 21
2.3.1 NetCDF . . . 21
2.3.2 Tools . . . 21

**3 Algorithmic Background** **22**
3.1 Segmentation Methods . . . 22
3.1.1 Thresholding . . . 23
3.1.2 Edge Detection . . . 24
3.1.3 Region Growing . . . 25
3.2 Invariant Theory . . . 27
3.2.1 Equivalence Classes . . . 27
3.2.2 Transformation Groups . . . 27
3.2.3 Invariant Features . . . 28
3.2.4 Non-Linear Mappings . . . 29
3.2.5 Calculating Invariants . . . 29
3.2.6 Group Average Invariants . . . 30
3.3 Classification . . . 31
3.3.1 Support-Vector Machines . . . 31
3.3.2 Model Selection . . . 36

**4 Gray-Scale Invariants** **38**
4.1 Two-Point Gray-Scale Invariants . . . 38
4.1.1 Invariants for 2D Signals . . . 38
4.1.2 Increasing Separability . . . 42
4.1.3 Multi-Channel Invariants . . . 42
4.1.4 Invariants for 3D Volume Data . . . 42
4.2 Three-Point Gray-Scale Invariants . . . 43

**5 Segmentation and Classification using Voxel-Wise Gray-Scale Invariants** **45**
5.1 Pixel/Voxel-Wise Gray-Scale Features . . . . . 46
5.1.1 Increasing Local Support . . . . . 46
5.2 Training and Classification . . . . . 47
5.2.1 Interclass Distances . . . . . 47
5.3 Runtime Issues . . . . . 48
5.3.1 Extracting Feature-Vectors . . . . . 48
5.3.2 Classifying . . . . . 49
5.4 Connected Component Labeling . . . . . 51
5.5 Exponential Kernel Functions . . . . . 52
5.6 Pseudo-Three-Point Invariants . . . . . 53
5.6.1 Three-Point Invariants . . . . . 54
5.6.2 Fourier Descriptors . . . . . 54
5.6.3 Implementing Pseudo Three-Point Invariants . . . . . 55

**6 Software Implementation** **58**
6.1 Libraries . . . . . 58
6.2 Software Description . . . . . 59
6.2.1 Tools . . . . . 60
6.2.2 Views . . . . . 60
6.3 Extracting Features . . . . . 61
6.4 Training . . . . . 62
6.5 Classification . . . . . 63
6.6 Segmentation . . . . . 64

**7 Experiments** **65**
7.1 Data Sets . . . . . 65
7.1.1 Angiogenesis . . . . . 65
7.1.2 Colon Tissue . . . . . 66
7.2 Measuring Experimental Results . . . . . 67

**8 Conclusion** **74**
8.1 Outlook . . . . . 74

# Chapter 0

# Introduction

In biological and medical research as well as in clinical praxis, the localization and classification cells is an everyday business. A vast number of research techniques and treatment methods require detailed information on the amount, type, localization and state of cells in a given probe of tissue or dilution.
The task of locating, classifying and analyzing cells is not simple, very time consuming and in most cases a human expert is needed. The demand for automation is continuously growing with the large number of applications in biotechnology and medical research. But so far this problem is not solved. Although there are various methods around, which perform quite well for simple tasks like counting or the segmentation of cells in dilution, most problems are still subject to basic research.

When it comes down to non-destructive analysis of tissue probes, image processing combined with pattern recognition is the method of choice. Much like the human expert, algorithms try to find and classify cells by their optical appearance. Since computer and human make use of the same source of information, in most cases translucent, fluorescent or laser scanning microscopes, automation can be achieved with little additional equipment.
Concerning the algorithms introduced in literature, most suffer from the fact that they have been developed for one special purpose only and cannot be easily generalized to other cell types or tissues. Besides that, segmentation is often an unsolved problem as well and many algorithms require manual preprocessing.

This work introduces a new general purpose algorithm using gray scale invariants for both, segmentation and classification of cells in 2D and 3D probes, together with some first, quite promising experimental results.
Motivated by work of [Schulz-Mirbach95] and [Burkhardt2001], gray scale invariants were very successfully applied for pollen recognition by [Ronneberger2002]. Due to the similar nature of the underling data, the algorithms developed for pollen recognition are suspected to perform as well for cell recognition. And indeed, the work of [Baumgartner2003] showed promising classification results for applying gray scale invariants on single, pre-segmented cells.

One major problem of cell classification in tissue probes is segmentation. In order to achieve good classification results, supervised-learning classifiers rely on proper segmented training samples and classification probes. Proper segmentation is hard to realize without knowledge about the object to segment. But the use of a-priori knowledge or manual segmentation is not suited for a full automatic general purpose

approach.
For this reason this work describes a self learning segmentation algorithm by use of gray scale invariants, which is capable of performing segmentation and classification in one step. Gray scale invariant features are extracted from the surrounding neighborhood of each pixel/voxel. In an interactive procedure a support-vector machine model is trained. Once this model has been obtained for the requested types of cells, segmentation and classification can be performed automatically without any further human interaction.

The first chapter gives a gentle introduction to a sample of medical and biological problems, which where used as references in this work and which delivered the data for the shown experiments. A short motivation of research interests is followed by a description of how the probes have been prepared and the data digitalized. In detail, all appearing cell types are described in form and relevant function.
The second chapter contains an overview of microscope techniques in the context of this work. The focus here lies on laser scanning microscopy.
The basics of segmentation, support-vector machines and gray-scale invariants are explained in chapter three as far as necessary for the comprehension of this work.
Chapter four then describes in detail the theory of the chosen feature extraction and proves invariance regarding rotation and translation. Further optimization and implementation techniques are discussed. The usage of these invariants for segmentation is described in chapter five.
The software implementation of previously introduced algorithms is presented in chapter six. Finally, in chapter seven, experimental results are presented followed by a summary and outlook in chapter eight.

# Chapter 1

# Biomedical Background

This chapter gives a short and gentle introduction to two current biomedical research areas, Angiogenesis and colon cancer research. Both are examples of methods which are based on localization and identification of cell types and states in 3D tissue probes. Datasets from these sources were used to conduct the experiments shown in chapter seven.

## 1.1 Angiogenesis

**Definition:** "Angiogenesis is the physiological or pathological process of sprouting new blood vessels from pre-existing vessels. In the physiological case, this is one process of growth and development as well as wound healing." [Schmidt-Thews] However non-sprouting angiogenesis by means of transluminal pillar formation has been also desribed [Kurz2003].

### 1.1.1 Motivation

Blood vessels form the largest network in human body and carry a vast variety of functions, from oxygen supply over nutrient flux to immune defense. A dysregulation in the formation of new vessels contributes to numerous malignant, ischemic, infectious and immune disorders. An overview is given in [Carmeliet2003].
One focus of recent Angiogenesis research is motivated by cancer therapy. An expanding tumor demands continued blood supply to keep growing. By releasing angiogenic growth factors, like VEGF (Vascular endothelial growth factor) for instance, tumors manipulate the physiological balance of vessel growth indicators. Comprehension of Angiogenesis mechanisms could lead to a pharmacological cancer treatment by preventing the sprout of new vessels, thus cutting off the tumors blood supply.

Much of the basic research done in this field concentrates either on the macroscopic evolution of vessel growth, like [Arlt2003], or the molecular mechanisms involved. The link between molecular and the entire network are cells building the vessel walls. Their size, shape and function is at the focus of recent research. [Kurz2003] Both sprouting and non-sprouting mechanisms involve movement and cleavage of cells within the structure of the vessel. An automatic detection and classification of cell types, states and orientation could provide a very useful tool for this task.

### 1.1.2 The Chicken Embryo Model

One well-established models for the study of vessel growth is the chorioallantioic membrane (CAM) of avians, especially the one of chickens. The rapid avian embryonic development and the ease of access to the membrane, which lies right under the shell, are some advantages in comparison to animal experiments.

#### Avian Enbryonic Development

The chicken embryo takes just 21 days from laying to hatching when incubated at 38,5 degree Celsius. The the alantiois (see fig. 1.1) and chorio membranes adhere to the for the Angiogenesis research interesting chorioallantioic embrane some time between the 4th and 6th day. A complete description of avian embryonic development can be found in [Smith97].

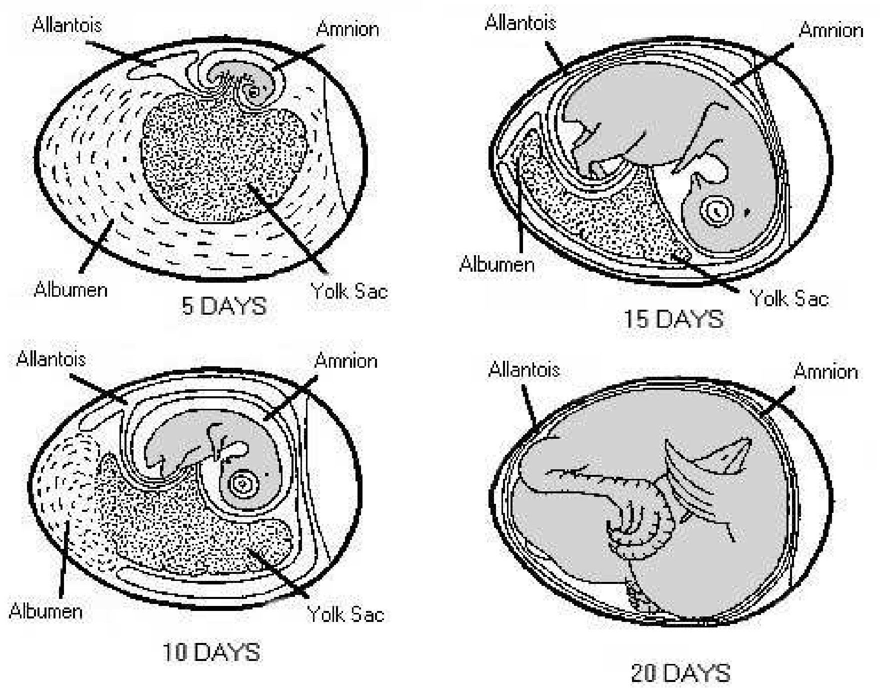



Figure 1.1: Development of a chick embryo. Taken from [Smith97]

#### Function of the Chorioallantioic Membrane

As shown in fig. (1.1), the membrane is embedding the embryo with the time and docks to the shell. This way, the chorioallantioic membrane becomes the embryo's external respiratory and excretory organ absorbing oxygen through the porous shell. Respiration requires a high rate of blood circulation, thus massive vessel growth starts within the chorioallantioic membrane. A window is cut in the shell around day two allowing an observation of the growing vessels.

### 1.1.3 Datasets

The 3D volume datasets were recorded from chorioallantioic membrane probes by use of a multi channel laser scanning microscope (see next chaper for details). The preparation of the membranes followed the description in [Kurz2003]. For the fluorescent imaging, probes were treated with BrdU or YoPro markers.

#### BrdU

This chemical is a base analog of thymidine, with the thymine substituted by bromouracil. By substituting for thymidine during DNA replication, BrdU is incorporated in newly synthesized DNA. Thus, BrdU antibodies linked to fluorescent molecules are used to visualize active DNA in living cells.

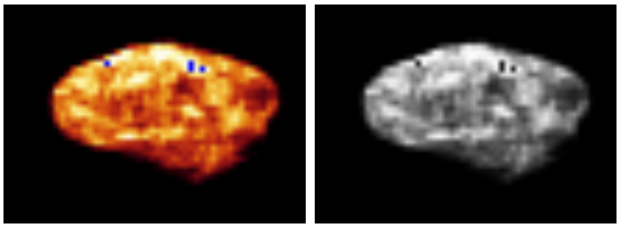


Figure 1.2: A LSM recorded nucleus marked with BrdU. Left in pseudo coloration, right in gray-scale.

#### YoPro

Binds to the DNA double helix. This works with inactive or dead cells, as well as with active ones. YoPro is usually used to visualize the entire nuclei of cells.

As a result of these fluorescent imaging techniques, the segmentation and classification

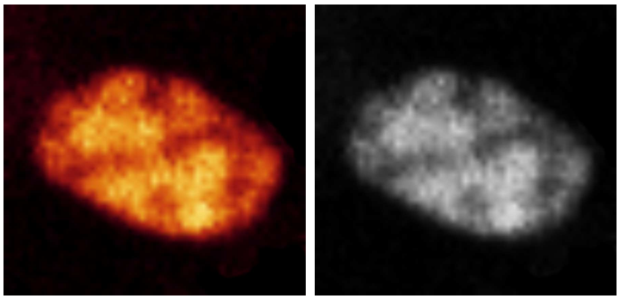


Figure 1.3: A LSM recorded erythrocyte nucleus marked with YoPro. Left in pseudo coloration, right in gray scale.

of cells in this context, can only be achieved through data retrieved from the nucleus shape, not the entire cell itself.

### 1.1.4 Blood vessels

In humans and also in the avian model blood vessels appear at all different kinds of scales. From the several centimeter thick arch of the aorta, which consists of hundreds of layers of endothelial, muscle and other cells, down to capillaries with their cross-sections are made up just by one single endothelial cell. In the given volume datasets from the chorioallantioic membrane probes, capillaries make up the vast majority of vessels.

Fig. 1.4 shows a typical cross-section of a capillary and surrounding tissue in a LSM scan with YoPro treated cells. The capillary is bounded by the a single endothelial cell.

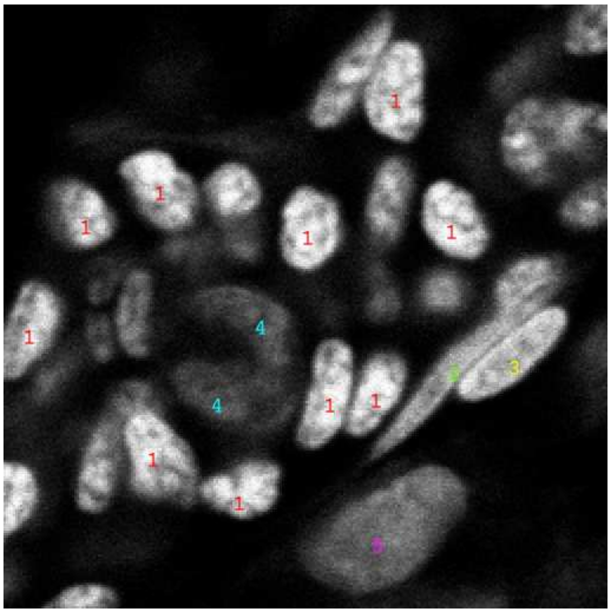


Figure 1.4: 1. erythrocyte (**Ery**), 2. endothelial cell (**EC**), 3. pericyte (**PC**), 4. fibroblast (**FB**), 5. macrophage (**M$\Phi$**).

The erythrocytes are located within the vessel and indicate blood flow, macrophages and pericytes lie outside. The fibroblasts are actually inside the capillary, but separated from blood by an endothelial plasmalemm. So in this case, the vessel is not just a "simple tube", but rather a more complex "double tubed" structure.

### 1.1.5 Cell types

As in fig. 1.4 shown, cells in the chorioallantioic membrane tissue probes basically can be referred to one of the five types: erythrocyte, endothelial, pericyte, fibroblast or macrophage.
The characteristics of these types and their appearance in 3D LSM datasets is discussed in the next few sections. All cells shown, were automatically segmented by the algorithm described in chapter five. The cell volume data is from the same source as in fig. 1.4.

The 3D surfaces were reconstructed with a polygon normal algorithm at a specific grey value threshold.

### Erythrocytes

Erythrocytes, also known as red blood cells, are the most common type of blood cell and are the vertebrate body's principal means of delivering oxygen to body tissues via the blood. The oxygen is carried by hemoglobin, a complex molecule containing heme groups whose iron molecules temporarily link to oxygen molecules in the lungs and release them throughout the body. The hemoglobin causes the red coloration of blood.
The diameter of a typical human erythrocyte is about 6-8 $\mu$m. Their very elastic structure enables red blood cells to pass capillary vessels of even smaller diameter. Human blood types are due to variations in surface glycoproteins of erythrocytes.
Unlike human erythrocytes, avian red blood cells are still containing a nucleus, when released from bone marrow to the circulatory system. This is a quite important fact, because only nuclei, but not the entire cell can be detected by the imaging techniques used in the context of this work.
Erythrocytes are not directly significant for the process of Angiogenesis, but for the observer the localization of red blood cells gives a good impression on the dimension and structure of vessels and their internal blood flow.

In LSM imaging, YoPro marked erythrocyte nuclei appear as dense, but fairly irreg-

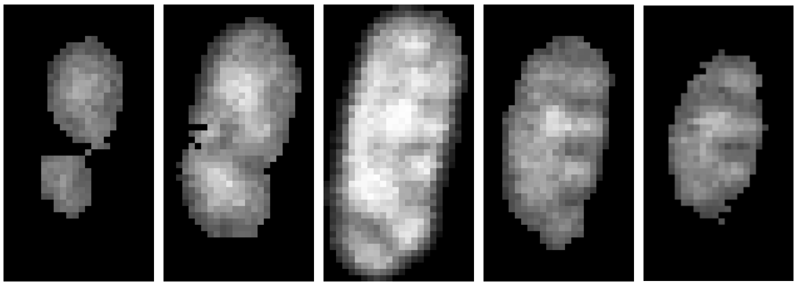


Figure 1.5: Series of LSM slices of a YoPro marked erythrocyte.

ularly colored structures of round or oval shape. Only few errors can be found and contrast to the background is good.

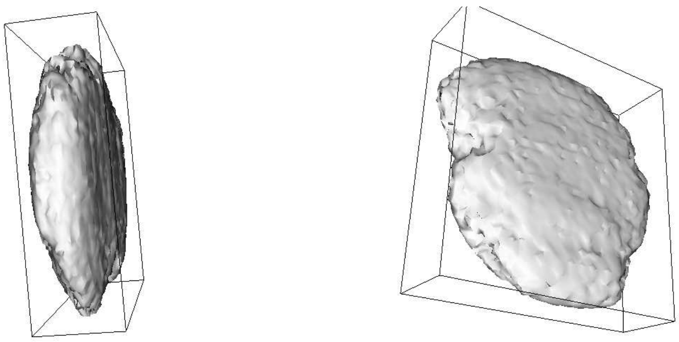


Figure 1.6: 3D surface reconstruction from LSM slices of a YoPro marked erythrocyte nucleus.

### Endothelial cells

Endothelial cells line the interior surface of blood vessels forming an interface between circulating blood in the lumen and the rest of the vessel wall. This is true for the entire circulatory system, from the heart to the smallest capillary. As seen in fig. 1.4, in small blood vessels and capillaries, single endothelial cells are often the only boundary present.

Endothelial cells control the passage of materials and white blood cells into and out of the bloodstream. They are involved in many aspects of vascular biology: the control of blood flow, blood clotting and the formation of new vessels - thus Angiogenesis.

The localization of endothelial cells, while Angiogenesis processes, could provide new insights on a key player of groth and remodeling.

In LSM imaging, YoPro marked endothelial cell nuclei appear quite dense and reg-

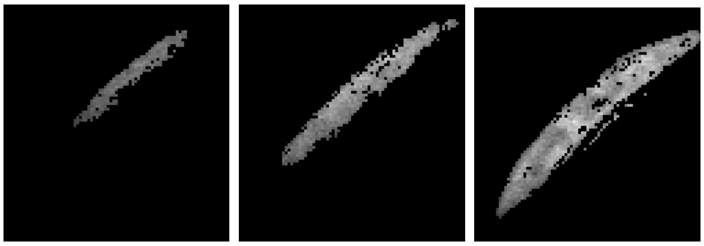


Figure 1.7: Series of LSM slices of a YoPro marked endothelial cell.

ularly colored. The outer bounds are not always detected as clear as that is the case for erythrocytes. Due to the close contact to pericytes, errors are not unusual. The overall shape appears flat, long and bent towards the vessel.

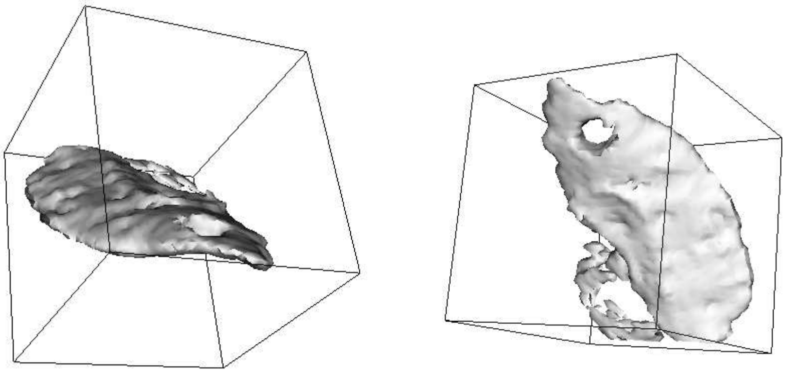


Figure 1.8: 3D surface reconstruction from LSM slices of a YoPro marked endothelial cell nucleus.

### Pericytes

By definition, pericytes only appear in combination with endothelial cells to which they are directly attached. This way, pericytes occasionally form a very tight second layer of the vessel wall, found in brain vessels e.g. In other cases, like fig. 1.4, pericytes appear rather sparsely distributed.
The main function of pericytes seems to be the manipulation of endothelial behavior, for instance concerning the passage of material from and to the blood stream. Thieir role in Angiogenesis, especially in the context of growth factors, has been discussed in [Hagedorn2004].
In LSM imaging, YoPro marked pericyte nuclei appear dense and regularly colored.

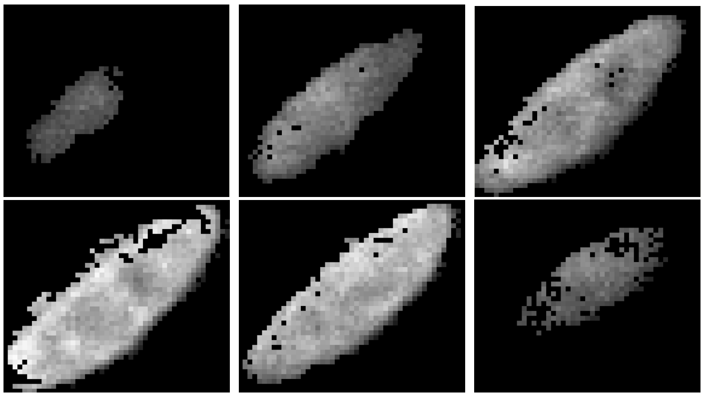


Figure 1.9: Series of LSM slices of a YoPro marked pericyte nucleus.

Due to the close contact to endothelial cells, errors are not unusual. The overall shape is the one of a flat disc.

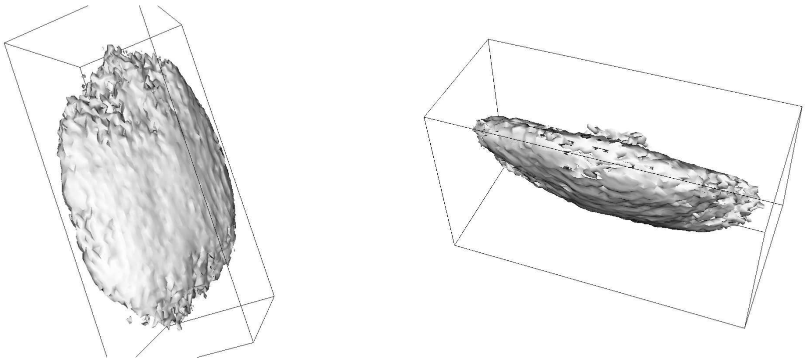


Figure 1.10: 3D surface reconstruction from LSM slices of a YoPro marked pericyte nucleus.

### Fibroblasts

Fibroblasts are active connective tissue cells. In the context of this work, the class of cells labeled as fibroblasts might also contain fibrocytes. Both are closely related and basically differ in the state of cell activity. Fibroblasts are responsible for connective tissue growth and can be found in the surrounding tissue of vessels, as well as structuring elements inside the vessels.
They also appear to be precursors of pericytes and produce extracellular material for the vessel wall. In LSM imaging, fibroblasts are not marked very well by YoPro. They

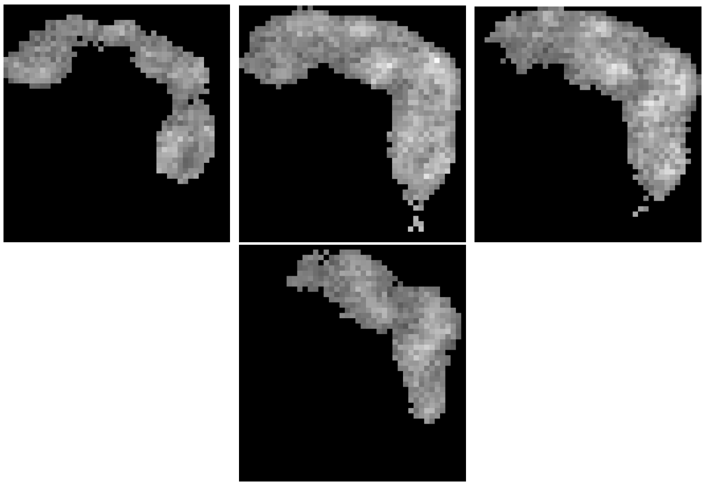


Figure 1.11: Series of LSM slices of a YoPro marked fibroblast nucleus.

appear as noisy clusters, which are hard to determine from the background. Thus, often only parts of the nuclei are imaged.

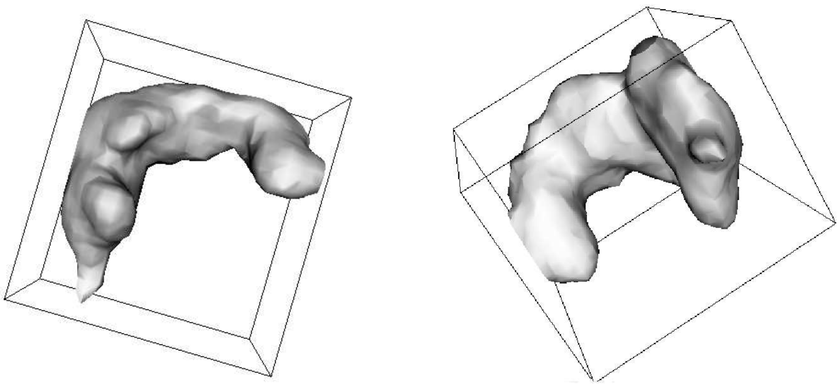


Figure 1.12: 3D surface reconstruction from LSM slices of a YoPro marked fibroblast nucleus.

### Macrophages

Macrophages are part of the immune system and evolve from white blood cells. They are capable of freely moving around tissue. The primary function of a macrophage is to clear foreign substances. They work by eating whatever they do not recognize as healthy tissue, including pathogens and dead cells. In fact, the etymology of macrophage is Greek "big eater".
Macrophages are also suspected to influence Angiogenesis process by secreting growth-factors and removing extracellular material. In LSM imaging, YoPro marked macrophage

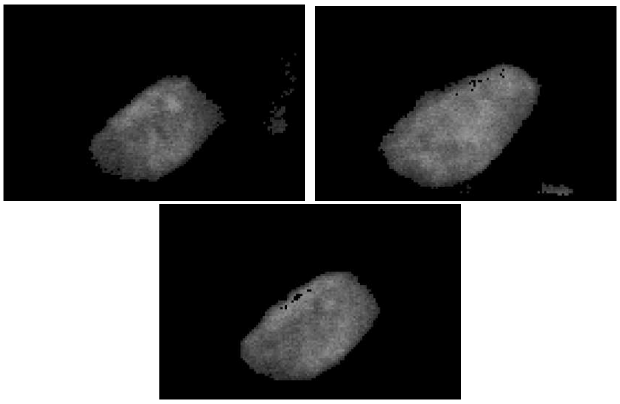


Figure 1.13: Series of LSM slices of a YoPro marked macrophage nucleus.

nuclei appear as fine regularly colored structures, which have a quite good contrast to the background noise. Macrophage nuclei have a large, somewhat round shape.

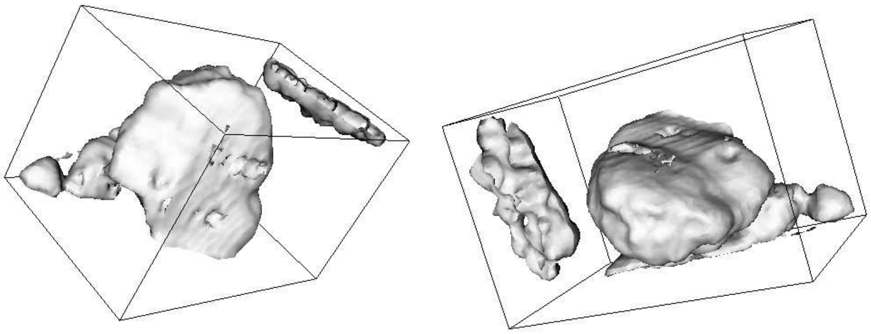


Figure 1.14: 3D surface reconstruction from LSM slices of a YoPro marked macrophage nucleus.

## 1.2 Colon Wall Histology

In anatomy of the digestive system, the colon is the part of the intestine from the cecum to the rectum. Its primary purpose is to extract water from feces.
Colon cancer is one of the most common and fatal types of cancer found in humans. Thus, much effort is put in histologic, cytologic and molecular cell biological research of the colon wall physiology.

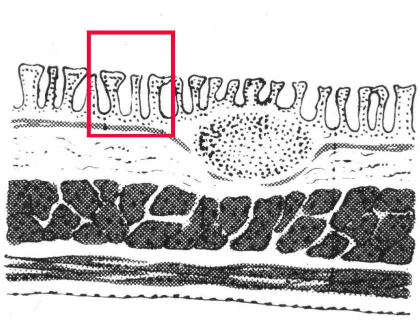


Figure 1.15: Schematic view of the human colon wall [Leonhardt]. The selected area indicates the region of the probes

### Motivation

Research on the colon adenoid's epithelium tissue, like [Cremer2003], often requires the segmentation of single epithel cells from 3D volume datasets. This is still done manually. Fig. 1.16 and 1.17 show the dense epithel tissue and give a first impression

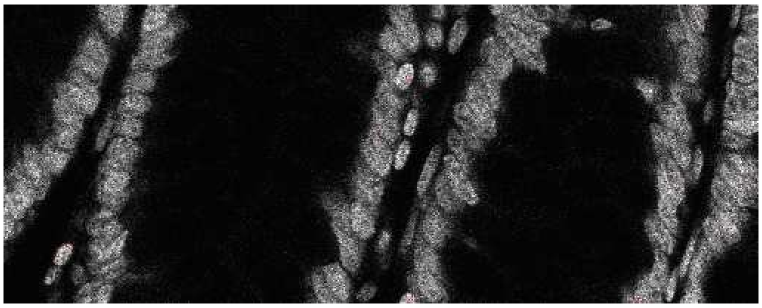


Figure 1.16: LSM slice of the colon villi, cell nuclei were marked with TO-PRO3.

of the difficulty for segmentation algorithms to isolate a single cell of the villi wall.

### Cell Types

The only cell type relevant in this context is the epithel cell of the adenoid's epithelium tissue bounding the colon inner wall. ToPro3 marked epithel cell nuclei appear dense and regular structured in LSM imaging with few errors.

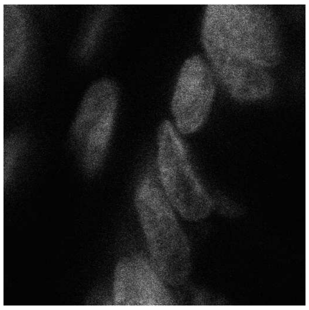


Figure 1.17: LSM close up on colon villi tissue.

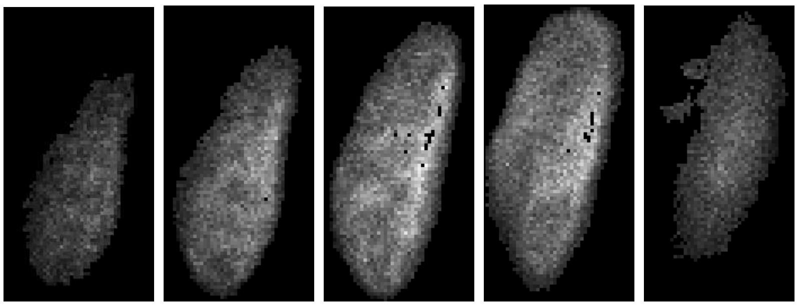


Figure 1.18: Series of LSM slices of an epithel nucleus marked with ToPro3.

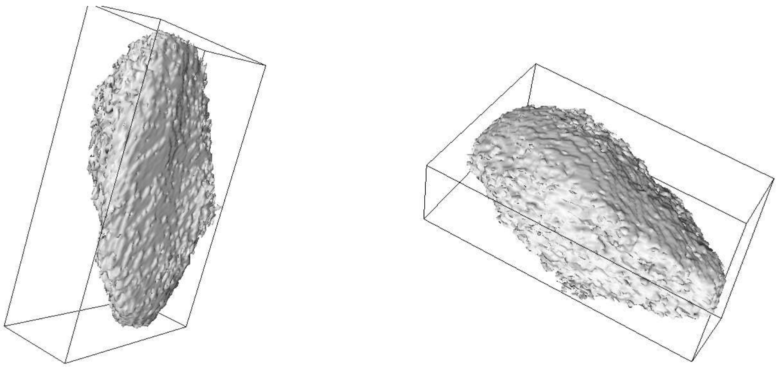


Figure 1.19: 3D surface reconstruction from LSM slices of a ToPro3 marked epithel nucleus.

# Chapter 2

# Data Retrieval

The probes described in the previous chapter have to be somehow magnified, imaged and digitized. This chapter gives a brief introduction to fluorescent microscopy in general, presenting some issues, which will be referred to later on and in a bit more detail, confocal laser scanning microscopy. Section three introduces the NetCDF file format and other aspects of volume data handling.

## 2.1 Fluorescent Microscopy

### Definition

Fluorescence is a luminescence phenomenon in which a molecule absorbs a high-energy photon and re-emits another in a lower state of energy (longer wave-length).

Fluorescent microscopy takes advantage of the fluorescence effect in order to image details of probes, which hardly could be retrieved by common light microscopy. Most light microscopes can be used for fluorescent microscopy, all it takes is a special light source and some filters.
Since most fluorescent substances are mostly excited by rather short-wave emissions, the light source usually has a wave-length somewhere in the blue to ultra violet spectrum, but other frequencies are also possible.
The light is projected by a dichroic mirror from the source on to (or through) the probe. The excited fluorescent molecules then emit photons themselves, which are collected by the objective lens. The dichroic mirror has the property of reflecting only certain frequencies and letting others pass. Thus, the light emitted by the fluorescent, with its shorter wavelength, can pass the mirror and reach the CCD camera or the ocular.

The advantage of fluorescent microscopy is, that only previously marked structures are imaged and reflections of the exciting light from unmarked structures are filtered by the dichroic splitter [Kapitza94].
One major drawback of conventional fluorescent microscopy is, that it is a quite sensitive technique. In thicker probes underlying and overlying structures outshine those in focus and cause huge background noise eventually covering much of the requested detail.

To avoid this problem, confocal laser scanning microscopy is used instead.

## 2.2 Confocal Laser Scanning Microscopy

The use of lasers as light source for fluorescent microscopy has several advantages compared to other kinds of exciting light. First, lasers beam only in one predetermined direction, can be focused to a single point with very high resolution, and have a fixed wavelength [ShSh97]. Most laser scanning microscopes (LSM) have more than one laser with different wavelengths. So different stainings can be applied simultaneously and recorded in different channels.

Just like ordinary light, the laser beam is reflected by the dichroic mirror and projected

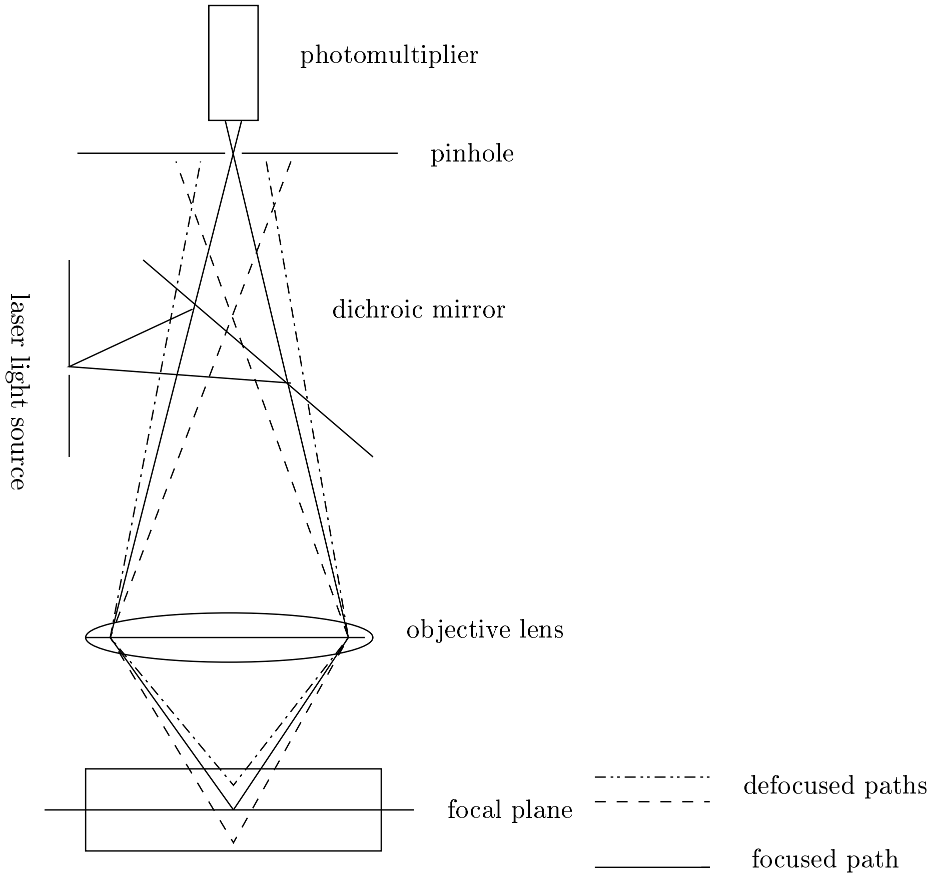



Figure 2.1: Schematic view of a confocal laser scanning microscope [ShSh97]

through the objective onto the probe. Unlike in conventional microscopy, the laser only exposes one point in the x-y-coordinate of the specimen at a time. By moving the beam over the probe line by line, the image is scanned and then reassembled.
The actual "trick" of laser scanning microscopes is, that it is also possible to have a resolution in direction of the z-axis. This is achieved by use of the **pinhole**.

The light emitted from the probe is collected by the objective and passes the dichroic mirror. Before reaching the photomultiplier, which is the actual imaging medium, beams have to pass the pinhole. Fig. 2.1 shows, how only focused paths and such close to them are able to pass the pinhole. Other defocused paths are hardly mapped.

By moving the probe in the z-axis, a series of focal images can be recorded slice by slice, where the pinhole size determines the thickness.
Resolution in direction of the z-axis is by far not as high as this is possible for the x-y-planes. This usually results in non-cubic voxels and a strong blur along z-direction on the point spread function (PSF).

## 2.3 Data formats

The confocal laser scanning of probes generates a series of single gray scale images, one for each x-y-plane and channel. The series for one channel is referred to as "stack". These stacks are then converted to real 3D volume data in form of an 3D array. For the storage of multichannel data, along with additional information like size, type of markers used, pinhole etc., the NetCDF format is used.

### 2.3.1 NetCDF

The network common data form (NetCDF) [Rew1997] is an open standard for self describing portable array based data. NetCDF provides fast access to arrays of many basic data types and different dimensionality. The `leica2netcdf` tool converts multichannel tiff stacks to the NetCDF format.
All data from 3D volumes to SVM models is stored in NetCDF format.

### 2.3.2 Tools

A variety of tools provide an easy handling of NetCDF files:

#### tiff2netcdf

`tiff2netcdf` converts a series of slices delivered from the microscope as tiff images into a NetCDF file. `tiff2netcdf` supports multi channel images.

#### smoothShrink

`smoothShrink` scales the size of data in NetCDF files with a smooth interpolation. Each axis can be scaled separately, thus the resolution problem of the z-axis can be reduced.

#### polvi

`polvi` is a NetCDF viewer which displays all channels slice by slice in a xy, yz or xz orientation.

#### nc2vtkvis

`nc2vtkvis` reconstructs a 3D surface illustration from NetCDF data.

# Chapter 3

# Algorithmic Background

This chapter sets the algorithmic basis for the detailed descriptions following in the next chapter. First the use of established segmentation algorithms is discussed, showing some experimental results and stating arising problems. Secondly, an introduction to general invariant theory is given. The third section gives an introduction to support-vector machines in general and some relevant topics are dealt with in more detail. Finally, gray-scale invariants are presented in section four, stating the ground for the actual segmentation and classification algorithm.

## 3.1 Segmentation Methods

The conservative approach to object detection in images or volume data involves two steps: segmentation and classification. First, regions of interest are partitioned from the background and then classified. Whereas the first step is crucial - even the best classifier will fail when confronted with poorly segmented, cut off or clustered samples. There are three major classes of segmentation methods: thresholding, edge detection and region growing algorithms. Each is introduced briefly and experimental results are discussed. For further details consult [Jähne2002] or [GonzalesWoods].
Discussion in the following section is reduced to 2D examples, but all algorithms can be easily extended to 3D data sets.

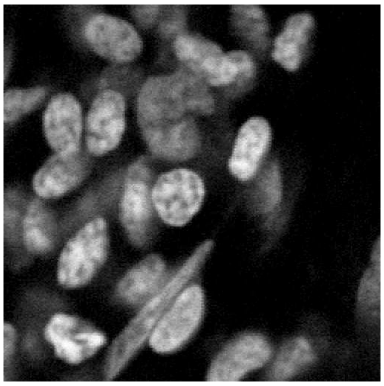


Figure 3.1: 2D slice of a YoPro marked probe used for the illustrations in this chapter

### 3.1.1 Thresholding

The probably most intuitive and simple segmentation algorithm is image thresholding. For a given gray-scale image a threshold $T$ is defined, such that all points $p(x, y)$ which hold $p(x, y) \geq T$ are considered to be foreground objects.

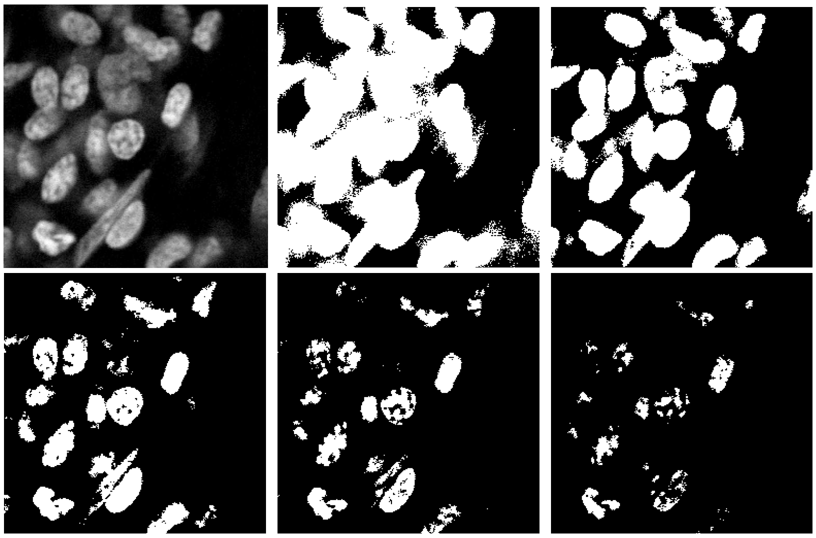


Figure 3.2: Threshold segmentation of YoPro marked probe. From left to right: original, threshold set at 10%, 30%, 50%, 60% and 70% of max gray value.

#### Discussion

Even though thresholding can turn out quite good results, this is certainly not true for this application. A quick look at the image histogram shows, that there is no reasonable value for $T$ to be set at. Although it is possible to isolate erythrocytes, endothelial cells and pericytes from background, it is not possible to split clusters of neighboring cells or to distinguish fibroblasts or macrophages from the background.

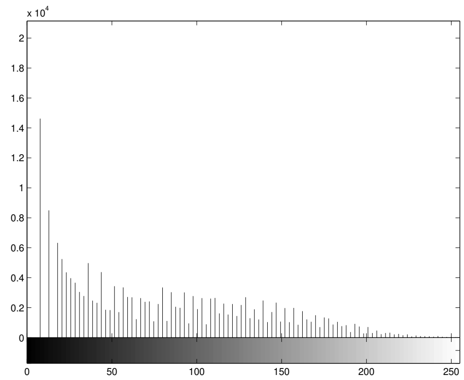



Figure 3.3: Histogram of a YoPro marked probe

### 3.1.2 Edge Detection

Another class of methods relies on edge detection algorithms. Segmentation is achieved by retrieving a preferably closed contour from foreground objects.
Edges are detected via the image gradient, using the first derivative of the image function $f(x, y)$ like the **Sobel** method or the second derivative like the **Laplacian of Gaussian**.

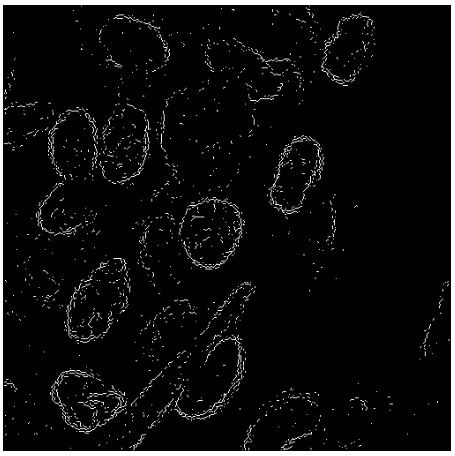


Figure 3.4: Result of Sobel edge detection

Figure 3.5: Result of Laplacian of Gaussian edge detection

#### Discussion

As figures 3.4 and 3.5 show, for most cases edge detection fails to detect anything like a closed contour. Hence the cells have strong internal structure and weak borders, this is not a very surprising result. Sobel works better than the Laplacian of Gaussian filter, but still only few erythrocytes with strong contrast are detected.

### 3.1.3 Region Growing

Region growing is a procedure that groups pixels into larger regions based on predefined criteria. The most popular method of this class is the **watershed algorithm**.
The watershed concept is based on the idea to interpret a 2D gray-scale image as 3D topographic landscape. Starting from local minima, this landscape is "flooded". These minima are called *catchment basins*. With the raising water, regions grow around the starting point. The crest lines of the topographic surface which separate two or more regions are called *watersheds*.

The watershed algorithm suffers from vast over-segmentation on strong structured

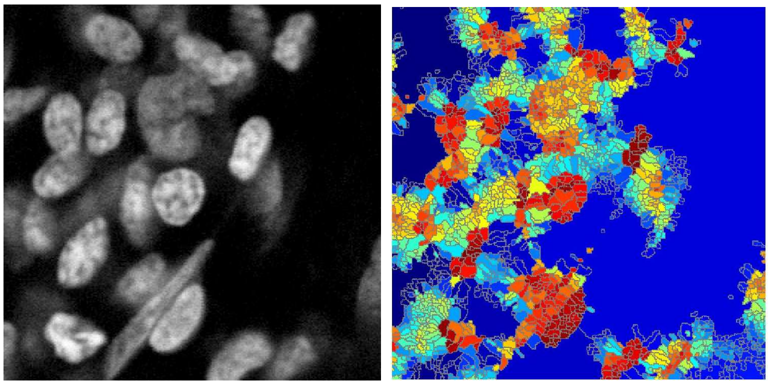


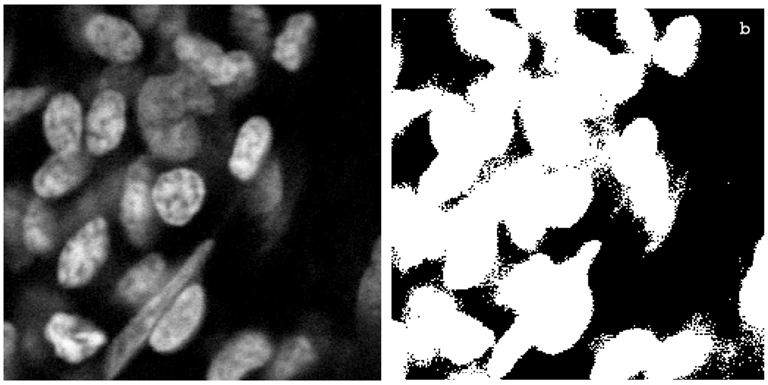
Figure 3.6: Result of watershed segmentation, the coloration indicates the (gray) level of water

data like the cell nuclei. There is no general algorithm or heuristic to overcome this and it is in no way intuitive when to unite single regions.

#### Seeded watershed

One popular region growing algorithm is the seeded watershed algorithm, which overcomes much of the drawbacks of simple watershed. [Wählby2003] introduced an adoption for cell segmentation which has proven to perform quite well on some data sets.

Fig. 3.7 shows some steps of this algorithm. The original image (a) is binarized




by the threshold technique to (b). In order not to loose too much information at this point the threshold is chosen to separate possible foreground from certain background.

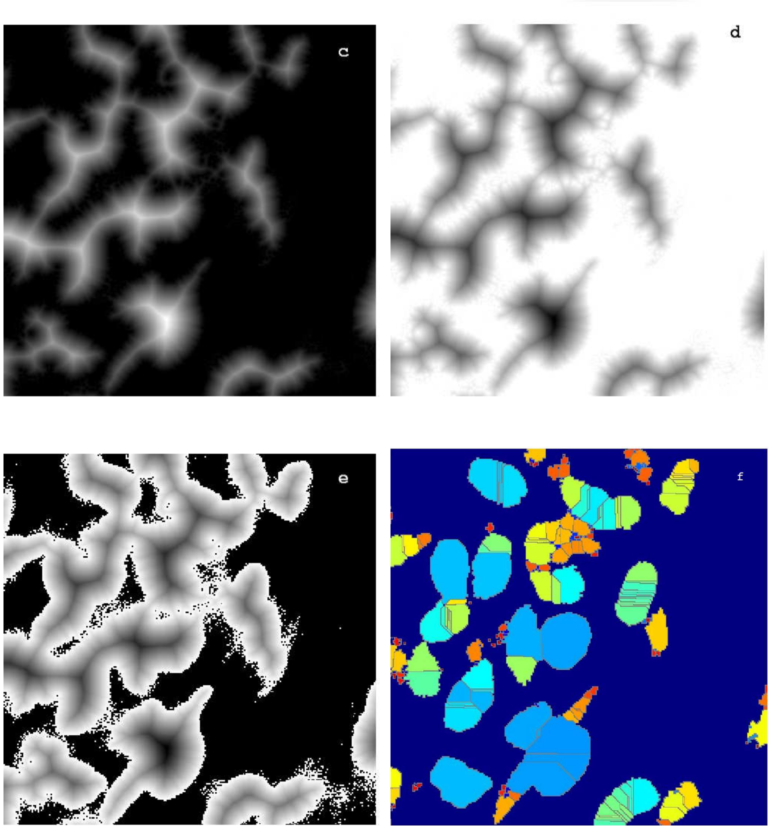



Figure 3.7: Result of seeded watershed segmentation, from left to right figures a,b,c,d,e and f. The coloration indicates the (gray) level of water.

Then, this result is distance transformed, for each foreground pixel, the distance to the background is calculated and coded in the gray values (c). In the inverse image(d), the background is set to $-\infty$ (e). The inner pixels, identified by distance transform, are used as "seeds" for watershed segmentation (f).

Even though the results of seeded watershed are by far less over-segmented than those by simple watershed, some major problems still remain unsolved. First, using thresholding for dividing fore- and background causes the same problems described before: either fibroblasts or macrophages are left out or strong noise reduces the overall segmentation performance.
Second, in the case of cell clusters, again, there is no reasonable algorithm to decide which regions to unite. [Wählby2003] propose a set of heuristics taking advantage of the overall round form of cells, but these are not adaptable to this case considering the form of pericytes i.e. Furthermore, fig. 3.7 (f) shows regions spreading over several cells.

### Discussion

The previous sections argued that none of the traditional segmentation algorithms solves the problem. Even though specialized algorithms like [Wählby2003] might perform well on one dataset, a generalization to even similar data is not implied.

## 3.2 Invariant Theory

In order to classify previously segmented objects, it has to be stated which objects are considered to be equal and which not. Especially different forms of appearance, so-called inner class variations, of objects belonging to the same class have to be handled. In most cases e.g. objects varying only in orientation or translation are considered to be equal. Thus, the mathematical representation of these objects should be **invariant** towards rotation and translation.

### 3.2.1 Equivalence Classes

An equivalence class $\epsilon$ for pattern $\{x_i\}$ is mathematically defined by a binary equivalence relation $\sim$ which holds:

$$x_1 \sim x_1 \quad \textbf{Reflexivity} \tag{3.1}$$

$$x_1 \sim x_2 \Rightarrow x_2 \sim x_1 \quad \textbf{Symmetry}$$

$$x_1 \sim x_2 \wedge x_2 \sim x_3 \Rightarrow x_1 \sim x_3 \quad \textbf{Transitivity}$$

Equivalence Classes are defined either by enumerating all members, which is hardly possible in most cases, or by a generating group $G$.

#### Groups

Generally speaking, groups are algebraic structures associated with a binary operation $\bullet$ which holds:

a) $a \bullet (b \bullet c) = (a \bullet b) \bullet c$

b) Existence of a neutral element $e$ with $a \bullet e = a$

c) Existence of the inverse $a^{-1}$ with $a \bullet a^{-1} = e$

In the case of image patterns, group operators define possible transformations like translation or rotation.

### 3.2.2 Transformation Groups

For a 2D signal (image) $I$ with elements $i_{x,y} \in I$ several practical relevant transformation groups exist.

#### Translation Group $\tau$

This is the group of a transformation operation closed to translation.

$$\tau_{k,l}(x, y) := (x + k, y + l) \tag{3.2}$$

This operation is commutative and separable $\tau_{k,l} = \tau_k \bullet \tau_l = \tau_l \bullet \tau_k$

### Rotation Group $r$

This group is closed to rotation. The rotation operator for image signals is defined in terms of multiplying the element coordinates $(x, y)^T$ with a rotation matrix $R$.

$$t' = Rt \quad \textbf{with } R = \begin{pmatrix} cos\phi & sin\phi \\ -sin\phi & cos\phi \end{pmatrix} \tag{3.3}$$

$R$ has to satisfy $R^T = R^{-1}$. In case of $\det(R) = -1$ $r$ is a reflection, in case of $\det(R) = 1$ a rotation. Under $r$-transformations distances are preserved.

### Euclidean Motion Group $c$

This group is closed to Euclidean motion, translation and rotation, which can be parameterized as follows

$$t' = Rt + a \quad \textbf{with } R = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \textbf{ and } a(a_1, a_2)^T \tag{3.4}$$

$R$ has to satisfy $R^T = R^{-1}$. In case of $\det(R) = -1$ $r$ is a reflection, in case of $\det(R) = 1$ a rotation. Under $c$-transformations distances are preserved.

## 3.2.3 Invariant Features

Given all kinds of group transformations $g$ on pattern $x$, one is looking for a representation which maps all $g(x)$ to their desired equivalence class $\epsilon(g(x))$. This is achieved by a function $T$, which maps all possible $g(x)$ to the same point in a feature space.

$$\forall g : T(g(x)) = \tilde{x} \tag{3.5}$$

$T$ is called the **invariant** regarding $g$. The features retrieved by the extraction of invariants are then used for classification.

### Necessary Conditions for Invariance

The necessary condition in order to achieve invariants towards a group transformation $g$ according [Burkhardt1979] is:

$$x_1 \overset{g}{\sim} x_2 \Rightarrow T(x_1) = \tilde{x_1} = \tilde{x_2} = T(x_2) \tag{3.6}$$

### Sufficient Conditions for Invariance

To achieve completeness for invariant mappings, $T$ has to hold

$$T(x_1) = T(x_2) \Rightarrow x_1 \overset{g}{\sim} x_2 \tag{3.7}$$

In most cases this condition is too strict, it is not practicable to have a distinct mapping for every theoretically possible pattern $x_i \in X$. A-priori knowledge is used to determine a subset $X_0 \subset X$ of likely patterns. If (3.7) holds for all $x_1, x_2 \in X_0$, separability is guaranteed for the classes.

### Characteristic of Invariants

Besides separability some other characteristics are desirable for invariants, too:

- **consistency** causes small changes in the pattern to have only small effects in feature space.
- **cluster preservation** transfers global characteristics from pattern to feature space.
- **robustness** against variation of environmental variables like the level of fluorescence.

### 3.2.4 Non-Linear Mappings

Another major constraint lies on invariants: the mapping into feature space has to be **non-linear**. To illustrate this fact, one may take the following lemma into consideration:

**Lemma 1** *"Let a linear system $S$ be given. Two signals $x_1, x_2$ are equivalent if and only if the difference $x_1 - x_2$ is an element of the Nullspace $N(S)$ of $S$." [Burkhardt1979]*

The correctness of this lemma follows directly from the definition of linearity. The lemma implies that by the use of linear systems only trivial bounded equivalence classes can be separated. This is expressed in the proposition [Burkhardt1979]:

**Proposition 2** ***"Linear systems can discriminate between frequencies or spectral components, but only non-linear systems are able to discriminate between equivalence classes"***

### 3.2.5 Calculating Invariants

For a formal calculation the notation of group transformations is extended to $g_{\vec{p}}(x)$, where $\vec{p}$ denotes the transformation parameter vector. All possible parameter combinations, $g_{\vec{p}}(x)$ can be illustrated as an "orbit" [Burkhardt1979] for every pattern. By

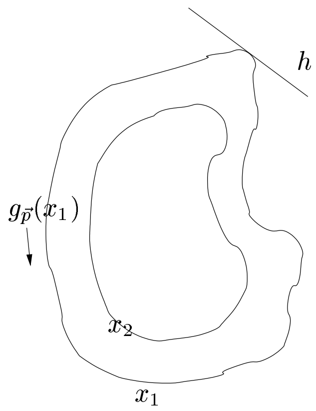



Figure 3.8: Parameter orbits generated for two patters, $x_1, x_2$

the definition of groups, these orbits can not intersect. Starting from the orbit model, there are basically three different approaches to find an invariant $T$ which maps an entire orbit to one point in feature space: normalization, derivation and integration.

#### Normalization

In trivial cases, normalization can be used to retrieve invariants. The basic idea is to find single characteristics of the orbit, like the point of maximal flection (denoted as $h$ in fig 3.8), which are able to identify the orbit.
These normalizations meet the necessary condition (3.6), but completeness is all but guaranteed.

#### Derivation

Since the only difference of group elements is encoded in the parameters $\vec{p}$, an invariant has to eliminate this influence. This can be achieved by derivation of the Image $X$ for arbitrary $\vec{p}$:

$$\frac{\delta T(g_{\vec{p}}(X))}{\delta p_i} = 0 \tag{3.8}$$

This differential equation can be solved using Lee-Theory approaches, but this is often very difficult. The derivation method meets the necessary condition (3.6), but completeness is not guaranteed.

#### Integration

The more practical approach is Haar-Integration over all degrees of freedom of the group transformation $G$,

$$T[f](X) = \int_G f((g_{\vec{p}}X))d\vec{p} \tag{3.9}$$

which eliminates the influence of $p$. $f$ denotes an arbitrary non-linear function.

For finite groups in discrete space the Haar-Integral transforms to a sum over all $N$ elements of $X$ and is calculated by

$$T[f](X)) = \sum_{i=0}^{N-1} f(x_i), \forall x_i \in X \tag{3.10}$$

This sum, often referred to as "group average", meets the necessary condition (3.6) for invariants.

### 3.2.6 Group Average Invariants

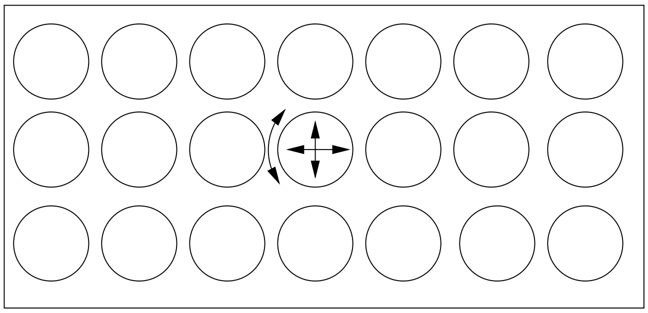
For Euclidean motion a 2D signal $X$ parameterized as in (3.4) with translation $t_0, t_1$ and rotation $\phi$, an invariant $T$ can be directly derived from (3.9)

$$T[f](X) = \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy \int_0^{2\pi} d\phi\ f(g_{x,y,\phi}X) \tag{3.11}$$

by integrating over all possible translations and rotations. Instead of transforming $X$ by group operations, an equivalent result is reached by $g$ transforming $f$. Integrating $f$ over all $(x, y)$ positions and all rotations of $\phi$, an invariant in form of a single real number is retrieved.

(3.12)

The degree of completeness of these invariants can be increased by combining several



$$= \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy \int_0^{2\pi} d\phi\ f(g_{x,y,\phi}X)$$

Figure 3.9: The evaluation of the function $f$ in all rotations is integrated over $X$.

different functions to a feature vector. However, this method has a major drawback: computational complexity.

## 3.3 Classification

After segmenting objects of interest and retrieving features, the next step would be the training of a classifier model followed by the actual classification based on this model. The classifier used in this context is the support-vector machine (SVM).

### 3.3.1 Support-Vector Machines

Support-vector machines were introduced in the mid nineties by [CortesVapnik95] and turned out as very efficient and easy to handle classifier which by now is the method of choice in many areas of classification and machine learning.
Principles from theory of statistical learning lead to a two class supervised learning classifier. In the trivial case, the set of labeled features vectors given

$$(\vec{x_1}, y_1), ..., (\vec{x_n}, y_n) \in \mathbb{R}^d \times \{-1, +1\}$$

is **linear separable**. The SVM calculates the hyperplane with the largest margin, which separates the feature-vectors $\vec{x_n}$ of the different labels $y_a = 1$ and $y_b = -1$. As fig. 3.10 shows, a distinct description of the hyperplane can be modeled by a few feature-vectors, called support-vectors (SV). The hyperplane is specified by the set of

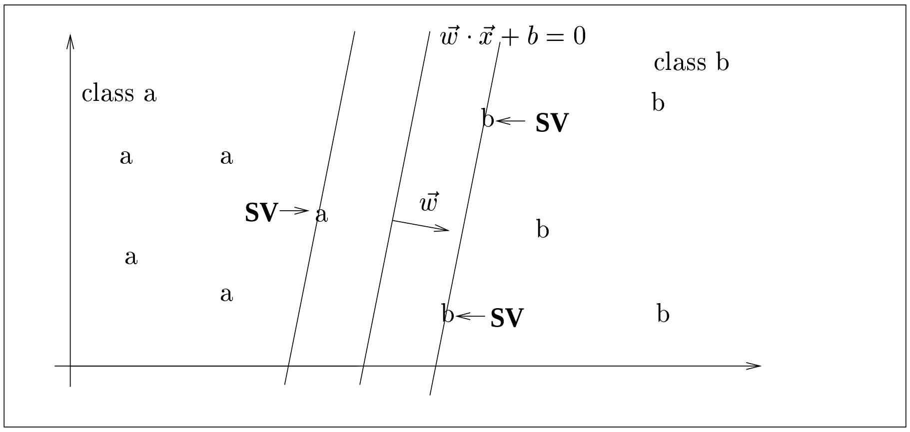



Figure 3.10: Hyperplane with largest margin separates the two classes.

vectors $\vec{x}$ which hold:

$$\text{Hyperplane} \quad \vec{w} \cdot \vec{x} + b = 0 \tag{3.13}$$

where $\vec{w} \in \mathbb{R}^d$ is referred as the normal vector of the hyperplane and $b \in \mathbb{R}$ denotes the scaled distance of the plane to the origin. The class $\hat{y} \in \{-1, +1\}$ of a vector is determined by:

$$\text{Classification} \quad \hat{y} = \text{sgn}(\vec{w} \cdot \vec{x} + b) \tag{3.14}$$

#### Calculating the Hyperplane

By normalizing the hyperplane margin to $min|\vec{w} \cdot \vec{x_i} + b| = 1$ the calculation can be reduced to an optimization problem which is solved by minimizing $\|\vec{w}\|^2$.

Thus, applying a general Lagrange functional, the resulting minimization problem looks like this:

$$\text{minimize} \quad L(\vec{w}, b, \alpha) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\vec{w} \cdot \vec{x_i} + b) - 1] \quad \text{with } \alpha_i \geq 0 \tag{3.15}$$

which can be transformed to the dual maximization problem:

$$\text{maximize} \quad L'(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x_i} \cdot \vec{x_j}) \tag{3.16}$$

$$\text{subject to } \alpha_i \geq 0 \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

Solving for $\alpha$ numerically via Karush-Kuhn-Tucker conditions (KKT) delivers the results for the desired hyperplane (3.13):

$$\vec{w} = \sum_{i=1}^{n} \alpha_i y_i \vec{x_i} \tag{3.17}$$

$$b = -\frac{\max_{y_i=-1}(\vec{w} \cdot \vec{x_i}) + \min_{y_i=1}(\vec{w} \cdot \vec{x_i})}{2)} \tag{3.18}$$

This numerical solution leads to a global optimum, hence the distinct optimal hyperplane is found. For vectors outside the hyperplane margin counts $\alpha_i = 0$ since none of theses vectors contribute to the solution. Vice versa,

$$\text{for all SV holds: } \alpha_i \neq 0 \tag{3.19}$$

### Deriving the Hyperplane Equation

Given $m$ training vectors $(\vec{x_i}, y_i)$ with $\vec{x_i} \in \mathbb{R}^n$ and $y_i \in \{\pm 1\}$, the decision function $f_{\vec{w},b} = \text{sig}((\vec{w}, \vec{x}) + b)$ is correct, if for all $(\vec{x_i}, y_i)$ holds:

$$y_i(< \vec{x_i}, \vec{w} > +b) > 0 \text{ all vectors are correctly classified}$$

With the margin normalized to 1, it directly follows that

$$y_i(< \vec{x_i}, \vec{w} > +b) \geq 1$$

Thus, for the two closest vectors belonging to different classes, it states that

$$\begin{aligned} < \vec{x_1}, \vec{w} > +b &= +1 \\ < \vec{x_2}, \vec{w} > +b &= -1 \end{aligned}$$

The distance between $\vec{x_1}$ and $\vec{x_2}$ is $< \vec{w}, (\vec{x_1} - \vec{x_2}) >\geq 2$. Taking the normalization to $\|\vec{w}\|$ into account, the real distance is given by

$$< \frac{\vec{w}}{\|\vec{w}\|}, (\vec{x_1} - \vec{x_2}) >= \frac{2}{\|\vec{w}\|} \tag{3.20}$$

This states, that the margin is maximized by minimizing $\|\vec{w}\|$.
The actual minimization however is done on the term $r(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2$, which states the distance of one of these vectors to the hyperplane and is a **convex** problem. Combined with the initial constraint $y_i(< \vec{x_i}, \vec{w} > +b) \geq 1 \quad \forall i = 1...m$, this directly leads to a Lagrangian approach.

### Lagrangian Approach

Following KKT conditions, the optimum of the Lagrangian expression 3.15 is determined by the saddle point which is retrieved by the derivative of $L(w, b\alpha)$ with respect to $b$ and $\vec{w}$:

$$\frac{d}{db}L(\vec{w}, b\alpha) = 0 \Rightarrow \sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\frac{d}{dw}L(\vec{w}, b\alpha) = 0 \Rightarrow \vec{w} = \sum_{i=1}^{m} \alpha_i \vec{x_i} y_i$$

This shows that only $\alpha_i > 0$ contribute to the solution $\Rightarrow$ **support-vectors**.
By substituting $w$ and $b$ by the constraints, the optimization problem can be reduced from three variables to a dual one variable problem maximizing $\alpha$ (3.16).
In the same way, $w$ can be substituted in the decision function as well, leading to

$$f(x) = \mathrm{sig}(\sum_{i=1}^{m} \alpha_i y_i (x \cdot x_i) + b) \tag{3.21}$$

### The Non-Linear Separable Case

In the non-trivial case, where train-vectors are not linear separable, of course no hyperplane can be found. To overcome this, feature-vectors are mapped into higher dimensional space $H$ by use of some non-linear function

$$\Psi(\vec{x}) : \mathbb{R}^d \to H, \tag{3.22}$$

$\Psi$ is chosen in a way, such that classes can be separated in $H$ by the trivial hyperplane SVM method.

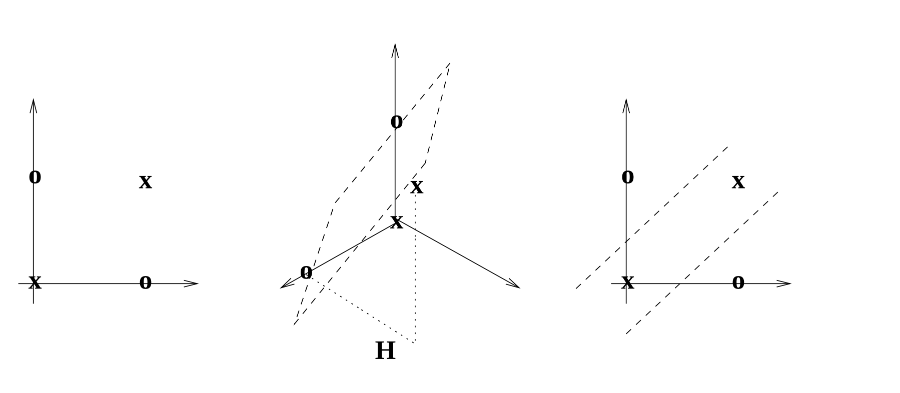



Figure 3.11: Separating classes in high dimentional space

### The Kernel-Trick

Neither the determination of an appropriate function $\Psi$ nor calculations in the space $H$ via $\Psi(\vec{x_i})$ itself are simple tasks. To avoid these difficulties, one uses a technique often referred to as the "kernel-trick".
By investigating the modified optimization problem in $H$,

$$L'(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\Psi(\vec{x_i}) \cdot \Psi(\vec{x_j})) \tag{3.23}$$

it turns out that only the **inner products** $\Psi(\vec{x_i}) \cdot \Psi(\vec{x_j})$ have to be calculated in $H$. These were introduced to measure distances or, more generally spoken, similarities. So rather than looking for some function $\Psi$, it is adequate to find an analog equivalence measure

$$K(\vec{x_i}, \vec{x_j}) = \Psi(\vec{x_i}) \cdot \Psi(\vec{x_j}) \tag{3.24}$$

in $H$. $K(\vec{x_i}, \vec{x_j})$ is referred to as **kernel-function**. In the linear case, this is exactly the inner product.

### Kernel-functions

The most popular kernel-functions are:

- $K(\vec{x}, \vec{x'}) = \vec{x} \cdot \vec{x'}$ linear
- $K(\vec{x}, \vec{x'}) = (\vec{x} \cdot \vec{x'})^d$ polynomial
- $K(\vec{x}, \vec{x'}) = tanh(\vec{x} \cdot \vec{x'} + \theta)$ sigmoid
- $K(\vec{x}, \vec{x'}) = exp(-\frac{\|\vec{x}-\vec{x'}\|^2}{2\sigma^2})$ RBF or gauss

The radial basis function (RBF), also known as gauss kernel, is the first choice in most cases because it combines good performance with a strong theoretic foundation. It can be proven that the RBF-kernel is equivalent to the inner product of an infinite dimensional space [Schölkopf2002].

### Kernel Design

Starting from a finite space $X = \{x_1, ..., x_r\}$, a kernel-function is represented by a symmetric $r \times r$ matrix which holds $K_{ij} = k(x_i, x_j)$.

Using Eigenvalue decomposition $K$ can be rewritten as $K = UVU^T$ with the $r \times n$ matrix $U = (\vec{u_1}^T; ...; \vec{u_r}^T)$ satisfying $UU' = I_n$ and $V = diag(\lambda_1, ...\lambda_n), \lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n > 0$.

The kernel-function $\Psi : X \to K \subseteq \mathbb{R}^n$ is now expressed as

$$\Psi(\vec{x_i}) = V^{\frac{1}{2}} \vec{u_i} \tag{3.25}$$

This results in the Gram-Matrix:

$$G_{ij} = (\Psi(\vec{x_i}), \Psi(\vec{x_j}))_K = (V^{\frac{1}{2}} \vec{u_i})^T (V^{\frac{1}{2}} \vec{u_j}) = \vec{u_i}^T V \vec{u_j} = K_{ij} \tag{3.26}$$

This shows that in fact the only restriction on kernels is that their Eigenvalues have to satisfy $\lambda \geq 0$. Thus, $K$ has to be **semi positive definite**. In general, a $K$ is a kernel iff $K$ holds $\sum_{i,j=1}^{m} \alpha_i \alpha_j K(\vec{x_i}, \vec{x_j}) \geq 0$. This is stated by the Mercer Theorem:

### Mercer Theorem

$K(\vec{x_i}, \vec{x_j}) = (\Psi(\vec{x_i}), \Psi(\vec{x_j}))$ iff for arbitrary $g(\vec{x})$ with $\int g(\vec{x})^2 dx < \infty$ holds:

$$\int K(\vec{x_i}, \vec{x_j}) g(\vec{x_i}) g(\vec{x_j}) dx_i dx_j \geq 0 \tag{3.27}$$

### Soft-Margin Hyperplanes

Even though the use of kernels makes it possible to separate all sets of feature-vectors just by adjusting the kernel parameters - in many cases it is not advantageous to achieve perfect separation. Outliers caused by noise should not dominate the modeling of the hyperplane. For this reason, the plane margin is "softened":
Changing the 3.16 constraints to $y_i(<\vec{x_i}, \vec{w}> + b) \geq 1 - \xi_i$ enables the misclassification of training samples. To limit the $\xi_i$, each maverick is penalized:

$$\min r(\vec{w}, \xi_i) = \frac{1}{2}\|\vec{w}\|^2 + \frac{c}{m}\sum_{i=1}^{m}\xi_i \tag{3.28}$$

The 3.16 constraints are now formulated as $0 \leq \alpha_i \leq \frac{\xi_i}{m}$ and $\sum_{i=1}^{m}\alpha_i y_i = 0 \quad \forall i = 1...m$

### Multi-Class SVM

So far only two-class problems have been discussed. And actually, since only one single hyperplane is used, SVMs are only capable of separating two classes. Fortunately, any multi-class problem can be easily broken down to several two-class problems. Essentially there are two ways of doing so: "One vs. One" or "One vs. Rest".

**One vs. Rest**. For each class there is one SVM trained against all other classes. For classification, vectors are tested against these models and the results of all SVMs are compared.

**One vs. One**. For $n$ classes $\frac{1}{2}n(n-1)$ pairwise SVMs are trained. Even though more models are trained, One vs. One is often faster in classification hence the overall number of SVs is lower due to smaller models.

### 3.3.2 Model Selection

With a powerful classifier at hand, like the support-vector machine described in the previous section, classification results depend on the training samples and a set of few training parameters. In case of the SVM, these parameters are the cost for outliers and kernel parameters like gamma.
By training, samples and parameters form a **model**. Since there is an infinite number of possible parameter combinations, the task of finding the best model is called **model selection**.

The first concern is how to measure models - what is a good model? In an intuitive way this question is easy to answer: a good model always gives a correct classification result. But in practice this turns out to be a bit more complicated: the number of training-vectors is limited and often requires expensive manual labeling, while the number of possible test-vectors is infinite. Thus, a good classifier not only recognizes all known samples, but also all unknown samples. This property is referred to as the **generalization** ability of a model.

If a model has a weak generalization ability, then the separation of classes is conducted too "close" to the training samples - **overfitting** appears. As a theoretical approach

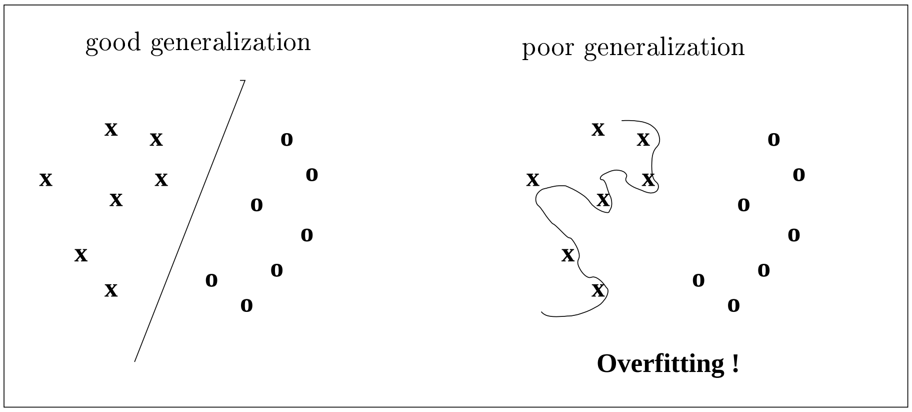



Figure 3.12: Effects of a model's generalization ability

to generalization ability [Vapnik79] introduced VC-Dimention as measure.

#### Finding a Model

A model generated by training is referred to as $h$ and the classification result of some vector $\vec{x}_i$ regarding $h$ is $h(\vec{x}_i)$. Performance for vectors $\vec{x_i}$ labeled with $y_i$ is detected in terms of a 0/1-loss function:

$$L(h(\vec{x}_i), y_i) =: \begin{cases} 1 & h(\vec{x}_i) \neq y_i \\ 0 & \text{otherwise} \end{cases} \tag{3.29}$$

In practice model selection is achieved by performing a grid search over a predefined array of parameter combinations, building a model for each and then estimating the model error rate $Err^n(h)$.
There is a vast number of estimation techniques available [Fehr2003], but Cross Validation has proved to be one of the most reliable and easiest to handle methods.

### Cross Validation

One advantage of Cross Validation is that this algorithm takes all available training vectors into account when estimating $\mathrm{Err}^n(h)$. The basic idea is to iteratively split the set of $S_n$ training samples into $m = \frac{n}{k}$ pairs of sets randomly. Each pair consists of a training set $S^{\text{train}}$ of size $k$ and a test set $S^{\text{test}}$

$$\forall \delta = 1, ..., m : \qquad S^{\text{train}}_{l_\delta} \bigcup S^{\text{test}}_{k_\delta} = S_n \quad \text{with } S^{\text{train}}_{l_\delta} \bigcap S^{\text{test}}_{k_\delta} = \varnothing$$

For all pairs, a model is trained and the samples from $S^{\text{test}}$ are classified. The use of 3.29 leads to the test error of each pair

$$\mathrm{Err}^{k_\delta}_{\text{test}}(h_\delta) = \frac{1}{k} \sum_{(\vec{x}_i, y_i) \in S^{\text{test}}_{k_\delta}} L(h_\delta(\vec{x}_i, y_i)) \tag{3.30}$$

Averaging over all test errors gives an approximation of the model error rate $\mathrm{Err}^n(h)$

$$\mathrm{Err}^n_{\text{cross}_m}(h) := \frac{1}{m} \sum_{\delta=1}^{m} \mathrm{Err}^{k_\delta}_{\text{test}}(h_\delta) \tag{3.31}$$

The quality of the $\mathrm{Err}^n$ approximation by $\mathrm{Err}^n_{\text{cross}_m}$ depends on the choice of $k$. A small $k$ reduces the negative bias on the estimate but increases calculation time. In the most expensive case, called Leave One Out (LOO), with $k$ is set to 1, the best results can be expected.

# Chapter 4

# Gray-Scale Invariants

Originating from the idea of Haar-Integral invariants (3.9) and the group average (3.11), gray-scale invariants were introduced by [Schulz-Mirbach95] in 1995.
Since then, gray-scale invariants have proven high separation performance, both in 2D gray-scale images [Burkhardt2001] and 3D gray-scale volume data [Ronneberger2002],[Schäl2000].

The basic goal of mapping all members of an equivalence class (3.1) to one point in feature space in order to generate a representation invariant (3.6) of group transformations is achieved by a Haar-Integration approach.

$$T[f](X) = \int_G f(gX)dg \tag{4.1}$$

The invariant $T$ is retrieved by integrating transformations $g$ of group $G$ over all possible parameters $p$ on a given pattern $X$. Choosing rather simple kernel functions $f$ is the key which made gray-scale invariants practicable.

## 4.1 Two-Point Gray-Scale Invariants

In most cases where gray-scale invariants are mentioned in this work, actually two-point gray-scale invariants are referred to. The two-point approach takes advantage of a family of simple kernel functions combining only two points for local support. This is illustrated by fig.4.1 for the 2D case.

### 4.1.1 Invariants for 2D Signals

Concerning 2D signals (images), usually feature representation invariant towards Euclidean motion (3.4) is demanded. For an image the group transformations rotation and translation are parameterized and substituted into (4.1)

$$T[f](X) = \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy \int_{0}^{2\pi} d\phi \; f(g_{x,y,\phi}X) \tag{4.2}$$

Instead of transforming the entire image $X$, the group transformation can be applied to the kernel function. This speeds up computation to linear complexity.
Figure (4.1) illustrates the two-point Haar-integration in the 2D case. Of course, in a discretized image, integration is replaced by summation and rotation results have to be interpolated.

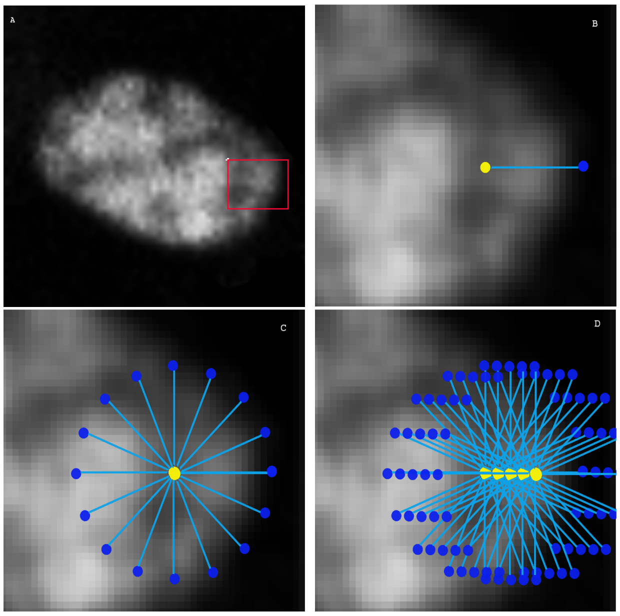



Figure 4.1: **Two-Point gray-scale invariants of an 2D image.** A) shows the magnified selection of a gray-scale cell image. B) the non-linear kernel function $f$ combining two pixels in a neighborhood. C) integrating $f$ over all angles, achieving invariants towards rotation. D) integrating $f$ over all translations, achieving invariants towards translation.

Studying fig.(4.1) leads to two further observations: First, a subset of the kernel functions $f(X)$ can be understood as combination

$$f(X) = f_a(X(\vec{0})) \cdot f_b(X(\vec{r})) \tag{4.3}$$

of two non-linear functions, $f_a$ and $f_b$ with a scope of $r$. Defining $A := f_a(X), B := f_b(X)$ equation (4.2) can be directly rewritten as

$$T[f](X) = \int_x \int_y A(x,y) \int_\phi B(x + r\cos(\phi), y + r\sin(\phi)) d\phi dx dy \tag{4.4}$$

Second, the rotation term can be expressed as convolution $(*)$ of the image with a circle $C$ of radius $r$, which leads to

$$T[f](X) = \int_x \int_y A(x,y) \cdot (B * C)(x,y) dx dy \tag{4.5}$$

$$C(x,y) = \begin{cases} 1 & \sqrt{x^2+y^2} = r \\ 0 & \text{otherwise} \end{cases}$$

### Fast Computation in Fourier Space

With convolution at hand, it is self-evident to perform the calculation in Fourier space. This way, a further substantial speedup is achieved.

Rewriting the discrete algorithm (4.6) by substituting fast fourier transformation (FFT) leads to the final formulation for two-point gray-scale invariant calculation.

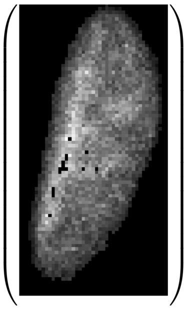
$$T = \sum_{x=0}^{N} \sum_{y=0}^{M} A(x,y) \cdot (B * C)(x,y) \tag{4.6}$$

$$T = \sum_{x=0}^{N} \sum_{y=0}^{M} A(x,y) \cdot FFT^{-1}(FFT(B) \cdot FFT(C))(x,y) \tag{4.7}$$

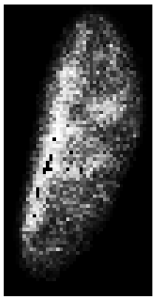
### Calculating Gray-Scale Invariants Step by Step

**1.** the original cell gray-scale values are transformed with a non-linear kernel function

$f_b$

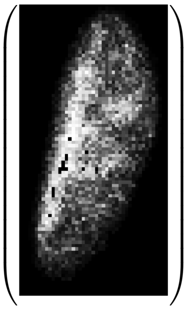


$=$

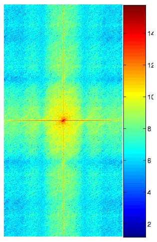


**2.** the result is mapped into fourier space (FFT)

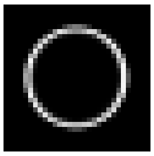
$FFT$



$=$

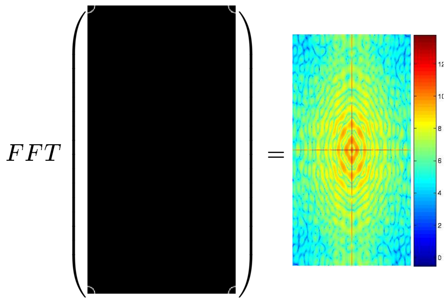


**3.** A circle of radius $r$ is generated

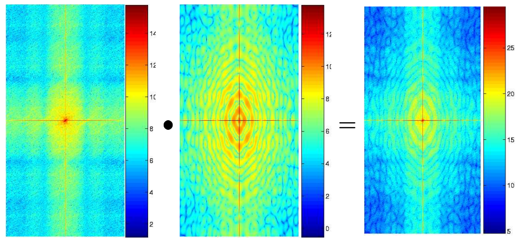


$$C(x,y) = \begin{cases} 1 & \sqrt{x^2+y^2} = r \\ 0 & \text{otherwise} \end{cases}$$

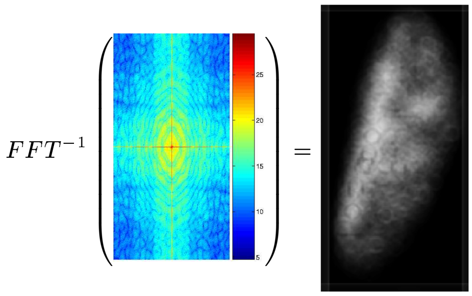
**4.** padding $C$ to the size of $X$ and centering of circle's zero frequencies before performing the FFT



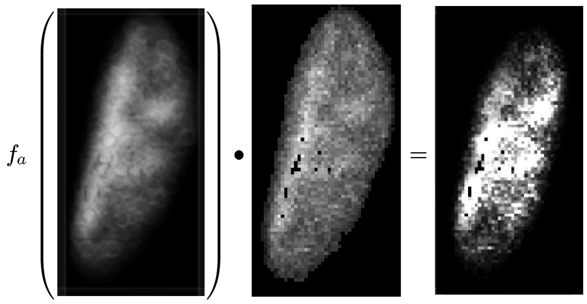

**5.** convolution by pointwise multiplication in fourier space $FFT(C) \cdot FFT(f(X))$



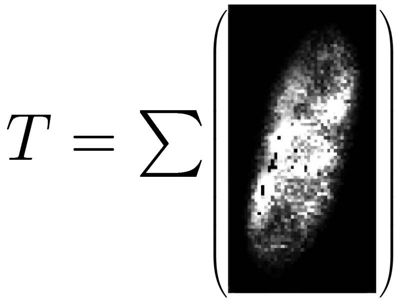

**6.** the result is transformed from fourier space ($FFT^{-1}$)




**7.** point wise multiplication with the non-linear transformation of the original cell gray-values




**8.** calculation of the invariant $T$ by summing over all pixels $(x, y)$



### 4.1.2 Increasing Separability

To increase the separability (??) of feature representation, several features are calculated and combined to a feature vector. This can be obtained either by choosing different non-linear mappings or, even more important, by varying the scope of local support.

In terms of the frequency domain, the range of the local support encodes the frequency spectrum of the feature retrieved from pattern texture. In other words, kernels with a large range describe the overall pattern shape (low frequencies), while those with close support encode local characteristics (high frequencies).
One way of adjusting the range is to scale the original pattern before extracting the features. Scalings of 1:1, 1:2, 1:4, 1:8, 1:16, 1:32, and 1:64 were suggested by [Ronneberger2002].

### 4.1.3 Multi-Channel Invariants

Another advantage of gray-scale invariants is that they can be easily extended to multichannel images or volume data.
Given a multi-channel pattern $X_n$ with channels $i, j \in \{0, \dots, n-1\}$, invariant features are retrieved by combining pixels over different channels:

$$A := f_a(X_i), B := f_b(X_j)$$

$$T = \sum_{x=0}^{N} \sum_{y=0}^{M} A(x,y) \cdot FFT^{-1}(FFT(B) \cdot FFT(C))(x,y) \tag{4.8}$$

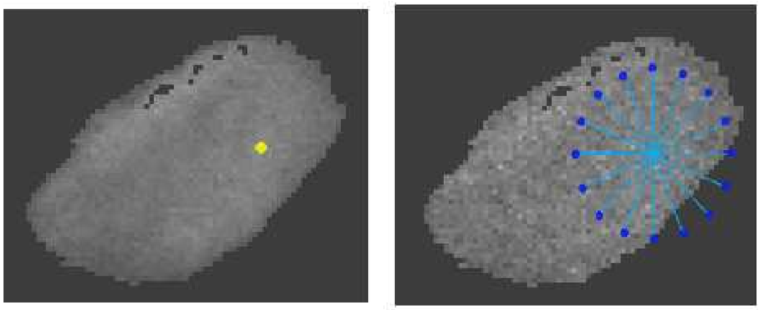


Figure 4.2: Combining pixels of different channels with a kernel function

### 4.1.4 Invariants for 3D Volume Data

The algorithm of two-point gray-scale invariants can be directly applied to 3D volume data. The additional parameters of Euclidean movement necessary for a $N \times M \times K$ gray-scale volume

$$T[f](X) = \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dz \int_{0}^{2\pi} d\phi \int_{0}^{\pi} d\theta \; f(g_{x,y,z,\phi,\theta} X) \tag{4.9}$$

are handled by replacing the circle by the surface of a sphere which holds:

$$S(x,y,z) = \begin{cases} 1 & \sqrt{x^2 + y^2 + z^2} = r \\ 0 & \text{otherwise} \end{cases}$$

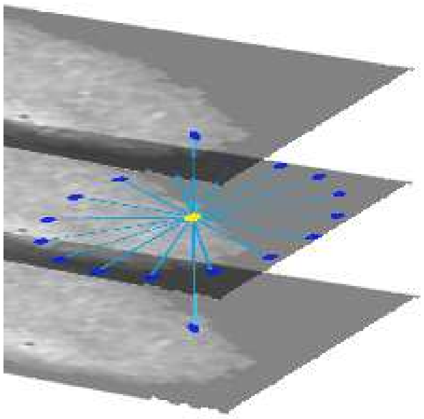


Figure 4.3: Two-Point gray-scale invariants for 3D Volumes. Kernel functions combine pixels in a 3D neighborhood.

## 4.2 Three-Point Gray-Scale Invariants

Starting from two-point gray-scale invariants, one could argue that instead of combining more and more simple features, separability could also be achieved by fewer but more complex features.
Intuitively, combining three pixels with two different kernel functions seems to be a straightforward approach: And even the formalization does not appear to cause further

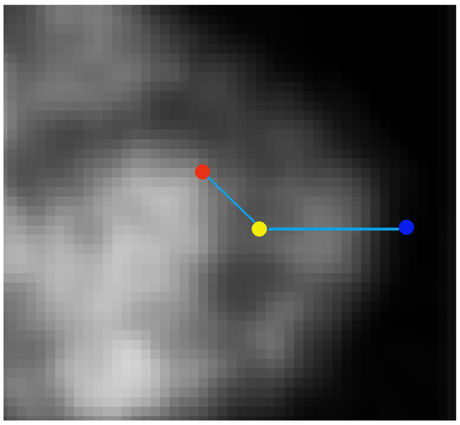


Figure 4.4: Three-Point Gray-Scale Invariants by combining 3 pixels with two kernel functions

problems,

$$f(X) = f_a(X(\vec{0})) \cdot f_b(X(\vec{r_1})) \cdot f_c(X(\vec{r_2})) \tag{4.10}$$

but, taking a look at the parameterization,

$$T[f](X) = \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy\ A(x,y) \int_{0}^{2\pi} d\phi\ B(x+r_1\cos(\phi), y+r_1\sin(\phi))C(x+r_2\cos(\phi+\Delta), y+r_2\sin(\phi+\Delta)) \tag{4.11}$$

one has to realize that the calculation of this feature cannot be expressed in terms of a cheap one-fold convolution anymore. For n-fold convolutions ($n \geq 2$) the calculation complexity grows tremendously, making three-point invariants rather useless for practical application.

# Chapter 5

# Segmentation and Classification using Voxel-Wise Gray-Scale Invariants

In traditional pattern recognition the classification of an object $X$ in a dataset $I$ is achieved by the following three steps: segmentation (see chapter 3.1), feature extraction (chapter 3.2 and 4) and classification (chapter 3.3).
It is evident, that bad performance at any point in this processing pipeline will cause pure recognition results. The combination of gray-scale invariants in the middle and support-vector machines at the end of the processing chain has proven to deliver a very good recognition rate of e.g. at around 98% on vast structured pollen data[Ronneberger2002]. Thus, the focus turns to the first step of segmentation.

As stated in chapter 3.1, segmentation of complex objects from datasets with structured background, like cells in tissue, is a difficult task. The main reason for this difficulty lies in the nature of the problem: in order to segment an object, it has to be recognized - but segmentation itself is the first step towards recognition.
To overcome this dilemma, usually some kind of a-priory knowledge about the objects is used to identify regions of interest. The more difficult the problem is, the more a-priory knowledge is needed. This leads to very specialized segmentation algorithms which cannot be generalized to even similar problems. Often manual pre-segmentation is required.

To avoid this in a general approach, the a-priory knowledge can be replaced by a self learning algorithm which automatically retrieves the same information from object examples.
Furthermore, in a discrete dataset $I$, segmentation is just the task of finding subsets of pixels/voxels $S_n \subset I$, which are then labeled by a self-learning classifier: $\Phi(S_n) = y_n$, with $y_n \in \{0, \dots, N\}$. If it is possible to label the pixel $x_i \in I$ directly, thus $\Phi(x_i) = y_n$, segmentation and classification are achieved in one step without expensive search for subsets.

The pattern recognition pipeline for such a self-learning segmentation approach looks

a bit different: first, features have to be extracted for each pixel. Secondly, training samples have to be selected manually and a classifier has to be trained. Then all pixels/voxels are classified. Part two and three can be iteratively repeated until the segmentation results are satisfying. Last, neighboring pixels/voxels of the same class are combined to one object.
Of course the training part has to be done only once to build a model. This general approach can be adopted to any further classes and datasets, just by extending or retraining the model.

## 5.1 Pixel/Voxel-Wise Gray-Scale Features

The pixel/voxel-wise calculation of two-point gray-scale invariants is adopted from object invariants (4.6). For the sake of simplicity, this derivation is shown for the case of 2D images - the 3D case is treated analogously.
Considering Euclidean motion, a feature representation of objects has to be invariant towards the orientation and location of a fixed set $S \subset I$ of member pixels $x_i \in S$ within the original image $I$. Any group transformation of the set does not affect the internal structure like neighborhood relations, distances between pixels or angels between three pixels of the set. Thus, from a pixel perspective, translation invariance is already encoded in its neighborhood. On the other hand, the representation of the surrounding structures gains more importance, hence these features are not averaged over the entire object in a pixel based approach.

Neglecting translation, (4.6) can be rewritten for pixel-wise invariants:

$$T(x,y) = A(x,y) \cdot (B * C)(x,y) \tag{5.1}$$

### 5.1.1 Increasing Local Support

To extend the representation of the surrounding structures (textures) of a pixel, several further adoptions are necessary. Increasing the local support transfers more information from the neighborhood into the features. This can be achieved in two ways: first, by adding further features with kernel functions of increased scope, and second, by extending the kernels to combine rather region-to-region than pixel-to-pixel relationships.

Increasing the scope of a kernel function is realized just by increasing the radius of the circle used for convolution. Since features are not averaged over the entire object, it is advantageous to have at least one feature encoding a neighborhood large enough to contain the entire object.
Since rescaling of the whole image in order to retrieve features with kernels of different scope is quite expensive, circles of different radius are used. But it has to be considered that the original down-scaling of the images is always combined with a interpolation of pixels. This maps the structure of a region to one pixel, generating the positive effect, that with an increasing scope, large-scale properties more and more dominate the extracted features (see chapter 4.1.2). To transfer this important effect to the fast circle scaling method, kernel functions need to be extended. Kernel functions combining regions with regions are constructed by the use of a gauss filter $\mathcal{G}$. Even better than interpolation, the gauss filter maps the environment of a pixel weighted by distance. By filtering both, the circle and the result of the non-linear transformation before convolution, the kernel combines a region with a region of local support.

$$T(x,y) = \mathcal{G} * A(x,y) \cdot ((\mathcal{G} * B) * C)(x,y) \tag{5.2}$$

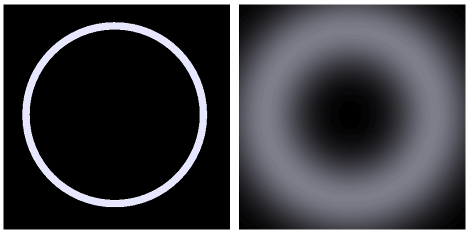


Figure 5.1: Gauss filtering a circle maps the environment of a pixel weighted by distance.

The degree of gauss filtering is closely related to the size of the scope. For kernels encoding local (high frequency) features, no or only a slight filter is applied. For kernels retrieving more global (lower frequency) features, filtering is intensified. Since Gauss

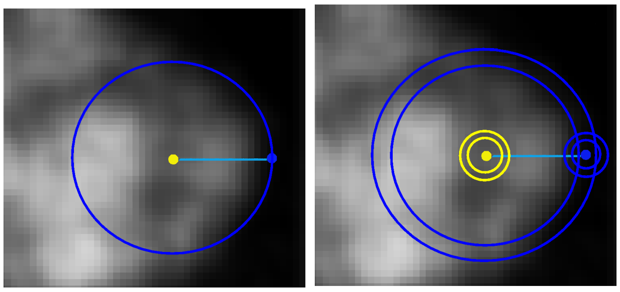


Figure 5.2: Two-Point gray-scale invariant. Left, the kernel combines a pixel with pixels in the neighborhood at scope $r$. Right, the kernel combines a region with neighboring regions at scope $r$ mapped to pixels by Gauss filtering.

filtering in discrete images is computed as convolution with a mask approximating the gauss function, the filter process can be computed as multiplication in fourier space at little additional cost.

## 5.2 Training and Classification

After extracting a vector of 8 to 14 invariant features for each pixel/voxel, training samples for each class are isolated and labeled. This is done manually by marking pixels in the original image/volume. Using the training samples, a model is learned by a support-vector machine (chapter (3.3.1)). The training parameters for the SVM are determined by cross-validation (3.30).
Once a model is learned, the feature-vectors of all pixels/voxels are classified against this model. The model can be retrained to cover more and more details by adding new samples. This interactive process is repeated until the model reaches a satisfying state. The model is then used to segment and classify objects in other datasets. But even then, the model can be expanded with new classes or further samples.

### 5.2.1 Interclass Distances

A nice side-effect of segmentation via pixel-wise classification is that the segmentation algorithm is not some black-box which makes selections hardly comprehensible for explicit examples.

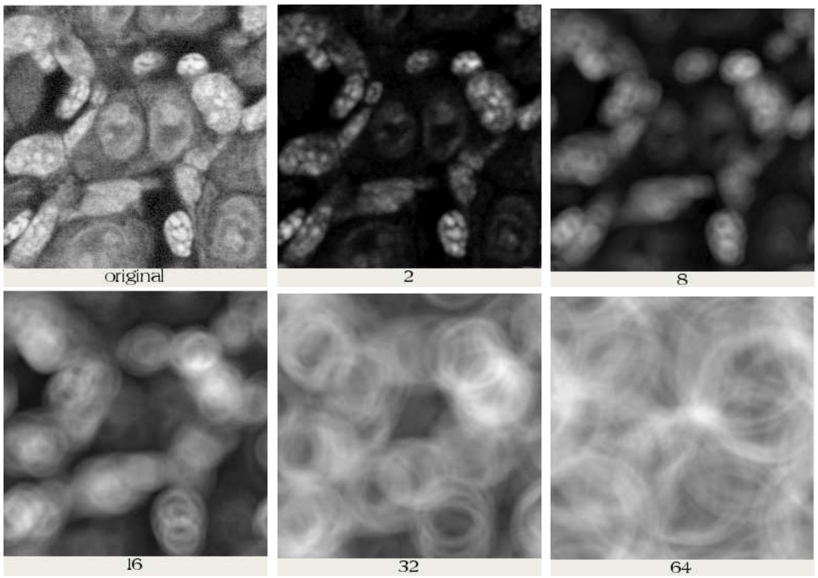



Figure 5.3: Showing the results of the convolution $((\mathcal{G} * B) * (\mathcal{G} * C))(x, y)$ before combining with the original pixel. The numbers indicate the size of the radius. The size of the used gauss is chosen at 2/3 of the radius.

Classifying with a SVM computes the confidence of class membership for every single feature-vector in terms of the $\alpha$-value (3.16). In One-Vs-Rest classification the confidence of one class is computed against all others, and interclass distances are detected by One-Vs-One multi-class classification. The confidence can be visualized as a gray-value pseudo colored image of $\alpha$:

## 5.3 Runtime Issues

Considering a typical LSM volume dataset with 100 slices of $256 \times 256$ pixels the suggested segmentation algorithm has to extract and classify roughly 7 million feature vectors. In such a scenario, computational complexity and runtime is an issue.
Basically there are two factors which have the greatest influence on complexity: feature extraction and classification. The actual training can be neglected, since only a portion of the overall vectors are used for training at all.

### 5.3.1 Extracting Feature-Vectors

The use of two-point invariants accelerated the feature calculation to a point where they became practical (4.6). Still, for each kernel function the fast fourier transformation of the entire volume has to be performed in both directions.

#### Speeding up the FFT

In this work the FFT is calculated by the FFTW library [Frigo1998]. The FFTW uses a prime factor decomposition algorithm [Rader1968] to find an optimal plan of factorizing the whole transformation problem. Thus, by padding the original volume to dimensions which are decomposable by preferably small prime factors, the FFT complexity if further reduced.

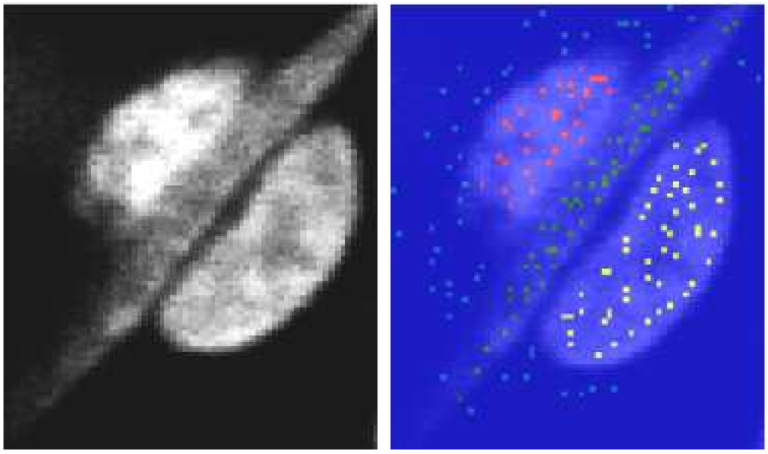


Figure 5.4: Left: section of the original gray-scale image or volume slice. Right: marked training samples for the different classes, blue: background, red: erythrocyte, green: endothel, yellow: pericyte.

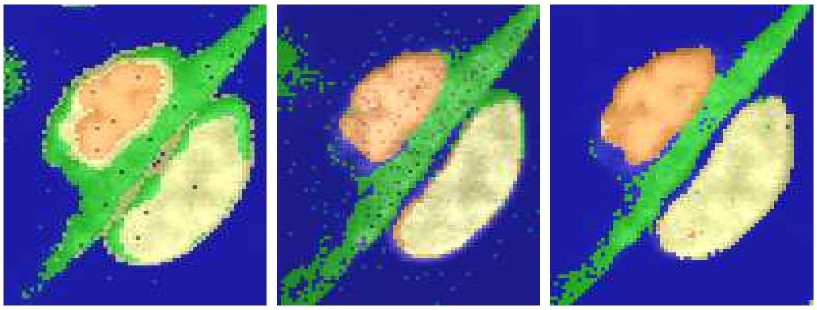


Figure 5.5: Shows the interactive training process. New samples are added from left to right. First: 37 training samples, second: 108. Third: 167. The black pixels indicate the support-vectors (SV).

Since the gauss filtering and convolution are cyclic transformations, the original volume has to be padded anyway by the size of the scope plus the gauss variance. Implemented in a lookup table, the padding is adjusted to a optimal prime decomposable size.

Another issue of FFT calculation is memory size. If memory is large enough, an average feature-vector of 12 features is computed for a $100 \times 256 \times 256$ dataset on a P4 2.8 GHz machine in about 2 minutes.

### 5.3.2 Classifying

A look at the classification algorithm of the SVM (3.14) reveals that for the classification of each feature-vector the kernel $k(x_1, x_2)$ has to be evaluated with all support-vectors (SV). This fact directly links computational complexity with the number of SV and the kernel. Both can be influenced.

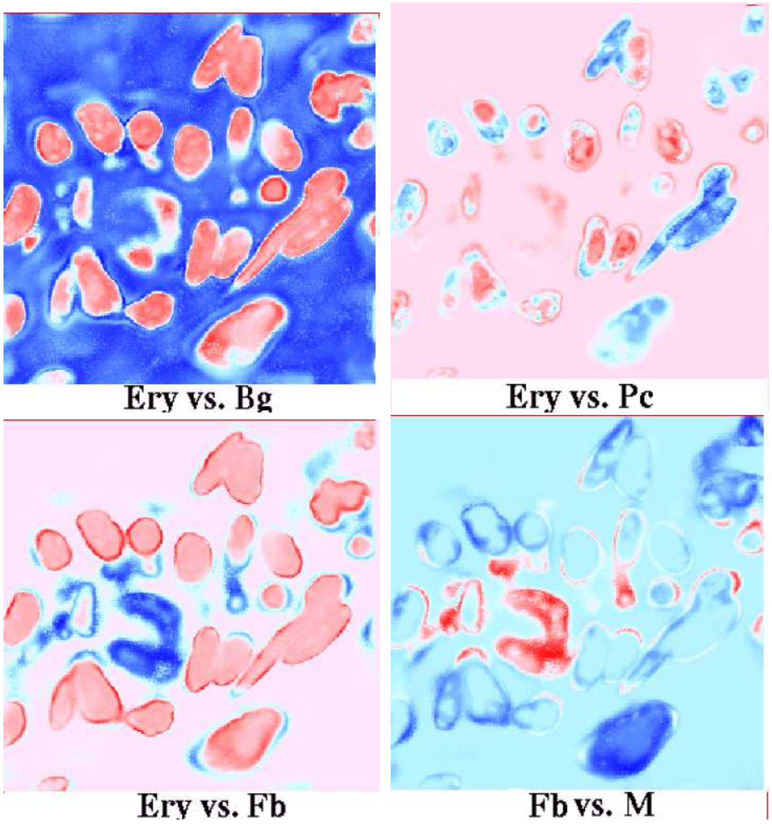



Figure 5.6: Visualizing the class confidence in One-Vs-One classification. Confidence in the first class is expressed as red-value , in the second class as blue-value. Background is mapped to white.

### Reducing the Number of SV

The number of SV can be reduced to some extent by increasing the penalty term $c$ (3.28) of the SVM. A higher penalty causes a shrinkage of the hyperplane margin, thus the number of vectors is the same. These vectors are the support-vectors.
Of course, this method underlies the tradeoff of quality and speed. A too excessive penalty will result in poor classification results.

### FastRBF Kernel

The choice of the kernel has the biggest impact on the classification complexity. The linear kernel,

$$K(\vec{x}, \vec{x'}) = \vec{x} \cdot \vec{x'}$$

identical to the inner product, is one of the fastest. But it cannot be assumed, that the classes are linear separable in any way. Thus, the RBF, or Gauss kernel,

$$K(\vec{x}, \vec{x'}) = exp(-\frac{\|\vec{x} - \vec{x'}\|^2}{2\sigma^2})$$

is the method of choice.
Unfortunately, the complexity of the RBF kernel is much higher. Especially the exponential term causes problems. It is quite expensive and $exp(-x)$ is close to zero for

large $x$ - nevertheless results below numerical accuracy are calculated at high costs.

To overcome this, the exponential function is approximated by a lookup table and linear interpolation. The new "fastRBF" kernel still holds (3.27) and is speeding up the classification process observably.

## 5.4 Connected Component Labeling

After classifying each pixel/voxel $x_i$ of the original dataset $I$, neighboring pixels/voxels with same label $y_i$ have to be combined to sets

$$S_n \subset I : \forall (x_i, y_i), (x_j, y_j) \in S_n : y_i = y_j \wedge \exists path(x_i, x_j)$$

representing classified objects.

Connected component labeling is a simple but effective recursive algorithm which achieves exactly that. Scanning over the dataset, once a non-background pixel is found, all neighboring pixels/voxels are called recursively. If they share the same label, they are combined to the same set and given a new label. This way no pixel is visited twice. In the end, all sets are copied into their own datasets - the segmentation

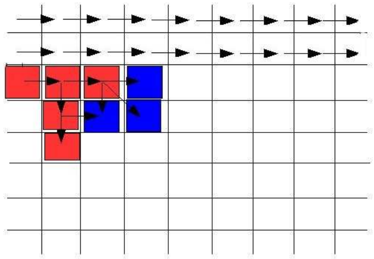


Figure 5.7: Connected component labeling illustrated in a 2D example.

is complete.

## 5.5 Exponential Kernel Functions

So far, very simple kernel functions $f_r(X) = f_a(X(\vec{0}) \cdot f_b(\vec{r})$ with scope $r$ and non-linear mappings $f_a(x), f_b(x) = x^2, x^3, x^4, x^5, \sqrt{x}$ were used. In general, this approach produced very satisfying results. But some data arrangements remain, where the results could be improved further. This phenomenon can be explained with the nature

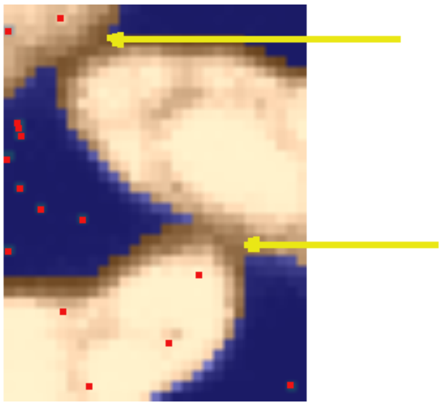


Figure 5.8: This selection shows that by use of simple kernel functions the algorithm tends to generate a "hull" around the actual objects, which causes two actually separated objects to merge. The red points indicate the training vectors in the selection

of gray-scale invariants - the extracted features are dominated by structures of higher pixel values. This effect is even amplified by kernel functions with high exponentials, spreading the gap between high a low valued pixels. Thus, higher valued pixels are somewhat "overshining" their neighborhood.
One possibility to compensate this effect could be to calculate some of the features on the inverse of the original data. This way, the critical regions are dominated by the

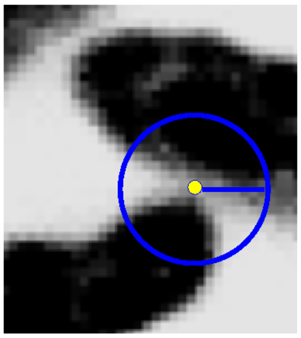


Figure 5.9: The inverse of the original data

background rather than by the objects.

Of course it is not very practical to keep an inverted dataset additional to the original, but the inversion can easily be included in a kernel function like

$$f_r(X) = x \cdot e^{-x'^2} \quad \text{exponential kernel function} \tag{5.3}$$

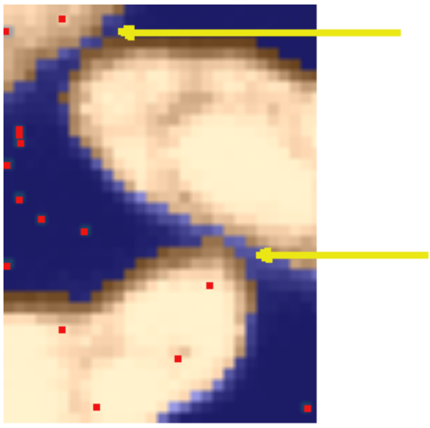


Figure 5.10: The same selection with exactly the same locations of training samples as in fig. (5.5), but with some features extracted using the exponential kernel function,the problem is solved.

## 5.6 Pseudo-Three-Point Invariants

There is another weakness coming up when adopting the two-point gray-scale invariants for pixelwise feature extraction. In contrast to object-wise calculated features, the invariants for single pixels are not summed up over the entire object. This is not only important to achieve translation invariance for objects, but it also restricts the degree of general invariance by encoding the neighborhood relations over the object. Directly speaking, the lack of this summation causes the point-wise features not only to be invariant towards rotation, but also towards any arbitrary permutation of its neighborhood.

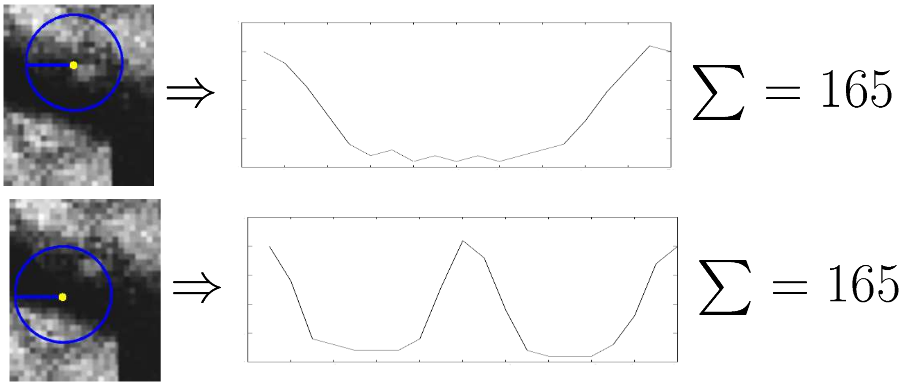



Figure 5.11: features calculated for two different pixels of varying classes. Values retrieved by the kernel function are interpreted as 1D function $f(x')$ with $\phi$ on the x-axis and the acording gray-value on the y-axis. The feature is calculated by summing over all values. Due to permutation invariance, the same feature value is calculated in both cases.

### 5.6.1 Three-Point Invariants

To overcome this problem, three-point invariants (4.10) could be used. By combining opposing values of the same kernel function, permutation variance can be obtained.

$$T(X) = f_r^{\phi}(X) \cdot f_r^{\phi+180}(X) \tag{5.4}$$

As discussed in (4.11), the drawback of three-point gray-scale invariants is their high

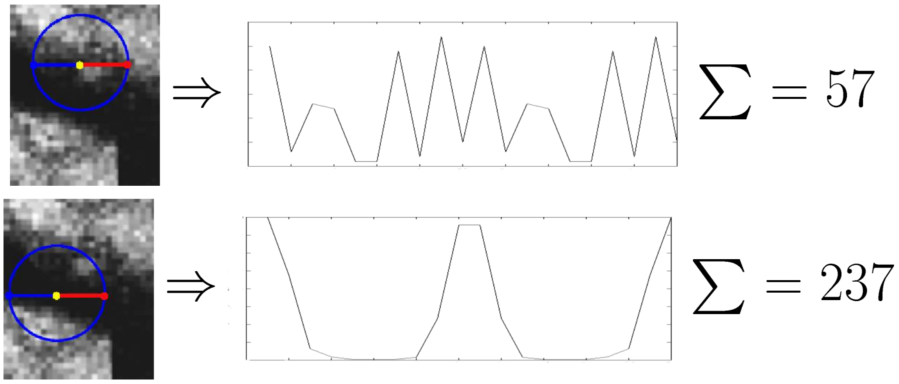



Figure 5.12: Simple three-point features solve the problem.

computational complexity. Thus, this solution is not practically applicable.

### 5.6.2 Fourier Descriptors

Seeking for a fast rotation invariant but permutation variant feature extraction sets the focus on the 1D representation of two-point invariants. A rotation in the original

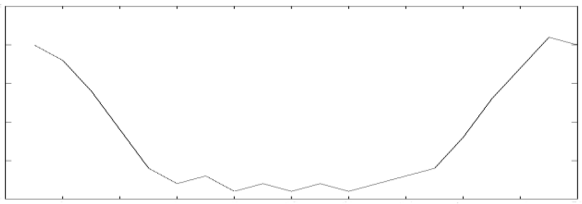


Figure 5.13: 1D representation of the values retrieved by the kernel function

dataset is equivalent to a translation in the 1D representation. Permutation has the same appearance in both representations.
A representation which fully covers the demanded characteristics is the fourier transformation. For the 1D data $f(x)$ the fourier transformation $FT(f(x))$ is variant towards permutation, but a rotation invariant descriptor $d(f(x))$is well known [Burkhardt1979]:

$$d(f(x)) = abs(FT(f(x))) \tag{5.5}$$

In terms of the FT, $f(x)$ is described as

$$f(x) = \sum_{k=-n}^{k=n} c_k \cdot e^{jk\omega x}$$

with $c_k = \frac{1}{T} \int f(x) \cdot e^{-jk\omega x} dx$. Taking the absolute value of $c_k$ eliminates the phase of the complex representation, thus causing rotation invariance. It has to be noted, that these features are not complete.

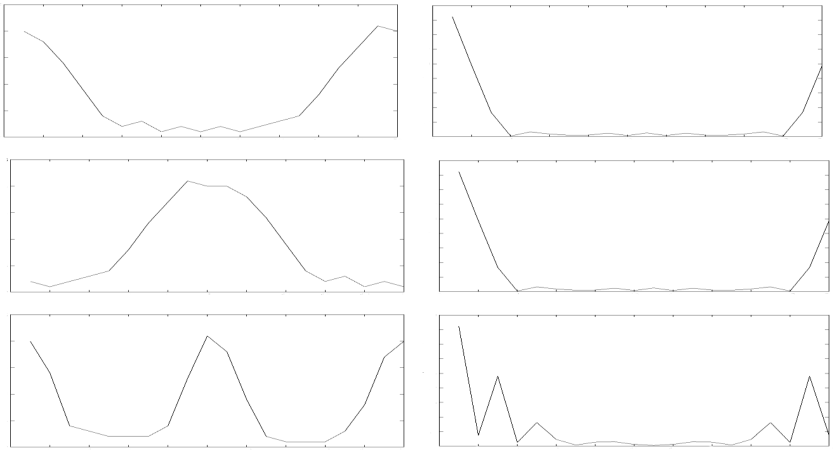


Figure 5.14: Starting from top, on the left: original signal, translation of the signal and permutation. The right colums shows the descriptors. Translation invariance and permutation variance of these descriptors is evident.

### 5.6.3 Implementing Pseudo Three-Point Invariants

The fourier descriptor approach covers all theoretical needs, but again the practical implementation causes problems. One question is, how to combine the FT with the convolution algorithm - and even more crucial: how to fourier transform the sphere surface in the 3D case?

Theses issues are solved by taking advantage of the FT's properties: hence,

$$e^{i\varphi} = cos(\varphi) + j \cdot sin(\varphi)$$

the complex FT representation can be broken down into a sine and a cosine part, which can be applied separately and then combined later on.

The sin and cos mappings are generally noted as points on the **unit circle**. Thus, returning to the 2D example, the fourier coefficients of $f(x)$ are retrieved by extending the convolution of the original $I$ from a single circle of constant value to several separate combinations of convolutions with a sine and a cosine circles at different frequencies.

$$C_s(x,y) = \begin{cases} \frac{y}{r} & \sqrt{x^2+y^2+z^2} = r \\ 0 & \text{otherwise} \end{cases}$$

$$C_c(x,y) = \begin{cases} \frac{x}{r} & \sqrt{x^2+y^2+z^2} = r \\ 0 & \text{otherwise} \end{cases}$$

To achieve rotation invariance, it is adequate to reduce this fourier approximation to the first coefficient which encodes the overall shape and orientation (lower frequencies). It is needless to note that this representation is not complete, but experiments showed an improvement of separability regarding two-point invariants

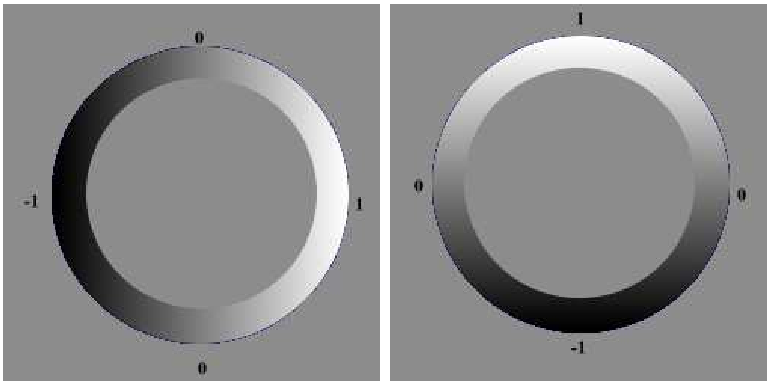



Figure 5.15: Sine $C_s$ and cosine $C_c$ circles.

Regarding the original two-point approach, the pseudo three-point invariants are retrieved by convolution with a circle, whose surface consists of the complex combination of the sine and cosine mappings.

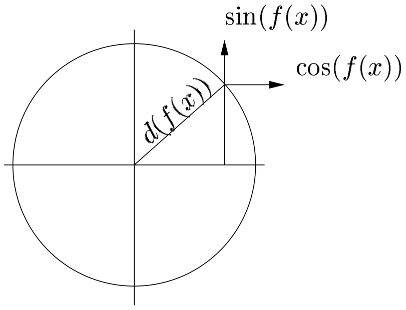



Figure 5.16: Complex combination of the sine and cosine transformations.

$$T(x,y) = A(x,y) \cdot \sqrt{(B * C_s)^2(x,y) + (B * C_c)^2(x,y)} \tag{5.6}$$

### Interpretation of Pseudo Three-Point Invariants

Due to the permutation variance property, the pseudo three-point invariants appear as a sort of edge-detector, which is a nice side effect for cell segmentation.

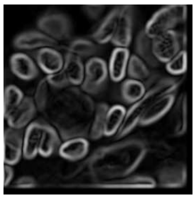
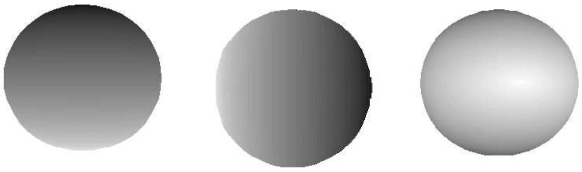


Figure 5.17: Edge detection with a pseudo three-point invariant kernel, scope 4

### 3D Pseudo Three-Point Invariants

For the 3D case, three spheres $S_x, S_y, S_z$, each with shadings from 1 to $-1$ in $x, y$, or $z$ direction are used for convolution.
These spheres are given by the general sphere-functions of the first order:

$$p_y = (\frac{3}{4\pi})^{\frac{1}{2}} \sin\theta \sin\phi, p_x = (\frac{3}{4\pi})^{\frac{1}{2}} \sin\theta \cos\phi, p_y = (\frac{3}{4\pi})^{\frac{1}{2}} \cos\theta \tag{5.7}$$



$$T(x,y,z) = A(x,y,z) \cdot \sqrt{(B * S_x)^2(x,y,z) + (B * S_y)^2(x,y,z) + (B * S_z)^2(x,y,z)} \tag{5.8}$$

### Proving the 3D Case

Proving invariance for the 3D case is beyond the scope of this work. In principle the geometric interpretation can be extended to the three 3D case, but this is still not trivial. [Danielson2001] gives an idea of the problem and a possible solution. For now, the 3D prove is replaced by intuition, leaving the actual prove for further work.

# Chapter 6

# Software Implementation

The algorithms described in the previous chapters are implemented in the software tool `3DSegment`. `3DSegment` provides a graphical interface which integrates the modularized parts of feature extraction, training, classification, segmentation and visualization combined with some enhancing tools.
`3DSegment` is implemented in C++, compiled with GCC 3.3.3 for posix conforming platforms (Linux/Unix). A porting to other platforms should be possible without major effort, but was not tested.

## 6.1 Libraries

`3DSegment` relies on a series of external libraries, which are briefly introduced.

### NetCDF

The network common data form (NetCDF) [Rew1997] is an open standard for self describing portable array based data. NetCDF provides fast access to arrays of many basic data types and different dimensionality. All data, from 3D volumes to SVM models is stored in NetCDF format.

### Blitz-Arrays

The Blitz-Library [Veldhuizen] provides fast and memory efficient arrays for real and complex data optimized for numerical applications. With blitz-arrays C++ reaches the speed level of `fortran` implementations. The volume and feature data is stored in Blitz Arrays.

### LibSVMtl

LibSVMtl [SVMtl] is a highly customizable C++ Support Vector Machine library based on Chi-Jen Lin's LIBSVM which is faster and easier to integrate than the original.

### FFTW

FFTW [Frigo1998] is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data. The FFTW is used for the convolution part of feature extraction.

### VTK

The Visualization ToolKit (VTK) [VTK] is an open source, freely available software system for 3D computer graphics, image processing, and visualization. VTK is used for the 3D reconstruction and visualization of the volume data.

### QT

Qt [QT] is a C++ toolkit for multi-platform GUI application development. The graphical interface of `3DSegment` is implemented with QT.

## 6.2 Software Description

`3DSegment` offers a variety of possible user interactions with 2D or volume data read from NetCDF files. The main window gives three views of the data: the 3D reconstruction is used for orientation purpose only - it requires strong filtering and is only approximating the original data.
The red plane indicates the position of the xy-slice of original data shown in the frame below the 3D reconstruction. The blue plane represents a yz-slice.

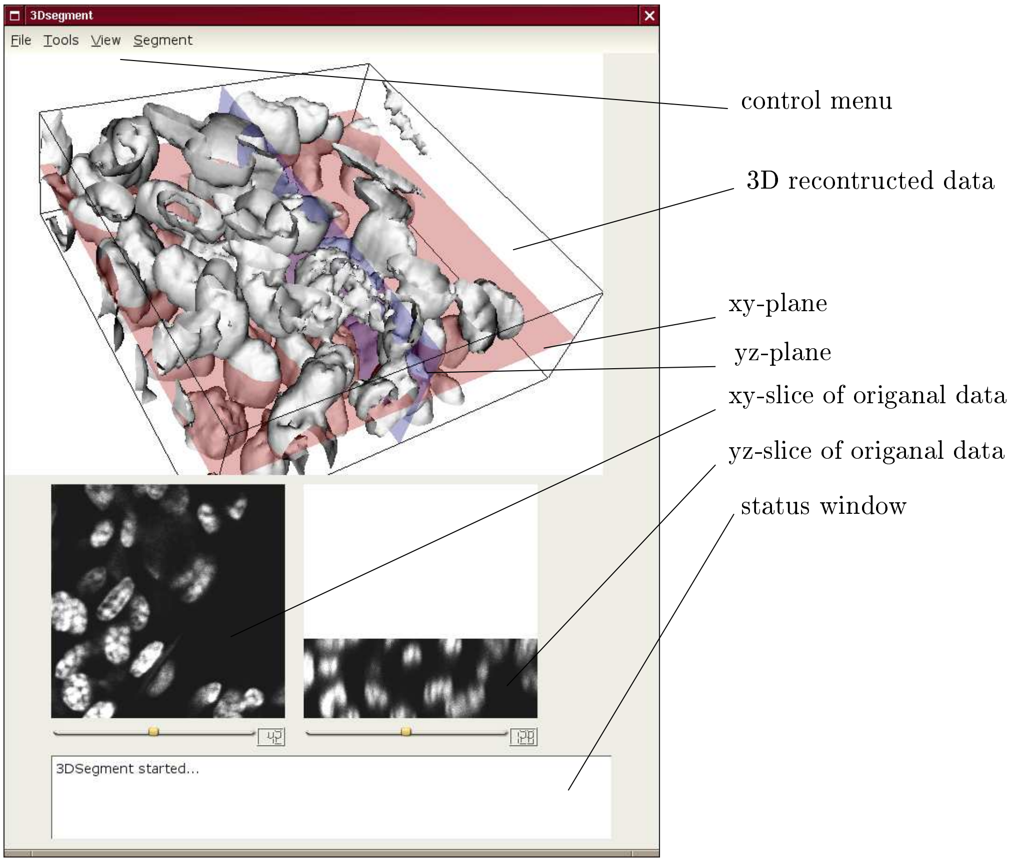



Figure 6.1: Main window of 3Dsegment

### 6.2.1 Tools

Several Tools for preprocessing and visualization are implemented in `3DSegment`. **2D Control** provides three sliders for the manipulation of slice imaging. **Filters** can be

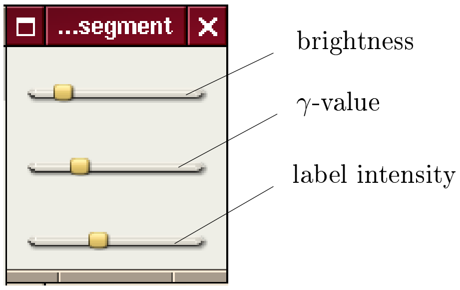



Figure 6.2: The label intensity regulates the blending of label information (colors) over the original gray-scale data

applied to the data and the threshold for 3D reconstruction can be set.

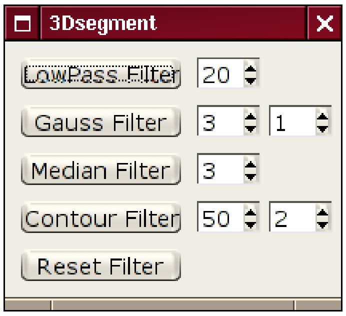



Figure 6.3: Several filtering options are provided

### 6.2.2 Views

Besides the three views in the main window, several other data illustrations are available: the xy and yz slices can be viewed at a 2:1 scale, all channels are imaged in the channel preview, the fourier transformation and even a 3D reconstruction for 3D-goggles.


Figure 6.4: Viewing all channels at a time.

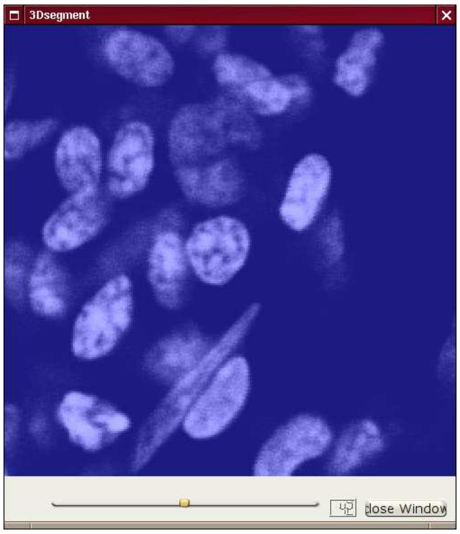



Figure 6.5: xy-slice scaled at 2:1.

## 6.3 Extracting Features

In order to extract the invariant two-point and pseudo three-point gray-scale feature vectors, the parameters like channels, scope, non-linear mapping and kernel type have to be chosen for each feature.
So far, this still has to done manually. The feature plot (fig. 6.7), which plots the values of training samples for each feature against all other features gives an indication which parameters might be useful. Clustering in the plot guarantees separability by the SVM - the reverse is of cause not true.

Another tool practicable for feature selection is the feature range plot. It shows the range of training sample values for each feature. The classes are plotted separately, the yellow line states the median and the two green lines identify the $\frac{1}{3}$ and $\frac{2}{3}$ confidence interval.

Finally it is possible to view the fourier transform for each feature of the entire dataset.

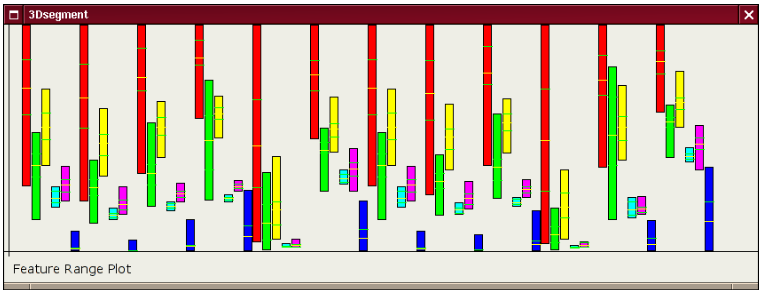



Figure 6.6: feature range plot.

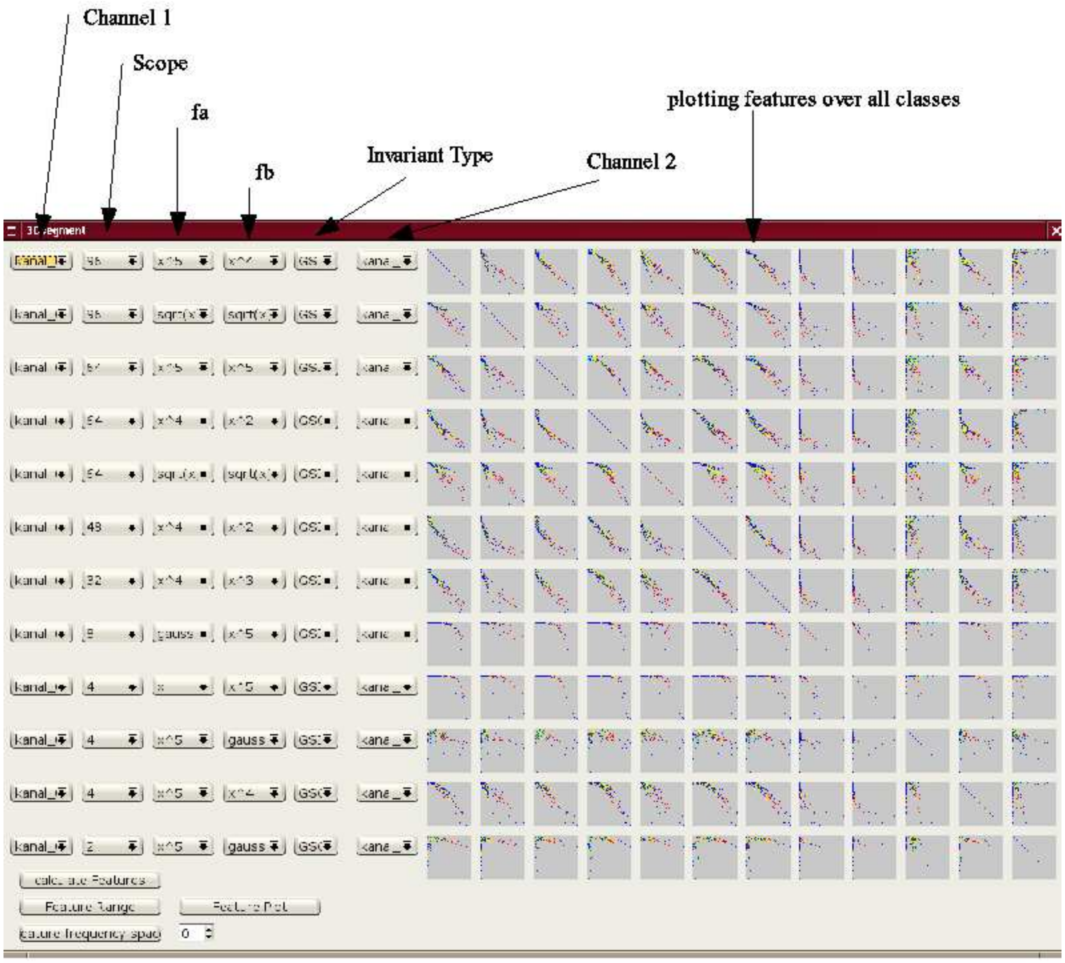



Figure 6.7: Feature extraction: selecting kernel type, non-linear functions, scope and channels.

## 6.4 Training

After the parameters have been set and features are extracted the training of a model can begin. First, training samples are labeled for each class.
The different class labels are set in the training dialog box. The labels are illustrated in different colors.

Models can be saved to and loaded from files. After the training process the number of support-vectors is displayed in the verbose window.

```
number of feature vectors:  307
training SVM ...
Number of classes:  6
Number of SV: 48
saving SV...
...  done!
```

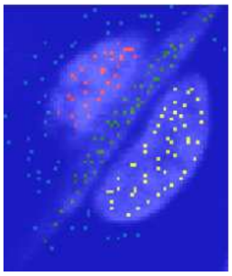


Figure 6.8: labeling traing sample by clicking into the the xy or yz sclice images.

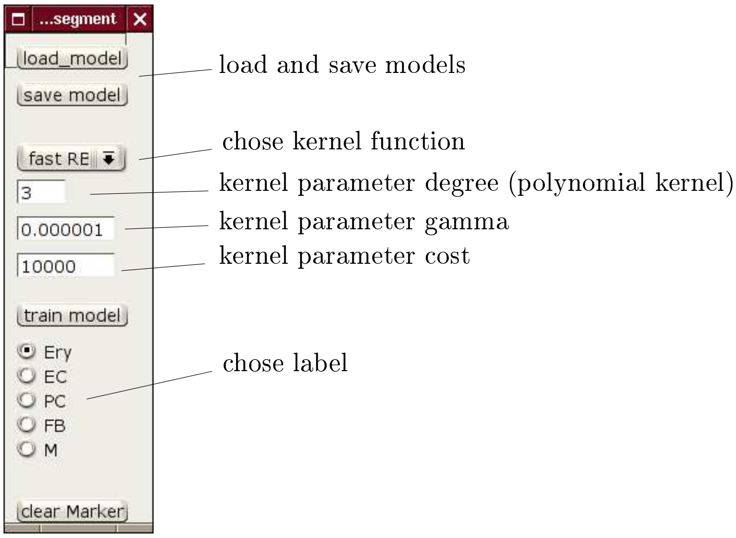



Figure 6.9: The training dialog.

## 6.5 Classification

Classification of all voxels is performed directly after training or loading a model. As mentioned in the chapter before, this is the most time consuming step. The classification progress is displayed in the verbose window:

```
classifying SVM ...2 3 1e-06 10000
...done:  455158 / 623539 / 413595 / 114439 / 284606 / 3678923 / 0
```

The output indicates the number of pixels mapped to each class. The class labels are displayed in different colors, training samples are indicated by a brighter coloring. Those samples which are actually supporting the model (support-vectors) are colored in black.
In an interactive procedure, new training samples can be selected and the model can be trained again. This can be repeated until the model fits the desired needs.

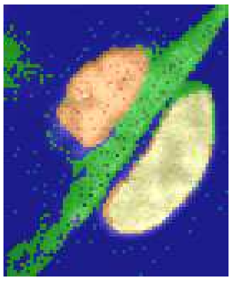


Figure 6.10: Classification results after the first run, support-vectors are colored in black, other training samples in a brighter coloration.

## 6.6 Segmentation

After fininshing the process of model training and classification, the labeled voxels have to be assigned to sets of connected objects. This is done by connected component labeling.
`3DSegment` markes objects with a bounding box in the 3D illustation, and with a

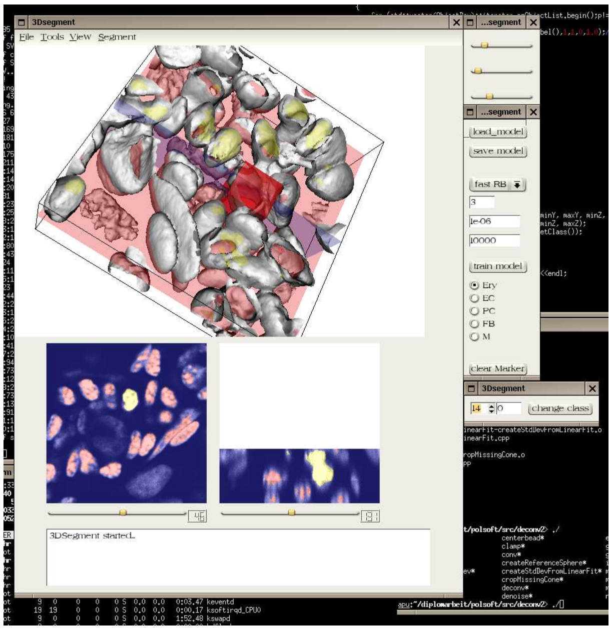


Figure 6.11: Main window of 3Dsegment showing segmented objects

red/yellow coloring in the 2D slices. Using the "Label objects" option, one is able to chose the object marked and an individual label can be asigned to objects.
The segmented objects can be saved to individual NetNDF files. The tool `polvi` shows a slice view of all channels for these files. `nc2vtkvis` reconstructs a 3D model of the segmented data.

# Chapter 7

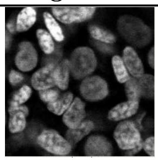
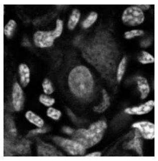
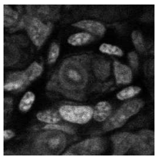
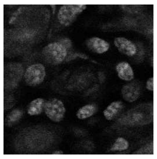
# Experiments

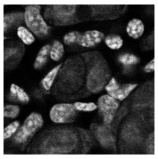
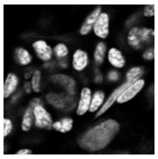
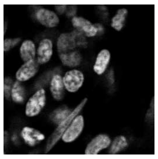
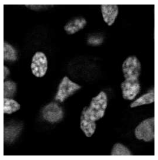
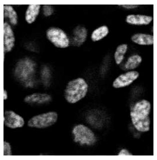
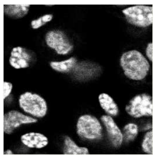
This final chapter shows some experimental results of previously introduced methods and algorithms. LSM data from (chapter 1.1) was used for the task of segmentation and classification in one step and cells from (chaper 1.2) were only segmented.

## 7.1 Data Sets

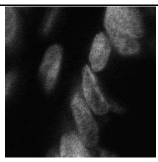
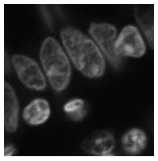
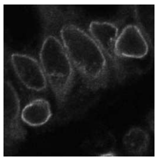
This table gives an overview of the used data sets.

### 7.1.1 Angiogenesis

| image | name | date | tinct | channel 1 | channel 2 |
|---|---|---|---|---|---|
|  | HKCAM.01.nc | 13/05/04 | YoPro | 40x zoom, 8x sampling | 40x zoom , 8x sampling |
|  | HKCAM.02.nc | 13/05/04 | YoPro | 40x zoom, 8x sampling | 40x zoom , 8x sampling |
|  | HKCAM.04.nc | 13/05/04 | YoPro | 40x zoom, 8x sampling | 40x zoom , 8x sampling |
|  | HKCAM.09.nc | 13/05/04 | YoPro | 40x zoom, 8x sampling | 40x zoom , 8x sampling |

| | | | | | |
|---|---|---|---|---|---|
|  | HKCAM.12.nc | 13/05/04 | YoPro | 40x zoom, 8x sampling | 40x zoom , 8x sampling |
|  | HKCAM.41.nc | 13/05/04 | YoPro | 40x zoom, 8x sampling | 40x zoom , 8x sampling |
|  | HKCAM.42.nc | 13/05/04 | YoPro | 40x zoom, 8x sampling | 40x zoom , 8x sampling |
|  | HKCAM.51.nc | 27/09/04 | YoPro | 40x zoom, 8x sampling | 40x zoom , 8x sampling |
|  | HKCAM.52.nc | 27/09/04 | YoPro | 40x zoom, 8x sampling | 40x zoom , 8x sampling |
|  | HKCAM.54.nc | 27/09/04 | YoPro | 40x zoom, 8x sampling | 40x zoom , 8x sampling |

### 7.1.2 Colon Tissue

| **name** | **date** | **tinct** | **channel 1** | **channel 2** |
|---|---|---|---|---|
| ColonNormal01.nc | 27/10/04 | ToPro |  | non |
| ColonLamin05.nc | 09/11/04 | ToPro and Lamin |  |  |

## 7.2 Measuring Experimental Results

The nature of the applied algorithms makes it difficult to quantify segmentation and classification results, since it is not possible to label each single vector of the datasets manually.
The only vectors with known labels are the training samples. Thus, a quantification for this subset can be reached by cross-validation or other model-selection techniques. These results give an indication on the artifact ratio.

A way of directly quantifying the separability of the extracted features is to take the confidence (5.2.1) of OneVsOne or OneVsRest classification into account. The standard deviation of the classes in the histogram derived from the $\alpha$-value gray-scale images combined with the class distance provides a good measure.

More indirect quantifying methods for segmentation quality include observations like the number of unsegmented cells, the amount of falsely classified cells or the appearance of cell clusters.

A combination of all three methods gives a measure for the quality of the segmentation and classification results.

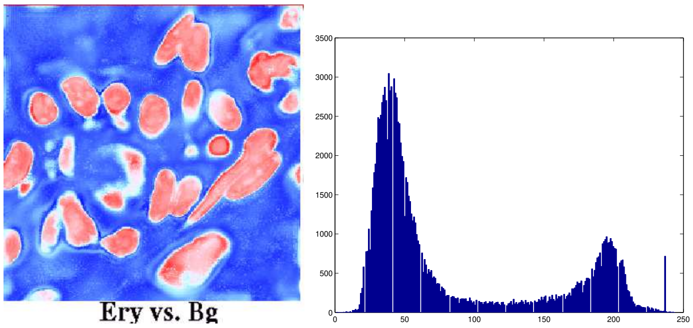



Figure 7.1: Left: $\alpha$-value gray-scale image of the One-Vs-One classification of Ery and background. Right: the histogram shows good separability of the two classes.

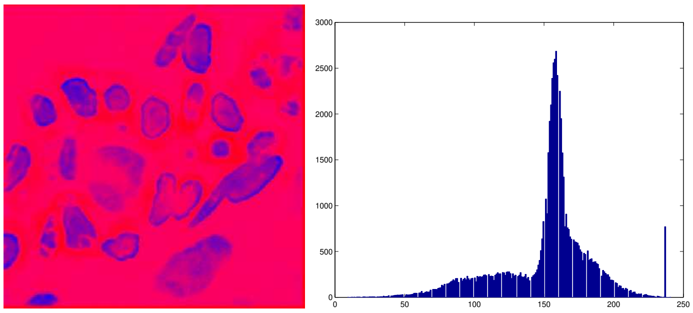



Figure 7.2: Left: $\alpha$-value gray-scale image of the One-Vs-Rest classification of the background against all other classes. Right: the histogram shows one peak with high confidence.

## Experiment 1

training and testing a model for probe HKCAM.41.nc

### SVM Parameter

| **kernel type** | **gamma** | **cost** |
|---|---|---|
| FastRBF | 0.000001 | 100000 |
| **train-vectors** | **SV** | **cross-validation**, $k = 50$ |
| 1756 | 277 | 96.81% |

### Feature Parameter

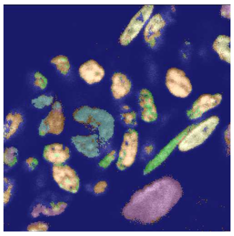
| # | **channel1** | **scope** | $f_a$ | $f_b$ | **invariant type** | **channel2** |
|---|---|---|---|---|---|---|
| 1 | 0 | 96 | $x^5$ | $x^4$ | 2-point | 1 |
| 2 | 0 | 96 | $\sqrt{x}$ | $\sqrt{x}$ | 2-point | 0 |
| 3 | 0 | 64 | $x^5$ | $x^5$ | 2-point | 0 |
| 4 | 0 | 64 | $x^4$ | $x^2$ | p3-point | 0 |
| 5 | 0 | 64 | $\sqrt{x}$ | $\sqrt{x}$ | 2-point | 0 |
| 6 | 0 | 48 | $x^4$ | $x^2$ | 2-point | 0 |
| 7 | 0 | 32 | $x^4$ | $x^2$ | 2-point | 0 |
| 8 | 0 | 16 | gauss | $x^5$ | 2-point | 0 |
| 9 | 0 | 4 | $x$ | $x^5$ | 2-point | 0 |
| 10 | 0 | 4 | $x^5$ | gauss | 2-point | 0 |
| 11 | 0 | 4 | $x^5$ | $x^4$ | p3-point | 0 |
| 12 | 0 | 2 | $x^5$ | gauss | p3-point | 1 |

### Results

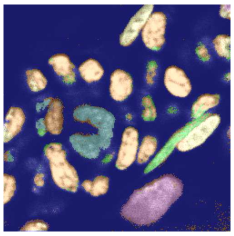
| **# of missed cells** | **# false classified cells** | **# partialy classified cells (> 10% missing)** | **# clusters** |
|---|---|---|---|
| 0 | 0 | 2 | 3 |

### Some Result Slices

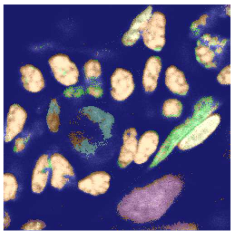
slice 29



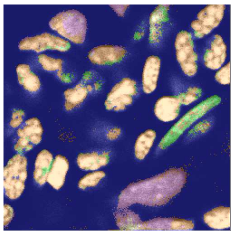
slice 56



slice 68

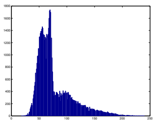


slice 87

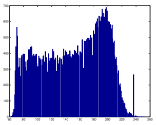


### Feature Separability One-Vs-Rest

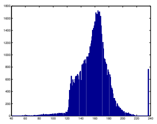
Ery



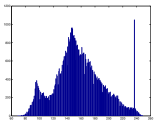
EC



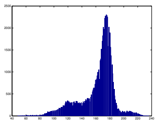
PC



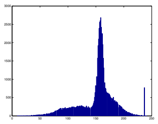
FB



MΦ



BG



### Experiment 2

training and testing a model for probe HKCAM.42.nc

**SVM Parameter**

| **kernel type** | **gamma** | **cost** |
|---|---|---|
| FastRBF | 0.000001 | 100000 |
| **train-vectors** | **SV** | **cross-validation**, $k = 50$ |
| 1303 | 304 | 95.01% |

**Feature Parameter**

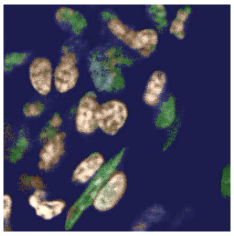
| # | **channel1** | **scope** | $f_a$ | $f_b$ | **invariant type** | **channel2** |
|---|---|---|---|---|---|---|
| 1 | 0 | 96 | $x^5$ | $x^4$ | 2-point | 1 |
| 2 | 0 | 96 | $\sqrt{x}$ | $\sqrt{x}$ | 2-point | 0 |
| 3 | 0 | 64 | $x^5$ | $x^5$ | 2-point | 0 |
| 4 | 0 | 64 | $x^4$ | $x^2$ | p3-point | 0 |
| 5 | 0 | 64 | $\sqrt{x}$ | $\sqrt{x}$ | 2-point | 0 |
| 6 | 0 | 48 | $x^4$ | $x^2$ | 2-point | 0 |
| 7 | 0 | 32 | $x^4$ | $x^2$ | 2-point | 0 |
| 8 | 0 | 16 | gauss | $x^5$ | 2-point | 0 |
| 9 | 0 | 4 | $x$ | $x^5$ | 2-point | 0 |
| 10 | 0 | 4 | $x^5$ | gauss | 2-point | 0 |
| 11 | 0 | 4 | $x^5$ | $x^4$ | p3-point | 0 |
| 12 | 0 | 2 | $x^5$ | gauss | p3-point | 1 |

**Results**

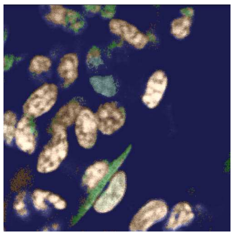
| **# of missed cells** | **# false classified cells** | **# partialy classified cells (> 10% missing)** | **# clusters** |
|---|---|---|---|
| 0 | 0 | 5 | 10 |

**Some Result Slices**

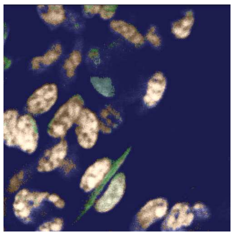
slice 29



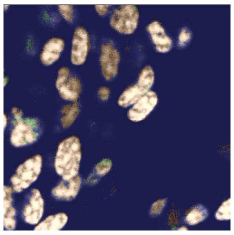
slice 56



slice 68

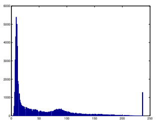


slice 87

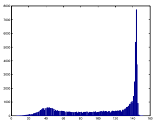


**Feature Separability One-Vs-One**

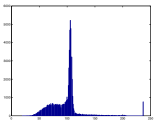
Ery-Vs-BG



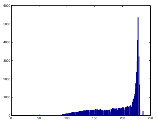
Ery-Vs-Ec



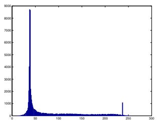
Ery-Vs-Pc



Ec-Vs-Pc



Pc-Vs-Bg



**Experiment 3**

training and testing a model for probe HKCAM.51.nc

**SVM Parameter**

| **kernel type** | **gamma** | **cost** |
|---|---|---|
| FastRBF | 0.000001 | 100000 |
| **train-vectors** | **SV** | **cross-validation**, $k = 50$ |
| 973 | 75 | 98.97% |

**Feature Parameter**

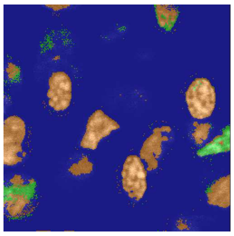
| # | **channel1** | **scope** | $f_a$ | $f_b$ | **invariant type** | **channel2** |
|---|---|---|---|---|---|---|
| 1 | 0 | 96 | $x^5$ | $x$ | 2-point | 0 |
| 2 | 0 | 96 | $\sqrt{x}$ | $\sqrt{x}$ | 2-point | 0 |
| 3 | 0 | 64 | $x^5$ | $x^2$ | 2-point | 0 |
| 4 | 0 | 64 | $\sqrt{x}$ | $\sqrt{x}$ | p3-point | 0 |
| 5 | 0 | 64 | $x^5$ | $x^5$ | 2-point | 0 |
| 6 | 0 | 48 | $x^5$ | $x$ | 2-point | 0 |
| 7 | 0 | 32 | $x^2$ | $x^4$ | 2-point | 0 |
| 8 | 0 | 8 | gauss | $x$ | 2-point | 0 |
| 9 | 0 | 8 | $x$ | gauss | 2-point | 0 |
| 10 | 0 | 4 | $x^5$ | $x^5$ | 2-point | 0 |
| 11 | 0 | 4 | $x^5$ | $x^5$ | p3-point | 0 |
| 12 | 0 | 2 | gauss | $x$ | 2-point | 0 |

**Results**

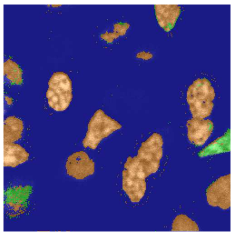
| **# of missed cells** | **# false classified cells** | **# partialy classified cells (> 10% missing)** | **# clusters** |
|---|---|---|---|
| 0 | 0 | 0 | 3 |

**Some Result Slices**

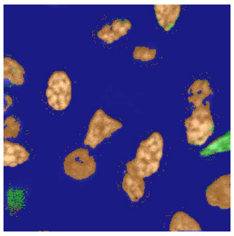
slice 13



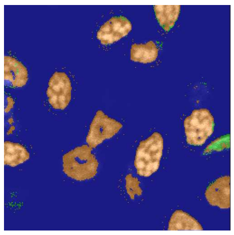
slice 18



slice 22

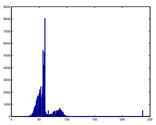


slice 27

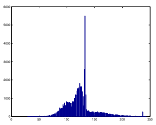


**Feature Separability One-Vs-One**

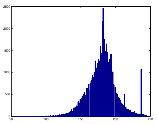
Ery-Vs-BG



Ery-Vs-Ec



Ec-Vs-Bg



### Experiment 4

Segmentation of ColonLamin05. This Experiment shows how overlapping cells can be segmented. The second channel shows a Lamin stain, which only markers the surface of nuclei. This information is used to train a bound around each cell, making segmentation possible.

| **kernel type** | **gamma** | **cost** |
|---|---|---|
| FastRBF | 0.000001 | 100000 |
| **train-vectors** | **SV** | **cross-validation**, $k = 50$ |
| 481 | 151 | 95.63% |

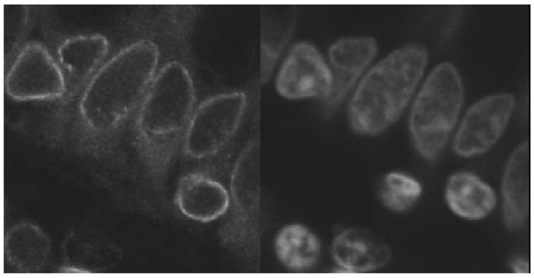


Figure 7.3: Left: Topro stain. Right: Lamin stain.

**Feature Parameter**

| # | **channel1** | **scope** | $f_a$ | $f_b$ | **invariant type** | **channel2** |
|---|---|---|---|---|---|---|
| 1 | 0 | 96 | $x^5$ | $x^2$ | 2-point | 1 |
| 2 | 0 | 96 | $\sqrt{x}$ | $x$ | 2-point | 0 |
| 3 | 0 | 64 | $x^5$ | $x^5$ | 2-point | 0 |
| 4 | 0 | 64 | $\sqrt{x}$ | $\sqrt{x}$ | p3-point | 0 |
| 5 | 0 | 4 | $x^5$ | $x$ | 2-point | 0 |
| 6 | 0 | 4 | $x^5$ | $x^5$ | p3-point | 0 |
| 7 | 1 | 96 | $x^5$ | $x^2$ | 2-point | 1 |
| 8 | 1 | 64 | gauss | $x$ | 2-point | 1 |
| 9 | 1 | 4 | $x^5$ | $x$ | 2-point | 1 |
| 10 | 1 | 96 | $x^4$ | $x^3$ | 2-point | 0 |
| 11 | 1 | 64 | $x^5$ | $x^2$ | 2-point | 0 |
| 12 | 1 | 4 | gauss | $x$ | 2-point | 1 |

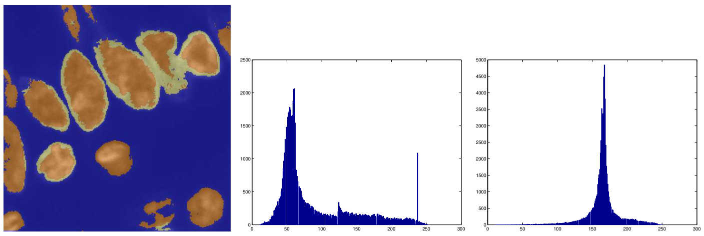


Figure 7.4: From left to right: segmentation results, cell-Vs-Bg and cell-Vs-Bound .

### Experiment 5

Segmenting and classification of probe HKCAM.52.nc using the model trained in experiment 3 on HKCAM.51.nc

#### SVM Parameter

are the same as in experiment 3, of course.

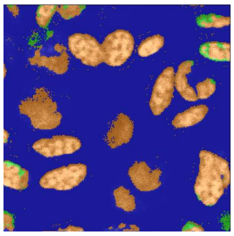
#### Feature Parameter

are the same as in experiment 3, of course.

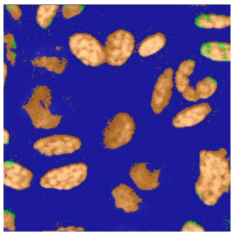
#### Results

| **# of missed cells** | **# false classified cells** | **# partialy classified cells (> 10% missing)** | **# clusters** |
|---|---|---|---|
| 0 | 0 | 3 | 9 |

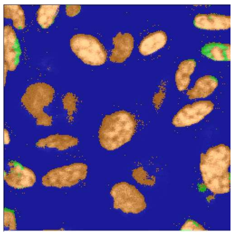
slice 11



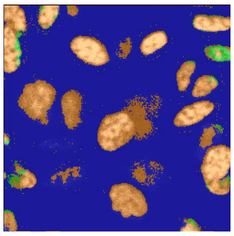
slice 15



slice 23

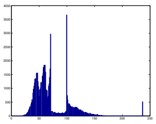


slice 29

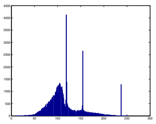


#### Feature Separability One-Vs-One

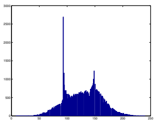
Ery-Vs-BG



Ery-Vs-Ec



Ec-Vs-Bg



#### Interpretation

Since only few of the provided datasets were recorded with the same parameters, this type of cross evaluation experiment could not be conducted as often as desired. HKCAM.52.nc contains hardly any cells other than Ery, so that the result shows only segmentation capabilities. This works quite well, although outliers occur.

### Experiment 6

Segmenting and classification of probe HKCAM.53.nc using the model trained in experiment 3 on HKCAM.51.nc

#### SVM Parameter

are the same as in experiment 3, of course.

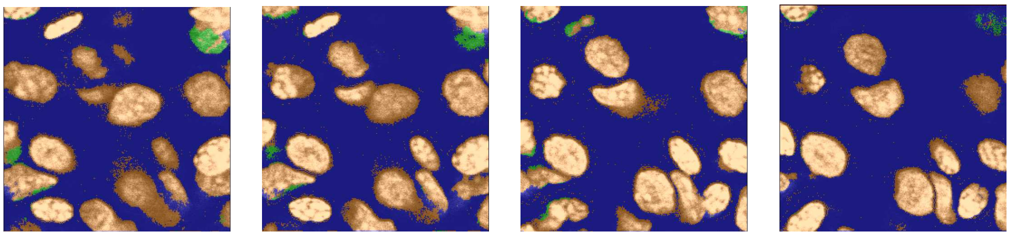
#### Feature Parameter

are the same as in experiment 3, of course.

#### Results

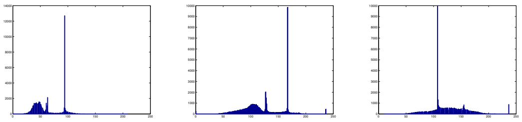
| # of missed cells | # false classified cells | # partialy classified cells (> 10% missing) | # clusters |
|---|---|---|---|
| 0 | 0 | 5 | 8 |

slice 12 slice 17 slice 27 slice 37



#### Feature Separability One-Vs-One

Ery-Vs-BG Ery-Vs-Ec Ec-Vs-Bg



# Chapter 8

# Conclusion

This work presented a new approach towards segmentation and classification of objects in a structured environment, like cells in tissue probes. Therefore a self-learning segmentation algorithm was introduced which is capable of performing segmentation and classification in one step.
The algorithm of a gray-scale invariant object representation was adopted to perform a pixel-wise extraction of rotation invariant features which were then used for training and classifying a pixel bases segmentation model.

The extension of the original two-point invariants with exponential kernel functions and then to pseudo three-point invariants lead to a decisive increase in separability, thus segmentation performance.

The results of the few experiments with 3D volume data of LSM probes from different tissues and cell types indicate the high potential of this new approach for segmentation as well as for classification. Further experiments with more labeled datasets are needed to verify the first results.

## 8.1 Outlook

The algorithms introduced in this work as well as the appendant implementation form a good basis for further research and applications. Yet, some issues have to be addressed in order to achieve progressing results:

First, calculation complexity is still the major drawback of the method. Further classification speedup could be achieved by a reduction of the number of support-vectors in the underlying model [Schölkopf2002]. Another approach would be to reduce the number of feature-vectors which have to be classified. This could be achieved by a preprocessing step which eliminates all voxels belonging with high probability to the background.

Secondly, the number of parameters which have to be chosen in order to receive good results is way to large for a practical application. The SVM parameters can be automatically chosen via model selection. For the feature parameters (non-linear mapping, scope, channels, invariant type) an automatic selection could be performed by gray value-cooccurrence-distribution (GVCD) computing optimal separating features.

Finally, it might be advantageous to extend the pixel-wise feature calculation to larger regions. Starting with a cheap over-segmenting preprocessing step, like watershed, in combination with a gray-scale invariant based learning algorithm which is trained to merge regions, could lead to a fast an powerful segmentation method.

# Bibliography


[Arlt2003] Arlt, C., Filmbasierte automatische Detektion und fraktale Analyse des Mikrozirkulationsnetzwerkes der Chorioallantois-Membran des Hühnereis, Dissertation TH Aachen, 2003.

[Baumgartner2003] Baumgartner, R., Erkennung von verschiedenen Zellkernen in konfokalen Laser Scanning Aufnahmen mittels grauwertbasierter Invarianten. Studienarbeit LMB, 2003.

[Burkhardt2001] Burkhardt, H., Siggelkow, S., Invariant features in pattern recognition - fundamentals and applications. In Kotropoulos, C., Pitas, I., editors, Nonlinear Model-Based Image/Video Processing ans Analysis, pages 269-3007. John Wiley & Sons,2001.

[Burkhardt1979] Transformationen zur lageinvarianten Merkmalgewinnung.Habilitationsschrift, Universität Karlsruhe, 1979.Fortschrittbericht (Reihe 10, Nr. 7) der VDI-Zeitschriften, VDI-Verlag.

[Carmeliet2003] Carmeleit, P., Angiogenesis in health and disease, Nature Medicine, 2003, 9(6):653-60.

[Chapelle2000] Chapelle, O. and Vapnik,V. ,(2000) Model Selection for Support Vector Machines.NIPS 12,pp.230-236.

[Cremer2003] Cremer, M., et al, Inheritance of gene density-related higher order chromatin arrangements in normal and tumor cell nuclei. The Journal of Cell Biology, 2003:809-820.

[CortesVapnik95] Cortes, C. and Vapnik, V. (1995) Support-Vector Networks. Machine Learning Journal, 20:273-297.

[Cossman] P.H. Cossmann, P.S. Eggl, H. Kurz:Three-dimensional analysis of DNA replication foci: a comparative study on species and cell type in situ. Histochem Cell Bio (2000) 113:195-205

[Danielson2001] Danielson, P., Lin, Q., A New Shape Space for Second Order 3D-Variations. IWVF4, LNCS 2059, pp. 145-154. 2001.

[DudaHartStrork] Duda, R., Hart, P., Stork, D., Pattern Classification, 2000, John Wiley & Sons.

[Faller] Faller, A., Der Körper des Menschen. Thieme 1999.

[Fehr2003] Fehr, J. Algorithmen zur Modellselektion. Studienarbeit am LMB, 2003. http://www.informatik.uni-freiburg.de/ fehr/

[Frigo1998] Frigo,M., Johnson, S., FFTW: AN ADAPTIVE SOFTWARE ARCHITECTURE FOR THE FFT, 1998 ICASSP conference proceedings (vol. 3, pp. 1381-1384).

[GonzalesWoods] Gonzales, R., Woods, R., Digital Images Processing. Prentice Hall, 2001.

[Hagedorn2004] Hagedorn, M. et. al, FEGF Coordinates Interaction of Pericytes and Endothelial Cells During Vasculogenesis and Experimental ngiogenesis, Developmental Dynamics 230:23-33, 2004.

[Hecht94] Hecht, E., (1994) Optik, Addison-Wesley.

[Jähne2002] Jähne, B. Digitale Bildverarbeitung. Springer.

[Kapitza94] Kapitza, H. (1994) Mikroskopieren von Anfang an. Carl Zeiss Jena GmbH.

[Kurz2003] Kurz, H., Burri, P., Djonov, V., Angiogenesis and Vascular Remodeling by Intussusception: From Form to Function. News Physiol Sci 18:65-70.

[Leonhardt] Leonhardt, H., Histologie, Zytologie und Mikroanatomie des Menschen. Thieme 1980.

[Lunts67] Lunts, A. and Brailovski, V. (1967). Estimation of error rates in discriminant analysis. Technometrics, 10:1-11.

[QT] The QT-WhitePaper. http://www.trolltech.com/pdf/whitepapers/qt33-whitepaper-a4.pdf

[Rader1968] C. M. Rader, Discrete Fourier transforms when the number of data samples is prime, Proc. of the IEEE, vol. 56, pp. 11071108, June 1968.

[Rew1997] Rew, R., et al, NetCDF User's Guide for C, Unidata Program Center, http://www.unidata.ucar.edu.

[Ronneberger2002] Ronneberger, O., Burkhardt, H., Schultz, E., General-purpose Object Recognition in 3D Volume Data Sets using Gray-Scale Invariants - Classifacation of Airborne Pollen-Grains Recorded with a Confocal Laser Scanning Microscope. Proceedings of the 16th International Conference on Pattern Recognition, Quebec, Canada, 2002.

[Schäl2000] Schäl, M., Siggelkow, S., Invariant grey-scale features for 3D sensor-data. In Proceedings of the International Conference on Pattern Recognition, volume 2, pages 531-535, Barcelona, Spain, September 2000.

[Schmidt-Thews] Schmidt, Thews, Lang, Physiologie des Menschen. Springer 2000.

[Schölkopf2002] Schölkopf, B., Smola, J., Learning with Kernels, 2002 MIT Press.

[Schulz-Mirbach95] Schulz-Mirbach, H. Invariant features for grey scale images. In Sager, G., Posch, S., Kummert, F., editors, 17. DAGM-Symposium Mustererkennung, Informatik aktuell, pages 1-14, Springer 1995.

[ShSh97] Sheppard, C., Shotton, D., (1997) Confocal Laser Scanning Microscopy, Springer.

[Smith97] Smith, T.,(1997) Avian Embryo, http://www.msstate.edu/dept/poultry/avianemb.htlm.

[SVMtl] Ronneberger, O., et al, SVM template library. http://lmb.informatik.uni-freiburg.de/lmbsoft/libsvmtl/index.en.html, 2004

[Vapnik79] Vapnik, V. and Tscherwonenkis, A. (1979). Theorie der Zeichenerkennung. Akademie Verlag, Berlin.

[Veldhuizen] Arrays in Blitz++, Proceedings of the 2nd International Scientific Computing in Object Oriented Parallel Environments (ISCOPE'98). Overviews the Blitz++ implementation of dense arrays.

[VTK] Visulization ToolKit Manual. http://www.vtk.org/doc/nightly/html/

[Wählby2003] Wählby, C., et al, Compining Intensity, Edge, and Shape Information for 2D and 3D Segmentation of Cell Nuclei in Tissue Sections.

# List of Figures


| | | |
|---|---|---|
| 1.1 | Avian enbryodic development | 9 |
| 1.2 | A nucleus marked with BrdU | 10 |
| 1.3 | An erythrocyte nucleus marked with YoPro | 10 |
| 1.4 | A LSM cross-sections of a capillary | 11 |
| 1.5 | Series of LSM slices of an erythrocyte nucleus marked with YoPro | 12 |
| 1.6 | 3D reconstruction of an erythrocyte nucleus | 12 |
| 1.7 | Series of LSM slices of an endothelial cell nucleus marked with YoPro | 13 |
| 1.8 | 3D reconstruction of an endothelial cell nucleus | 13 |
| 1.9 | Series of LSM slices of a pericyte nucleus marked with YoPro | 14 |
| 1.10 | 3D reconstruction of a pericyte nucleus | 14 |
| 1.11 | Series of LSM slices of a fibroblast nucleus marked with YoPro | 15 |
| 1.12 | 3D reconstruction of a fibroblast nucleus | 15 |
| 1.13 | Series of LSM slices of a macrophage nucleus marked with YoPro | 16 |
| 1.14 | 3D reconstruction of a macrophage nucleus | 16 |
| 1.15 | Schematic view of the human colon | 17 |
| 1.16 | LSM slice of the colon villi | 17 |
| 1.17 | Colon villi tissue | 18 |
| 1.18 | Series of LSM slices of an epithel nucleus marked with ToPro3 | 18 |
| 1.19 | 3D reconstruction of a epithel nucleus | 18 |
| 2.1 | A confocal laser scanning microscope | 20 |
| 3.1 | 2D slice of a YoPro marked probe | 22 |
| 3.2 | Threshold segmentation of YoPro marked probe. | 23 |
| 3.3 | Histogram of a YoPro marked probe | 23 |
| 3.4 | Sobel edge detection | 24 |
| 3.5 | Laplacian of Gaussian edge detection | 24 |
| 3.6 | Watershed Segmentation | 25 |
| 3.7 | Seeded watershed segmentation | 26 |
| 3.8 | Calculating Invariants | 29 |
| 3.9 | Group avarage invariant | 30 |
| 3.10 | SVM largest margin hyperplane | 31 |
| 3.11 | Mapping features to high dimentional space | 33 |
| 3.12 | Overfitting | 36 |
| 4.1 | Two-Point gray-scale invariants of a 2D image | 39 |
| 4.2 | Multi channel gray-scale invariants | 42 |
| 4.3 | Two-Point gray-scale invariants for 3D volume data | 43 |
| 4.4 | Three-Point Gray-Scale Invariants | 43 |
| 5.1 | Gauss filtered circle | 47 |
| 5.2 | Two-Point invarinat with a gauss filter kernel | 47 |

5.3 Results of convolutions a different scopes . . . . . . . . . . . . . . . . . 48
5.4 Marking training samples for each class . . . . . . . . . . . . . . . . . 49
5.5 Interactive training process . . . . . . . . . . . . . . . . . . . . . . . . 49
5.6 Visualizing the class confidence . . . . . . . . . . . . . . . . . . . . . . 50
5.7 Connected component labeling . . . . . . . . . . . . . . . . . . . . . . 51
5.8 Exponential kernel functions: the hull problem . . . . . . . . . . . . . 52
5.9 Exponential kernel functions: inverse data . . . . . . . . . . . . . . . . 52
5.10 Exponential kernel functions: solving the hull problem . . . . . . . . . 53
5.11 Pseudo three-point invarinats: examples . . . . . . . . . . . . . . . . . 53
5.12 Pseudo three-point invarinats: real three-point invariiants . . . . . . . 54
5.13 Pseudo three-point invarinats: 1D representation . . . . . . . . . . . . 54
5.14 Pseudo three-point invarinats: fourier descriptors . . . . . . . . . . . . 55
5.15 Pseudo three-point invariants: implementing fourier descriptors . . . . 56
5.16 Pseudo three-point invariants . . . . . . . . . . . . . . . . . . . . . . . 56
5.17 Pseudo Three-Point Invariants: edge detection . . . . . . . . . . . . . 56

6.1 Software: main window . . . . . . . . . . . . . . . . . . . . . . . . . . 59
6.2 Software: 2D Control . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
6.3 Software: Filters . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
6.4 Software: Channel Preview . . . . . . . . . . . . . . . . . . . . . . . . 60
6.5 Software: xy-slice scaled at 2:1 . . . . . . . . . . . . . . . . . . . . . . 61
6.6 Software: feature range plot . . . . . . . . . . . . . . . . . . . . . . . . 61
6.7 Software: extracting features . . . . . . . . . . . . . . . . . . . . . . . 62
6.8 Software: labeling traing samples. . . . . . . . . . . . . . . . . . . . . . 63
6.9 Software: training dialog box. . . . . . . . . . . . . . . . . . . . . . . . 63
6.10 Software: Classification. . . . . . . . . . . . . . . . . . . . . . . . . . . 64
6.11 Software: segmenting objects . . . . . . . . . . . . . . . . . . . . . . . 64

7.1 Histogram of a $\alpha$-value gray-scale image One-Vs-One . . . . . . . . . . 67
7.2 Histogram of a $\alpha$-value gray-scale image One-Vs-Rest . . . . . . . . . 67
7.3 Experiment 4: Lamin stain. . . . . . . . . . . . . . . . . . . . . . . . . 71
7.4 Experiment 4: Lamin stain. . . . . . . . . . . . . . . . . . . . . . . . . 71

# Abbreviations

| | |
|---|---|
| EC | endothelial cell |
| Ery | erythrocyte |
| FB | fibroblast |
| LOO | leave one out estimation |
| LSM | laser scanning microscope |
| MΦ | macrophage |
| KKT | Karush-Kuhn-Tucker conditions |
| PC | pericyte |
| PSF | point spread function |
| SV | support-vectors |
| SVM | support-vector machine |