# Models and Algorithms for Image-Based Analysis of Microstructures

Dipl.-Inform. Oliver Wirjadi

Vom Fachbereich Informatik der Technischen Universität Kaiserslautern zur Verleihung des akademischen Grades Doktor der Naturwissenschaften (Dr. rer. nat.) genehmigte Dissertation.

| | |
|---|---|
| Dekan: | Prof. Dr. Karsten Berns |
| Vorsitzender der Promotionskommission: | Prof. Dr.-Ing. Jens Schmitt |
| Berichterstatter: | Prof. Dr. Thomas Breuel |
| | Prof. Dr.-Ing. Andreas König |
| Tag der wissenschaftlichen Aussprache: | 13.02.2009 |

(D 386)

Technische Universität
Kaiserslautern

**Abstract**

Modern digital imaging technologies, such as digital microscopy or micro-computed tomography, deliver such large amounts of 2D and 3D-image data that manual processing becomes infeasible. This leads to a need for robust, flexible and automatic image analysis tools in areas such as histology or materials science, where microstructures are being investigated (e.g. cells, fiber systems). General-purpose image processing methods can be used to analyze such microstructures. These methods usually rely on segmentation, i.e., a separation of areas of interest in digital images. As image segmentation algorithms rarely adapt well to changes in the imaging system or to different analysis problems, there is a demand for solutions that can easily be modified to analyze different microstructures, and that are more accurate than existing ones. To address these challenges, this thesis contributes a novel statistical model for objects in images and novel algorithms for the image-based analysis of microstructures.

The first contribution is a novel statistical model for the locations of objects (e.g. tumor cells) in images. This model is fully trainable and can therefore be easily adapted to many different image analysis tasks, which is demonstrated by examples from histology and materials science. Using algorithms for fitting this statistical model to images results in a method for locating multiple objects in images that is more accurate and more robust to noise and background clutter than standard methods. On simulated data at high noise levels (peak signal-to-noise ratio below 10 dB), this method achieves detection rates up to 10% above those of a watershed-based alternative algorithm.

While objects like tumor cells can be described well by their coordinates in the plane, the analysis of fiber systems in composite materials, for instance, requires a fully three dimensional treatment. Therefore, the second contribution of this thesis is a novel algorithm to determine the local fiber orientation in micro-tomographic reconstructions of fiber-reinforced polymers and other fibrous materials. Using simulated data, it will be demonstrated that the local orientations obtained from this novel method are more robust to noise and fiber overlap than those computed using an established alternative gradient-based algorithm, both in 2D and 3D. The property of robustness to noise of the proposed algorithm can be explained by the fact that a low-pass filter is used to detect local orientations. But even in the absence of noise, depending on fiber curvature and density, the average local 3D-orientation estimate can be about 9° more accurate compared to that alternative gradient-based method.

Implementations of that novel orientation estimation method require repeated image filtering using anisotropic Gaussian convolution filters. These filter operations, which other authors have used for adaptive image smoothing, are computationally expensive when using standard implementations. Therefore, the third contribution of this thesis is a novel optimal non-orthogonal separation of the anisotropic Gaussian convolution kernel. This result generalizes a previous one reported elsewhere, and allows for efficient implementations of the corresponding convolution operation in any dimension. In 2D and 3D, these implementations achieve an average performance gain by factors of 3.8 and 3.5, respectively, compared to a fast Fourier transform-based implementation.

The contributions made by this thesis represent improvements over state-of-the-art methods, especially in the 2D-analysis of cells in histological resections, and in the 2D and 3D-analysis of fibrous materials.

# Contents

# List of Abbreviations

| | | |
|---|---|---|
| **AUC** | Area under the curve | 33 |
| **CRF** | Conditional random field | 6 |
| **DT-MRI** | Diffusion tensor magnetic resonance imaging | 72 |
| **FFT** | Fast Fourier transformation | 73 |
| **FIR** | Finite impulse response | 84 |
| **GRP** | Glass fiber-reinforced polymer | 67 |
| **H&E** | Hematoxylin and eosin | 38 |
| **HMM** | Hidden Markov model | 6 |
| **IIR** | Infinite impulse response | 85 |
| $k$**NN** | $k$-nearest neighbor classifier | 39 |
| $\mu$**CT** | Micro-computed tomography | 1 |
| **LI** | Labeling index | 40 |
| **MAP** | Maximum a-posteriori | 9 |
| **MCMC** | Markov chain Monte Carlo | 20 |
| **MLP** | Multi-layer perceptron | 11 |
| **MRI** | Magnetic resonance imaging | 71 |
| **MRF** | Markov random field | 5 |
| **MSE** | Mean square error | 29 |
| **NN** | Nearest neighbor | 86 |
| **PSNR** | Peak signal-to-noise ratio | 28 |
| **ROC** | Receiver operating characteristic | 30 |
| **RJMCMC** | Reversible jump Markov chain Monte Carlo | 25 |
| **RSA** | Random sequential adsorption | 57 |
| **SAM** | Scanning acoustic microscopy | 65 |
| **SVM** | Support vector machine | 14 |

# Chapter 1

# Introduction

With the availability of digital imaging technologies and the large amounts of data they produce comes the need for automated image analysis methods. Two examples are the assessment of microscopic images in histology [174], and the design and development of new materials in engineering, where the availability of micro-computed tomography ($\mu$CT) enables the 3D-characterization of the microstructure of materials [12]. A pathologist, for instance, will only be able to label examples of a few species of cells observed in microscopic images. Therefore, these analysis tasks share the property that only a limited amount of user interaction is possible. Furthermore, they require algorithms for the automatic analysis of large amounts of data.

Standard image processing algorithms suitable for analyzing microstructures are available in various software package, e.g. ImageJ [138]. These usually contain implementations of a variety of preprocessing, segmentation and analysis methods, which need to be recombined for every analysis task anew, requiring both time and expert knowledge. Work on automating this process exists, e.g. for the classification of textures[132], and a software that integrates such procedures is under development [130]. Furthermore, a number of specialized algorithms for the analysis of microstructures have been proposed, e.g. for automatic grading of tumors [1, 16, 30, 73, 84, 98, 114, 141] in histology or for the characterization of fiber-reinforced materials [78, 81, 102, 119, 139, 167] in materials engineering. Most of these rely on segmentation of the microstructures [1, 16, 30, 73, 84, 98, 114, 126, 172], which can be difficult, especially when operating at the resolution limits of an imaging system.

Therefore, there is a need for the development of novel methods which improve over existing ones in these aspects. That is, methods need to be found that are trainable, i.e., which are capable to automatically adjust to new image data, and which avoid image segmentation. While adhering to these constraints, the three topics covered in this thesis contribute to different aspects of the image analysis of microstructures, especially in histology and materials science.

**Statistical multiple object localization** In order to count or to describe the spatial arrangement of tumor cells, metal particles or granulates in images, it is not necessary to perform a segmentation of these objects; rather, it would be sufficient to know their coordinates. This is the problem addressed by the first contribution of this thesis. A trainable statistical model of the

locations of such collections of objects in images is developed. This model can be used to describe the spatial arrangement of these particles. To apply this model to the localization of objects in images, suitable algorithms will be introduced and evaluated. The localization results obtained by this method are useful quantities e.g. in the analysis of meningioma tumor cells that will be presented in Ch. 2, where the labeling index can be computed from them. The labeling index, which indicates the fraction of cells in mitotic phase, is required for tumor grading according to the standards set by the World Health Organization [113].

The term "object localization" does not have a precisely defined meaning within the computer vision literature; most commonly, it has been used in conjunction with "object detection", i.e., deciding whether or not an object was present in an image [95], but other meanings exist, e.g. estimating the pose of known object shapes in images [166]. In the context of object detection, object localization implies that not only a decision on object presence in an image is made, but that in case of a positive detection, also the most likely position of the searched object is computed, see e.g. [95]. In this thesis, rather than detecting the presence or absence of single objects in images, the problem of determining the positions of many similar objects in one image will be considered, e.g., fibers in planar cuts through fiber-reinforced materials or cells in histological resections. To make this differentiation from other meanings of localization obvious, the term "multiple object localization" shall be used wherever this is of importance.

The approach taken in this thesis uses methods from machine learning [21, 40] and interaction point process theory [106, 121] to construct a fully trainable statistical model. This construction makes the proposed method adaptable, which will be shown by applications to image data of different microstructures. Previous methods addressing this problem were frequently based on image segmentation algorithms [1, 16, 126, 172], which usually do not adapt well to changing imaging conditions such as lighting or noise.

Like Markov random fields [59], interaction point processes are suitable for modeling the dependencies within collections of spatial random events. But they do not impose a fixed graph structure. In this sense, point process models are well suited for handling randomly positioned objects in images [36]. This thesis demonstrates how to combine these models with trainable classifiers like multi-layer perceptrons [20] and support vector machines [24], making point process models applicable to a wide range of computer vision problems for the first time. The original idea to apply point process methods in computer vision is due to Baddeley and van Lieshout [8]. Their and all following publications in that area relied on parametric models for the images' gray value statistics being available [3, 4, 39, 72, 91, 128, 154]. In this sense, the novel method that will be developed in Ch. 2 extends the work of Baddeley and van Lieshout by reformulating it to make it fully trainable, thereby making the method applicable to a larger variety of image data.

Applying this statistical model to the task of computing the locations of multiple objects in images enables automation of applications in materials science and neuropathology. In contrast to most segmentation-based image analysis systems, such trainable algorithms have the built-in capability to adapt to different domains. Furthermore, it will be shown to be more accurate than an established image processing tool [138] on simulated and real world data.

**Fiber orientations from orientation space** The second of the three topics covered in this thesis is an automatic method for computing the fiber orientation distribution from microscopic or $\mu$CT-images of fibrous materials such as fiber-reinforced polymers or paper. This distribution, or moments thereof, are required for computing mechanical properties of such materials, see e.g. [25, 66, 83]. The algorithm that will be introduced in Ch. 3 uses anisotropic filters to construct the so-called orientation space representation directly from 2D and 3D-grayscale images. In an analogy to scale space, orientation space expands images into higher dimensional representations by applying banks of anisotropic filters. They contain information about the local orientations of structures in images, and have found some applications in image segmentation [62]. The method will be shown to be more accurate than an existing gradient-based method for computing fiber orientations [17, 51, 112, 90, 134, 144], and to be more robust to factors such as noise and fiber overlap.

Most existing systems for assessing the fiber distribution in such materials can be characterized as either stereological [32, 78, 81, 80, 102, 139] or segmentation-based [6, 38, 176] methods. Stereological approaches draw conclusions on the 3D-microstructure from measurements in 2D-sections [7]. Preparation of such sections often destroys the specimen, or, if tomographic data is available, these methods ignore the three dimensional information contained in the data. Fiber orientations obtained by image segmentation, which uses the three dimensional image structure, is often difficult due to the quality of the available image data. E.g., a current $\mu$CT scanner can acquire images with pixel side lengths of about 1 $\mu$m, while typical glass fiber diameters in reinforced polymers are in the range of 10 to 20 $\mu$m. Carbon fibers are even thinner. When analyzing microstructures at these scales, an unambiguous separation of the structures of interest will not be possible, regardless of which segmentation method is used.

This difficult and error-prone segmentation step can be avoided by using filters to compute local orientation, an idea that goes back to the work of Granlund and Knutsson [64, 89]. The method proposed in this thesis is similar to theirs, but uses different filter kernels and sampling methods. This allows for the use of a novel efficient implementation, also introduced with this thesis, and more finely resolved orientation measurements. In this method, response to anisotropic filters is used as a measure for the local orientation structure in images. By computing orientation descriptors used in spherical statistics [48] and rheology [163], the results of the proposed method can be used both as a descriptive tool for analyzing the orientation distribution in a specimen and as input for subsequent simulation studies on fibrous materials.

**Separation of anisotropic Gaussian filters** To reduce the computational burden for computing the local orientations in that new method for assessing 2D and 3D-fiber orientation distributions, the third contribution of this thesis is a novel and fast algorithm for anisotropic Gaussian image filtering. By using an optimal factorization of the Gaussian filter kernel, this filter operation can be implemented very efficiently, outperforming state-of-the-art implementations. The basic principle behind the approach taken in this thesis is that of separable filters, where a sequence of lower dimensional filter operations is used to compute the filter. Not only is this filter operation required for performing the fiber orientation analysis outlined above, but it can also be used for adaptive image smoothing [33, 87, 101, 149, 175].

Fast linear filter implementations by separated filter kernels have long been known and are widely used. The existence of the separation introduced in Ch. 4 has not been known until the appearance of Geusebroek, Smeulders and van de Weijer's papers in 2002 and 2003 [60, 61]. There, a non-orthogonal separation of the 2D-anisotropic Gaussian kernel has been derived. By setting criteria for when such a non-orthogonal separation is useful, and by solving the $d$-dimensional separation problem, an optimal separation is derived. Therefore, the material in Ch. 4 both explains and generalizes the results of Geusebroek and coworkers.

By making use of this non-orthogonal separation of the Gaussian kernel, the three dimensional anisotropic Gaussian convolution is implemented by three one-dimensional convolutions. Each of these can be performed using recursive infinite impulse response filters [177], leading to an implementation which is faster than both truncated convolution and filtering in the Fourier domain. Its runtime is independent of kernel size and linear in the number of pixels.

The methods developed in this thesis enable accurate multiple object localization that can adapt to a variety of image data, fiber orientation estimation of fibrous materials with very dense fiber systems and in noisy 2D and 3D-images, and fast anisotropic Gaussian image filtering. In the following, each of these three novel contributions will be detailed in a dedicated chapter, including all derivations and evaluations. While each of these chapters presents a self-contained topic, the fiber orientation estimation approach in Ch. 3 will profit from the fast implementations of anisotropic Gaussian filters discussed in Ch. 4.

# Chapter 2

# Statistical Model for Multiple Object Localization

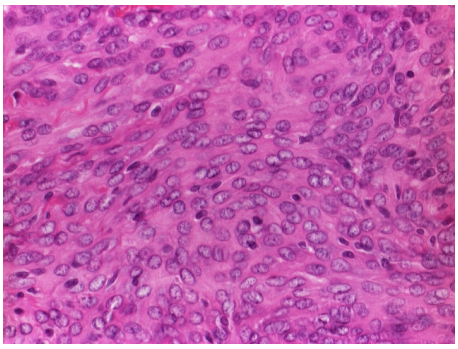
## 2.1 Introduction

Localization of multiple objects in images enables the computation of characteristics such as spatial distributions or object frequencies, which is relevant e.g. in the computer-aided grading of histological resections (Fig. 2.1(a) and 2.1(b)) or image analysis-based studies of material properties (Fig. 2.1(c) and 2.1(d)). Such medical and materials engineering problems have frequently been solved using fine-tuned image segmentation methods, e.g. thresholding [1, 16] or the watershed algorithm [126, 172]. In contrast, trainable systems, once designed, have the potential to be trained to perform well to a variety of data. An example is the neural network system for cell counting in histological preparations proposed in [151]. The method introduced in this chapter is such a trainable system, designed for computing the locations of objects in images. As user input, it requires one or more training images and a set of known object locations in these images. A full, pixel-accurate segmentation of the objects of interest does not need to be available.

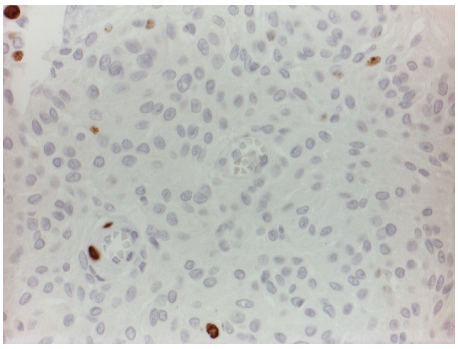
The developed system constitutes a statistical model of non-overlapping circular objects in the plane that can be trained from given images and object locations, and algorithms for localizing objects under the trained model in previously unseen images. A quantitative evaluation of this method on simulated and real data will provide a clear picture of when the use of interaction point process models is advantageous over conventional image processing methods. It will be checked whether the proposed trainable model is indeed transferable across various problem instances, and how well it performs compared to customized standard processing methods.

Statistical modeling of the spatial arrangement of objects has actively been used e.g. for object tracking in video, but these have mostly been ad-hoc models. Examples are area overlap between objects [79, 180] or non-integrable potential functions [179] to avoid detection of overlapping objects in subsequent video frames.

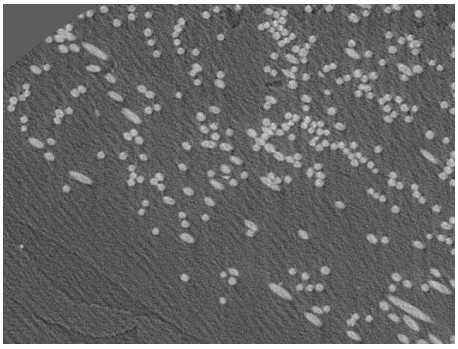
A more rigorous and well-established method for modeling spatial random processes are Markov random fields (MRF), where random variables are arranged on some lattice and restrictions are



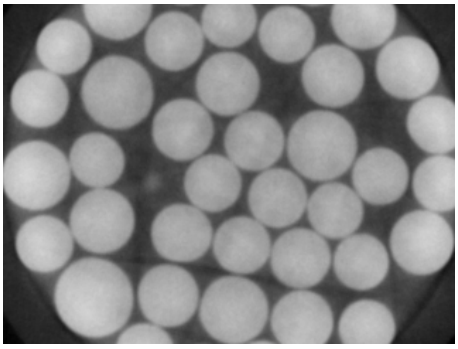
(a) Meningioma tumor cells stained with H&E dye (image source: Uni Saarland).



(b) Meningioma tumor cells treated with Ki-67 antigen (image source: Uni Saarland).



(c) Slice through a glass fiber-reinforced polymer (image source: IVW Kaiserslautern).



(d) Slice through a probe of sinter particles (image source: BAM Berlin).

Figure 2.1: Examples of the localization problem type treated in this chapter: Computing the coordinates of collections of similar objects in 2D-images. For detailed descriptions of each dataset refer to Sec. 2.7.

placed on the conditional probability density functions at one lattice point given its neighbors [59]. The MRF model has proven to be especially well suited for image denoising [19]. These models can be extended by separating the set of random variables into "observable" and "hidden" ones, which is then called a hidden Markov model (HMM). With this extension, these models are suitable for various image processing tasks, see e.g. [53]. Recently, the conditional random field (CRF) model has drawn a lot of attention [92, 165]. These are more general than HMMs in the sense that they allow to model arbitrary dependencies between observed variables [92]. All three, MRF, HMM and CRF, share the property that they are defined fixed graphs. Therefore, their use for modeling freely moving objects is limited.

The method developed in this chapter is based on spatial point processes, which model random point patterns. A special case of these processes are pairwise interaction point processes [106, 121], in which the joint probability density function includes contributions from each point and from interactions between any two points in the process. The use of interaction point processes in image analysis has been advocated by Descombes and Zerubia [36]. They argue that point processes have an advantage over MRFs because they can model relations between objects, rather than pixels or lattice sites.

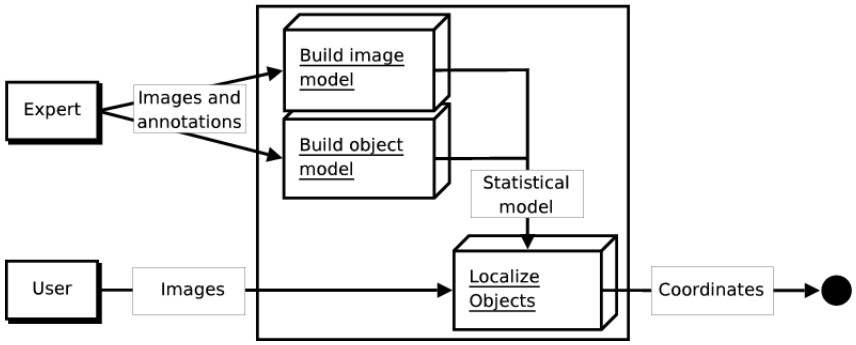



Figure 2.2: Overview of the components and problem setup for localizing circular objects in images. The model is a posterior density of object locations combining a spatial statistics prior term with statistical classifiers such as multi-layer perceptrons or support vector machines. Given new user images, Metropolis-Hastings sampling or greedy search can find the locations of objects under the previously trained model.

An early application of these spatial point processes in computer vision can be found in a paper by Baddeley and van Lieshout from 1992 [8]. They proposed to estimate object locations $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ in an image $f$ by sampling from their posterior distribution given by

$$p(X|f) \propto p(f|X)p(X). \tag{2.1}$$

Here, $p(f|X)$ is a parametric likelihood term that followed from an assumed image noise model, and $p(X)$ is an interaction point process density for the spatial arrangement of the $n$ objects in $X$. A number of publications using spatial point processes for vision, including the present chapter, follow this general model. Drot et al. segment images by fitting non-overlapping triangles and describe the relation between these triangles by a hardcore model, which is one type of spatial point process [39]. For networks of lines, the "Candy" point process model was developed and applied to fit lines to image edges [91, 154]. A further point process model consisting of rectangles and line segments was applied to segment elevation maps [128]. More closely related to the problem setting in the present chapter are applications of marked point processes to counting cells in confocal microscopy images, which have been published in a series of papers by M. Hurn et al. [3, 4, 72].

All of these applications of interaction point processes either involved novel prior terms $p(X)$ that were adjusted to the specific task [39, 91, 128, 154], or used likelihood functions $p(f|X)$ that modeled the given imaging system [3, 4, 72]. But all of these methods can be reduced to the basic model in Eq. (2.1) [3, 4, 39, 72, 91, 128, 154]. The current chapter extends this model by introducing statistical classifiers as a method for making these models fully trainable, i.e., abolishing the need for a known likelihood term $p(f|X)$ and demonstrates how to apply this technique to the multiple object localization problem.

To solve the multiple object localization problem, the setup for this chapter will be the following: Given images containing objects such as those in Fig. 2.1 and a list of known object locations

created by an expert, train a system for locating similar objects in other images. A trained system can then be applied to localize objects in images. The proposed system is schematically shown in Fig. 2.2. Learning methods operate in separate "training" and "user" phases. When training the proposed model, an expert of the field provides image data along with annotations. For most part, these annotations will be the coordinates of objects, represented by the center points. Additionally, information such as object size could also be supplied. Once all parameters are set up, the system is ready to compute coordinates from new, previously unseen images.

The core of the method combines knowledge of the shape, size, frequency and spatial constellation of objects ("object model") and of the appearance and noise characteristics of the image ("image model"). Methods from point process theory, introduced below, will be used to model object locations. The specific form of random process used here describes events with random locations which have a minimal distance parameter. A previously known model for the image contents will turn out not to be sufficient for images containing any amount of noise. To solve this shortcoming, classification algorithms will fill the gap. This combination of a trained model and spatial statistics will turn out to be robust to the type of defects under which simpler approaches fail.

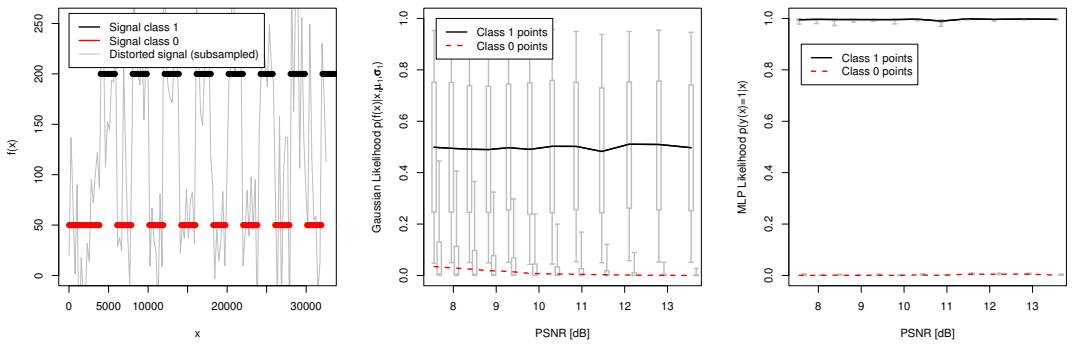
The description of the complete system is spread over four sections in this chapter. Sec. 2.2 first introduces the proposed model, a posterior probability of object locations $X$ given an observed image $f$, and shows how and why it differs from the one in (2.1). Then, concrete techniques from the machine learning and point process literature will be introduced to fill each of the three model components in Fig. 2.2. In Sec. 2.3, statistical classifiers as image models will be discussed, followed by a description of suitable interaction point processes for the object model in Sec. 2.4. Finally, Sec. 2.5 proposes two specific algorithms for fitting the model to images to obtain estimates of object locations. The remainder of the chapter is then dedicated to evaluation, applications and discussion.

## 2.2 The proposed model

This section introduces a posterior density model for object locations in images. Let $\mathcal{X}$ be a random process with realizations $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in [0,1]^d$. These are the locations of $n$ objects in an image $f$. The number of objects $n$, which is not known a-priori, is the realization of a random variable $\mathcal{N}$. E.g., $X$ could describe the $n$ coordinates of all particles in Fig. 2.1(d). The joint density $p_{\mathcal{X},\mathcal{N}}(X, N)$ is not known, but inference can be used to estimate the object locations $X$ from a given image $f$. This is the task of this chapter. To keep notation uncluttered, probability density functions will not carry an index variable. The corresponding random variables will be clear from the arguments, e.g., $p(X)$ is understood as the short from for $p_{\mathcal{X}}(X)$. Given an image $f$, the posterior of $X$ and $n$ is rewritten as

$$p(X, n|f) \propto p(f|X, n)p(X, n) \approx p(X|n)p(n) \prod_{i=1}^{n} p(f(\mathbf{x}_i)|\mathbf{x}_i). \tag{2.2}$$

Note the crucial step here, which is the reduction to the pointwise location likelihood $p(f(\mathbf{x}_i)|\mathbf{x}_i)$. This approximation deliberately ignores all image background (locations that are not contained




(a) A two-level piecewise constant function in independent additive Gaussian noise (PSNR 10.04 dB).

(b) The Gaussian likelihood at various noise levels, ordered by the two signal levels ("class 0" and "class 1").

(c) The likelihood term computed using MLPs, trained independently for each noise level.

Figure 2.3: Evaluation of the alternative models for computing the likelihood term on an exemplary 1D-signal. While the likelihood values of the Gaussian model vary strongly and start overlapping between the two classes below a PSNR of 12 dB, the likelihood term $p(y(\mathbf{x}) = 1|\mathbf{x})$ computed using a MLP is robust to noise. This 1D-experiment demonstrates the potential improvements on noisy data when switching from a fixed to a trainable model for $p(f|\mathbf{x})$ in Eq. (2.2).

in $X$), which has two main advantages: It leads to efficient algorithms as it is not necessary to integrate over the whole image $f$, and it enables the use of patch-based models by integrating the neighborhoods of the locations $\mathbf{x}_i \in X$, cf. Sec. 2.3 for details. The minimum error rate estimator under a zero-one loss function is the maximum a-posteriori (MAP) estimator [40],

$$(X^*, n^*) = \operatorname{argmax} p(X, n|f). \tag{2.3}$$

The remainder of this chapter proposes, discusses and evaluates different models for the individual terms in (2.2) and algorithms for implementing the maximization in (2.3).

To understand why it may be advantageous to integrate trainable models $p(f(\mathbf{x}_i)|\mathbf{x}_i)$ into the system, which is one of the main points of this chapter, consider the original image model used by Baddeley and van Lieshout [8] and also in later applications of point processes in computer vision, e.g. [3]. For this initial image model, assume that objects appear with a constant gray value $\mu_1$ in images and that the data contains independent additive Gaussian noise with variance $\sigma_n^2$. Furthermore, the image background must have some mean gray value different from $\mu_1$. Under these preconditions, (2.2) can be expressed as

$$p(X, n|f) \approx p(X|n)p(n) \prod_{i=1}^{n} p(f(\mathbf{x}_i)|\mathbf{x}_i) = p(X|n)p(n) \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{1}{2\sigma_n} \left(f(\mathbf{x}_i) - \mu_1\right)^2\right). \tag{2.4}$$

In contrast to this formulation, which for reasons discussed earlier factorizes $p(f|X, n)$ only at object locations, Baddeley and van Lieshout assumed solid circular objects with known radius

and applied this model by factorizing over all pixels in image $f$ [8]. That is computationally expensive. The approach taken in the present chapter factorizes only over the object locations $X$, leading to a computationally less expensive optimization problem.

To demonstrate the performance of the Gaussian likelihood function in (2.4), consider the example of a 1D signal in Fig. 2.3(a): A piecewise constant function in additive, independent normally distributed noise. Decreasing the variance of the additive noise leads to peak signal-to-noise ratios (PSNR) between 7 and 14 dB. The PSNR as a measure of signal or image quality will be defined in Sec. 2.6.1. The values of the Gaussian likelihood term in (2.4) at different noise levels are shown in Fig. 2.3(b). For increasing levels of noise (low PSNR), the class 1-likelihood of points truly belonging to the class 0-level of the signal increases. This is caused by the Gaussian's parameter $\sigma_n$, which increases with noise level. This would inevitably lead to erroneous localizations in the overall system. On the other hand, if a trainable classifier such as a multi-layer perceptron (MLP) could be used, which can adapt to noise levels, errors caused by this likelihood term could be avoided (Fig. 2.3(c)). Such alternative image models will be reviewed in Sec. 2.3.

Other variants of pixel-wise likelihood terms, which have also been used in conjunction with spatial statistics, are Poisson noise models [72] and models of point spread functions in confocal microscopy [4]. As these parametric models are not capable of adapting to high noise levels or other, not necessarily random image defects, they are all similar to the Gaussian likelihood in this sense. Therefore, the next step is to find more flexible, non-parametric models for the likelihood term $p(f(\mathbf{x}_i)|\mathbf{x}_i)$ in (2.2).

### 2.2.1 Incorporating statistical classifiers as image models

Methods that have proven to be powerful tools in computing probabilities in a form similar to what is required here for an image model are classifiers that compute class-posterior probabilities. In classification, the task is to predict a class label $y$, usually an integer valued quantity, from an observation vector $\mathbf{x}$ [40]. Instead of merely predicting a class label $y$, many classifiers compute class-posterior probabilities of the form $p(y(\mathbf{x}) = k|\mathbf{x})$. Examples for such classifiers are naive Bayes classifiers, multi-layer perceptrons and classification trees [21], as well as (postprocessed) support vector machines (Sec. 2.3.2). Class-posterior probabilities can be reduced back to a class decisions e.g. by picking the class $k$ that maximizes the posterior, i.e., $k^* = \text{argmax}\, p(y(\mathbf{x}) = k|\mathbf{x})$.

In order to use such statistical classification models in (2.2), the localization problem must be reduced to a classification problem in a suitable form. In its simplest setting, object localization is a two-class problem with $k = 0$ encoding background and $k = 1$ encoding object locations. When more than two object types exist, $k = 0$ will still denote background locations, but the set of object locations $X$ will then contain all locations $\mathbf{x}$ for which $y(\mathbf{x}) \neq 0$. Then, the set $X$ of object locations can also be written as

$$X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} = \{\mathbf{x}|y(\mathbf{x}) \neq 0\}. \tag{2.5}$$

This being an equivalent definition, it follows that $y$ is a sufficient statistic for $f$, i.e., any information on the points $X$ contained in the image $f$ is also contained in the (unknown) function $y$.

With this observation, (2.2) can be rewritten to

$$p(X, n|f) = p(X, n|y) \propto p(X, n)p(y|X, n) \approx p(X|n)p(n) \prod_{i=1}^{n} p(y(\mathbf{x}_i) \neq 0|\mathbf{x}_i). \tag{2.6}$$

In this equivalent form, the image likelihood term from (2.2) is replaced by $p(y(\mathbf{x})|\mathbf{x})$. From the perspective of the overall system, not much has changed. The only difference lies in the way in which the image likelihood term is computed. Even though the form of the function $y$ used above is unknown, machine learning methods are capable of constructing the density $p(y(\mathbf{x})|\mathbf{x})$ from labeled data in their training process. Two such classifiers, a special form of the multi-layer perceptron and support vector machines, will be introduced next.

## 2.3 Classifiers used as image models

As argued in the previous section, any classifier that is capable of computing class-posterior probabilities would be a suitable method for the image model in (2.2). This section reviews two such classifiers with a focus on their applicability within the model introduced with this chapter. The two methods are convolutional neural networks, as this form of the multi-layer perceptron is particularly well suited for classification in images [100], and support vector machines with a postprocessing step to compute class-posteriors, as this classifier's training algorithm, in contrast to most other classifiers, is globally optimal [24].

In order to train image models $p(y(\mathbf{x})|\mathbf{x})$, the classifiers must be given access to the image $f$. This is done in the following way. At a coordinate $\mathbf{x}$, a square image patch of $M \times M$ pixels is extracted. $M$ is chosen to be some pixels larger than the expected size of objects of interest in $f$. These $M^2$ pixel values are the actual input for the classifiers. Therefore, in the following, writing coordinate vector $\mathbf{x}$ as input to a classifier should be understood as a shorthand notation for a $M \times M$ pixel image patch at coordinate $\mathbf{x}$. While the input to convolutional neural networks remains two dimensional, these values need to be concatenated into a vector in $\mathbb{R}^{M^2}$ in case of the support vector machine method.

### 2.3.1 Multi-layer perceptrons

A multi-layer perceptron (MLP) consist of units that are connected by weighted arcs [20]. Each unit computes a linear combination of the values at its input connections, using the weights associated with each arc, followed by a non-linear transfer function. The thus computed value is then passed on to other units which are again connected by an outgoing arc. By combining large numbers of such units and arranging them in layers, where the units in layers do not have any cross-connections, powerful discriminative classifiers can be constructed. Their parameters, the weights, can be fit to model some training data using a variety of algorithms.

For the purposes of this chapter, the important property of MLPs is that they minimize an error function with respect to some training data. Let $x$ be the input to a MLP. The network will model a function $y(x)$, prescribed by a set of training samples $\{(x_i, y_i)\}_{i=1\ldots n}$, drawn independently from their unknown joint density $p(x, y)$. Using suitable error measures and problem encodings, the

output $y$ of a MLP has an interpretation as class-posterior probability. To see why this is true, assume that the training process converged to the global optimum under a sum-of-squares error function. This error is minimized by the conditional expectation $E[y|x]$ [20]. Thus, given an input $x$, the output value of an optimally trained MLP has the value

$$y(x) = E[y|x] = \int yp(y|x)dy. \tag{2.7}$$

Using a 0/1 coding, i.e. $y \in \{0, 1\}$, the conditional expectation simplifies to

$$= 0 \cdot p(y = 0|x) + 1 \cdot p(y = 1|x) = p(y = 1|x). \tag{2.8}$$

For a K-class problem, every class $k = 0, \ldots, K-1$ is assigned to one distinct output unit $y_k$, so that, by the same argument as above, $y_k(x) = p(y = k|x)$. Note that similar arguments can be made for other machine learning methods as well. Therefore, other approaches such as logistic regression or some types of decision trees could also be used. For the support vector machine (SVM), which will also be used in this chapter, it is more difficult to obtain estimates of the class-posterior probabilities, see Sec. 2.3.2.

#### Convolutional neural networks

One problem with the standard MLP layout is that the number of parameters (weights) increases with the number of connections. Further, the topology of fully connected MLPs does not model the dependencies between neighboring pixels which are inherent to images. As outlined above, the input to classifiers in this section are 2D-image patches containing $M \times M$ pixel values, and this image structure should be used if possible. One way to handle image structures in classifiers is to derive a number of features from images, which are then given to the classifier as input. These features often include scale and edge information, which in turn model the local dependencies. Selection of appropriate features is a difficult problem. To circumvent such issues, shared weights are appropriate. Sharing weights in neural networks means that there exist groups of connections with the same weight in the network. The most prominent and maybe most successful MLP-variant using the concept of weight sharing is the convolutional neural network, proposed by LeCun et al. [100], which have proven to be efficient classifiers [150].

To see how weight sharing and convolution are connected, consider a discrete input image $f(x, y)$ at some coordinate $(x_0, y_0)$ at which a convolutional neural network is to be applied. The input contains a neighborhood around pixel $(x_0, y_0)$, e.g., a $5 \times 5$ square region centered in $(x_0, y_0)$. This results in a matrix $[f(x_0 + i, y_0 + j)]_{i,j=-2,\ldots,2}$ of input values. Each element of this matrix is connected to a hidden layer unit $z$ of the network by an arc with associated weight $w_{ij}$,

$$z(x_0, y_0) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} w_{ij} f(x_0 + i, y_0 + j). \tag{2.9}$$

Weight sharing implies that other units in the same layer as $z(x_0, y_0)$ are connected to other image pixels, say $(x_0+1, y_0)$, but use equivalent weights $w_{ij}$. When shifting these weights over the whole area of image $f$, the matrix $Z$ of second layer unit activations is equivalent to a 2D-finite convolution of $f$ with the matrix of weights $W = [w_{ij}]_{i,j=-2,\ldots,2}$,

$$Z = \left[ \sum_{i=-2}^{2} \sum_{j=-2}^{2} w_{ij} f(x_0 + i, y_0 + j) \right]_{x_0, y_0} = W * f. \tag{2.10}$$

The weights $w_{ij}$ are part of the regular neural network training procedure. Each layer contains not one, but a configurable number of convolutional feature matrices $Z$. In this chapter, $5 \times 5$ fields of weights as in the example above, are used in two consecutive layers of convolutional features. Following [150], these convolutional layers are each followed by $2 \times 2$ subsampling and are finally combined to form the overall output via one fully connected standard MLP layer.

In this network layout, the first two layers can be interpreted as a set of trainable image features followed by a fully connected hidden layer. Thus, the need for hand selected input features is avoided, resulting in a classifier system that can directly be applied to image data. As a consequence of the subsampling layers, these MLPs are robust to shifts in the input data. See [100] for details on the general properties of convolutional neural networks, and [150] for the specific architecture that will be used here. Especially, the training strategy from [150] is pursued, in which the amount of training data is artificially increased by adding geometrically transformed variants of the original data to the training set (randomly rotated and sheared patches), leading to better generalization performance of the classifier.

At this point, return to the exemplary evaluation of the likelihood term in (2.2) on the 1D signal shown in Fig. 2.3(a). Where the likelihood function derived from an additive Gaussian noise model leads to unstable likelihood values, the convolutional neural network model just introduced yields stable and well separated values for the likelihood term, cf. Fig. 2.3(c).

#### Training using stochastic gradient descent

Online or stochastic gradient descent training, where a MLP's parameters are updated for each sample, is known to outperform gradient descent batch training, especially for large training sets [22]. For applying the convolutional neural network model to the multiple object localization task, it will be applied to image patches sampled at and around object locations, and patches sampled from the image background. Together with the geometric transformations described above, this leads to training sets containing several ten thousands of examples (e.g., for the application to a glass fiber reinforced polymer that will be described in Sec. 2.7, approximately 75000 training examples have been generated). Therefore, online training will be used in this chapter. The most widely used training method for MLPs is backpropagation training with fixed learning rate $\epsilon$ and

weight update rule

$$w^{t+1} = w^t - \epsilon \frac{\partial E}{\partial w}, \tag{2.11}$$

where $E$ is the error function. Especially for convolutional neural networks, LeCun et al. use a stochastic diagonal Levenberg-Marquardt update rule [100]. Instead of using a fixed learning rate $\epsilon$ as in (2.11), they effectively compute an individual step size for each weight. The corresponding update rule

$$w^{t+1} = w^t - \frac{\frac{\partial E}{\partial w}}{|\frac{\partial^2 E}{\partial^2 w}| + \mu} \tag{2.12}$$

results from the Newton method, but uses only the diagonal elements of the Hessian matrix [15]. $\mu$ is a positive constant to deal with regions of low curvature. In contrast to other Newton methods, this update rule can be applied in online mode. The forward and backward propagation formulas using either fixed (2.11) or adaptive (2.12) step size are not fundamentally different in convolutional neural networks than in standard MLPs. Due to weight sharing, merely the indices change, and the update rules need to average over all connections sharing one weight. For backpropagation training, these technical details are given in App. A.

In experiments, adaptive weight updating did not lead to significant convergence rate improvements during training, and determination of suitable stopping rules turned out to be more difficult than for standard backpropagation training. Therefore, for the experiments that will be described later in this chapter, a fixed step size weight update with early stopping was used.

### 2.3.2 Support Vector Machines

The support vector machine (SVM) is one the most widely used algorithms for classification and regression. There exists a wide body of literature. Unlike the multi-layer perceptrons introduced above, SVMs do not result in class-posterior probabilities. Nevertheless, there exist ways to implement modifications of SVMs such that it results in an estimate of the posterior probability of class label given sample. To introduce the general principles of SVMs and especially to show how to estimate posterior probabilities, a brief review of SVMs is given next. For a comprehensive introduction to SVMs, refer to the tutorial by C. Burges [24].

Most classifiers, like the multi-layer perceptron model described above, solve the classification problem by optimizing their parameters such as to perform best on a given set of training data. This general strategy is referred to as empirical risk minimization. The support vector machine, on the other hand, is an example of a large-margin classifier, which pursues a different optimization strategy. SVMs find a $d$-dimensional plane $H$ which separates two classes (with labels -1 and 1) such that the distance of the samples $\mathbf{x}_i \in \mathbb{R}^d$ to $H$ is maximized. This distance to $H$ is called the margin.

Decision functions that are non-linear functions of the data are found by applying a function $\Phi : \mathbb{R}^d \to \mathcal{H}$, where $\mathcal{H}$ is an Euclidean space with a dimension that is usually much larger than $d$. $\Phi$ is chosen such that there exists a kernel function $K$ with $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^t \Phi(\mathbf{x}_j)$, i.e., inner products in the high dimensional space $\mathcal{H}$ can be computed via $K$. One function $K$ with this

property is the radial basis function (RBF) kernel,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma\right). \tag{2.13}$$

This property leads to the "kernel trick": Functions that depend on the data only via inner products of $d$-dimensional vectors are transformed to the high dimensional space $\mathcal{H}$ by substituting kernels such as (2.13) for the inner products. One function that fits into this scheme is the objective function of SVMs. Let $y_i \in \{-1, 1\}$ be the known class label of a sample vector $\mathbf{x}_i$ in the training set. For a $C$-SVM, where $C \in \mathbb{R}$ is a parameter controlling the number of outliers, the hyperplane $H$ through the origin with normal vector $\mathbf{w}$ (a shift from the origin is omitted here for simplicity of presentation) that maximizes the margin is found by solving the following quadratic program [24].

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & 0 \le \alpha_i \le C \\ & \sum_i \alpha_i y_i = 0 \end{aligned} \tag{2.14}$$

Here $\alpha_i > 0$ are Lagrangian multipliers enforcing the condition that the sample $\mathbf{x}_i$ lies on the positive side of $H$ if and only if $y_i = 1$ (except for some outliers limited by parameter $C$). New observations are classified by checking on which side of $H$ they lie.

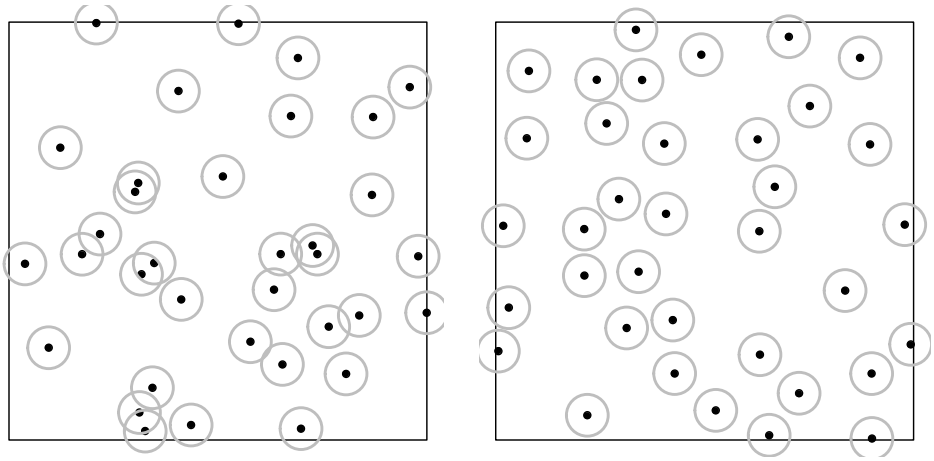
#### Using SVMs to compute class-posterior probabilities

The $C$-SVM described above assigns labels $-1$ or $+1$ to samples according to their position relative to a hyperplane $H$. This decision rule is given by the sign of the function $g(\mathbf{x}) = \sum_i y_i\alpha_i K(\mathbf{x}_i, \mathbf{x})$. What is required to use SVMs in the statistical localization model (2.2) are class-posterior probabilities $p(y|\mathbf{x})$.

Among the different algorithms for approximating class-posterior probabilities using SVMs, one widely used method (also implemented in the LibSVM software package [26]), is originally due to J.C. Platt [133]. It fits a sigmoid function with parameters $A$ and $B$,

$$p(y|\mathbf{x}) \approx \frac{1}{1 + \exp(Ag(\mathbf{x}) + B)}, \tag{2.15}$$

to the decision boundary. To get an unbiased estimate, labels obtained by cross-validation need to be used [133]. For the experiments reported later in this chapter, the algorithm reported in [110] was implemented based on the SVM-light software package [77], using 5-fold cross validation. It is an improved Newton algorithm for fitting (2.15) to SVM decision boundaries.



(a) Poisson process (b) Softcore process ($\kappa = \sigma = 0.1$)

Figure 2.4: Sample realizations of spatial point processes in the unit square with an associated circle of radius 0.05. While circular objects attached to a Poisson process overlap, pairwise interaction point processes, such as the softcore model, are suitable for modeling objects in images.

## 2.4 Point processes as object models

After having introduced two classifiers as possible image models, a form of object model that is suitable for describing circular objects in the plane will be introduced. As was argued earlier in this chapter, point process models are applicable for modeling randomly positioned objects in images. To provide the required background, this section provides a brief introduction to point process theory and their simulation. There exist excellent overviews of the field, such as the books by Møller and Waagepetersen [121] or van Lieshout [106].

Point processes describe random events which have a spatial distribution. If the events of a point process can be observed in a space $\mathcal{S}$, then it is said to be defined on that space. Point processes can be defined on quite general spaces (separable complete spaces [106], which basically implies that points can be counted anywhere within that space). For all material in this chapter, it will be the real plane with Euclidean metric, i.e., $\mathcal{S} \subset \mathbb{R}^2$. The simplest case, which will be used here, is the unit square, $\mathcal{S} = [0, 1]^2$.

Formally, a point process is a mapping $\mathcal{X}$ from probability space to the space of all countable point configurations in the plane. $\mathcal{X}$ will be assumed to be locally finite, i.e., the number of points in any bounded subset of $\mathcal{S}$ is finite, and simple, i.e., points will occur only once. Returning to the examples from Fig. 2.1, the centers of the objects (cells, fibers, metal particles) are assumed to be realizations from point processes in the plane. This section introduces one form of spatial point process which is a suitable model for such coordinates.

The completely random point process is the Poisson process, which is the random distribution of non interacting points: Let $W \subset [0, 1]^2$ be an observation window with area $\mu(W)$, and $N(W)$ denotes the number of points in $W$. Then $\mathcal{X}$ is a Poisson process with intensity $\beta$ if

1. the number of points in any observation window is a Poisson random variable, $N(W) \sim$

Poiss($\beta\mu(W)$), and

2. the number of points in disjoint sets are independent random variables.

A point process is called homogeneous if its distribution is invariant under translations. All processes investigated in this chapter are homogeneous. Fig. 2.4(a) shows an example of a realization from a Poisson process with an intensity of $\beta = 35$ in the unit square $[0, 1]^2$. Assume circular objects with radius 0.05 as in the plot. Then, a Poisson process is not a satisfactory model because it leads to significant overlap of objects.

Pairwise interaction processes, on the other hand, can model point patterns with regularities. These are special cases of the more general class of Markov point processes. Informally, these lift the independence assumption of Poisson processes, and allow points to interact. For a formal definition, see [106]. These interactions could be attractions between points, which lead to clustering, or inhibition between points, avoiding them to get too close. A pairwise interaction process takes into account interactions between at most two points, and has a conditional density of the form

$$p(X|n) = Z_\theta \prod_i \beta \prod_{i \neq j} h_\theta(\mathbf{x}_i, \mathbf{x}_j) = Z_\theta \exp\left\{ -\sum_i \log(\beta) - \sum_{j \neq i} \varphi(\mathbf{x}_i, \mathbf{x}_j;\ \theta) \right\}. \tag{2.16}$$

Here, $Z_\theta$ is the normalizing constant, $\beta$ remains the intensity parameter of a Poisson process, $\mathbf{x}_i \in [0, 1]^2$ are coordinates, $\theta$ is the parameter vector of the interaction potential $h(\cdot)$ and $h_\theta(\mathbf{x}_i, \mathbf{x}_j) = \exp\{-\varphi(\mathbf{x}_i, \mathbf{x}_j;\ \theta)\}$ by definition. Specification of different pairwise interaction models always reduces to choosing a form of $\varphi(\cdot)$. The potential needs to be chosen such that the $Z_\theta$ will be finite. A sufficient condition is that $\varphi(\mathbf{x}, \mathbf{y};\ \theta)$ is a positive, monotonically decreasing function of the distance $\|\mathbf{x} - \mathbf{y}\|$ between two points $\mathbf{x}, \mathbf{y} \in [0, 1]^2$ [121].

Computation of $Z_\theta$ would require integrating over all possible point configurations. For parameter estimation and simulation of $p(X|n)$, unnormalized densities are sufficient, see below, making it unnecessary to ever evaluate $Z_\theta$.

The density of a pairwise interaction point process in (2.16) is defined relative to a Poisson process with intensity $\beta$ [121]. It follows that $n$, the number of points on which that density is conditioned, is itself a Poisson random variable with parameter $\beta$. Therefore, the prior $p(n)$ in (2.2) is given by

$$p(n) = \frac{\beta^n}{n!} \exp(-\beta). \tag{2.17}$$

This is the prior that is used to model the number of objects in an image throughout this chapter. The parameter $\beta$ is the same one as in (2.16).

### 2.4.1 Objects of equal size – softcore interaction point processes

Depending on the form of the potential function $h$ in the pairwise interaction density (2.16), the object model introduced above can deal with different object shapes. This section introduces a potential for equally large circular objects with little overlap. Under this assumption, the softcore

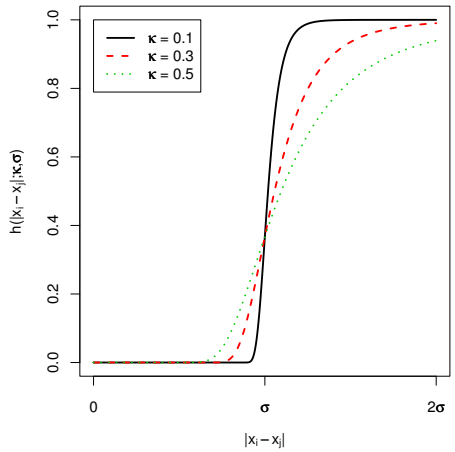



Figure 2.5: The softcore potential for fixed $\sigma$ and different values of $\kappa$, demonstrating the effects of the parameters of a softcore model. The parameter $\sigma$ governs how close to each other points from a softcore point process may get, and when $\kappa \to 0$, no overlap is allowed between two circular objects with radius $\sigma/2$ each. For some fixed value of $\kappa$, this potential can therefore be used to model circular objects with known radii by choosing appropriate values for $\sigma$.

point process with interaction potential

$$h(\mathbf{x}_i, \mathbf{x}_j;\ \kappa, \sigma) = \exp\left(-\left(\frac{\sigma}{\|\mathbf{x}_i - \mathbf{x}_j\|}\right)^{2/\kappa}\right) \tag{2.18}$$

is a more suitable model for the object locations $X$ than the Poisson process (Fig. 2.4(b)). It is an inhibitory point process with infinite interaction. From the application point of view, this potential function has a convenient parameterization: $\kappa$ determines the function's steepness, and as $\kappa \to 0$, the softcore potential approaches the form of a hardcore potential, i.e., a step function. This would correspond to an exact object size specification. With the potential's smooth shape, this restriction is less rigorous and allows for some size variations. The other parameter, $\sigma$, determines how close points may get, cf. Fig. 2.5. The sample realization in Fig. 2.4(b) in the unit square was generated using Markov chain Monte Carlo sampling, see Sec. 2.4.2, with parameters $\kappa = 0.1$ and $\sigma = 0.1$.

### Parameter estimation

The first step in training a softcore model for object locations is to infer the model parameter $\sigma$ from training data. The potential's steepness parameter $\kappa$, which is a modeling parameter for the amount of allowed overlap between neighboring objects, will not be inferred from data. It is set to $\kappa = 0.4$ for the remainder of this chapter. In experiments, it did not have a large impact on the localization results. For the parameter estimation methods discussed next, assume that a set $X$ of object locations in an image is given. Likelihood-based parameter estimators for pairwise interaction point processes are described in [9] and [37]. The general problem of these likelihood methods is that computation of the normalizing constant is intractable. It requires

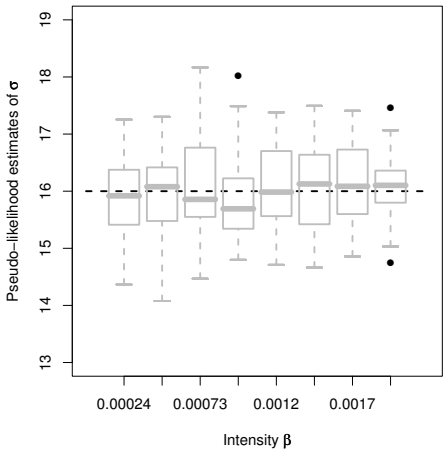



Figure 2.6: Evaluation of the pseudo-likelihood estimation method regarding parameter $\sigma$ of the softcore model at different densities $\beta$, the potential's remaining parameter $\kappa$ was fixed to 0.4. The boxplots are centered around the true value, indicated by the dashed line in this figure, which shows that the pseudo-likelihood method is suitable for estimating the parameter $\sigma$ of a softcore interaction point process at different densities. An increasing number of hardcore points in a $320 \times 320$ pixel box were simulated with a hardcore distance of 16. Each of the boxplots shows the distribution of the distance estimated in 50 realizations. The shown intensities $\beta$ correspond to fields from $n = 25$ to 200 points.

integration over all possible point configurations and depends on the parameters. Therefore, a number of approximate methods have been proposed and the result of a simulation study by Diggle et al. was that the pseudo-likelihood method is the most generally applicable one [37]. For an account of numerical approximation schemes for maximum likelihood estimation, and a treatment of Bayesian parameter estimation for Markov point processes, see [121, Ch. 6].

Pseudo-likelihood estimators were proposed by Besag [18], who introduced them as consistent parameter estimators in Markov random fields. The principle mechanism in any pseudo-likelihood estimator is the following. Instead of estimating parameter vector $\theta$ from the full likelihood $p(X|\theta)$, maximize a function of the conditionals $p(\mathbf{x}_i|\{\mathbf{x}_j\}_{i\neq j}, \theta)$ over $\theta$. This simplification has the important consequence that the normalizing constant cancels out. For a pairwise interaction process with density given in (2.16), the conditional density of a point at $\mathbf{u} \notin X$,

$$\lambda_\theta(\mathbf{u};\ X) := p(\mathbf{u}|X, n, \theta) = \frac{p(X \cup \{\mathbf{u}\}|n, \theta)}{p(X|n, \theta)} = \beta \prod_{i=1}^{n} h(\mathbf{u}, \mathbf{x}_i;\ \theta), \tag{2.19}$$

is known as the Papangelou conditional density. This term will reappear shortly in the contexts of Markov chain Monte Carlo simulation in Sec. 2.4.2 and of algorithms for approximating (2.3) in Sec. 2.5. Using $\lambda_\theta$, the pseudo-likelihood function PL for interaction point processes is defined as follows [37, 121].

$$\mathrm{PL}(\theta;\ X) := \exp\left(-\int \lambda_\theta(\mathbf{x}, X)d\mathbf{x}\right) \prod_{i=1}^{n} \lambda_\theta(\mathbf{x}_i|X \setminus \{\mathbf{x}_i\}). \tag{2.20}$$

PL is a concave function of $\theta$ and $\theta^{\mathrm{PL}} = \operatorname{argmax}_\theta \mathrm{PL}(\theta)$ is a consistent estimator, cf. [76]. Note that evaluation of (2.20) still requires numerical integration. For the results that will be presented later in this chapter, a quadrature integration scheme at 400 regularly distributed grid points along with the samples from the observed set $X$ was used. Details are described in [9] and the required numerical integration and optimization procedures are implemented in the *GNU R* [136] library *spatstat* [10].

To demonstrate the applicability of this estimation method in practice, consider Fig. 2.6. The estimation performance for the distance parameter $\sigma$ is adequate considering the wide range from quite sparse to dense point field realizations.

### 2.4.2 Markov chain Monte Carlo simulation

This section demonstrates how to generate realizations from interaction processes. Simulation is not the topic of this chapter, but the method introduced here is the basis for an algorithm that fits point fields to images, and which will be described in Sec. 2.5.1. An established tool for generating random samples from complex distributions such as those encountered in this chapter is Markov chain Monte Carlo (MCMC) simulation. The idea is to repeatedly and randomly modify the state of a system, in this case the positions of points in the plane, and to most likely accept those changes that lead to likely states according to the desired distribution. Consider for example the softcore model introduced above. MCMC algorithms are capable of constructing random samples such as those in Fig. 2.4(b) for a given set of model parameters ($\beta$, $\sigma$, $\kappa$).

For a detailed introduction to MCMC and the Metropolis-Hastings algorithm, see [31], different variants thereof for the case of interaction point processes are described e.g. in [106, Ch. 3] or [121, Ch. 7]. Let $p(X)$ be the the probability density function of a point field $X$ from (2.16). The corresponding unnormalized density will be denoted by $h$, defined by

$$p(X) = Z_\theta h(X), \tag{2.21}$$

which is used during the sampling process. MCMC simulations generate random sequences $X^1, X^2, \ldots$ of point fields using a transition function $p(X^t, X^{t+1})$. The transition function is a density giving the probability of moving from a given state at time $t$ to a new state (a new point configuration) at time $t+1$. If this sequence could be constructed such that for any $t > t_0$, the distribution of the point fields $X^t$ was the given $p(X)$, then any $X^t$ beyond time $t_0$ could be used as an independent sample from $p(X)$. This can be achieved if the transition function obeys

$$\int h(X^t) p(X^t, X^{t+1}) \mathrm{d}X^t = \int h(X^{t+1}) p(X^{t+1}, X^t) \mathrm{d}X^{t+1}, \tag{2.22}$$

a property which is called the detailed balance. In other words, it must be equally likely to move between two states in either direction (reversibility). The proof can be found e.g. in [31]. This leaves open the question of how to find a transition function that fulfills the detailed balance condition. One widely applicable answer is the Metropolis-Hastings algorithm. There, the transition function $p(X^t, X^{t+1})$ from above is modified to perform the transition only with probability

$\alpha(X^t, X^{t+1})$,

$$p(X^t, X^{t+1}) := q(X^t, X^{t+1})\alpha(X^t, X^{t+1}), \tag{2.23}$$

where $q$ is the probability of generating the new state $X^{t+1}$. The choice of parameters $\alpha$ is what makes the Metropolis-Hastings algorithm work. To derive $\alpha$, let $u^t$ and $u^{t+1}$ be random vectors and $g(X^t, u^t)$ be a differentiable, deterministic and injective function that computes a new proposal state, i.e., $g(X^t, u^t) = (X^{t+1}, u^{t+1})$. Similarly, let $g^{-1}(X^{t+1}, u^{t+1}) = (X^t, u^t)$ denote the corresponding transition in reverse direction. Next, substitute $g$ and (2.23) into (2.22).

$$\begin{aligned} &\int h(X^t)q(X^t, X^{t+1})\alpha(X^t, X^{t+1})\mathrm{d}X^t \\ &\qquad = \int h(X^{t+1})q(X^{t+1}, X^t)\alpha(X^{t+1}, X^t)\mathrm{d}X^{t+1} \end{aligned} \tag{2.24}$$

$$\begin{aligned} \Leftrightarrow &\int\int h(X^t)q\left(X^t, g_1(X^t, u^t)\right)\alpha\left(X^t, g_1(X^t, u^t)\right)\mathrm{d}X^t\mathrm{d}u^t \\ &\qquad = \int\int h(X^{t+1})q\left(X^{t+1}, g_1^{-1}(X^{t+1}, u^{t+1})\right)\alpha\left(X^{t+1}, g_1^{-1}(X^{t+1}, u^{t+1})\right)\mathrm{d}X^{t+1}\mathrm{d}u^{t+1} \end{aligned} \tag{2.25}$$

Here, $g_1$ and $g_1^{-1}$ denote the state components of $g$ and $g^{-1}$, respectively. A change of variables on the right hand side leads to

$$= \int\int h\left(g(X^t, u^t)\right)q\left(g(X^t, u^t), X^t\right)\alpha\left(g(X^t, u^t), X^t\right)|J|\,\mathrm{d}X^t\mathrm{d}u^t, \tag{2.26}$$

where $J = \frac{\partial g(X^t, u^t)}{\partial X^t \partial u^t}$ is the Jacobian that enters due to the rules of multidimensional integration. By setting $\alpha(X^{t+1}, X^t) = 1$, i.e., choosing the largest possible acceptance probability, the Hastings ratio $r$ is found as the value of $\alpha(X^{t+1}, X^t)$ that forces the detailed balance condition in (2.22) to hold.

$$\alpha(X^t, X^{t+1}) = \frac{h(X^{t+1})q(X^{t+1}, X^t)}{h(X^t)q(X^t, X^{t+1})} \cdot \left|\frac{\partial g(X^t, u^t)}{\partial X^t \partial u^t}\right| \tag{2.27}$$

In this expression, $\alpha$ may exceed one. The probability for performing the transition from state $X^t$ to $X^{t+1}$ is therefore set to $r(X^t, X^{t+1}) = \min\left\{1, \alpha(X^t, X^{t+1})\right\}$.

To translate these equations into an algorithm for simulating a point field under a softcore interaction model, consider some initial point field $X^0$ with $n$ points. The idea is to modify the points in $X^0$ by moving one point $\mathbf{x}_i \in X^0$ by a random vector $\delta$, where the index $i$ is chosen randomly. This change is then either accepted or rejected according to the mechanism that was just described, and the procedure continues from this, possibly new, point field.

Note that point processes $X$ are commonly represented by sets, as simple point processes may not contain any duplicate points. For the purposes of the Metropolis-Hastings algorithm introduced next, however, it is necessary to access and modify an individual point $\mathbf{x}_i$ at position

$i$ in a sequence. Therefore, the algorithm operates on an ordered sequence $\tilde{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ of the points in the point field $X$.

The random transition component at time $t$ is $u^t = (i, \delta)$. Except for an addition in dimension $i$, $g$ is the identity in all remaining $n-1$ dimensions and the Jacobian therefore reduces to $|J| = 1$. Furthermore, for a symmetric distribution of $\delta$ ($p(\delta) = p(-\delta)$), the $q$-terms in (2.27) will cancel out. This leads to the following Metropolis-Hastings algorithm.

1. Initialize $X^0$ to some first configuration of $n$ points.
2. At time $t$, draw a random element $\mathbf{x}_i$ from $\tilde{X}^t$.
3. Randomly select a vector $\delta \in \mathbb{R}^2$ from a uniform distribution in a circle of radius $\delta_{\max}$.
4. Compute the Hastings ratio $r = \min\left(1, \frac{h(X^t \setminus \mathbf{x} \cup (\mathbf{x}+\delta)}{h(X^t)}\right)$.
5. Draw a uniformly distributed random number $u$ between zero and one.
6. If $u < r$, set $X^{t+1} = X^t \setminus \{\mathbf{x}_i\} \cup (\{\mathbf{x}_i + \delta\})$, otherwise, set $X^{t+1} = X^t$.
7. Continue at step 2.

Because of the detailed balance being fulfilled by construction, this algorithm will converge to the invariant distribution $p(X)$, i.e., beyond some point $t_0$, each generated sample $X^t$ is an independent sample from $p(X)$. Note, however, that this algorithm samples from the conditional $p(X|n)$. What is required to sample from the joint density $p(X, n)$ is reversible jump Markov chain Monte Carlo (RJMCMC) [65], a device allowing one to define densities on spaces of fixed but unknown dimension. More practically, one needs to add proposal densities which add or delete points in order to search the state space for different $n$. Details on RJMCMC will be given in Sec. 2.5.1.

### 2.4.3 Objects of varying sizes – marked softcore interaction point processes

The softcore potential described above models objects of fixed size with its distance parameter $\sigma$. While this model is appropriate for treating a wide range of object types, cf. Sec. 2.7, it lacks the capability to express properties of individual objects. Such properties could be the size, shape, or orientation of an object. Depending on the application, these additional parameters may be useful either in determining the object locations, or because they themselves represent a quantity of interest. With marked point processes [106, 121], the point process literature provides an appropriate tool for modeling such object properties.

Marked point processes are point processes with "marks" attached to each individual point. The marks are random vectors, themselves, and their distribution can be either dependent on or independent of the point process. Formally, the point processes introduced above are generalized to marked point processes $X = \{(\mathbf{x}_i, m_i)\}_{i=1,\ldots,n}$, i.e., sets of tuples which each contain a point $\mathbf{x} \in [0,1]^2$ and a vector-valued mark $m \in \mathbb{R}^{d_m}$ ($d_m$ denotes the dimensionality of the marks).

Building on the material earlier in this chapter, the special case of marked pairwise interaction processes [121] with conditional density

$$p(X|n) = Z_\theta \prod_i \beta p(m_i) \prod_{i \neq j} h_\theta((\mathbf{x}_i, m_i), (\mathbf{x}_j, m_j)) \tag{2.28}$$

is of particular interest. Note that this density belongs to the special case where the distribution $p((\mathbf{x}, m)) = \beta p(m)$ factorizes, i.e., the marks are independent of the point process. Here, $\beta$ is once again the intensity parameter of the underlying Poisson process.

The marks $m$ are not limited to any specific form, and in fact the restriction made here for the marks to be real-valued vectors is a restriction made by choice. By attaching a mark $m$ to each point $\mathbf{x}$, the posterior model from Eq. (2.2) needs to be modified. Later, marked point processes will be applied to the localization of circular objects of varying sizes. There, the mark will describe object size and the image likelihood term depends on the marks. Therefore, rewrite (2.2) as

$$p(X, n|f) \propto p(f|X, n)p(X, n) = p(X|n)p(n) \prod_{i=1}^{n} p(f|\mathbf{x}_i, m_i), \tag{2.29}$$

with $p(X|n)$ given by (2.28). In the same spirit as in Sec. 2.2.1, define a function $y : [0, 1]^2 \times \mathcal{M} \rightarrow \{0, 1\}$ on the joint coordinate and mark space. $y$ assumes the value 1 whenever an object with mark $m \in \mathcal{M}$ is located at $\mathbf{x}$ in some given image $f$. Then, $y$ is again a sufficient statistic for $X$, leading to the revised model

$$= p(X|n)p(n) \prod_{i=1}^{n} p(y(\mathbf{x}_i, m_i) \neq 0|\mathbf{x}_i, m_i). \tag{2.30}$$

Compared to the unmarked point process introduced earlier, to apply this marked point process model requires modifications of the models and algorithms described above.

First, the likelihood term $p(y(\mathbf{x}, m) \neq 0|\mathbf{x}, m)$ depends not only on the object location $\mathbf{x}$, but also on its mark $m$. For this purpose, the classifiers introduced in Sec. 2.3 are again suitable. As above, $M \times M$ pixel image patches at coordinate $\mathbf{x}$ are extracted from image $f$. Additionally, the mark $m$ is added to the classifier's input. For SVMs, this is achieved by adding $d_m$ entries to the $M^2$ input vector. Due to the special architecture of convolutional neural networks, that is not possible, there. Instead, the input $m$ bypasses the convolutional layers and is connected directly to the hidden layer, cf. Sec. 2.3.1.

Second, MCMC sampling of marked point processes needs to sample not only the coordinates $\mathbf{x}$, but also the marks $m$. To this end, random shifts of $\delta_m$ of $m$ are generated in step 3 of the algorithm listed in Sec. 2.4.2. As the restriction of $m$ being vector-valued was made, these shifts $\delta_m$ are vectors from the same space as $m$, $\mathbb{R}^{d_m}$. As the distribution of the marks is contained in the marked point process density (2.28), and consequently also in the unnormalized density $h$ defined in Sec. 2.4.2, the Metropolis-Hastings rule will guarantee that this slightly modified MCMC sampler will sample from the marked point process density (2.28).

**Softcore interaction potential revisited**

One example of the marked point process model that was just introduced is to remain with circular objects, as earlier in this chapter, but to lift the restriction of all objects having equal radii. To achieve this, the radii are attached as marks to each point. Let $r_i \in \mathbb{R}$ be the radius of an object at $\mathbf{x}_i$, resulting in the marked point process $X = \{(\mathbf{x}_i, r_i)\}_{i=1,\ldots,n}$ with $\mathcal{M} = \mathbb{R}$. Assuming objects with little overlap as above, the softcore model is modified to

$$h((\mathbf{x}_i, r_i), (\mathbf{x}_j, r_j);\ \kappa) = \exp\left(-\left(\frac{r_i + r_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}\right)^{2/\kappa}\right). \tag{2.31}$$

Hence, the softcore potential fulfills the same function as in the case of circular objects with fixed radii, but the allowed distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$ depends on the marks. Note that the object sizes $\{r_1, r_2, \ldots, r_n\}$ need not be known, and any prior knowledge should be expressed by their prior distribution $p(r_i)$ in (2.28). Other, more complex types of marks for describing geometric arrangements in images have also been used [39, 107, 128].

## 2.5 Model-based multiple object localization

What remains to be developed are methods for applying the statistical model described up to this point. In the following, it will be assumed that the model $p(X, n|f)$ has been fit to some training data, using pseudo-likelihood estimation for the object model and MLP or SVM training for the image model. Hence, given an image $f$, $p(X, n|f)$ can be evaluated for any $n$ points in the set $X$.

The first of the two algorithms introduced here is an extension of the MCMC sampler from Sec. 2.4.2. Instead of generating samples from $p(X|n)$, it simulates from the joint posterior density $p(X, n|f)$. Since the number $n$ of objects is unknown, the algorithm needs to be extended to "jump" between states in different dimensions, i.e., it needs to add and remove object locations to and from $X$. MCMC samplers of this form have been used in [8] for object detection. Furthermore, an annealing schedule is added to the sampler in order to guide the sequence into an optimal state [86]. M. van Lieshout proposed and analyzed such annealing algorithms for object recognition [105]. Similar algorithms have also been used elsewhere, e.g. [39, 154].

MCMC sampling with annealing is computationally expensive. Furthermore, it is a randomized algorithm by definition, making the exact results irreproducible.

As an alternative, a second, deterministic algorithm will be proposed, which is similar to J. Besag's iterated conditional modes (ICM) [19]. The ICM method was developed in an image restoration context. It iteratively updates pixel labels in cycles by assigning pixel labels according to the maximum conditional density in that point. The algorithm introduced below is similar to ICM, as it iteratively adds points to the set of solutions. At each iteration, the point with the maximal conditional probability given all previously selected points is chosen, and will therefore be referred to as greedy algorithm. Such algorithms have also been used in [8] to fit point process models to images. Due to $n$ being unknown, the question of determining when to stop adding points to $X$ is crucial.

### 2.5.1 Localization by reversible jump Markov chain Monte Carlo

Next, the algorithm from Sec. 2.4.2 is extended to a reversible jump Markov chain Monte Carlo (RJMCMC) algorithm [65] for sampling from the posterior $p(X, n|f)$. Instead of only randomly moving points within the set of locations $X$, RJMCMC also adds and removes points, thus "jumping" between different dimensions of $X$. Good introductions to RJMCMC can be found in [69] and [164].

In the case of the MCMC sampler introduced above, $p(X|n)$ was defined on $\mathbb{R}^d$, with the number $n$ of points being known, and the points in $X$ have been restricted to the unit square $[0,1]^2$. $X$ takes values from a space that is usually denoted by $N_{lf}$ [106, 121], which is the set all locally finite point configurations. For an exact definition of $N_{lf}$, see [106, 121]. Let $W$ again be the bounded observation window, i.e. $W = [0,1]^2$. Then, $N_{lf}$ can be split into a set of locally finite point configurations in $W$, $N_{lf} = \bigcup_n N_{lf,n}$, where each $N_{lf,n}$ represents the set of point configurations containing $n$ points, i.e., $N_{lf,n} = \{X|N_X(W) = n\}$. Therefore, $p(X|n)$ operates on such a space $N_{lf,n}$, and as $n$ is known, $N_{lf,n}$ has a fixed dimension. Sampling from the joint density $p(X, n)$ with $n$ being unknown requires the concept of union spaces, introduced by P. Green [65]. For the joint density $p(X, n)$, this union space has the form $\bigcup_n (N_{lf,n})$. Then, for any given $n$, $p(X|n)$ can again be evaluated. Therefore, it is possible to add points to or to delete points from $X$, thus changing $n$, which leads to a new conditional density defined in the corresponding subspace.

At each time step, the RJMCMC sampler picks one of three actions (move, birth or death) with probability $p_{\text{move}}$, $p_{\text{birth}}$ and $p_{\text{death}}$. The detailed balance condition must hold for any of the three transition types. This is achieved by performing each individual transition according to the Hastings ratio, which requires some attention to the form of the transition functions $g$ being chosen. As for the MCMC sampler in Sec. 2.4.2, let the points in $X$ be arranged in an ordered sequence $\tilde{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$.

- *Move event*: Let $u_{\text{Move}} = (i, \delta)$, where $\delta$ is a random vector. Then the definition of $g$ and the Hastings ratio of the fixed dimension MCMC from Sec. 2.4.2 can again be used.

- *Birth event*: Let $u_{\text{Birth}} = (i, \mathbf{x}^n)$. A birth event raises the dimension $n$ of $\tilde{X}^t$ to $n+1$ of $\tilde{X}^{t+1}$. For the algorithm to work correctly, the so-called dimension matching condition [65] must hold, which states that the sum of dimensions of the state vector $\tilde{X}$ and the random transition vector $u$ must be the same in the forward and reverse directions. Therefore, define $g_{\text{Birth}}(\tilde{X}, u_{\text{Birth}}) := (\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}^n, \mathbf{x}_i, \ldots, \mathbf{x}_n, i)$. The dimension matching condition is fulfilled (the state sequence dimension grows by one while the random vector component in the reverse direction loses the novel coordinate $\mathbf{x}^n$, see below). As $g_{\text{Birth}}$ performs merely a permutation of its vectorial argument, the Jacobian disappears from the Hastings ratio ($|J| = 1$).

- *Death event*: Following the same scheme as for the birth event, let $u_{\text{Death}} = (i)$ and $g_{\text{Death}}(\tilde{X}, i) := (\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n, i, \mathbf{x}_i)$. The dimension matching condition is fulfilled and $|J| = 1$, as can be seen by the same arguments as made above.

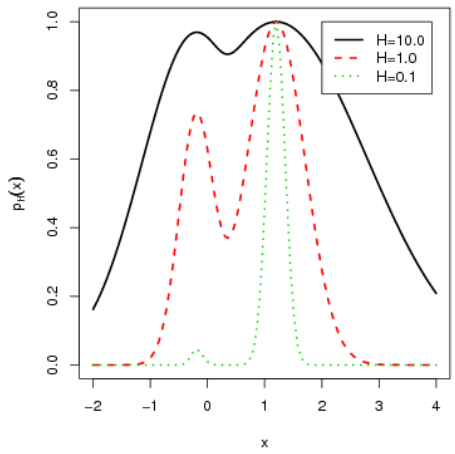



Figure 2.7: A bimodal Gaussian mixture example of a temperature modified posterior, scaled to maximum 1, which illustrates how annealing drives the sampling sequence towards the global maximum of the distribution. Decreasing parameter $H$ during an annealing process leads to the probability mass being concentrated narrowly around the location of the maximum.

Note that here $g_{\text{Birth}}^{-1} = g_{\text{Death}}$ and $g_{\text{Death}}^{-1} = g_{\text{Birth}}$, by definition. When using the Hastings ratio (2.27) in the RJMCMC sampler, the chain will converge to the desired distribution $p(X, n|f)$. After some $t_0$ initial steps, any element produced by the chain will be a sample from that distribution. But which sample will maximize the posterior? Simulated annealing [86] is a proper tool to avoid making that choice explicit. Van Lieshout modified the posterior distribution in (2.2) to

$$p_H(X, n|f) \propto \{p(f|X, n)p(X|n)p(n)\}^{1/H}, \tag{2.32}$$

where $H > 0$ is the temperature parameter [105]. The effect of decreasing $H$ from large to low values is visualized in Fig. 2.7. Early in the sampling process ($H >> 0$), many unlikely configurations $X$ are generated, thus exploring the state space. Towards the end ($H \to 0$), proposals will only be accepted if they are close to the global maximum of the distribution.

### 2.5.2 Localization by greedy search

An alternative localization algorithm is introduced next which, in contrast to RJMCMC, is entirely deterministic. In image restoration, the task is to recover image data $\hat{f}$ from a noisy observation $f$. Assuming that the image pixels form a Markov Random Field (MRF) [59], an iterative algorithm for recovering the original pixel values, iterated conditional modes (ICM), was proposed by J. Besag [19]. At each iteration of ICM, each pixel is assigned the value maximizing its posterior probability given the values of all neighboring pixels. This procedure is repeated until convergence.

For the problem of computing object locations under the model $p(X, n|f)$, an analogous algorithm chooses, at iteration $t$, object locations from the conditional $p(\mathbf{x}|X^t, n^t, f)$. For the purpose of this section, let $F(i, j) = f(i\Delta_x, j\Delta_y)$ be a quantized image with $0 \leq i\Delta_x, j\Delta_y \leq 1$ and $i, j \in \mathbb{Z}$. $\Delta_x$ and $\Delta_y$ are the quantization steps in the horizontal and vertical directions, respectively.

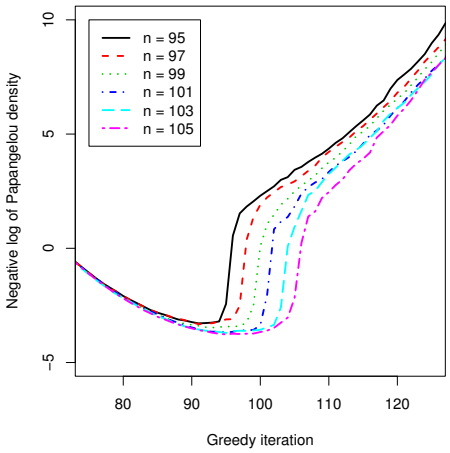



Figure 2.8: Negative logarithm of the Papangelou conditional density over iterations of the greedy algorithm, averaged over 25 trials. The prior for $n$ was set to a mean intensity of $\beta = 100$ objects for all six experiments, while the simulated number of objects varied between 95 and 105. These functions have a change in the sign of slope when reaching the true object count, which is a suitable feature to determine termination of the greedy object localization algorithm.

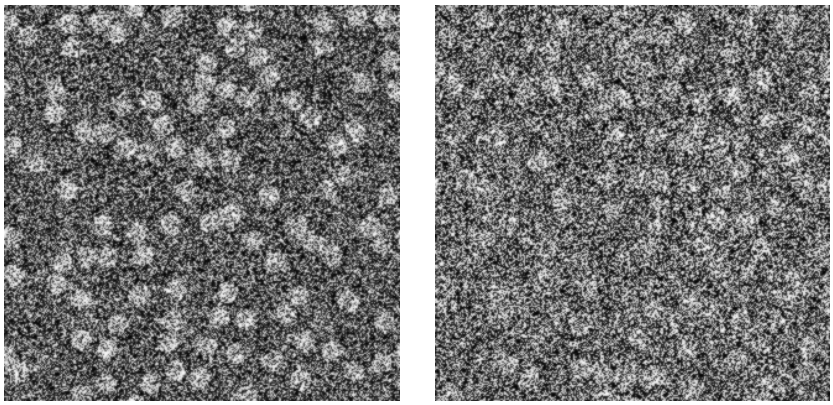
Suppose that at iteration $t$, the set $X^t$ contains $n^t$ points. The algorithm proceeds iteratively, selecting the point $(i,j)^{t+1} = \operatorname{argmax} p((i,j)|X^t, n^t, f)$. Selection of a new point is a linear time operation in the number of pixels, i.e., each iteration is in $\mathcal{O}(\Delta_x^{-1} \cdot \Delta_y^{-1})$. Therefore, it is a greedy algorithm that selects points according to the Papangelou conditional density, cf. Sec. 2.4.1, given by

$$p(\mathbf{x}|X^t, n^t, f) = \frac{p(X^t \cup \{\mathbf{x}\}|n^t + 1, f)}{p(X^t|n^t, f)}. \tag{2.33}$$

Because of $n$ being unknown, the stopping rule is an important point when applying this greedy algorithm. Consider Fig. 2.8, which shows the negative logarithm of (2.33) for six different values of $n$. For each value of $n$, the corresponding curves show a sign change in the first derivative in the corresponding iteration.

For the results reported below, the termination condition was a decrease of the Papangelou conditional density in five consecutive iterations. Using such a stopping rule, this greedy multiple object localization algorithm automatically chooses a number $n$ of objects according to the posterior probability $p(X, n|f)$.

This concludes the description of the proposed model of multiple objects in the plane, which combines point process models with machine learning and localization algorithms. This combination results in a trainable system for multiple object localization. Next, this system will be evaluated.



(a) PSNR 10.63 dB (b) PSNR 9.14 dB

Figure 2.9: The images used for experimental evaluation are 320×320 pixels large, containing $n$ solid discs with radius 8 pixels each, where $n$ is a Poisson random number with mean $\beta = 100$. Independent normally distributed noise was added to each pixel, followed by a $3 \times 3$ low-pass filter given by Eq. (2.34). Depending on the amount of additive noise, this process results in images at different noise levels, which is measured in terms of the PSNR.

## 2.6 Results

Experiments on simulated data are performed to quantitatively evaluate and compare the proposed models for the image likelihood and methods for optimization against each other, and also against selected alternative control methods. The insights gained from these experiments will be used to settle on one combination of the above described algorithms, resulting in a system that is then ready to be applied to real data, later in this chapter.

### 2.6.1 Design of experiment

The experimental evaluation conducted in this section will be performed on simulated data. White, non-overlapping discs on black background are used to model a multiple object localization problem. White Gaussian noise is added to each pixel, followed by a low-pass filter given by

$$\begin{pmatrix} 0.05 & 0.1 & 0.05 \\ 0.1 & 0.4 & 0.1 \\ 0.05 & 0.1 & 0.05 \end{pmatrix} \tag{2.34}$$

.

This procedure produces images with correlated noise such as those in Fig. 2.9. Changes to the variance of the additive noise component result in images at different noise levels. For quantifying image quality, the peak signal-to-noise ratio (PSNR) is an established measure, especially in the context of image compression, see e.g. [158, Ch. 1]. Let $f_{\max}$ denote the maximal signal value, i.e.,

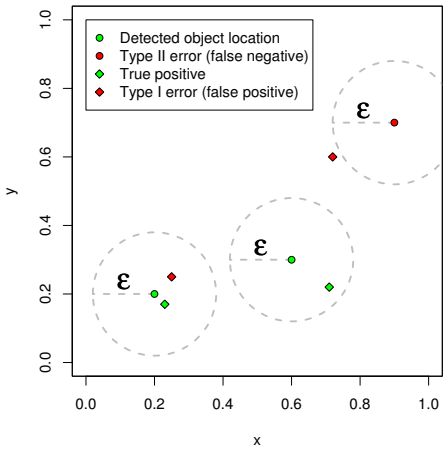



Figure 2.10: For evaluating localization performance, estimated object locations are counted as true or false detections, depending on whether or not they are within a distance $\epsilon$ of a true object location. Only one estimated location can be assigned to an object at a time.

the highest pixel value in a noiseless image. The PSNR,

$$\mathrm{PSNR} = 10 \log_{10} \frac{f_{\max}^2}{\mathrm{MSE}}, \tag{2.35}$$

relates $f_{\max}$ to the pixel-wise mean square error (MSE) between noiseless and distorted image. $f_{\max} = 200$ for the experiments on simulated data reported in this section. This particular choice of $f_{\max}$, and a corresponding value $f_{\min} = 50$, allowed to store the sum of the original pixel values and the random noise component in 8 bit data format without truncation, in most cases.

All experiments reported in this section were performed on 320×320 pixel images containing non-overlapping solid white discs with radius 8 pixels, generated using random sequential adsorption (RSA) [161]. In RSA simulation, random object sites are sequentially added to $X$ under the condition that the new points do not intersect any of the existing points' discs in $X$. This process repeats until reaching the required number of objects, $n$, or until no more points can be added. $n$ is a Poisson random variable with mean $\beta = 100$.

### 2.6.2 Error measures

Result of the simulation process outlined above is an image $f$ and a set $X^*$ of object locations. Similarly, the statistical multiple object localization method proposed in this chapter yields a set $X$ of estimated object sites. Fig. 2.10 illustrates the types of errors that may occur.

These errors can be identified by solving the minimum cost assignment problem on the distance matrix between the points in $X^*$ and $X$. The Hungarian algorithm solves this problem in polynomial time, resulting in a unique assignment between true and estimated points [88]. If $|X^*| \neq |X|$, either some estimated or true object locations remain unassigned, resulting in false positive or false negative errors, respectively. Furthermore, by introducing a distance threshold $\epsilon$, only those estimated locations that hit one of the simulated objects will be accepted. As the

radius of the simulated objects is 8 pixels, $\epsilon = 7$ is used throughout this section. For the results on real-world application data that will be reported in Sec. 2.7, this threshold $\epsilon$ will be chosen according to the object size in each dataset. An open-source implementation of the Hungarian algorithm by B.M. Clapper[1] was used for computations.

The receiver operating characteristic (ROC) is an established graphical method for comparing the performance of different classifiers [70, Ch. 9]. It analyzes the trade-off between true and false positives as a function of the system's sensitivity. Usually, the class-posterior probability is thresholded at different levels. The higher this threshold, the less false positive errors are registered, since only very few observations will be accepted as positive classifications.

To transfer this concept to multiple object localization, an analogous parameter governing the trade-off between true and false positives is required. By modifying the prior $p(n)$ from a small to a large number of expected objects in image $f$, the size of the resulting set $X$ will increase. As both the Metropolis-Hastings and the greedy algorithms choose points with high-probability first, a behaviour similar to that of thresholding a class-posterior probability in classification tasks is expected. However, note that especially for the RJMCMC sampler the functional dependence between prior $p(n)$ and the resulting number of points $|X|$ will not necessarily be monotonic.

Therefore, the ROCs shown below were forced to be monotonically increasing by discarding points for which the true positive rate was decreasing with an increasing false positive rate.

Within these limitations, the ROC is a suitable tool for comparing different localization algorithms. The area under the ROC curve (AUC) is a common scalar figure for comparing the performance, assuming a value of 1 for a perfect classifier. The AUC will be used below to achieve a ranking of different methods.

### 2.6.3 Control methods

Three standard image analysis algorithms will be used as references for judging the quality of the results of the proposed method. Especially in the medical image processing literature, thresholding [1, 152, 73, 124], morphological segmentation (watershed transform and others) [16, 30, 56, 84, 98, 115, 123], and MLPs [151, 159] are frequently used for object localization and counting. None of these methods has an explicit model of object geometries or the spatial arrangement of objects in images. Therefore, these will be used as control methods against the proposed statistical object localization method, both as control experiments against the state of the art and in order to demonstrate the benefits of using spatial models.

- *Isodata thresholding* – Isodata thresholding is an iterative algorithm for binarizing images. After some smoothing, it is applied to obtain a binary image, connected components are identified and the corresponding coordinates computed. This procedure is implemented using the ImageJ software package [138], with the exact source code listed in App. B.

- *Watershed segmentation* – Watershed segmentation is a morphological image transformation that segments an image into disjoint regions. Starting from a binarized image, a distance map from all foreground pixels to the background region is computed. This distance map is used

[1] `http://www.clapper.org/software/python/munkres/`

by the watershed transformation to estimate the extent of individual objects. Watershed segmentation has the capability to separate overlapping objects. This control method is also implemented using the ImageJ software package [138], with the exact source code listed in App. B.

- *Nonmaximum suppression* – The output of the convolutional neural network or support vector machine classifiers are approximations of the image likelihood $p(f|\mathbf{x})$ at each pixel $\mathbf{x}$. Without applying a spatial constraint in form of a prior, the object locations can be found by identifying the local maxima in this likelihood function. Nonmaximum suppression finds these local maxima by suppressing all pixels that are not at least as likely as all pixels in their surround.

For the isodata and watershed controls, ImageJ (version 1.41c) was chosen for its wide-spread use in medical image analysis.

### 2.6.4 Localizing objects with equal sizes

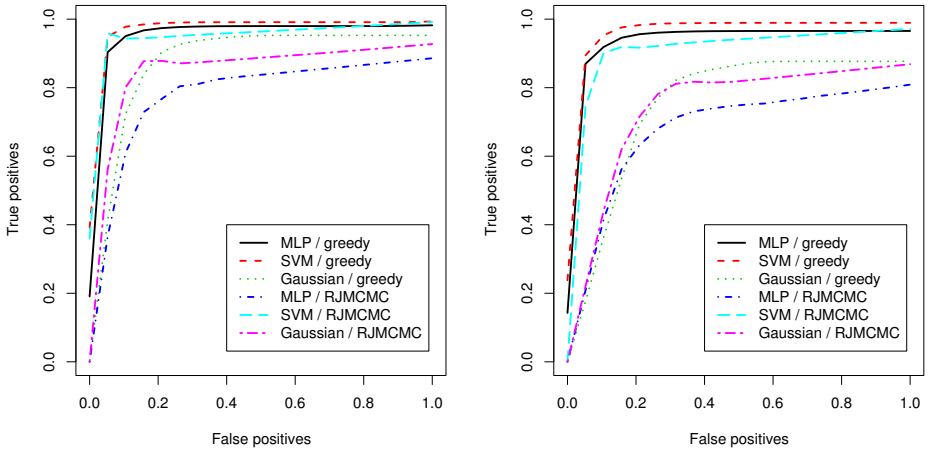
This section presents quantitative localization results for circular objects with equal sizes. The presentation of the empirical results will be split in two parts. First, the combinations to compute the image likelihood term (Gaussian noise model, MLP, SVM) and to select object locations according to the spatial model (RJMCMC sampler with annealing, greedy) are tested among each other. Second, the variant of statistical multiple object localization chosen based on these results is compared to the three control methods introduced above. All evaluations use the ROC measurements introduced in Sec. 2.6.2.

#### Experimental details

$320 \times 320$ pixel training and testing images containing a mean number $\beta = 100$ of objects were generated as described in Sec. 2.6.1. Square image patches containing the raw gray values were extracted from the images. These $28 \times 28$ pixel large image patches provide the input to the likelihood estimation methods. No feature extraction was performed. For the SVM model, this results in 784-dimensional input vectors. The training data consisted of positive examples at the true object locations and their immediate neighbours. Negative background examples were collected by randomly sampling patches from all remaining background pixels (no closer than three pixels to any object location). As described in Sec. 2.3.1, all patches are randomly distorted (rotation and shear) to improve generalization performance.

The MLP model used here was a convolutional neural network, cf. Sec. 2.3.1, with a total of 50 convolutional features followed by a fully connected layer of 20 units. Backpropagation training with fixed rate and early stopping was used (35 iterations). For the SVM model, the SVM-light software [77] with the extensions outlined in Sec. 2.3.2 was used with a polynomial kernel (degree 2).

The annealing schedule of the RJMCMC sampler was tuned such as to yield similar runtimes as the deterministic greedy algorithm. The temperature parameter $H$ in (2.32) decreased from




(a) PSNR 9.90 dB, $\epsilon = 7$ (b) PSNR 9.48 dB, $\epsilon = 7$

Figure 2.11: Performance comparison of different implementations of the statistical multiple object localization method, at two noise levels. True and false positive rates were evaluated on 100 test images, ROCs were obtained by varying the intensity parameter $\beta$ from 0 to 200. Overall, the greedy algorithm for selecting the points $X$ outperforms the RJMCMC sampler, with a slight advantage for the likelihood function computed using the SVM.

4.0 to 0.3. The runtime of either algorithm remained below 30 seconds per image on a 2.4 GHz CPU.

### Different implementations of the proposed method

Fig. 2.11 shows the performances of different implementations of the proposed method, leading to three observations, which hold across all noise levels, two of which are shown here.

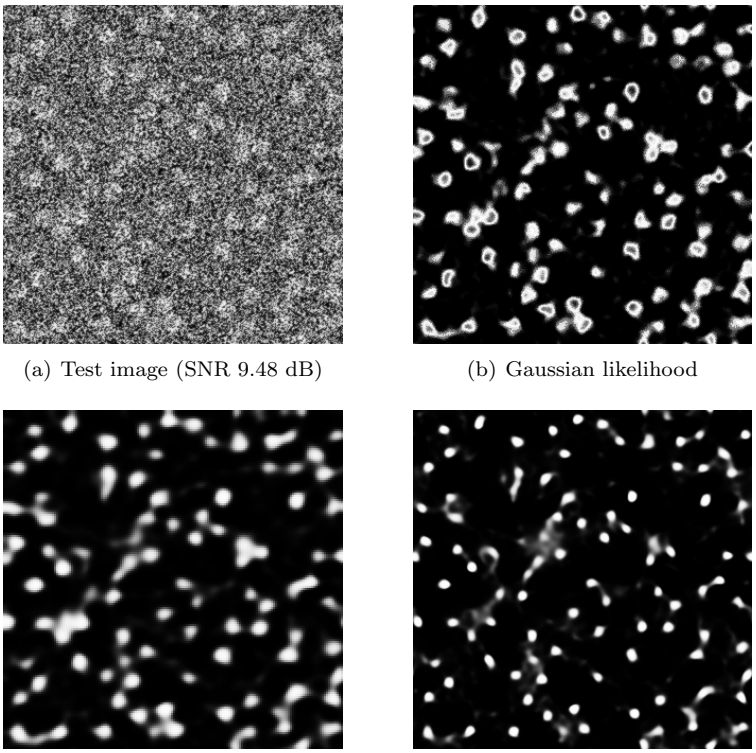
1. Trained image models are superior to the fixed likelihood model based on an additive noise assumption, especially when the noise level increases.

2. Among the two tested trainable image models, SVMs outperform MLPs.

3. The deterministic greedy algorithm outperforms the stochastic RJMCMC sampler, but the difference in performance also depends on the chosen representation of the image model.

These results justify the combination of machine learning and spatial interaction point process models proposed in this chapter. By using trainable models such as support vector machines or multilayer perceptrons for the image model in (2.2), spatial statistics models become applicable to more complex computer vision tasks.

One peculiar feature about both plots in Fig. 2.11 is that while the greedy algorithm generally outperforms the RJMCMC sampling method, the margin of improvement is larger for the MLP-computed likelihood than for the SVM-computed and Gaussian ones. To explain this difference, consider Fig. 2.12, which depicts a test image at a noise level of 9.48 dB along with the three



(a) Test image (SNR 9.48 dB) (b) Gaussian likelihood

(c) MLP likelihood (d) SVM likelihood

Figure 2.12: The likelihood maps computed using the three alternative methods described in this chapter have different characteristics. Note that the SVM method results in narrow peaks at object locations. In practice, this narrows down the state space explored during RJMCMC sampling, thus explaining the image model-dependent differences between the greedy and RJMCMC algorithms in Fig. 2.11.

differently computed likelihood maps. The SVM method results in narrower peaks at the true object locations.

The greedy algorithm performs line-scans to add one point $\mathbf{x}$ to the set $X$ at a time, considering every pixel in every iteration. In every step, it optimizes the trade-off between the likelihood of an object location and the spatial arrangement of the current points in the set by an exhaustive search (Sec. 2.5.2). Thus, only points with locally maximal likelihood are selected.

RJMCMC sampling, on the other hand, maintains a set of points $X$, and randomly adds, deletes, or moves points (Sec. 2.5.1). The annealing mechanism is responsible for easing the sequence of generated states into solutions that are more probable according to the given model. When using the SVM-computed likelihood model in Fig. 2.12(d), the effective search space is smaller compared to the MLP-computed one. Thus, more sampling steps are required to reach a good point configuration when using MLPs as image models. This explains why the performance difference between greedy search and RJMCMC sampling is smaller for the SVM-based model.

Considering the area under the curve (AUC) as a performance measure leads to an ordering of the methods, which is consistent across both tested noise levels, cf. Tab. 2.1. Clearly, the SVM-

| Method | 9.90 dB AUC | | | Rank | 9.48 dB AUC | | | Rank |
|---|---|---|---|---|---|---|---|---|
| SVM / greedy | 0.960 | ± | 0.00405 | 1 | 0.958 | ± | 0.00229 | 1 |
| MLP / greedy | 0.946 | ± | 0.00236 | 2 | 0.927 | ± | 0.00446 | 2 |
| SVM / RJMCMC | 0.947 | ± | 0.00236 | | 0.902 | ± | 0.00299 | 3 |
| Gaussian / greedy | 0.873 | ± | 0.00377 | 4 | 0.748 | ± | 0.00644 | 4 |
| Gaussian / RJMCMC | 0.846 | ± | 0.00368 | 5 | 0.739 | ± | 0.00382 | 5 |
| MLP / RJMCMC | 0.763 | ± | 0.0102 | 6 | 0.651 | ± | 0.0139 | 6 |

Table 2.1: Ranking the different implementations of statistical multiple object localization in terms of the AUC shows that SVM-computed likelihood with greedy point selection performs best. The difference in AUC between "MLP / greedy" and "SVM / RJMCMC" at the 9.90 dB noise level is not statistically significant. Therefore, these two methods are tied in rank at that noise level.

computed likelihood function uniformly outperforms the MLP one in these two experiments.

As the performance gain is small (0.014 and 0.031 in terms of the AUC measure in Tab. 2.1), all remaining experiments will be performed using the MLP as image model, nevertheless. The reason for the MLP being the method of choice is the following: For the results in Fig. 2.11, the SVM could discriminate between the object and non-object classes based on vectors of gray values alone. Real-world images such as those that will be considered in Sec. 2.7 contain more complex object shapes. For optimal performance, SVMs require the selection of features that encode these object properties well [35]. Convolutional neural networks, on the other hand, can be applied directly, making them more versatile than SVMs.

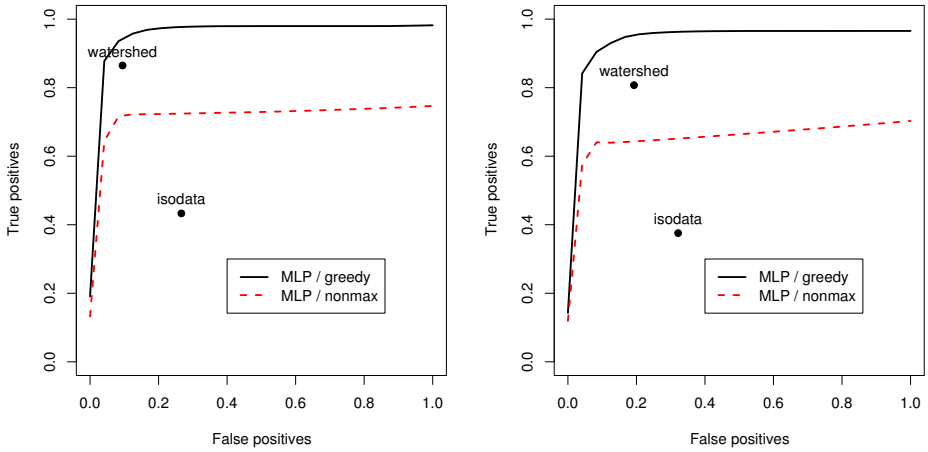
**Comparing to standard algorithms**

Next, statistical object localization is evaluated against the three standard image processing methods listed above. For the nonmaximum suppression method, ROCs can be computed by thresholding the likelihood map and accepting only those local maxima that exceed this threshold. The isodata and watershed methods, on the other hand, result in a set of coordinates. There, true and false positives are evaluated in the same manner as for all other methods, and represented by points in the ROC plots.

Fig. 2.13 shows that applying spatial statistics to the MLP output significantly improves localization accuracy. While the isodata thresholding method fails, watershed segmentation of circular particles can reach up to the performance of the proposed methods (Fig. 2.13(a)), but its accuracy decreases at higher noise levels (Fig. 2.13(b)).

The good performance of the watershed method can be explained as follows. By applying the watershed transformation on a distance map, it implicitly applies a spatial constraint: Particles may not be too close to one another. In that sense, it is more related to the proposed method than the isodata and nonmaximum suppression methods. As the spatial constraints are only implicitly enforced in the watershed method, and as it can not adjust automatically to higher noise levels as the proposed method during the training phase, its performance decreases from some point on.

This last statement is illustrated more explicitly in Fig. 2.14, where the ROCs of the proposed and the controls are shown over a range of noise levels, rather than at only two selected ones.




(a) PSNR 9.90 dB, $\epsilon = 7$ (b) PSNR 9.48 dB, $\epsilon = 7$

Figure 2.13: Performance comparison of statistical multiple object localization against three control methods. The true and false positives were computed as the mean over 100 test images, ROCs were obtained by varying the intensity parameter $\beta$ from 0 to 200. The watershed and isodata methods do not have a corresponding parameter that could be varied, and therefore their localization performances are shown as points in this plot. The spatial model leads to superior localization performance. Watershed segmentation can reach up to the performance of the proposed approach.

It shows that applying spatial statistics to the outputs of an MLP uniformly outperforms other localization approaches at all tested noise levels.

### 2.6.5 Localizing objects with varying sizes – preliminary results

This sections presents some early results on localizing objects with varying sizes, based on the marked pairwise interaction softcore process described in Sec. 2.4.3. Images of solid white circular objects with a uniform distribution of the object radii, $p(r)$, were generated (Fig. 2.15). A convolutional neural network was trained on one of these images with the additional object size parameter passed directly to the hidden layer, as described in Sec. 2.4.3. The size parameter $r$ was rounded to integer values, encoded as a binary array, and normalized to the interval $[-1, 1]$, independently in each dimension. Thus, an object radius of 3.2, for instance, would be encoded as a vector $(-1, -1, 1, -1, -1, \ldots, -1)^t$, where the length of this vector is chosen as the maximal expected object radius. As the distance parameter $\sigma$ follows from the point marks, pseudo-likelihood estimation of the softcore parameters is not required in this case. The point process intensity $\beta$ was set to the point density in the training image.

To fit this model to an image $f$, a modified variant of the RJMCMC sampler was used, cf. Sec. 2.4.3. That is, in a "move" event, the mark $r_i$ of the selected point $i$ is randomly shifted and the initial mark of a new point is drawn randomly from a uniform distribution.

The result of one run of this extended sampler is shown in Fig. 2.15. Setting the error threshold to $\epsilon = 12$ results in 10.1% false positive and 71.0% true positive rate in that image. Comparing the cumulative distribution functions of the true and estimated object radii in Fig. 2.15(b) shows

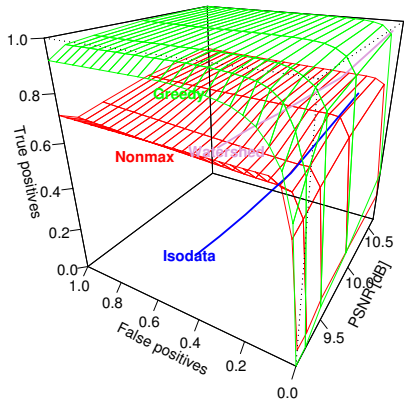



Figure 2.14: Statistical multiple object localization performs superior to simple nonmaximum suppression of the likelihood function computed using a MLP and to segmentation-based localization, not only at the two noise levels depicted in Fig. 2.13, but uniformly across a wide range of noise levels.

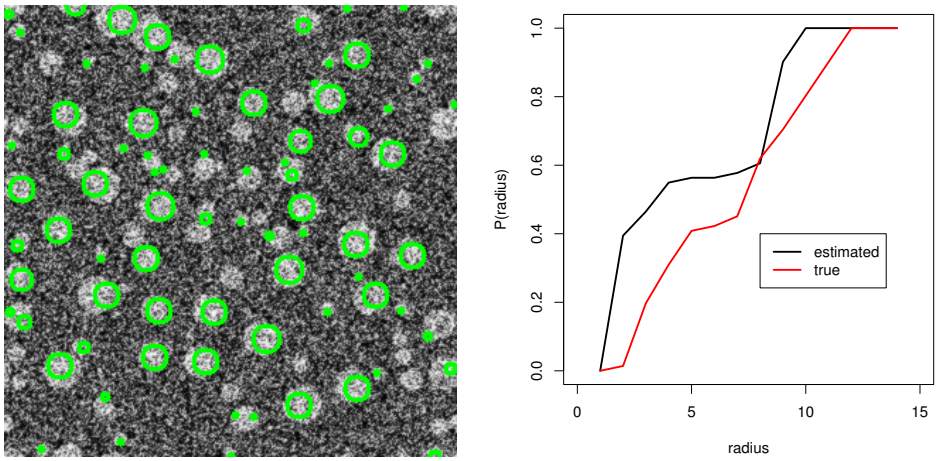
that the sampler underestimates the object sizes, which can also be seen in the corresponding marker image. The correlation coefficient of the estimated radii and their matched true objects (again using $\epsilon = 12$) is 0.838.

Clearly, these results remain below the localization accuracies of uniformly sized circular objects reported earlier in this chapter. The reason for this performance drop lies in the larger state space that needs to be searched during sampling. Adding the marks increases the searched space from 2 spatial dimensions to 3 dimensions. By the curse of dimensionality [21], this leads to an exponential increase in volume of the searched interval. Consequently, the random initializations and shift operations performed during RJMCMC sampling have a drastically lower probability of reaching a high density state.

## 2.7 Applications

To demonstrate the practical applicability of the proposed localization method not only on simulated, but also on real-world data, this section shows applications of the fixed object size model in histology and materials research. All images used in this section were hand-labeled. To make best use of the labeled data, cross-validation is applied to estimate performances: The proposed method is trained on $m-1$ images and the generalization error on the remaining image is computed. The final error measure is the mean of $m$ cross-validation measurements. Three application examples have been selected, one from the medical field and two from materials science.

The first concerns the localization and classification of meningioma tumor cells in histological resections. Meningiomas are tumor cells of the meninges (linings of the brain). They are usually benign, but histological grading is required to predict the risk of recurrence after surgery. Sec. 2.7.1 applies statistical multiple object localization to microscopical images of such tumor cells and introduces a method for classification of the detected nuclei.




(a) Localization of circular objects of varying sizes.

(b) Cumulative distributions of true and estimated object sizes.

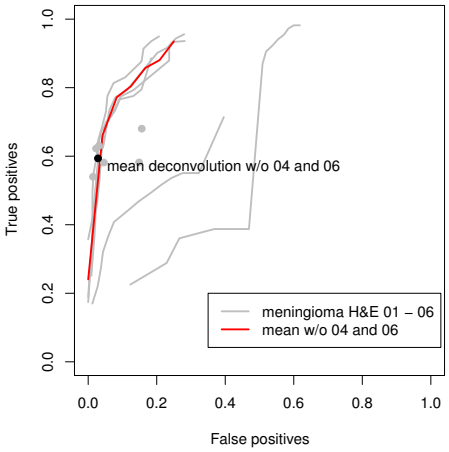
Figure 2.15: Example of fitting objects of varying sizes to an image using RJMCMC with annealing. The sampler underestimates the true object size, but captures the low count of objects with radii 5 to 7 well.

Next, the proposed method is applied to slices from micro computed tomography ($\mu$CT) data of a fiber-reinforced polymer. Fiber-reinforced polymers consist of a matrix of plastics (e.g. epoxy) and inserted fibers (e.g. glass or carbon), which provide the structural strength of these lightweight materials. The distribution of these fibers throughout a specimen is of interest in materials engineering, either for improving production processes, or for failure analysis after a probe has undergone mechanical stress. In either case, $\mu$CT-imaging is the only means for in-situ analyses, and Sec. 2.7.2 demonstrates how to locate fibers in such data.

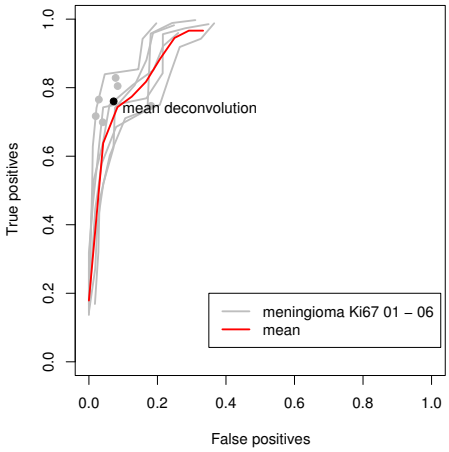
Detecting the locations of copper particles in $\mu$CT-images at the beginning of a sinter process is the topic of the third application (Sec. 2.7.3). In general, sintered materials are produced from powders (e.g. ceramics or metals) by applying heat and/or pressure until they adhere to one another. They have a large surface area and by compressing the particles into dies, sintered materials are readily produced in different shapes. Therefore, they find their applications e.g. in catalysts and filters. The goal of some recent studies was to investigate the behaviour of particles during the sinter process, a task that requires detection of locations of individual particles.

### 2.7.1 Localization and classification of meningioma cells

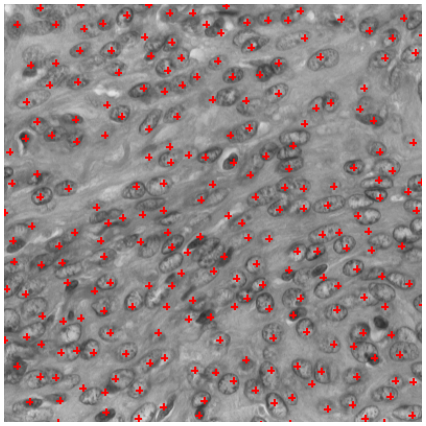
Computerized image analysis has emerged as a powerful tool for objective and reproducible quantification of histological features. DNA ploidy measurement, quantification of immunohistochemical markers, nuclear quantification, texture analysis of chromatin, and morphological diagnostics based on algorithms applied to multiple descriptors of tumor cells are the main application areas of computerized microscopy in pathology. For example, according to the grading system of the World Health Organization (WHO) of brain tumors, quantification of histologic features (mitotic index, cellular density, and Ki-67 labeling-index) are essential in the grading of meningioma cells



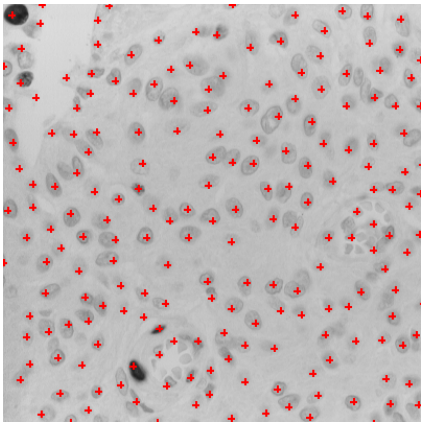

(a) Meningioma H&E ($\epsilon$=20)




(b) Meningioma Ki-67 ($\epsilon$=20)



(c) Locations in meningioma H&E 01 (color image converted to grayscale for better visualization; image source: Uni Saarland)

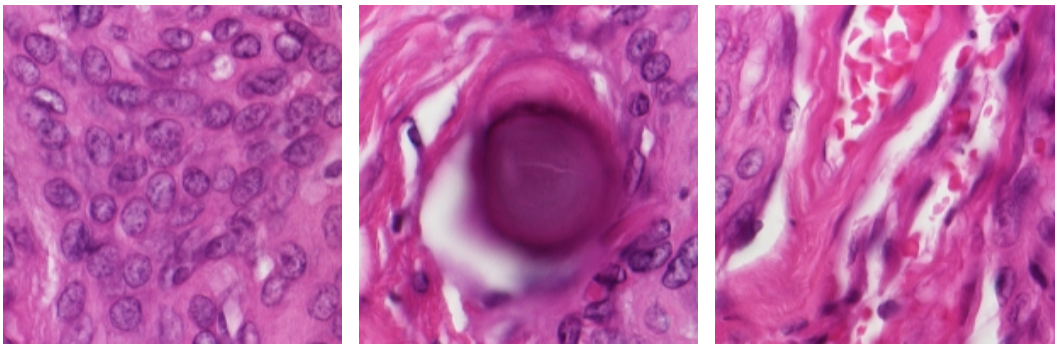


(d) Locations in meningioma Ki-67 04 (color image converted to grayscale for better visualization; image source: Uni Saarland)

Figure 2.16: Results from cross-validation experiments on the two "meningioma" application examples against an alternative method [84]. Performance of the two is similar on resections treated with Ki-67 antigen, where cells are clearly differentiable by color. Cells stained with standard H&E dye are detected with higher accuracy by the system proposed in this chapter. "H&E 04" and "H&E 06" contain cells and deformations that are not present in any of the remaining images, cf. Fig. 2.17 and are therefore ignored in the cross-validation average.

[113]. Computerized image analysis may enable an objective, standardized, and time-saving assessment of these prognostic features. However, pixel-based methods at present are still afflicted by segmentation and classification problems. Different segmentation [84] and classification [85, 171] methods for Ki-67 antibody-labeled meningioma cells have been proposed.

The data for this application example consists of six images of each H&E-stained and Ki-67 antigen labeled images. H&E (hematoxylin and eosin) is a standard dye in histology, which stains tissue and cell nuclei, making them visible in light microscopy. Ki-67, on the other hand, is a



(a) Typical image area containing mostly meningioma cells and lymphocytes

(b) H&E 04: A large calcified area

(c) H&E 06: Erythrocytes and distorted cells

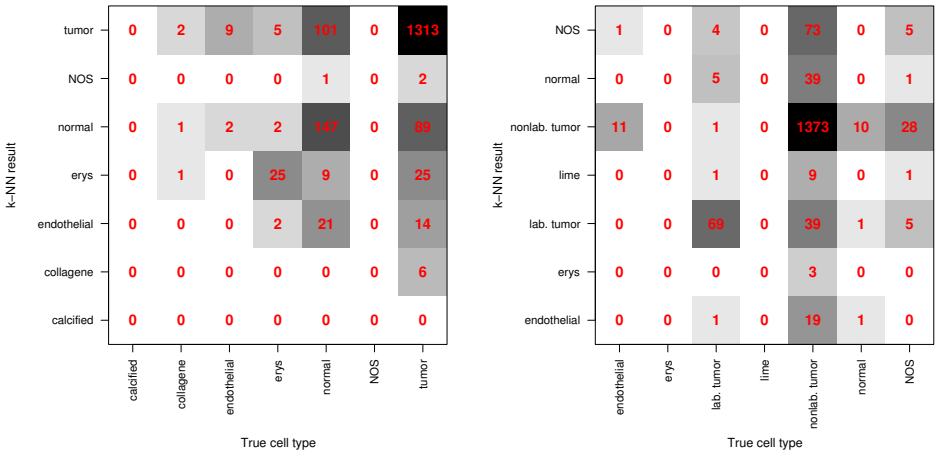
Figure 2.17: From the six images in the meningioma H&E dataset, images 04 and 06 contained cell types and deformations that were not present in any of the other images. This made them unsuitable for the cross-validation experiments reported in this section, where the object localization method was trained on all but the image on which test errors were measured.

protein that marks cells that are proliferating.

The images are $640 \times 480$ pixels large, and contain 370 cells on average. For all twelve images, an experienced pathologist provided cell locations and the corresponding cell types. While in the H&E-stained images, there are mainly meningioma tumor and normal cells, in the Ki-67 images, there exist both meningioma cells that have been labeled by the antigen (brown appearance), and meningioma cells that have not been labeled. Additionally, there are again non-tumor cells visible. The non-tumor cells fall into smaller categories, for a full list see the annotations in Fig. 2.18.

The task here is not only localization of cells, but also classification into one of a total of seven groups of cells. Therefore, a two step procedure is applied. First, the localization method from above is used to locate the cells. For this purpose, the proposed localization model is trained on all but one image at a time, using only a grayscale representation of the original images. In the results, the greedy algorithm from Sec. 2.5.2 detects object locations. At each of these locations, a high-dimensional, Haar-like feature descriptor (also including color), see [171], is extracted. Then, each detected point is assigned to one of seven cell types using $k$-nearest neighbor ($k$NN) classification with all but the samples from the current image. The $k$NN classification algorithm here assigns cell types by picking the most frequently observed class label among the $k$ training data points that are closest in feature space to such a detected point in terms of Euclidean distance. As multiple object localization, not object classification, is the topic of this chapter, $k$NN is chosen here for its simplicity and generally good classification performance [120]. The parameter $k$ was chosen experimentally, but did not have a large impact on the classification accuracy (e.g., the classification accuracy in the H&E dataset, see below, did not vary by more than 3% for values of $k$ between 1 and 13). In the following $k = 5$ will be used, which gave a good trade-off between accuracy and runtime.

Localization performance for the H&E and Ki-67 data is given in Fig. 2.16(a) and 2.16(b), re-




(a) Classification performance in meningioma H&E 01 images (overall accuracy 83.6%)

(b) Classification performance in meningioma Ki-67 images (overall accuracy 85.1%)

Figure 2.18: Confusion matrices for a $k$NN classifier ($k = 5$) applied to local image features at the locations detected as described above. Abbreviations used here: "NOS" – not otherwise specified, "erys" – erythrocytes, "nonlab." / "lab." – (non-)labeled tumor cells. On the H&E data, the number of false positives, i.e., normal cells that have been classified as tumor cells, is large. A more sophisticated classifier may be needed. The $k$NN classifiers performance in differentiating between labeled and non-labeled tumor cells in the Ki-67 data, on the other hand, is much better. This can be explained by the clear differentiability of these two cell types in color space, caused by the Ki-67 antigen and subsequent counterstaining with hematoxylin, cf. [84].

spectively. Clearly, the overall performance is better in the Ki-67 case. For the H&E series, observe the results for images No. 04 and 06, which are outliers in these results. These images contain different cell types that are not present in any other images, see Fig. 2.17. These image-specific cell types could not be learned in the present cross-validation experiments, and the corresponding two ROC curves are therefore omitted when computing the mean localization performance on the H&E data.

Next, the $k$NN classifier implementation available in GNU R [136] (Version 2.4.1) was used to assign each of the detected points to one class of cells. The results are given in the form of confusion matrices in Fig. 2.18. The true detections were compared against the labels of their matching correspondences in the given ground truth data. Once again, the overall performance on the Ki-67 data is better than on the H&E data. Using the Ki-67 antigen, the immuno-positive cells become clearly differentiable in color space, which eases the classification task.

Using this combination of multiple object localization and classification, the labeling-index (LI), i.e., the ratio of the number of immuno-positive cells and the total number of cells, can be computed. This ratio indicates the fraction of cells in mitotic phase (cell separation), and is used by pathologists as a prognostic feature for disease outcome [113]. Kim et al. described a method for measuring the LI by separating the contributions of labeled and non-labeled cells into separate images using color deconvolution [141], and segmenting the cell types separately using the watershed method [84]. They also provide source codes of an ImageJ plugin implementing

| Dataset | Ground truth | Color deconvolution [84] | This chapter |
|---|---|---|---|
| Ki-67 01 | 12.65 | 19.29 | 4.20 |
| Ki-67 02 | 6.09 | 6.46 | 4.97 |
| Ki-67 03 | 2.81 | 3.23 | 2.64 |
| Ki-67 04 | 3.54 | 2.97 | 0.95 |
| Ki-67 05 | 6.43 | 7.79 | 4.83 |
| Ki-67 06 | 4.56 | 5.54 | 4.64 |
| Mean error | | -1.53 $\pm$1.06 | 2.31 $\pm$1.29 |

Table 2.2: Labeling index (LI), indicating the fraction of cells in mitotic phase, computed on the six images from the meningioma Ki-67 dataset using the proposed method and a control method. The proposed method underestimates the LI, but the current experiment is too small to make a final judgement on the significance of these results.

their method[2].

Results comparing the performance of the method in [84] in terms of localization accuracy and labeling index are shown in Fig. 2.16 and Tab. 2.2, respectively. For the results on the H&E data in Fig. 2.16, the color deconvolution parameters for H&E dye were used together with the identical segmentation procedure of [84] applied to the corresponding deconvolved channel. The localization accuracy of the method proposed here is higher than that of the color deconvolution method, especially in the H&E data, where the contrast between cells and tissue is lower than in the Ki-67 case.

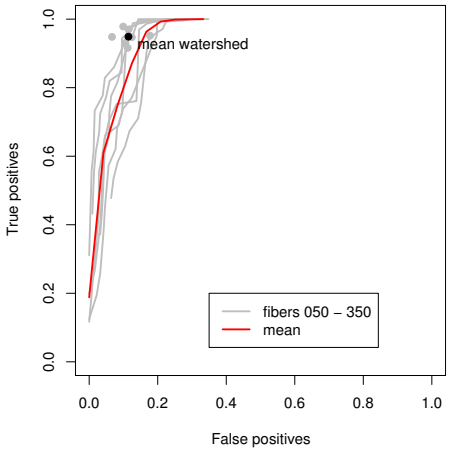
Classification accuracies from the 6-fold cross-validation experiments are similar for the Ki-67 data. The proposed method underestimates the labeling index, cf. Tab. 2.2, but the high mean error rate in this experiment can be attributed mostly to one image (Ki-67 01). As object localization, rather than tumor cell classification, is at the focus of the present chapter, the main conclusion from Tab. 2.2 is that assessment of prognostic histological features using the proposed method is feasible. The cell classification performance using a simple $k$NN classifier is comparable to that of the dedicated segmentation-based method reported in [84] (-1.5% and 2.3% error), but a larger image database would be required for a final judgement.

### 2.7.2 Glass fiber reinforced polymer

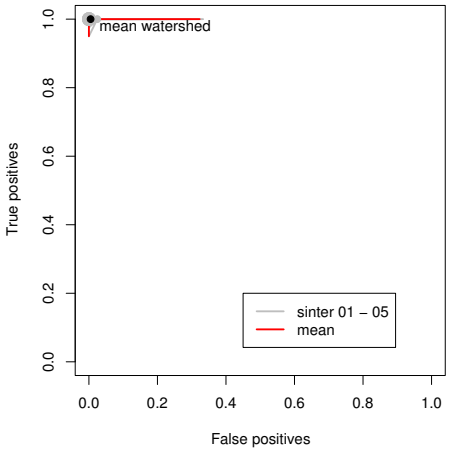
Fiber-reinforced polymers play an important role wherever low weight and high structural load are design requirements, e.g., in automotive and aerospace engineering. While the polymer matrix is incompressible, the fibers contribute their high tensile strength to the overall material properties. Therefore, information about the spatial and orientation distributions of fibers within fiber-reinforced polymers is relevant for assessment of the mechanical properties of a specimen.

For references and a more comprehensive treatment of fiber-reinforced polymers, cf. Ch. 3, which introduces a method for measuring the 3D-fiber orientation distribution from $\mu$CT data. In the present context, statistical multiple object localization is applied to locate glass fibers in CT slices through a glass fiber-reinforced polymer (GRP). This information is essential to locate low fiber-density areas within a specimen (a potential source of material failure when put under stress),

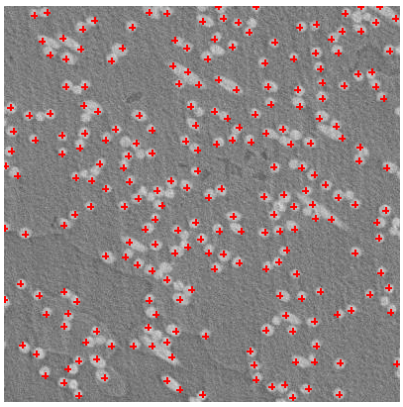
[2] `http://wwwalt.med-rz.uniklinik-saarland.de/neuropathologie/morph.html`



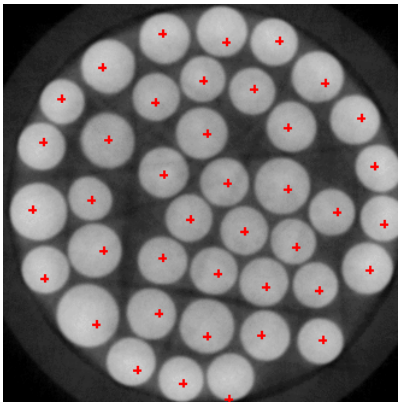

(a) Glass fiber reinforced polymer ($\epsilon = 7$)




(b) Sinter particles ($\epsilon = 30$)



(c) Locations found in fiber slice 250 (image source: IVW Kaiserslautern).



(d) Locations found in sinter slice 5 (image source: BAM Berlin).

Figure 2.19: Results from the cross-validation experiments on the "sinter" and "fiber" application examples. On this high-quality image data with good contrast and without any background clutter, the performances of watershed image segmentation and the method proposed in this chapter are almost equivalent.

and may also be used in conjunction with stereological methods to infer the 3D-fiber orientation distribution, see [80].

A sample of a GRP was provided by IVW Kaiserslautern (Institut für Verbundwerkstoffe) and imaged using phase-contrast $\mu$CT at ESRF Grenoble (European Synchrotron Radiation Facility) with a spatial resolution of 3.5 $\mu$m in each direction. The object localization model was trained using leave-one-out cross validation, where one of the total 7 images was left out and used for computing errors at a time. The results from this experiment are shown in Fig. 2.19(a). The mean number of objects per image was 243.

There is no clear gain of using the statistical object localization method instead of simple watershed segmentation. The performance of the two methods is comparable. Analyzing the errors of either method on detail reveals that both methods introduce some false positive locations

along the in-plane fibers. This was to be expected as these cases are not modeled.

### 2.7.3 Sinter particles

The sintering of metal and ceramic powders is routinely used in the production of high surface density materials such as filters and catalysts. Nevertheless, some of the processes that occur are not fully understood, to date. While the process has long been described by 1D and 2D-particle models [44], there is a known discrepancy between predicted and measured volume shrinkage [125], which has been attributed to 3D-particle motions that are not observable in 2D. Recently, a series of studies has appeared that use time series of $\mu$CT-images to observe these 3D-motions (e.g. [125, 126, 172]), leading to first 3D-models of sintered metal powders [99].

All these studies rely on the locations of individual sinter particles being available, and standard algorithms such as watershed segmentations have been used to compute those [172]. As this is a multiple object localization task in the sense of the present chapter, the localization of sinter copper particles has been selected as a further application example. The probe for this experiment was provided by BAM (Bundesamt für Materialforschung und -prüfung) and imaged by $\mu$CT with a spatial resolution of 14 $\mu$m in each direction.

Again, the object localization model was trained using leave-one-out cross validation, where one of the total 5 images was left out and used for computing test errors. The results from this experiment are listed in Fig. 2.19(b), and the locations found in one of the images are shown in Fig. 2.19(d). The expected number of objects per image was 40.

For this imaging situation of clearly defined, large particles on a homogeneous background, localization of the sinter particles is almost perfect, both for the watershed image segmentation and the proposed statistical localization methods.

## 2.8 Discussion

In this chapter, a novel combination of machine learning and point process densities was proposed and applied to the task of locating multiple circular objects in images. For circular objects with a fixed size, the novel method was shown to perform at least as well as standard image processing algorithms that are in use today. Whenever the imaging situation gets more difficult, statistical object localization performs better than the tested standard algorithms. This was observed consistently both on simulated and application data. In experiments on simulated data, the performance gain of the proposed method over standard algorithms increased with noise level. In applications on real world data, the localization performance of the current and the watershed methods were similar on the “fiber” and “sinter” data sets. For the “meningioma H&E” application, where cluttered background and textured cell surfaces complicate the task, statistical object localization showed more accurate localization results.

The use of statistical classifiers as image models eliminates the need to derive new likelihood functions for every application, as was demonstrated by applying the proposed model to different image data. This is an improvement over previously published applications of point processes to vision tasks.

Among the three tested control methods (isodata thresholding, watershed segmentation, non-maximum suppression), the watershed algorithm was the most accurate one, performing similarly as the proposed method at low noise levels in the simulation experiment, and in the "fiber" and "sinter" experiments. The watershed algorithm implicitly uses spatial constraints. As it produces compact regions, false positive detections can be ruled out. As the amount of distortions in an image increases, the smoothness of the distance map (on which the watershed method constructs its segments) lessens and therefore localization performance descreases.

The proposed system can be seen as a statistical framework for object localization, in which the image model and the fitting algorithms can be exchanged independently. The performance evaluation in Sec. 2.6.4 lead to the choice of combining convolutional neural networks with greedy mode finding. This choice for the convolutional neural network model was made for the versatile applicability to image data of this specific type of MLP. In fact, a SVM with sigmoid function fitted to the decision boundary to compute probabilities performed slightly better than the MLP model on simulated data, but would require to extract appropriate feature vectors from images when applied to real world data. This was not the task of this chapter, but the presented experiments show that if appropriate features were available, SVMs could be applied within the proposed framework.

By choosing the softcore interaction model with Euclidean metric, the current method specializes on the localization of approximately equally sized circular objects. A preliminary experiment with marked point processes applied to circular objects of varying sizes showed the feasibility of incorporating object parameters, in this case radii, into the localization process. Yet, with 71% true positives, localization performance was far below the results on objects with equal sizes. This was explained by the increased dimension of the search space Therefore, different localization algorithms will be required for handling marked point processes, which do not suffer from problems in high dimensional spaces.

The most promising application domain of the current system lies in the field of histology, as was shown on two datasets of differently prepared histological resections of meningioma tumor cells. There, the method proposed in this chapter could improve localization performance by utilizing its capability to adapt to complex object shapes and cluttered background.

The method described in this chapter represents the first application of interaction point process theory to computer vision that is fully trainable and therefore applicable not to isolated vision tasks, but rather to a broader range of multiple object localization problems.

# Chapter 3

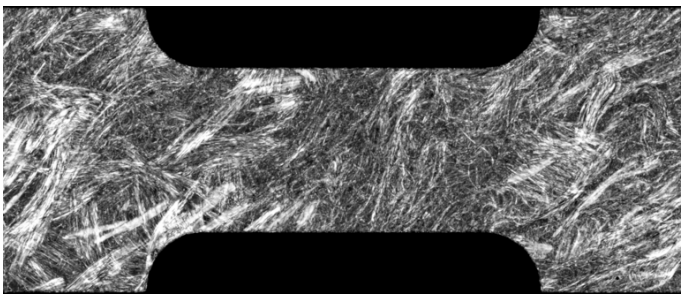
# Estimation of Fiber Orientations using Linear Filters

## 3.1 Introduction

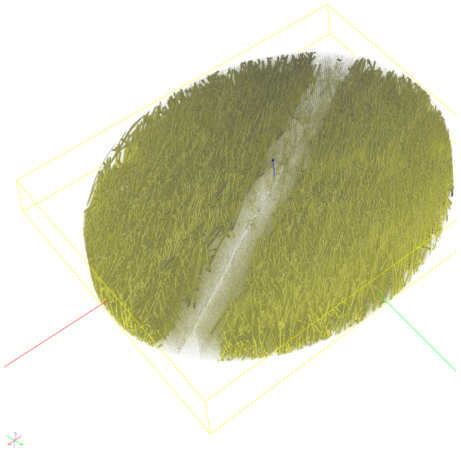
The previous chapter introduced a trainable method for localizing circular objects in 2D-images. One application example was the detection of fibers in planar cuts through a tomographic reconstruction of a glass fiber-reinforced polymer. Using this method, the spatial distribution of fibers can be assessed. But, as has already been outlined in Sec. 2.7.2, the spatial fiber distribution is only one factor that influences a fiber-reinforced material's mechanical properties, see e.g. [55, 71].

The fiber orientation distribution is one such other factor influencing the properties of various fibrous materials. In medicine, for example, the effects of collagen fiber orientation on bone strength have been investigated [117, 118]. There, it was found experimentally that collagen fiber orientation is the main factor influencing bovine cortical bone tensile strength and bending properties. In civil engineering, it is known that the fibers in ultra high performance concretes should be aligned with the direction of main tensile stress [116, Ch. 3]. In materials science, there has also been work towards understanding the influence of fiber orientations on materials' properties. E.g., Fu and Lauke developed models for fiber lengths and orientations to study their effects on the tensile strength of fiber-reinforced polymers [55]. For glass fiber-reinforced materials, the effect of local orientations on stiffness and thermal expansion was experimentally investigated in [71]. An important factor for the fiber orientation in the production of fiber-reinforced polymers is the molding process, which has been investigated in simulation studies such as [109].

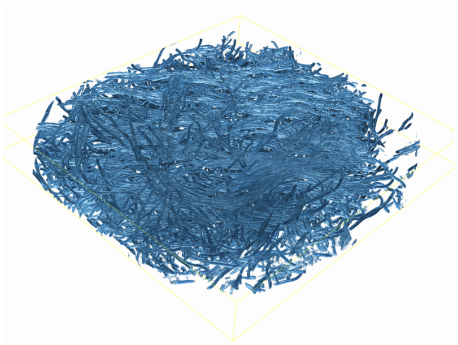
The class of materials with a fibrous microstructure is very wide, including e.g. wood, concrete, polymers, paper and textiles. Fiber-reinforced materials play an important role in manifold applications. In civil engineering, ultra high performance concretes, which are very dense and contain metal, carbon, glass or natural fibers for reinforcement, are currently an active field of academic research and commercial development [116]. Examples for applications of fiber-reinforced plastics include construction [11], and optimization of aircraft weight by applying carbon fiber-reinforced polymers in fuselages [153].



(a) Top-view of a glass fiber-reinforced polymer imaged by scanning acoustic microscopy (image source: Fraunhofer EMI, Freiburg)



(b) Tomographic reconstruction of two fused pieces of glass fiber-reinforced polymer (image source: IVW Kaiserslautern)



(c) Tomographic reconstruction of a carbon paper (image source: Fraunhofer ITWM (dept. SMS), Kaiserslautern)

Figure 3.1: Visualizations of the three data sets used as application examples in this chapter. In all of these images, segmentation of individual fibers using conventional morphological or region growing methods is difficult and error-prone.

Three examples of fibrous materials that will be analyzed in this chapter are two fiber-reinforced polymers and carbon paper, which is used as a gas diffusion layer in certain fuel cells (Fig. 3.1). Apart from dedicated machinery such as described in [167] or a system using the diffusion of light [160], most work on determining fiber orientation distributions has focused on images. These can be obtained e.g. by ultrasound or tomographic methods [156]. Purely visual inspection of such images has been performed, e.g. for examining short fiber reinforced plastics [82], and interactive systems for identifying fibers in images have been proposed [5, 157].

Automatic analysis of fiber systems in such images is a long-standing topic of active research, e.g. [32, 78, 81, 80, 102, 139]. As 3D-data has not always been available, stereology provided spatial information from planar sections of materials [7]. Under some appropriate assumptions on the fiber system, stereological approaches to measuring 3D-fiber orientations infer information on the three-dimensional arrangement of fibers by observing their shape or number in two-dimensional sections through a specimen. These sections can be obtained by confocal microscopy [32, 102], scanning acoustic microscopy [78], X-ray imaging [139], or by microscopy of polished and etched

microsections [80, 81]. Given a segmentation of individual fibers in 3D-images obtained by $\mu$CT, the fibers' orientation, spatial distribution or shape are accessible. Therefore, various segmentation methods have been applied to fibers, e.g. [6, 38, 176].

Results from stereological fiber orientation estimators are mean results. E.g. the "rose of directions" computed in [81] represents the direction of the fibers' tangent vector in the typical fiber point. Consequently, these systems do not allow for detection of local orientation measures. Segmentation-based approaches, on the other hand, can also be evaluated locally. Unfortunately, image quality is often low when imaging fiber-reinforced materials, making image segmentation a difficult task.

Other approaches to measuring fiber orientations utilize gradient information. The first derivative is useful for edge detection and has been used to compute 2D-fiber orientation distributions [57]. Second order derivatives are suitable to detect ridge-like structures such as fibers [43]. This approach to orientation detection can be applied either to an image's auto-correlation function [122], or directly to a grayscale image, e.g. [34, 51, 144]. The advantage of these gradient-based over segmentation-based methods is that they can detect subtle changes in an image's gray values, even when some segmentation algorithms may fail on that data.

In a similar spirit, filter-based methods utilize filter responses to select local orientations. For two-dimensional images, matched [27] or quadrature [46, 62] filters have been proposed for orientation estimation. M. van Ginkel's thesis [62] treats the concept of orientation spaces, which will also be used in the present chapter. Using anisotropic filters, orientation space accumulates evidence for the preferred local orientation in images. This concept will be rigorously defined in Sec. 3.2. Quadrature filters have been used for this purpose for their theoretical property of zero response when applied to a constant signal. Westin et al. applied them to locally estimate 3D-orientation of bones in images [169]. Another common type of orientation-selective filters are Gabor wavelets, which have also been used to construct orientation space representations of 2D-images [28, 29].

All of these papers use orientations obtained from filtering as inputs for further algorithms, usually image segmentation or noise smoothing, not as relevant quantities on their own. An exception to this rule is the recent work by Sandau and Ohser [143], who obtain 3D-orientation distributions of fibers by measuring the length of oriented chords that can be inscribed to a binary structure. The present chapter uses anisotropic Gaussian filters in 2D and 3D to compute fiber orientation distributions. This novel, easily implementable approach to fiber orientation measurement will be shown to deliver accurate results, and its performance will be compared to the gradient-based approach outlined above. Results from the proposed method are different forms of orientation measures such as discrete distributions on the sphere, orientation tensors or mean fiber directions.

The orientation tensor, a moment matrix of the fiber orientation distribution, is frequently used when simulating mechanical properties of fiber reinforced materials [25, 66, 83, 163]. Therefore, results from the proposed method will be suitable for subsequent simulation studies.

Before proceeding, note that throughout this chapter the following distinction between the terms direction and orientation will be made. A vector $\mathbf{v} \in \mathbb{R}^d$ is a direction vector in the usual sense that it is normalized, i.e. $\|\mathbf{v}\| = 1$. Consequently, a direction is a point on the sphere $S^{d-1}$.

A fiber, on the other hand, is inherently symmetric. Consider a direction vector $\mathbf{v}$ that describes the alignment of that fiber in some fixed point. In this situation, no distinction between $\mathbf{v}$ and $-\mathbf{v}$ can be made, and a fiber will therefore be said to have an orientation, rather than a direction, indicated by a point on the half-sphere. For the practically significant cases of $d = 2$ and $d = 3$, this will by convention be the semicircle on the positive side of the $x_1$ axis and the hemisphere on the positive side of the $x_1x_2$-plane, respectively.

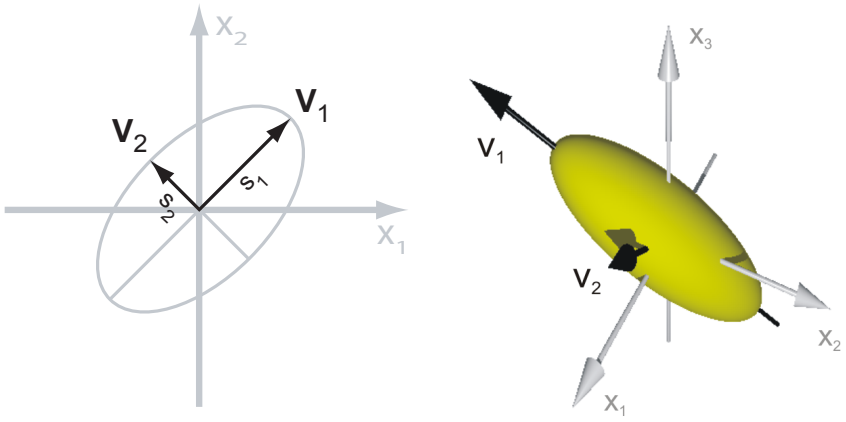
## 3.2 Gaussian orientation space

This section introduces the basic mechanism used in this chapter to measure orientations, the Gaussian orientation space. Orientation spaces [62] are similar to the well-known scale spaces [111] in image processing: To obtain the orientation space of an image, anisotropic filters with varying alignment are successively applied to the image and the filter responses are collected and indexed with the corresponding orientation parameters. Throughout this chapter, it will be assumed that the fibers in a given 2D or 3D-image $f$ all have equal diameter $2r$, which is valid for most fiber-reinforced polymers. Hence, coordinates $\mathbf{x}$ and directions $\mathbf{v}$ are defined in $\mathbb{R}^2$ or $\mathbb{R}^3$. The only exceptions are the two orientation distribution descriptors covered in Sec. 3.4.1, which are restricted to $\mathbb{R}^3$.

The method proposed in this chapter uses anisotropic Gaussian convolution filters $g_{\mathbf{v}}$, where the vector $\mathbf{v}$ ($\|\mathbf{v}\| = 1$) describes the filtering direction. The choice of Gaussian filters is motivated by practical considerations. Firstly, anisotropic Gaussian filters possess an intuitive parameterization, and secondly, fast implementation of these filters is possible, see Ch. 4. This is an advantage over quadrature filters, which have been used for this purpose e.g. in [45, 64, 75, 89, 62]. Different forms of this type of filter have been proposed, and depending on the specific filter design, parameterization and implementation can be cumbersome. Quadrature filters possess a theoretical property that is not shared by Gaussian filters: Their response is zero for constant signals, thus identifying empty image areas. For the present work using Gaussian orientation space, the same is achieved using a gray value threshold, see Sec. 3.4. Even though threshold-based binarization is usually not suitable for segmenting individual fibers in CT reconstructions of glass or carbon fiber-reinforced polymers, it is a well suited method for roughly identifying pixels belonging to the polymer matrix. This is sufficient as the purpose of this chapter is to compute mean characteristics of images.

In general, an orientation space is a high dimensional representation of an image, in which a vector of filter responses to differently oriented filter masks is attached to each pixel. Especially for 3D-images, the amount of thus generated data is prohibitively large. Therefore, only the predominant orientation for every location $\mathbf{x}$ in an image $f$ is recorded by choosing the maximum filter response across a set of tested directions $\mathbf{v}$. This results in the reduced orientation space representation

$$O(\mathbf{x}) = (r(\mathbf{x}), \mathbf{v}(\mathbf{x})), \tag{3.1}$$




(a) 2D (b) 3D

Figure 3.2: Level sets of a prolate Gaussian filter have the shape of prolate spheroids. The filter's major axis is aligned to the eigenvector $\mathbf{v}_1$ corresponding to the largest eigenvalue $\lambda_1$ of the filter kernel's covariance matrix $\Sigma$. These shapes are suitable for detecting fiber orientations.

where

$$r(\mathbf{x}) = \max_{\mathbf{v}} \left[ f * g_{\mathbf{v}} \right](\mathbf{x}) \tag{3.2}$$

$$\mathbf{v}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{v}} \left[ f * g_{\mathbf{v}} \right](\mathbf{x}). \tag{3.3}$$

Here, $*$ denotes convolution. This shows that for $O$ to be a meaningful representation of the image $f$, the result of convolution must be maximized when the filter fits the local image structure well. For the reasons outlined above, the proposed method uses anisotropic Gaussian filters of the form

$$g(\mathbf{x}) \propto \exp\left( -\frac{1}{2} \mathbf{x}^t \Sigma^{-1} \mathbf{x} \right). \tag{3.4}$$

The positive definite, symmetric matrix $\Sigma$ describes the shape and orientation of the filter. For detecting fibers in images, prolate filter kernel shapes will be used, which have one elongated axis of symmetry. Assuming bright fiber structures on dark background, (3.2) then reaches its maximum when the elements of $\Sigma$ are set such that the kernel's filtering direction $\mathbf{v}$ is aligned with the local fiber structure. To see how $\Sigma$ relates to $\mathbf{v}$, consider its eigen decomposition

$$\Sigma = VDV^{-1} = VDV^t = \begin{pmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_d \end{pmatrix} \begin{pmatrix} s_1 & & \\ & \ddots & \\ & & s_d \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^t \\ \vdots \\ \mathbf{v}_d^t \end{pmatrix}, \tag{3.5}$$

with eigenvalues $s_i$ in descending order, and corresponding eigenvectors $\mathbf{v}_i$, $i = 1, \ldots, d$. Since $\Sigma$ is symmetric, $V$ is orthonormal, i.e., $V^{-1} = V^t$, and all $\mathbf{v}_i$, have unit length and are mutually

orthogonal, see e.g. [63].

For a prolate Gaussian kernel, the vector $\mathbf{v}_1$ indicates the direction of its axis of least inertia (Fig. 3.2). Consequently, by setting $\mathbf{v}_1 = \mathbf{v}$, $s_2 = \ldots = s_d = r$, $s_1 = 2r$ and choosing arbitrary direction vectors $\mathbf{v}_2, \ldots, \mathbf{v}_d$ with $\mathbf{v}_i^t \mathbf{v}_j = 0$, $\forall i \neq j$, the matrix $\Sigma$ of a prolate Gaussian filter kernel for detecting fibers of known radius $r$ can be computed using (3.5). The process of determining $d-1$ directions orthogonal to $\mathbf{v}_1$ could be implemented by Gram-Schmidt orthogonalization [63, Ch. 5]. The resulting kernel's aspect ratio of 2:1 works well in practice, smaller aspect ratios should be used only for highly curved fibers. As $V$ is orthonormal, the direction vectors $\mathbf{v}_2, \ldots, \mathbf{v}_d$ all lie in a hyperplane with normal vector $\mathbf{v}_1$. For the practically relevant cases $d = 2, 3$, Sec. 4.4.2 gives explicit formulas for $\Sigma$ that avoid the need to explicitly compute $\mathbf{v}_2$ and $\mathbf{v}_3$. Other factorization of $\Sigma$ will be discussed in Ch. 4.

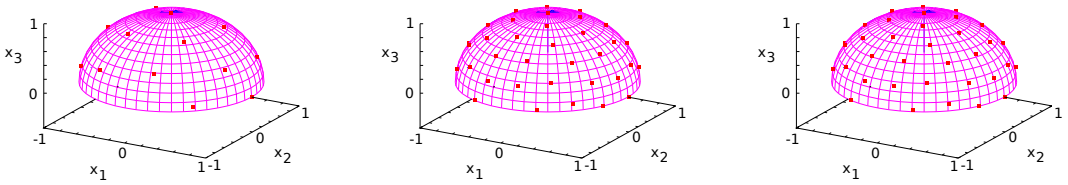
## 3.3 Sampling on the hemisphere

Computation of the reduced Gaussian orientation space, introduced above, requires discretization of the directions $\mathbf{v}$ when performing the search for the maximal filter response. In practice, one computes (3.1) for a fixed number $n$ of directions $\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^n$. Due to the symmetries of anisotropic Gaussian filters, the filter response to a fiber oriented along $\mathbf{v}$ is identical for the two directions $\mathbf{v}$ and $-\mathbf{v}$. Therefore, filtering must only be performed in directions sampled from the semicircle or the hemisphere in 2D or 3D-images, respectively. In order to avoid systematic errors in this process, it is important to sample these points uniformly. By representing a direction $\mathbf{v} \in \mathbb{R}^2$ on the semicircle in polar coordinates $\alpha \in [0, \pi[$, $n$ equal angular steps yield adequate sampling directions $\mathbf{v}^i$. However, the task of finding $n$ suitable directions that uniformly cover the hemisphere is difficult, and different strategies have been used in the literature.

In a preceding publication, Robb et al. parameterized $\mathbf{v} \in \mathbb{R}^3$, $\|v\| = 1$, in terms of polar coordinates $(\theta, \varphi)$ and derived a formula for quantizing the upper hemisphere [140]. Choosing the number of quantization steps along the colatitude $\theta$ and the longitude $\varphi$ results in different $n$. This had the disadvantages that the shapes of the sampling intervals varied strongly between equator and north pole, and that the point $(0, 0, 1)^t$ could never be sampled precisely.

Faas and van Vliet used an icosahedron, i.e., a polyhedron with 20 triangular faces [45]. Placing one sampling direction on each vertex of the icosahedron, they obtain 12 filtering directions $\mathbf{v}^i$, $i = 1, \ldots, 12$. To increase the number of samples, they imposed a hexagonal grid on each of the 20 triangular faces, and projected the center of each face's tessellations onto the sphere, see [45] for details. This allowed them to flexibly choose the number of samples, but it is only an approximation as the samples within each face are placed regularly in the plane, not on the sphere.

The authors of [143] have recently extended the so-called chord length transformation to 3D, which also requires them to perform uniform sampling on the upper hemisphere. For this purpose, they placed points on the hemisphere's surface in planes parallel to the $x_1x_2$-plane, achieving an approximately uniform distribution [127]. An algorithm for equal area partition of the unit sphere into zones with small diameters, the recursive zonal equal area partition, was described in [104]. That algorithm is very efficient, but it is not easily modified to work only on the hemisphere.




(a) Coarse sampling: 18 points, 33.1° mean distance to nearest neighbor

(b) Medium sampling: 50 points, 19.7° mean distance to nearest neighbor

(c) Fine sampling: 98 points, 13.8° mean distance to nearest neighbor

Figure 3.3: Sampling points on the upper hemisphere computed as described in App. C. These points serve as filter directions in 3D when discretizing the maximization in (3.2). With an increasing number of samples, the angular resolution of the orientation space representation increases.

### 3.3.1 Approximately uniformly distributed points on the hemisphere

To compute appropriate sampling points for implementing the maximization in (3.1), the present chapter follows the work of Fliege and Maier [49, 50], who computed sets of integration nodes on the sphere using numerical methods. They began by maximizing the pairwise Euclidean distance between $n$ direction vectors $\mathbf{v}^i \in \mathbb{R}^3$, using a combination of simulated annealing and the Newton method. Next, they computed the weight of each of these $n$ points to be used for numerical integration. These weights also served as a measure of quality of the detected arrangements on the sphere.

This approach has advantages over the aforementioned sampling schemes: The number $n$ of sampling points can be chosen (not any number $n$ is possible, though, see [50]), and since the solutions are invariant to rotations, they can include $(0, 0, 1)^t$ as a fixed point. Furthermore, their distributions are uniform on the sphere with high numeric precision. Results for up to 1600 points on the unit sphere are available from the authors of [50]. Unfortunately, their original results are not applicable here, for they represent uniformly distributed points on the sphere, rather than the hemisphere.

Therefore, the approach in [50] was modified by simultaneously optimizing the pairwise distances of the $n$ direction vectors $\mathbf{v}^i$ and their counterparts on the lower hemisphere, $-\mathbf{v}^i$. This results in $n$ approximately uniformly distributed points on the upper hemisphere. Furthermore, the three coordinate axis directions $(1, 0, 0)^t$, $(0, 1, 0)^t$ and $(0, 0, 1)^t$ are fixed, therefore modifying only the remaining $n - 6$ vectors $\mathbf{v}^i$ and $-\mathbf{v}^i$ when maximizing the pairwise distances. While this constraint could have a negative impact on the quality of the optimization result, there is an immediate benefit. These three coordinate directions are frequently used to make a quick judgement about the isotropy or anisotropy of the structures in a given image. Therefore, it will be useful from a practical point of view to be able to assign weights to these three directions.

All computational details, lists of the resulting vectors and of the corresponding cubature weights are deferred to App. C. The optimization procedure was conducted for $n = 18, 50, 98$, which will be referred to as coarse, medium and fine resolution, respectively. Fig. 3.3 visualizes

the distribution of the resulting points across the hemisphere for these three resolution levels. These three sets of filtering directions allow for the user to choose a trade-off between runtime and accuracy of the achieved results, which will be investigated in Sec. 3.6.

## 3.4 Computation and interpretation of the orientation tensor

Using the results of this chapter up to this point allows for computation of the reduced orientation space $O(\mathbf{x})$ in each image pixel $\mathbf{x}$. However, this representation of an image is not useful on its own. This section therefore introduces methods for averaging and summarizing the orientation information contained in $O(\mathbf{x})$.

Orientation tensors, originally introduced by Tucker and Advani [163], are frequently used for simplifying the computation of directional averages when simulating mechanical properties of fibrous materials, see e.g. [25, 66, 83]. Let $p$ be a density defined on the sphere $S^{d-1}$ with $p(\mathbf{v}) = p(-\mathbf{v})$. Orientation tensors are then defined as the moments of the distribution $p$. The second order orientation tensor $a_{ij}$,

$$a_{ij} = \int_{S^{d-1}} v_i v_j p(\mathbf{v}) d\mathbf{v}, \tag{3.6}$$

is $\mathbf{v}$'s correlation matrix and therefore represents an ellipsoidal approximation of $p$'s shape. Here, $v_i$ denotes the $i$'th component of $\mathbf{v}$, and $i, j = 1, \ldots, d$.

The use of $a_{ij}$ from an image analysis perspective is twofold. It comprises a convenient method for averaging directional information over images or image areas, see below, and it can directly be used in subsequent simulation studies using methods such as those described in the references given above. Therefore, a number of authors have used $a_{ij}$ to describe the orientation of fibers in images. Among those are approaches for computing $a_{ij}$ using stereological methods [42, 68, 102, 103], and by integrating (3.6) over the responses of quadrature filters instead of $p$ [89].

The question then arises how to compute $a_{ij}$ from the reduced Gaussian orientation space $O(\mathbf{x})$. The solution is to use sample averages over all pixels belonging to the fiber system $B$, which is identified by a global gray value threshold $t_0$,

$$B = \{\mathbf{x} | f(\mathbf{x}) > t_0\}, \tag{3.7}$$

assuming that fibers appear with larger gray values than the background. In practical experiments setting $t_0$ to 10% above Otsu's threshold [129] was sufficient for all datasets tried. To choose a high threshold $t_0$ is appropriate here as it eliminates pixels close to fiber edges, where orientation estimates are least stable. In a window $W$, $a_{ij}$ is then estimated by

$$\hat{a}_{ij} = \frac{1}{|W \cap B|} \sum_{\mathbf{x} \in W \cap B} v_i(\mathbf{x}) v_j(\mathbf{x}). \tag{3.8}$$

In matrix-vector notation, this leads to a sample second order orientation tensor $T_W$,

$$T_W := \frac{1}{|W \cap B|} \sum_{\mathbf{x} \in W \cap B} \mathbf{v}(\mathbf{x}) \mathbf{v}^t(\mathbf{x}). \tag{3.9}$$

For averaging over $B \cap W$ to yield correct sample orientation tensors, a few conditions have to be met. In stochastic geometry, random fiber processes are described by sets of finite smooth curves. Their orientations are characterized by the tangential vectors of these curves, which posses distributions that are random measures on the circle or sphere. For a rigorous definition, see [155]. Then, for a system of non-overlapping fibers, the distribution of the tangential direction vectors, $p(\mathbf{v})$, is a length-weighted distribution, since each fiber contributes proportionally to its length to the overall probability mass. Under the assumption made above that all fibers have equal diameter $2r$, this is also true for the sample mean over $\mathbf{v}\mathbf{v}^t$ in (3.9), as the number of pixels that a single fiber contributes to $B$ is proportional to its length.

A potential problem in evaluating (3.9) is the edge treatment. When observing stochastic processes in finite areas $W$, realizations of large objects, in the present case long fibers, have a higher probability of intersecting the boundary than small objects. This can lead to a bias in the estimated quantity. Different methods to avoid this problem have been proposed, see e.g. [7, Ch. 3] for an overview. These methods, however, all rely on individually segmented objects being available.

As the present chapter introduces a method for orientation estimation that does not rely on the segmentation of fibers, there is no immediate remedy for the edge correction problem. Therefore, when estimating $T_W$ using (3.9), one has to either choose a sufficiently large observation window $W$, or the orientation distribution of an individual fiber must be independent of its length. Whether an observation window $W$ can be considered sufficiently large depends on the length of the fibers.

### 3.4.1 Descriptors derived from the orientation tensor

The sample second order orientation tensor $T_W$ contains orientation information about the fibers in window $W$. As it is computed as the sum of outer products, it is a symmetric, positive definite matrix. Analogously to the matrix $\Sigma$ in Sec. 3.2, the orientation tensor's eigen decomposition,

$$T_W = \Gamma \Lambda \Gamma^{-1} = \Gamma \Lambda \Gamma^t = \begin{pmatrix} \gamma_1 & \cdots & \gamma_d \end{pmatrix} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix} \begin{pmatrix} \gamma_1^t \\ \vdots \\ \gamma_d^t \end{pmatrix}, \tag{3.10}$$

with orthonormal matrix $\Gamma$ and diagonal matrix of eigenvalues $\Lambda$, can be used to extract the orientation information contained in $T_W$ [48]. Assume again that the eigenvalues are sorted in descending order, $\lambda_1 \geq \ldots \geq \lambda_d$ (all $\lambda_i$ are positive due to $T_W$ being positive definite). The quadratic form of $T_W$, $\mathbf{x}^t T_W \mathbf{x} = 1$, describes an ellipsoid. Therefore, the *mean fiber orientation* $\mu \in \mathbb{R}^d$ is given by the eigenvector $\gamma_1$ corresponding to the largest eigenvalue $\lambda_1$ of $T_W$,

$$\mu = \gamma_1. \tag{3.11}$$

The mean orientation, however, is relevant only for fiber systems that are aligned to one orientation. Such fiber orientation distributions will be referred to as being of cluster-type, as the tangent directions of such fiber systems are clustered around one point on the half-sphere. To characterize other distribution shapes, the remaining eigenvalues of $T_W$ must also be taken into account. Two nonparametric descriptors of spherical distributions for the 3D-orientation tensor, denoted by $T_W^{(3)} \in \mathbb{R}^{3\times3}$ to avoid confusions, are described in [48]. These 3D-descriptors, called the *shape* and *strength* parameters of spherical distributions, are computed from the eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 > 0$ of $T_W^{(3)}$ as

$$\gamma = \frac{\log(\lambda_1/\lambda_2)}{\log(\lambda_2/\lambda_3)}, \tag{3.12}$$

$$\zeta = \log(\lambda_1/\lambda_3). \tag{3.13}$$

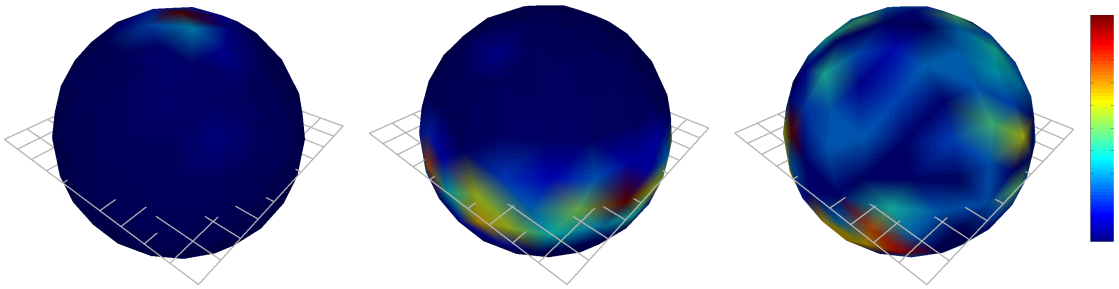
These two numbers allow to differentiate between cluster-type, girdle-type and isotropic spherical distributions (cf. Fig. 3.4):

- *Cluster-type* distributions on the sphere are characterized by shape values larger than one ($\gamma > 1$), for the ellipsoid described by the second order orientation tensor will then have two approximately equally long minor axes ($\log(\lambda_2/\lambda_3) \to 0$), and the major axis in direction $\gamma_1$ is longer than the minor axes ($\log(\lambda_1/\lambda_2) \to \infty$).
- *Girdle-type* distributions concentrate their probability mass along a great circle of the sphere. Fiber-systems corresponding to such distributions are arranged in layers and fiber orientations vary either within or among these layers. The ellipsoid described by the corresponding $T_W^{(3)}$ will have an oblate shape, $\lambda_1 \approx \lambda_2$, and therefore $\log(\lambda_1/\lambda_2) \to 0$. Consequently, girdle-type distributions can be identified by values $\gamma < 1$.
- The second order orientation tensor of an *isotropic distribution* has three approximately equally large eigenvalues, i.e. $\lambda_1 \approx \lambda_2 \approx \lambda_3$. Therefore, the strength parameter $\zeta$ tends to zero, regardless of the shape $\gamma$.

These two descriptors allow for an easy characterization of spherical distributions. When a cluster-type fiber distribution was identified, the mean orientation $\mu$ summarizes the corresponding fiber system. The mean orientation $\mu$ of girdle-type distribution lies in-plane with the fibers. These two parameters are capable of differentiating between these three principal forms of directional distributions. However, due to the limitations of the second order orientation tensor, they do not cover all possible distribution types. Especially mixtures of these basic forms can not be detected, e.g., multimodal distributions or mixtures of cluster and girdle-types. Descriptors derived from moments of higher order would be required.

## 3.5 Fiber models for evaluating the proposed method

This section introduces two models for fibers in 2D and 3D-images, which will be used below to evaluate the quality of the proposed method for fiber orientation estimation. The first model is



(a) A cluster-type orientation distribution ($\gamma = 74.56$,$\zeta = 1.87$)

(b) A girdle-type orientation distribution ($\gamma = 0.05$,$\zeta = 1.67$)

(c) An orientation distribution of neither of the previous two types ($\gamma = 7.61$,$\zeta = 0.30$)

Figure 3.4: Histograms on the sphere for realizations from three types of orientation distributions, showing how $\gamma$ and $\zeta$ relate to different forms of distributions. The shape parameter clearly distinguishes between cluster and girdle-type distributions ($\gamma = 74.6$ and 0.05, respectively). The third one, which is neither a girdle nor a cluster-type is characterized as an isotropic spherical distribution ($\zeta = 0.3$).

an overlapping fiber process with a parameter governing the degree of curvature of each fiber. It is suitable for comparing the proposed method against control methods in terms of the directional accuracy that is measured in each pixel. The influences of noise, fiber curvature and density can be assessed. Applications such as the analysis of fiber-reinforced polymers usually require a characterization of the fiber orientation distribution, and methods for computing these have been described in the previous sections. As sampling of curved fibers according to a fixed probability density function on the sphere is difficult, this first fiber model is not suitable for an experimental evaluation of these measures. Instead, these evaluations will be performed using a second model of straight fibers, consisting of a system of non-overlapping cylinders. Using this model, experimental validation of the proposed method for characterizing fiber orientation distributions is possible.

Regardless of whether the modeled fibers are straight or not, a measure for fiber density is required. In stochastic geometry, the standard density measure of random fiber processes is the length density $L_A$ or $L_V$ in $\mathbb{R}^2$ or $\mathbb{R}^3$, respectively [155]. The length density is the total length of all fibers in an observation window divided by the area or volume of that window.

### 3.5.1 Curved fibers

Next, an algorithm for generating 2D and 3D-images of curved fibers is given. It generates fiber paths by sampling direction vectors from distributions on the unit circle or sphere. The von Mises and von Mises-Fisher distributions are often referred to as the equivalents of the normal distribution on the unit circle [47] or unit sphere [48], respectively. In fact, the von Mises-Fisher distribution is a generalization of the von Mises distribution. Both have two parameters, a mean direction $\mu \in \mathbb{R}^d$ and a spread parameter $\kappa \in \mathbb{R}$, comparable to the mean and variance of the normal distribution. For $\kappa = 0$, these distributions are equivalent to the uniform distribution on the circle or sphere. As $\kappa \to \infty$, their probability mass concentrates around the mean direction $\mu$.

These distributions are now used for defining Markov chains along fiber paths. Let $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$

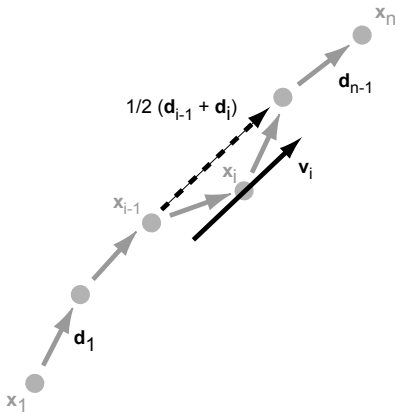



Figure 3.5: To evaluate the performance of orientation estimation, simulated fiber systems are used. The first model of curved fibers in $\mathbb{R}^2$ or $\mathbb{R}^3$ consists of $n$ points $\mathbf{x}_i$ and unit length difference vectors $\mathbf{d}_i$ between consecutive points. The tangential direction $\mathbf{v}_i$ at a point $\mathbf{x}_i$ is parallel to the vector from $\mathbf{x}_{i-1}$ to $\mathbf{x}_{i+1}$.

be a sequence of points in $\mathbb{R}^2$ or $\mathbb{R}^3$ with the property that $\|\mathbf{x}_i - \mathbf{x}_{i-1}\| = 1$, $i = 2, 3, \ldots, n$. These points describe the path of a fiber. To each point $\mathbf{x}$, attach a direction vector $\mathbf{d}$, $\|\mathbf{d}\| = 1$, such that

$$P = (p_1, p_2, \ldots, p_n) = ((\mathbf{x}_1, \mathbf{d}_1), (\mathbf{x}_2, \mathbf{d}_2), \ldots, (\mathbf{x}_n, \mathbf{d}_n)) \tag{3.14}$$

represents a fiber path and the vectors between adjacent points, $\mathbf{d}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$. As all $\mathbf{d}_i$ are unit vectors, the fiber's tangential direction at point $\mathbf{x}_i$ is parallel to the line from $\mathbf{x}_{i-1}$ to $\mathbf{x}_{i+1}$, $\mathbf{v}_i = 1/2(\mathbf{d}_{i-1} + \mathbf{d}_i)$ (Fig. 3.5). $P$ is a random variable with joint probability density function $p(P)$, which will be assumed to be a Markov-1 process with conditional density

$$p(p_i | (p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n)) = p(p_i | p_{i-1}). \tag{3.15}$$

This Markov-1 chain can model everything from smooth, almost straight fibers to paths with highly irregular shapes. Assume stationarity of $P$. Then $p(p_i|p_{i-1})$ depends only on the difference vectors $\mathbf{d}_i$ between consecutive points along a fiber path. Choosing the von Mises-Fisher distribution with parameter $\kappa$ for the conditional density leads to

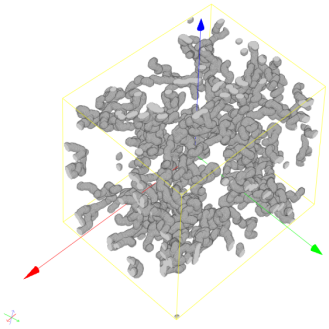
$$p(p_i|p_{i-1}) = \frac{\kappa}{4\pi \sinh \kappa} \exp\left\{\kappa \mathbf{d}_i^t \mathbf{d}_{i-1}\right\}. \tag{3.16}$$

Realizations of simulated fiber processes are generated starting from a uniformly distributed random point $p_1 = (\mathbf{x}_1, \mathbf{d}_1)$. At each $\mathbf{x}_i$, a sample from the Fisher or von-Mises Fisher density is generated, using $\mathbf{d}_i$ as mean vector, thus specifying the next point $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{d}_i$. The parameter $\kappa$ influences the degree of curvature of the thus realized fibers (Fig. 3.6). This process is repeated until reaching the desired fiber length $n$. A given length density, i.e., the total fiber length in an observation window $W$ normalized by the size of $W$, can be achieved by generating multiple fibers.

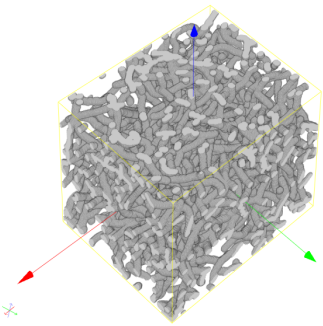
(a) 2D, $\kappa = 10$, $L_A = 0.025$ pixel$^{-1}$ (inverted gray values for visualization)

(b) 2D, $\kappa = 100$, $L_A = 0.075$ pixel$^{-1}$ (inverted gray values for visualization)

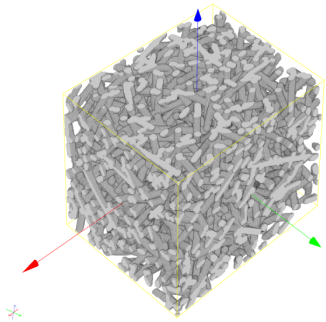
(c) 2D, $\kappa = 5000$, $L_A = 0.0125$ pixel$^{-1}$ (inverted gray values for visualization)



(d) 3D, $\kappa = 10$, $L_V = 0.001$ pixel$^{-2}$ (inverted gray values for visualization)



(e) 3D, $\kappa = 100$, $L_V = 0.003$ pixel$^{-2}$ (inverted gray values for visualization)



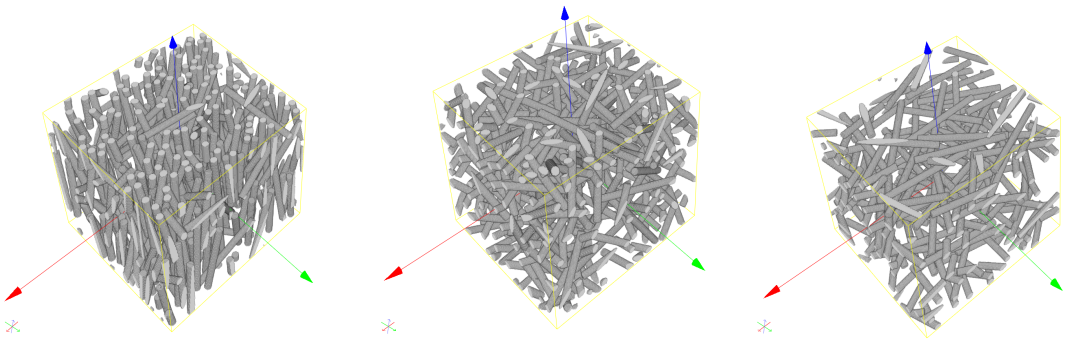
(f) 3D, $\kappa = 5000$, $L_V = 0.005$ pixel$^{-2}$ (inverted gray values for visualization)

Figure 3.6: Realizations of the curved fiber process for different values of parameter $\kappa$: (a, b, c) in the plane, and (d, e, f) in 3D. The images are $256^2$ and $256^3$ pixels large, respectively. Each fiber is 128 pixels long with a radius of 5 pixels. The length densities $L_A$ and $L_V$ are given by the total fiber length divided by the area or volume of the image, respectively.

### 3.5.2 Straight fibers

To evaluate the accuracy of measurements obtained from the 3D-orientation tensor $T_W^{(3)}$, see Sec. 3.4, requires simulated fibers for which the orientation distribution of the entire fiber system is known. As the previous model described only the amount of flexibility of a fiber, it is not suitable for this purpose. Instead, a random sequential adsorption (RSA) [161] model of non-overlapping cylinders will be used. RSA is a general technique to simulate random processes of objects, and it has already been used in Sec. 2.6.1 to generate 2D-test data for object localization. Starting from an empty set, RSA randomly adds objects until the space is packed such that no further object can be added. Here, the objects are oriented cylinders of a given length $l$ and diameter $2r$.

Only 3D-fiber systems are considered in this section. The RSA process is equipped with an orientation distribution $p$, defined below, which the method proposed in this chapter approximates by means of the orientation tensor. In Sec. 3.6, realizations of this model are used to test the



(a) $\beta = 0.1$, $L_V = 0.002$ pixel$^{-2}$ (inverted gray values for visualization)

(b) $\beta = 1.0$, $L_V = 0.0015$ pixel$^{-2}$ (inverted gray values for visualization)

(c) $\beta = 10$, $L_V = 0.001$ pixel$^{-2}$ (inverted gray values for visualization)

Figure 3.7: Realizations of the second type of fiber model used for evaluations, a hardcore cylinder process. Cylinders are generated by random sequential adsorption, with periodic edge treatment, and are shown here with different sets of parameters. The images are $256^3$ pixels large, each cylinder is 128 pixels long with a radius of 5 pixels.

accuracy of this method. Therefore, it is important that the following RSA algorithm generates cylinder systems that adhere to the given $p$.

1. Draw a direction $\mathbf{v}$ according to $p(\mathbf{v})$.
2. Draw a coordinate $\mathbf{x}$ from $U([0,1]^3)$.
3. If a cylinder of length $l > 0$ and radius $r > 0$ attached to $\mathbf{x}$ in direction $\mathbf{v}$ does not intersect any existing cylinders, insert it and go to step 1. Otherwise, go to step 2.

Note that, once drawn, the direction $\mathbf{v}$ is never modified. The maximal number of retrials $m$ for steps 2 and 3 above terminates the algorithm. For the images in Fig. 3.7 and for the experiments in the sections below it was $m = 10000$. To influence the density of the resulting fiber system, these three steps are repeated at most $n$ times. This algorithm was implemented and used to model glass fiber-reinforced polymers by M. Schöneberger [146].

The spherical von Mises-Fisher distribution is not suitable for testing the performance of $T_W^{(3)}$ as its concentration parameter $\kappa$ merely allows for distribution shapes between isotropic and cluster-type. Instead, the model relies on the one-parameter distribution introduced in [145] for modelling an acoustic trim. Its sole parameter $\beta > 0$ varies the shape of the distribution from a girdle shape along the equator ($0 < \beta < 1$), via a uniform distribution on the sphere ($\beta = 1$) to a cluster shape of vectors mostly parallel to $(0,0,1)^t$ ($\beta \to \infty$). Using $\theta$ and $\varphi$, to denote the colatitude and longitude of $\mathbf{v}$'s polar representation, respectively, the density is given by

$$p(\mathbf{v}) = p(\theta, \varphi) = \frac{1}{4\pi} \frac{\beta \sin\theta}{(1 + (\beta^2 - 1)\cos^2\theta)^{3/2}}. \tag{3.17}$$

Further properties of this density are listed in [145]. The resulting process can be interpreted as a marked point process $P = \{(\mathbf{x}_i, \mathbf{v}_i)\}_{i=1,\dots,n}$ with a density defined on $\mathbb{R}^3 \times S^2$.

## 3.6 Results

This section presents quantitative results on the accuracy of the proposed method on 2D and 3D-data, compared to a control method based on image gradients, introduced below. The range of evaluations includes robustness to noise, fiber curvature and density, as well as an analysis of the efficiency of the proposed orientation distribution parameters $\zeta$ and $\gamma$. But first, a control method from the literature is introduced

### 3.6.1 Fiber orientations from the Hessian matrix

To put the performance of the method presented in this chapter into perspective, a method from medical image processing will be used for reference. There, filtering in directions obtained from partial second derivatives has been used to improve the quality of images depicting vessels [17, 51, 112, 90, 134, 144]. These directions can also be used to measure fiber orientation. First derivatives are commonly used for edge detection, as they indicate changes in a function. Second derivatives are useful to detect ridges [43], such as fiber profiles.

Let $\sigma$ be the scale parameter of an isotropic Gaussian convolution filter $g_\sigma$. Lindeberg introduced the concept of scale-space as a method for detecting image structures at unknown scales [111]. For this purpose, he defined a normalized partial derivative as

$$f_{x_i} := \sigma^\gamma \frac{\partial}{\partial x_i} f * g_\sigma. \tag{3.18}$$

Here, $\gamma$ is a parameter required to normalize $f_{x_i}$ across different scales such that the maximum over $\sigma$ can be used to detect edges in a scale-independent manner. For the purpose of finding fibers with known diameter $2r$, the scale is known, $\sigma = r$, and this normalizing factor will therefore be omitted.

Analogously to (3.18), the Hessian matrix at scale $\sigma$ for a $d$-dimensional image is defined as

$$H = \begin{pmatrix} f_{x_1x_1} & \cdots & f_{x_1x_d} \\ \vdots & \ddots & \vdots \\ f_{x_dx_1} & \cdots & f_{x_dx_d} \end{pmatrix}, \tag{3.19}$$

where

$$f_{x_ix_j} := \frac{\partial^2}{\partial x_i \partial x_j} f * g_\sigma \tag{3.20}$$

are the partial second derivatives. As the order of differentiation is unimportant, $H$ is symmetric. The second order gradient in direction $\mathbf{v}$ can be computed as $\bigtriangledown^2_{\mathbf{v}} = \mathbf{v}^t H \mathbf{v}$. As the curvature along a fiber is low, $\bigtriangledown^2_{\mathbf{v}}$ will be minimized when $\mathbf{v}$ is the tangent fiber direction at a fiber point $\mathbf{x}$. For a more detailed discussion of the geometric interpretation of $H$, see [43]. In order to compute $\mathbf{v}$ from $H$, an eigen analysis similar to the one for orientation tensors in Sec. 3.4.1 is performed. As the curvature is lowest in the fiber direction, $\mathbf{v}$ will be chosen as the eigenvector corresponding to

(a) Additive Gaussian noise variance 50

(b) Additive Gaussian noise variance 150

(c) Additive Gaussian noise variance 250

Figure 3.8: Visualizations of different noise levels used for evaluating the robustness of local fiber orientation estimation, in 2D.

the smallest eigenvalue of $H$.

This observation leads to estimated local orientations $\mathbf{v}$ by computing the Hessian and its eigenvector corresponding to the smallest eigenvalue in every pixel $\mathbf{x}$. These results will be used below to control the accuracy of the method proposed in this chapter in comparison to this standard method.

### 3.6.2 Accuracy of local orientation estimation

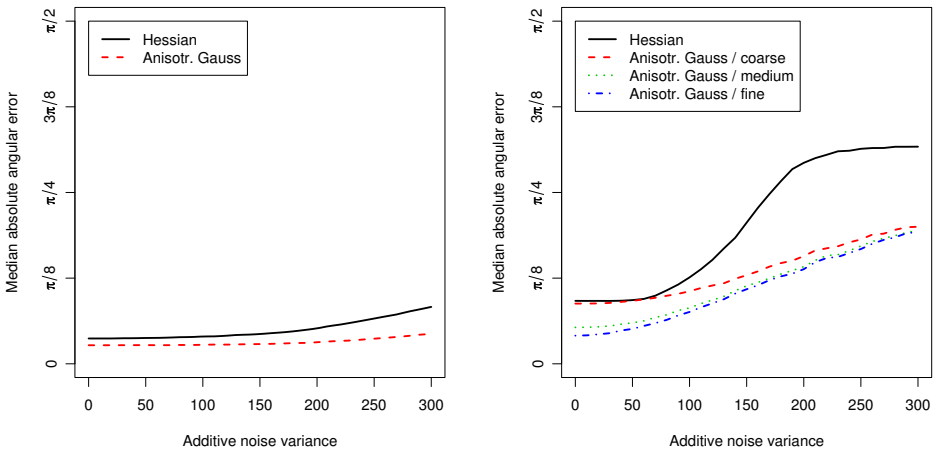
This first part of the evaluation section analyzes the accuracy of orientation estimates at fiber locations. For this purpose, the curved fiber model is used, for which at each location $\mathbf{x}$ along a fiber path $P$ the tangent direction vector $\mathbf{v}$ is known, cf. Sec. 3.5.1. The angular difference between this true orientation $\mathbf{v}$ and the estimated $\hat{\mathbf{v}}$, computed either using reduced Gaussian orientation space or from the Hessian matrix, is the error measure used throughout this section. It is given by the inner product,

$$\delta = \arccos(\mathbf{v}^t \hat{\mathbf{v}}). \tag{3.21}$$

In the graphical presentations, the median of $\delta$ over all fiber path pixels in a number of trials is shown. The number of trials was 10 for the 2D-experiments, and 5 for the 3D-experiments at each tested set of parameters. This error measure will be evaluated with respect to three parameters: Additive independent Gaussian noise ($\sigma^2$), fiber curvature ($\kappa$), and fiber length density ($L_A$, $L_V$). As the number of fibers varied across the different experiments, the number of angular difference measurements over which the median was computed was also not constant. Therefore, this number, denoted by $m$, will be given for each plot separately.

2D and 3D-realizations of the Markov process were generated in images of size $320 \times 320$ and $128 \times 128 \times 128$ pixels, respectively. The fiber radius was fixed to $r = 5$ pixels in either case, and consequently $\sigma = 5$ was used for the Hessian method, and the lengths of the minor axes of the anisotropic Gaussian convolution filters were $s_2 = 5$.

The control method using the eigenvector corresponding to the smallest eigenvalue of the Hessian matrix as direction estimate does not require any quantization. In contrast, the orientation




(a) 2D ($L_A = 0.039$, $\kappa = 1000$, $m = 40000$) (b) 3D ($L_V = 0.0031$, $\kappa = 10000$, $m = 32000$)

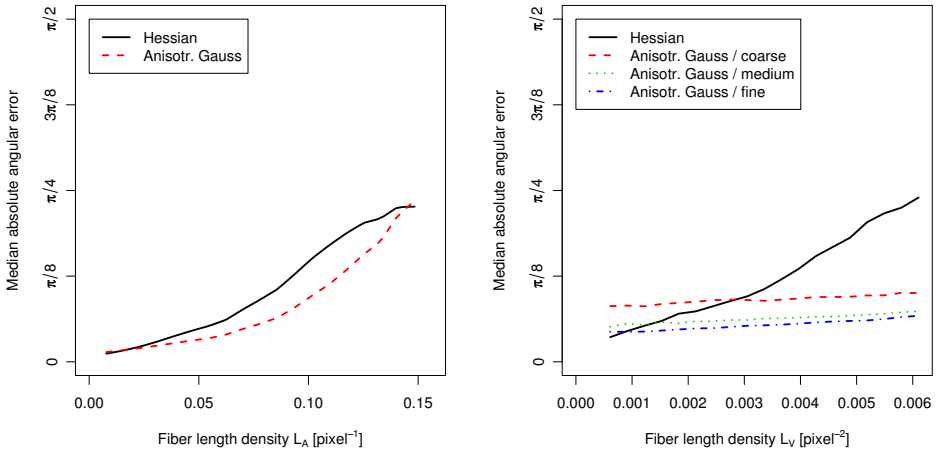
Figure 3.9: Accuracy of the computed directions along $m$ simulated fiber points with respect to independent, additive noise. The orientation estimates obtained along the fiber paths using anisotropic Gaussian filters are more robust to noise than those obtained from the Hessian matrix, both in 2D and 3D. Exemplary visualizations of the effects of additive noise are shown in Fig. 3.8.

space must be sampled at a fixed set of points. For all 2D-experiments, the upper semicircle was quantized in 180 steps. The bank of anisotropic 2D-Gaussian filters therefore has a constant angular resolution of $1^\circ$ or $\pi/180$. Computations of the reduced 3D-orientation space are performed at the three resolution levels described in Sec. 3.3, referred to as coarse, medium and fine resolution. These points are approximately uniform with mean angular resolutions of $33.1^\circ = 0.18\pi$, $19.7^\circ = 0.11\pi$ and $13.8^\circ = 0.08\pi$, respectively, see App. C for details.

The first experiment analyzes the angular error with respect to noise (Fig. 3.9). For this purpose, 2D and 3D-images were used, each containing 25 and 50 rather straight fibers of length 160 and 128 pixels, respectively. For this set of parameters, the proposed method is more precise in detecting the local orientation structure, both in 2D and 3D. While the difference is low for no or little noise, it increases with noise variance. This is observed in 2D and 3D, but the difference is less pronounced in planar images. Moving between resolution levels of the Gaussian orientation space method in 3D, note that the accuracy differences remain below the respective differences in mean nearest neighbor distance of the sampling points. Moreover, the gain in accuracy from using more sampling points on the hemisphere vanishes with an increasing amount of additive noise.

Anisotropic Gaussian filters not only detect the image structures to which they are aligned. They are also low-pass filters, which may explain their good performance on noisy images. Derivative filters, on the other hand, are high-pass filters. Even though the Hessian is computed here using derivate-of-Gaussian filters at scale $\sigma$, the amount of effective smoothing is lower than for the anisotropic Gaussian filters.

For the second experiment, the noise variance was fixed to $\sigma^2 = 50$, a level at which the anisotropic Gaussian and the Hessian methods both performed well, see above. Then, the fiber length densities were varied by increasing the number of fibers from 5 to 50 and from 10 to




(a) 2D ($\kappa = 1000$, $\sigma^2 = 50$, $m = 8000 \ldots 80000$)

(b) 3D ($\kappa = 10000$, $\sigma^2 = 50$, $m = 6400 \ldots 64000$)

Figure 3.10: Accuracy of the computed directions along $m$ simulated fiber points with respect to fiber density. The local orientations along fiber paths computed from the Hessian matrix are more strongly influenced by the fiber density. Realizations at different fiber length densities are shown in Fig. 3.6.

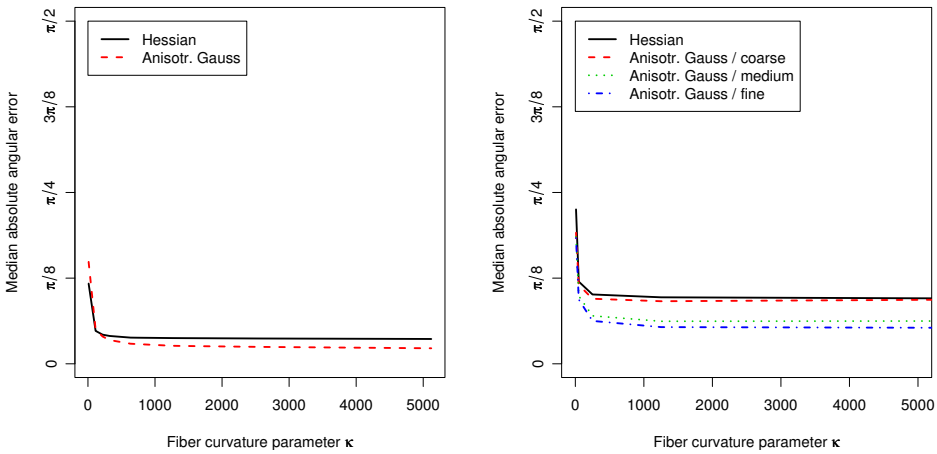
100 in the 2D and 3D-data, respectively (Fig. 3.10). With increasing fiber density, the simulated fibers overlap and intersect more strongly, making the orientation detection problem more difficult. While the local orientations derived from the Hessian matrix are slightly more accurate than those computed from Gaussian orientation space for very low fiber densities, the proposed method is more accurate than the control method at higher fiber densities. The results for the Hessian method are similar in 2D and 3D. The accuracy of anisotropic Gaussian filters, on the other hand, seems to be less subjective to the fiber density in 3D than in 2D-images.

This different behavior for 3D-data is not a property of the anisotropic Gaussian filters, but rather a property of the data. In 3D, the amount of fiber intersections and overlap is much lower, a fact that is also expressed in the fiber length densities: $L_V$ remains one order of magnitude below $L_A$ in these experiments. Higher values of $L_V$ were not tested as the Hessian control method's median angular error was already $\pi/4 = 45°$ at the densities investigated here.

The two experiments reported so far used almost straight, overlapping fibers. As the proposed method is based on anisotropic kernels with fixed aspect ratio 2:1, it should perform best for straight fibers. Using the curvature parameter $\kappa$ of the Markov fiber process from Sec. 3.5.1, the influence of fiber curvature on the accuracy of local orientation estimates was investigated in a third experiment (Fig. 3.11). Results obtained using the Gaussian orientation space method are again more accurate than the gradient-based approach for a wide range of the fiber curvature parameter $\kappa$. Both methods fail for highly curved fibers.

The conclusion of this last experiment is that the use of prolate filter kernels is not a hindrance when analyzing curved fibers. For highly curved fibers, the method does fail as expected, but this is also true for the control method which does not rely on such filters.

All evaluations presented so far measured the absolute angular error $\delta = \arccos(\mathbf{v}^t \hat{\mathbf{v}})$. While




(a) 2D ($L_A = 0.039$, $\sigma^2 = 50$, $m = 16000$) (b) 3D ($L_V = 0.0031$, $\sigma^2 = 50$, $m = 32000$)

Figure 3.11: Accuracy of the computed directions along $m$ simulated fiber points with respect to $\kappa$, a measure for the bending of fibers in these simulations. Both the proposed and the control method are limited by the curvature of fibers. This was to be expected for the orientation space approach described in this chapter, as the length of the filter kernel puts a limit on the amount of curvature that can be handled. These results show that this is not a disadvantage compared the control method. Different degrees of curvature are shown in Fig. 3.6.

$\delta$ is suitable for judging the amount of error in the local orientation estimates, it does not allow for a judgment of possible systematic errors. No bias was observed in any of the experiments. To demonstrate this in at least one selected case, the estimated direction vectors along fiber paths, $\hat{\mathbf{v}}(\mathbf{x})$, were centered with respect to their known counterparts, $\mathbf{v}(\mathbf{x})$. Projection of the resulting 3D-error vectors into the plane gives a visual impression of the spread and position of these errors (Fig. 3.12).

### 3.6.3 Effectiveness of the 3D-orientation distribution parameters

Given the local orientations $\mathbf{v}(\mathbf{x})$ which have been evaluated in the previous section, the orientation tensor $T_W$ can be computed. In the following, the hardcore cylinder model introduced above will be used to evaluate the effectiveness of the distribution parameters $\gamma$ and $\zeta$ that have been described in Sec. 3.4.

Systems of non-overlapping cylinders in $256 \times 256 \times 256$ pixel images were simulated, containing between $n = 50$ and 190 cylinders. For each value of $n$, the orientation distribution's parameter $\beta$ was set to $\beta = 0.01, 0.1, 1.0, 10.0$, resulting in sets of images containing three types of orientation distributions: Two cluster-type distributions ($\beta = 0.01, 0.1$), one isotropic ($\beta = 1.0$) and one girdle-type distribution ($\beta = 10.0$).

As the proposed spherical distribution descriptors $\gamma$ and $\zeta$ are capable of describing exactly these three types, they were evaluated with respect to $\beta$ (Fig. 3.13). The resulting values, especially the strength parameter, are almost independent of the chosen sampling. Furthermore, the fiber density or number of cylinders does not have a large impact on the results.

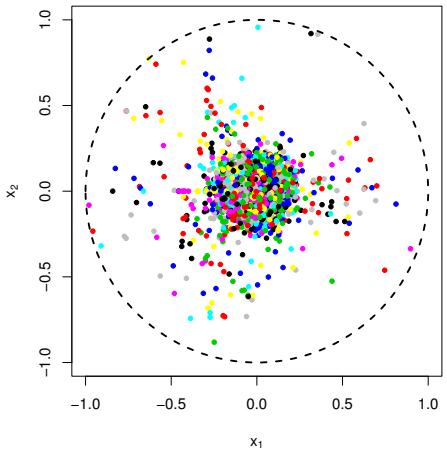


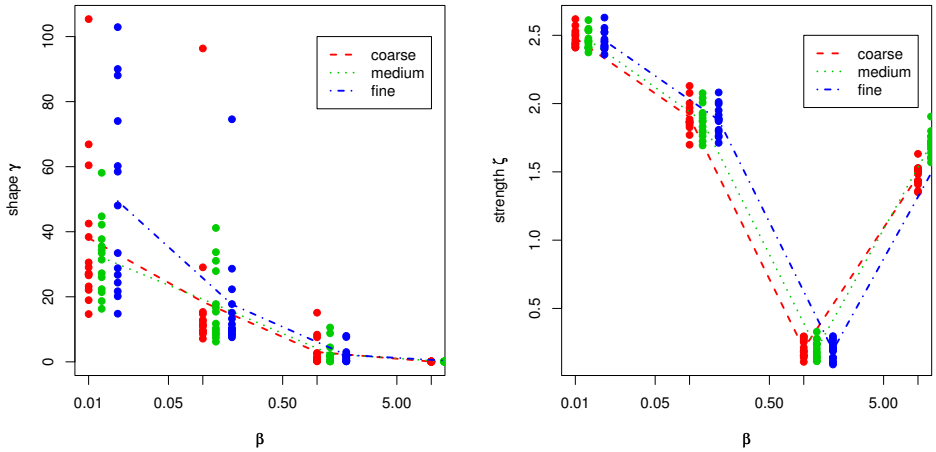

Figure 3.12: A closer examination of the difference vectors of the proposed method of a selected 3D-case: Orthogonal projections of the difference vectors into the plane for medium sampling, no noise, and $\kappa = 1000$. The colors stand for individual fibers, but are not unique. The difference vectors are not biased in any direction (the mean is at (-0.024, -0.0014) in the projection plane).

Combining the results of $\gamma$ and $\zeta$, it is concluded that these two parameters are well suited for characterizing the shape of the 3D-orientation distribution: For $\beta = 0.01, 0.1$, they both assume values larger than one, clearly indicating a cluster-type distribution. In such cases, the mean orientation derived from $T_W$ is a suitable quantity for describing the direction of fiber alignment. In the case of an isotropic fiber orientation ($\beta = 1.0$), the strength parameter $\zeta$ tends towards zero, independent of sampling or fiber density. The remaining case of a girdle-type density, with fibers arranged in parallel planes, is also detected well by these measurements, with the shape parameter $\gamma$ going to zero for all tested resolution levels and fiber densities.

These evaluations have been performed in noiseless images. As good robustness of the local orientations computed using anisotropic Gaussian filters to noise and other distortions has already been demonstrated in the previous section, these findings on the effectiveness of the proposed distribution parameters are expected to be transferable to real-world data.

## 3.7 Applications

This section presents three applications of the proposed method for detecting the fiber orientation from image data (Fig. 3.1). First, 2D-microscopic images showing layers of a glass fiber-reinforced polymer at different depths will be investigated. The manufacturers suspect a change in the fiber orientation distribution from outer to inner fiber layers of this specimen. The second problem concerns a sample of a glass fiber-reinforced polymer that consists of two parts which have been fused. It will be investigated whether this fusing process had any measurable impact on the arrangement of fibers adjacent to the welding seam. Analysis of special carbon paper that finds its use as a gas diffusion layer in polymer electrolyte fuel cells is the topic of the third application example.




(a) The shape parameter $\gamma$ for different sampling resolutions and orientation distributions.

(b) The strength parameter $\zeta$ for different sampling resolutions and orientation distributions.

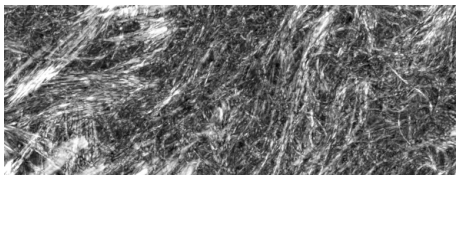
Figure 3.13: Evaluation on the hardcore cylinder model using different parameters and resolutions. The shape and strength parameters were computed in $256^3$ images and are plotted here against the anisotropy parameter $\beta$ of the density in Eq. (3.17). Note the logarithmic scale on the abscissa. The three curves are offset against each other for better visualization. Measurements have been taken at the same values $\beta = 0.01, 0.1, 1.0, 10.0$ in all cases. The dots in these plots correspond to measurements from individual images, each containing a different number $n$ of cylinders.

### 3.7.1 Sheet molding compound

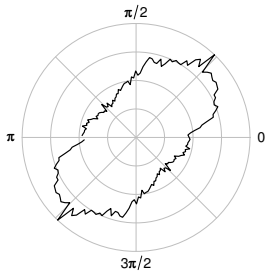
Sheet molding compounds (SMC) are thin layers of fiber-reinforced polymers which can be shaped by compression molding. These sheets are produced with some predefined fiber orientation to provide the required plasticity and stability when molding the final structure. A specimen of a glass fiber-reinforced SMC was produced and imaged using scanning acoustic microscopy (SAM) at Fraunhofer EMI, Freiburg. The SAM images show the 2D-microstructure of the SMC sample at different depths relative to the top-view surface shown in Fig. 3.1 at the beginning of this chapter.

The sample investigated here, which is part of a larger study, was designed to contain fibers oriented at 45° (in plane). Due to the production process, it is suspected that fibers in the outer layers of this SMC sample posses a highly anisotropic orientation distribution with mean orientation at 45°, and that the fibers tend towards a more isotropic orientation distribution deeper within the specimen.

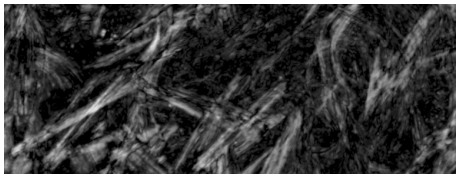
Two images depicting parallel layers 0.3 mm apart have been analyzed using the proposed method (Fig. 3.14). At both depths, the orientation distributions are anisotropic with the desired mean orientation around 45°. An increase in isotropy at the lower level could not be detected. On the contrary, the orientation distribution measured near the surface (Fig. 3.14(b)) spreads farther around the expected mean orientation than the corresponding distribution deeper within the sample (Fig. 3.14(d)). At the time of writing, an analysis of this and other effects on a large series of samples was still under way.



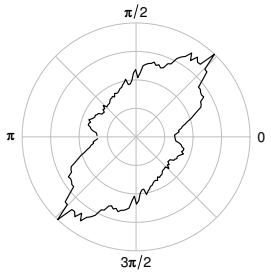
(a) Glass fibers near the surface




(b) Orientation distribution near the surface



(c) Glass fibers 0.3 mm deeper inside




(d) Orientation distribution 0.3 mm deeper inside

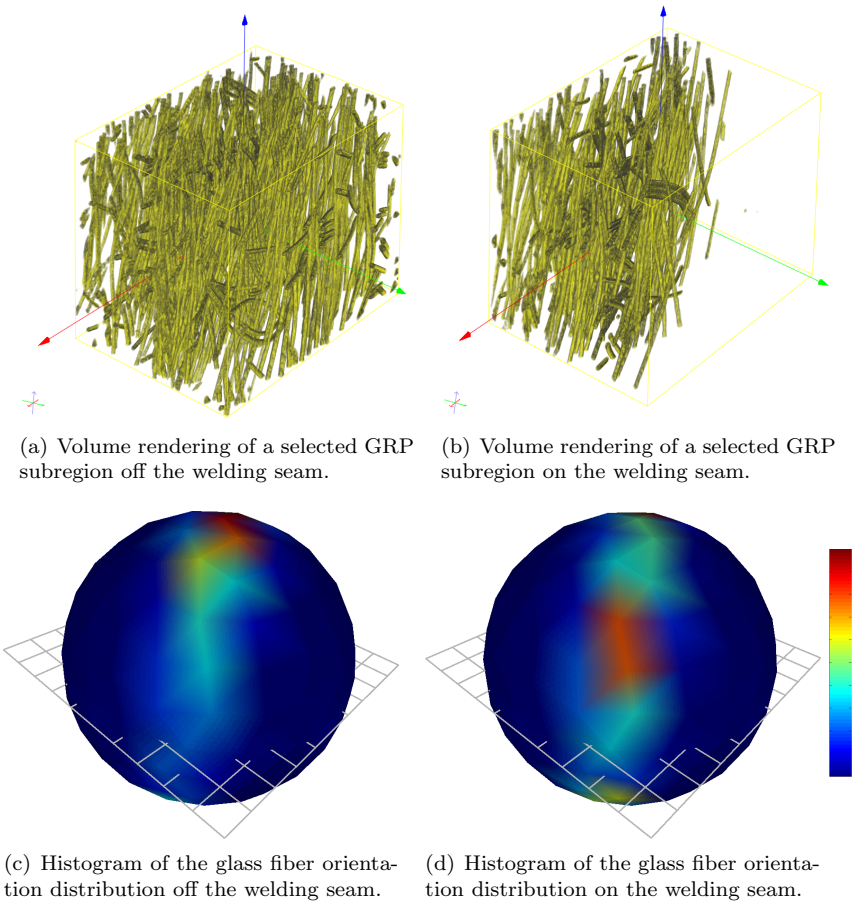
Figure 3.14: Fiber orientation distributions in images of a glass fiber-reinforced polymer specimen, imaged by scanning acoustic microscopy in two different depths. At both depths, the fibers show an anisotropic distribution with most fibers oriented at 45°.

### 3.7.2 Glass fiber-reinforced polymer

The sample investigated here was provided by the Institute for Composite Materials (IVW), Kaiserslautern, and imaged in phase-contrast mode at the European Synchrotron Radiation Facility (ESRF), Grenoble. Therefore, the data has little noise and high edge contrast along the glass fibers. The specimen contains glass fibers with a diameter of about 30 $\mu$m imaged with a lateral resolution of 3.5 $\mu$m. The total size of the imaged area, containing both parts and the welding seam, was about $7 \times 7 \times 3$ mm$^3$. Out of this larger sample, several $1.4 \times 1.4 \times 1.4$ mm$^3$ subregions at and off the welding seam were extracted.

Orientation histograms from Gaussian orientation space for each subregion have been computed, two of which are shown in Fig. 3.15. The strength parameter $\zeta$ varies around values of 1.5 for all subregions, clearly indicating an anisotropy of the fibers that is also visible in the image data. $\gamma$, the spherical distribution descriptor for the shape, does differ among the different subregions. For the samples taken off the welding seam, it varies between approximately 0.3 and 0.6, while the $\gamma$-values of the sample along the welding seam are found to be in the range from 0.1 to 0.3. While this observation can not be associated with any statement of its significance, it does hint at a farther spread of the glass fibers' orientations near or at the welding seam.

This can also been seen from the orientation histograms in Fig. 3.15, where more fibers oriented away from the $x_3$-axis are found in the sample taken on the welding seam. The mean orientation



(a) Volume rendering of a selected GRP subregion off the welding seam.

(b) Volume rendering of a selected GRP subregion on the welding seam.

(c) Histogram of the glass fiber orientation distribution off the welding seam.

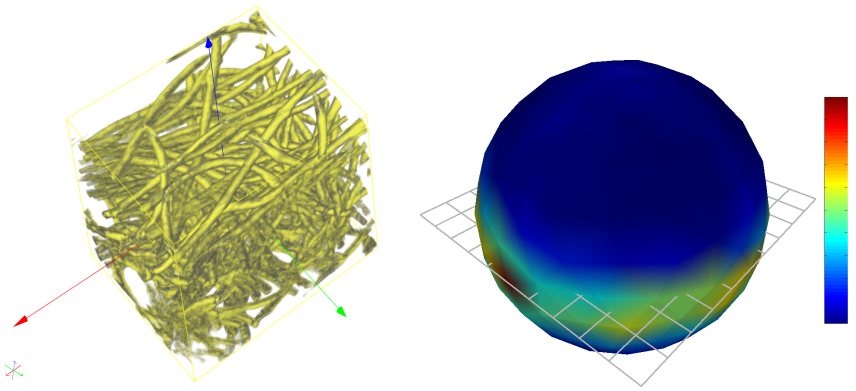
(d) Histogram of the glass fiber orientation distribution on the welding seam.

Figure 3.15: Visualizations of data and results for the glass fiber-reinforced polymer (GRP) application example on and off the welding seam. While the glass fibers are mostly oriented parallel to the $x_3$-axis (vertical direction in these images), many fibers along the welding seam are rotated away from that direction.

was found to lie close to $(0, 0, 1)^t$ in all cases, but as the orientation distributions show a girdle-type form in the histograms, this mean orientation is not a meaningful figure in this case.

### 3.7.3 Carbon-coated paper

A specimen of a carbon paper gas diffusion layer was imaged using $\mu$CT with a lateral resolution of 0.7 $\mu$m at the European Synchrotron Radiation Facility (ESRF), Grenoble. For a detailed account of this specific dataset, and for simulation results concerning the material's physical properties such as permeability and relative diffusivity, see [14]. To obtain an image region suitable for evaluation, a cubic volume of $231 \times 231 \times 231$ $\mu$m$^3$ was chosen for image analysis out of the original sample ($1.43 \times 1.43 \times 0.34$ mm$^3$). The fiber diameter in this dataset is approximately 11 $\mu$m. This evaluated region and the resulting histogram of fiber orientations are shown in Fig. 3.16. The paper fibers show a clear planar arrangement in the $x_1x_2$-plane, with a preferred orientation parallel to the $x_1$-axis.



(a) Volume rendering of the evaluated region of the carbon paper specimen.

(b) Orientation histogram of the carbon paper fibers.

Figure 3.16: Visualization and Gaussian orientation space results for the carbon paper example. The orientation histogram shows a girdle-type fiber orientation distribution. Within the plane in which the fibers are aligned, a preferred direction can be seen in the histogram.

The resulting mean orientation vector $\mu = (-0.75, 0.66, -0.07)^t$ accordingly lies in the $x_1x_2$-plane. The shape $\gamma = 0.15$ and strength $\zeta = 0.66$ indicate a girdle-type orientation distribution, but as the value of $\zeta$ is smaller than one, a significant amount of fibers with non-planar tangent direction vectors must exist.

## 3.8 Discussion

This chapter described a novel method for computing the orientation distribution of 2D and 3D-fiber systems from image data, which does not require segmentation of individual fibers. Experimental evaluations on simulated data have shown that the proposed approach is more robust to noise and fiber density (overlap) than an alternative method that is frequently used in the image processing literature. Using the sample second order moments of the local orientation vectors obtained by this method, the second order orientation tensor can be computed. This tensor is frequently used for simulating mechanical or other physical properties of anisotropic fibrous materials. Furthermore, it allows for non-parametric descriptors of the orientation distribution's properties (mean orientation, shape) to be computed. These descriptors were experimentally shown to be accurate and powerful descriptive tools for analyzing fiber-reinforced polymers and other fibrous structures, especially when combined with visualizations of the measured 3D-orientation distributions.

The basic operation required to implement the proposed approach is image filtering with anisotropic Gaussian kernels, which is repeatedly performed to detect local image structures. Consequently, it is essential to efficiently implement this filter operation. No runtime or implementation details have been reported in this chapter, as the following chapter will introduce a factorization of the Gaussian kernel that leads to a highly efficient implementation of anisotropic Gaussian filtering. Timing results of the method proposed in the present chapter will also be presented there,

compared to an alternative implementation.

Even when using such an optimized implementation, computation of the reduced Gaussian orientation space remains an expensive computational problem. The control method that was used to put the performance of the proposed approach into a relation to established algorithms requires a smoothing operation followed by several differentiation filters and one eigen decomposition of a $d \times d$ matrix per pixel. Summing these operations together, computation of local orientations using this alternative method remains significantly faster than the proposed method using anisotropic filter banks. Note however that this aforementioned control method has a high memory cost, requiring $\mathcal{O}(d^2)$ memory per pixel.

It is worth paying the price of long runtime of the presented method, though, as it results in more precise and more stable results than the aforementioned other method. While the analysis of the microstructure of materials is beginning to be routinely used in some industry sectors, it still remains an offline procedure. Considering this circumstance, this chapter contributes a reliable and suitable method to the collection of available tools for analyzing the microstructure of fibrous materials.

# Chapter 4

# A Non-Orthogonal Separation of the Anisotropic Gaussian Convolution Filter

## 4.1 Introduction

The method for computing the orientation distribution of fibers in 2D and 3D-images that was introduced in the previous chapter requires a large number of anisotropic Gaussian filtering operations. This chapter describes a novel separation of the anisotropic Gaussian filter kernel, which generalizes a previously known result, and leads to a faster implementation of the corresponding convolution operation.

The commonly used isotropic Gaussian filter has one degree of freedom, which is the kernel width. The $d$-dimensional anisotropic Gaussian filter, on the other hand, has $d(d+1)/2$ degrees of freedom. Geometrically, these variables govern the spread of the filter in $d$ directions and its orientation. When using this type of filter, a mechanism must be devised for choosing those parameters. Applications of anisotropic Gaussian convolution filters can be grouped into three categories, according to how they approach this parameter choice.

The first method applies filter banks, i.e., sets of filters with different parameter settings, for constructing orientation spaces. This was the approach in Ch. 3, where Gaussian kernels of prolate shapes were used to measure local orientations in images of fibrous microstructures.

A second group of methods computes the filter parameters from images. Yang et al. align anisotropic Gaussian kernels with local gradient directions to avoid smoothing across edges [175]. Following that general idea, Knossow et al. applied anisotropic Gaussian filters to 2D-images for smoothing in a tracking application [87], and Sijbers et al. aligned anisotropic Gauss kernels with the directions of a second moment matrix derived from an image's power spectrum to smooth 3D-images obtained by magnetic resonance imaging (MRI) [149]. Crum et al. apply anisotropic 3D-Gaussian filters to compensate for anisotropic pixel grids in registering medical MRI data [33].

The third and last group of applications of anisotropic Gaussian filters are those where the filter

parameters follow from the imaging system itself, which is the case for diffusion tensor MRI (DT-MRI) [170]. That imaging modality measures the diffusion of water, which relates directly to local anisotropies. Gaussian filters can be aligned to that tensor, resulting in one of the best-performing smoothing filters among those tested in [101] for smoothing DT-MRI images.

Regardless of how the filter parameters are set, all of these applications of anisotropic Gaussian filters can profit from the separation scheme and corresponding implementation that will be described in this chapter.

Apart from Gaussian filters, a variety of anisotropic image filters have been proposed. Anisotropic diffusion [168] filters along edges by effectively deforming the filter mask according to local gradients. Steerable filters [52] have drawn a lot of attention, which generate a filter basis from which the filter response at an arbitrary orientation can be computed as a linear combination of the elements in that basis. The derivative of Gaussian filter is the most prominent steerable filter. Anisotropic Gaussian filters are not steerable, methods for constructing approximate steerable filter bases have been described by Perona [131].

The concept of separable filters is different from that of steerable filters. Separable filters are factorizations of convolution filters into a series of (usually) lower-dimensional filter kernels. Fast filtering is achieved by applying a sequence of these lower dimensional convolutions, rather than filtering a $d$-dimensional image with a $d$-dimensional filter mask.

Several approximate separation schemes for efficient anisotropic filtering have been proposed. Andersson et al. developed a general strategy for separating filters into a sequence of convolutions that involves a numerical optimization to obtain the filter coefficients [2]. Lakshmanan proposed a separable anisotropic filter for smoothing weather radar images [93]. An approximate separation of the 3D-anisotropic Gaussian kernel with 2 fixed and one variable filtering directions was proposed in [172], but it is accurate only for a certain range of kernel orientations. Recently, Lam and Shi introduced an interpolation-free implementation of the 2D-anisotropic Gaussian filter that is suitable for some kernel aspect ratios [94].

The first accurate factorization of the anisotropic Gaussian convolution kernel in $\mathbb{R}^2$ into a product of two one-dimensional Gaussian kernels was described by Geusebroek, Smeulders and van de Weijer [60, 61]. Their result demonstrated that fast and accurate anisotropic 2D-filtering could be implemented by filtering along one axis-parallel direction and along a second, non-orthogonal direction that can be derived from the filter parameters.

This chapter introduces a factorization of the $d$-dimensional anisotropic Gaussian filter kernel that is optimal in the required number of memory accesses and interpolation operations. The novel filter separation has an intuitive geometric interpretation, and it will turn out that the separation of the two dimensional Gaussian kernel proposed in [60, 61] is a special case of this result. Through this fundamental understanding of optimal factorizations of the anisotropic Gaussian filter, efficient implementations of the three dimensional Gaussian filter are accessible, which lead to practically significant speedups of both adaptive smoothing and orientation space computations.

### 4.1.1 Separable filters

To see how and why separable filtering would lead to a speedup of anisotropic Gaussian convolution, let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ denote vectors and $g : \mathbb{R}^d \to \mathbb{R}$ the $d$-dimensional Gaussian kernel given by

$$g(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}\mathbf{x}^t\Sigma^{-1}\mathbf{x}\right\}. \tag{4.1}$$

$\Sigma$ is the $d \times d$ covariance matrix, and $|\Sigma|$ its determinant. To convolve an image $f : \mathbb{R}^d \to \mathbb{R}$ with filter $g$ means to integrate over the point-wise product of $f$ and $g$ at all locations $\mathbf{x}$,

$$(f * g)(\mathbf{x}) := \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\mathbf{y})g(\mathbf{x}-\mathbf{y})\; dy_1\, dy_2\, \dots\, dy_d, \tag{4.2}$$

where $*$ denotes the linear convolution operator and $y_1, y_2, \dots, y_d$ are the $d$ components of vector $\mathbf{y}$. Without forestalling much of the material on different implementation strategies for convolutions that will be covered in Sec. 4.5, the motivation for what follows is to achieve an efficient implementation of this integral.

The simplest implementation of (4.2) replaces the continuous integrals by sums of discretized representations of $g$ and $f$ and truncates sums at some point, say at $\pm w \in \mathbb{Z}$. This results in a computational complexity of $\mathcal{O}(n^d \cdot w^d)$ for the $n^d$ pixels in a discrete image. By utilizing the convolution theorem and the fast Fourier transformation (FFT), the integrals in (4.2) can be evaluated in $\mathcal{O}(n^d \log n^d)$ time [135]. As $w$ is not a negligible factor for filters with large spread, this is an improvement over the implementation using sums truncated at $\pm w$.

Imagine that the integrals in (4.2) could be unraveled to a series of decoupled one-dimensional integrals as in

$$(f * g)(\mathbf{x}) = \int_{-\infty}^{\infty} g_d(u_d - v_d)\; \dots \int_{-\infty}^{\infty} g_2(u_2 - v_2) \int_{-\infty}^{\infty} g_1(u_1 - v_1)f(\mathbf{v})\; dv_1\, dv_2\, \dots\, dv_d, \tag{4.3}$$

where $\mathbf{u}$ and $\mathbf{v}$ are $\mathbf{x}$ and $\mathbf{y}$ transformed to a new coordinate system, respectively, using a yet to be specified mapping, see Sec. 4.2. Eq. (4.3) consists of $d$ one-dimensional convolutions, rather than one $d$-dimensional convolution as in (4.2). A filter $g$ for which a convolution integral in the form of (4.3) exists is called separable. If discrete variants of these one-dimensional convolutions could be implemented with $\mathcal{O}(1)$ complexity per pixel, then then the $d$-dimensional discrete convolution could be performed in $\mathcal{O}(n^d)$ time.

The following section first demonstrates how the Gaussian kernel $g(\mathbf{x})$ can be factorized. This factorization will then be substituted into Eq. (4.2), which leads to the sought after form of a separated convolution integral. In the sequel of the present chapter, an optimal transformation from $\mathbf{x}$ to $\mathbf{v}$ is derived, and different algorithms for implementing this separated anisotropic Gaussian filter will be discussed and analyzed.

## 4.2 Separating the Gaussian convolution integral

Isotropic and axis-aligned Gaussians, where $\Sigma$ is diagonal, factorize readily along the $d$ coordinate directions. For the general, anisotropic Gaussian considered in the present chapter, many different factorizations are possible. E.g., the separation of possibly anisotropic Gaussians along their orthogonal major axes is a standard procedure in statistical pattern recognition, usually in the setup of the "whitening transform", see e.g. [41, Ch. 2]. This chapter, however, considers more general separations of the Gaussian convolution filter along arbitrary, possibly non-orthogonal axes in $\mathbb{R}^d$.

The following two results build the basis for what follows in this chapter. While the first of these describes a general mechanism for separating the Gaussian filter kernel by factorizing its covariance matrix, the second one demonstrates that such a separation leads to a separated convolution filter. Sec. 4.3 then introduces a factorization that that is optimal for implementing (4.2).

**Proposition 1 (Non-orthogonal factorizations of the Gaussian)** *Let $\Sigma$ be the $d \times d$ symmetric, positive definite covariance matrix of the Gaussian function. For any decomposition $\Sigma = VDV^t$ into square matrices $D$ and $V$, where $D$ is diagonal with positive entries and $V$ has determinant 1, there exists a factorization of the $d$-dimensional Gaussian into $d$ one-dimensional Gaussians. The separation directions and the 1D-Gaussians' variance parameters are defined by $V$ and $D$, respectively.*

**Proof 1** Assuming that the required factorization of $\Sigma$ exists, the Gaussian function $g$ from (4.1) can be rewritten as

$$g(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}\mathbf{x}^t\Sigma^{-1}\mathbf{x}\right\} = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}\mathbf{x}^t(V^{-t}D^{-1}V^{-1})\mathbf{x}\right\}. \quad (4.4)$$

Since $|V| = 1$, it follows that $|D| = |\Sigma|$ and

$$g(\mathbf{x}) = \frac{\exp\left\{-\frac{1}{2}\mathbf{x}^t(V^{-t}D^{-1}V^{-1})\mathbf{x}\right\}}{(2\pi)^{d/2}|D|^{1/2}} = \frac{\exp\left\{-\frac{1}{2}(V^{-1}\mathbf{x})^tD^{-1}(V^{-1}\mathbf{x})\right\}}{(2\pi)^{d/2}|D|^{1/2}}.$$

After a linear change of coordinates from $\mathbf{x}$ to $\mathbf{v} = (v_1, \ldots, v_d)$ with $\mathbf{v} := V^{-1}\mathbf{x}$, this becomes

$$g(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|D|^{1/2}} \exp\left\{-\frac{1}{2}\mathbf{v}^tD^{-1}\mathbf{v}\right\}.$$

$D$ is known to be a diagonal matrix with positive entries, denoted by $d_1^2, \ldots, d_d^2$, such that $|D|^{1/2} = d_1 \cdot \ldots \cdot d_d$. It follows that $D^{-1}$ is diagonal with positive entries as well, namely $D^{-1} = \mathrm{diag}(\frac{1}{d_1^2}, \ldots, \frac{1}{d_d^2})$. $D^{-1}$ exists as $|D| = |\Sigma| > 0$. Therefore, the matrix product is in fact just a weighted sum of squares,

$$g(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}d_1 \cdot \ldots \cdot d_n} \exp\left\{-\frac{1}{2}\sum_{i=1}^{d}\frac{v_i^2}{d_i^2}\right\}.$$

According to the laws of exponentiation, this can be factorized into

$$g(\mathbf{x}) = \frac{1}{\sqrt{2\pi}d_1}\exp\left(-\frac{1}{2}\frac{v_1^2}{d_1^2}\right)\cdot\ldots\cdot\frac{1}{\sqrt{2\pi}d_d}\exp\left(-\frac{1}{2}\frac{v_d^2}{d_d^2}\right) = g_1(v_1)\cdot\ldots\cdot g_d(v_d), \tag{4.5}$$

where each $g_i(v_i) := \frac{1}{\sqrt{2\pi}d_i}\exp\left(-\frac{1}{2}\frac{v_i^2}{d_i^2}\right)$ is an ordinary 1-dimensional Gaussian of mean 0 and variance $d_i^2$. As $\mathbf{v} = V^{-1}\mathbf{x}$, this factorization has the required form. ∎

**Corollary 1 (Separations of the Gaussian convolution integral)** *Let $\Sigma$ be the $d \times d$ symmetric, positive definite covariance matrix of the Gaussian function. For any decomposition $\Sigma = VDV^t$ into square matrices $D$ and $V$, where $D$ is diagonal with positive entries and $V$ has determinant 1, the $d$-dimensional Gaussian convolution integral can be computed by a sequence of $d$ one-dimensional Gaussian convolution integrals in not necessarily orthogonal directions given by the columns of $V$.*

**Proof 2** For a general function $f : \mathbb{R}^d \to \mathbb{R}$, the convolution with the Gaussian $g$ is defined as

$$(f * g)(\mathbf{x}) := \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty} f(\mathbf{y})g(\mathbf{x}-\mathbf{y})\; dy_1\, dy_2\, \ldots\, dy_d.$$

As $V$ is not singular ($|V| = 1$), a change in coordinates from $\mathbf{x}$ to $\mathbf{u} := V^{-1}\mathbf{x}$ and from $\mathbf{y}$ to $\mathbf{v} := V^{-1}\mathbf{y}$ gives

$$(f * g)(\mathbf{x}) = \int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(V\mathbf{v})g\left(V\mathbf{u} - V\mathbf{v}\right)|V^{-1}|\; dv_1\, dv_2\, \ldots\, dv_d,$$

where the factor $|V^{-1}|$ enters due to the coordinate change. Using Proposition 1, and $|V^{-1}| = |V|^{-1} = 1$, the Gaussian function in $V\mathbf{u} = \mathbf{x}$ can be factorized, yielding

$$(f*g)(\mathbf{x}) = \int_{-\infty}^{\infty} g_d(u_d - v_d)\int_{-\infty}^{\infty} g_{d-1}(u_{d-1} - v_{d-1})\ldots\int_{-\infty}^{\infty} g_1(u_1 - v_1)f(V\mathbf{v})\; dv_1\, \ldots\, dv_{d-1}\, dv_d. \tag{4.6}$$

Note that the factor $V$ in the $d$-dimensional Gaussian $g\left(V\mathbf{u} - V\mathbf{v}\right)$ cancels out when substituted into Eq. (4.4). This is the standard form of a separated convolution integral. Using $\mathbf{v}^i$ to denote the $i$'th column of $V$, the matrix-vector product in the argument of $f$ can be expanded to $V\mathbf{v} = v_1\mathbf{v}^1 + v_2\mathbf{v}^2 + \ldots + v_d\mathbf{v}^d$. When integrating over $v_i$, all $v_j$ remain constant ($j \neq i$). The $i$'th integral in (4.6) therefore convolves a 1D-Gaussian $g_i$ with image $f$ along the direction $\mathbf{v}^i$, as demanded by the corollary. ∎

The result of Corollary 1 can be written very compactly using the notation of *directional convolutions*,

$$g * f = g_d *_{\mathbf{v}^d} \ldots g_2 *_{\mathbf{v}^2} g_1 *_{\mathbf{v}^1} f,$$

where the *directional convolution operator* $*_{\mathbf{v}}$ indicates convolution along direction $\mathbf{v}$, defined as

$$(g *_{\mathbf{v}} f)(\mathbf{x}) := \int_{-\infty}^{\infty} g(\lambda) f(\mathbf{x} - \lambda \mathbf{v}) \, d\lambda. \tag{4.7}$$

This integration is one-dimensional, even though the direction is specified by a vector $\mathbf{v} \in \mathbb{R}^d$. With these results at hand, the problem of separating the $d$-dimensional anisotropic Gaussian convolution reduces to the problem of finding useful factorizations $\Sigma = VDV^t$. These factorizations are not unique and can be constructed using elementary matrix operations [63]. One such factorization of $\Sigma$ is the eigen decomposition that has already been discussed in the context of Gaussian orientation spaces in Sec. 3.2.

When using the eigen decomposition of $\Sigma$, the columns of $V$ correspond to the major axes of the hyper-ellipsoid $\{\mathbf{x} : \mathbf{x}^t \Sigma^{-1} \mathbf{x} = 1\}$. As $\Sigma$ is symmetric, these $d$ column vectors are mutually orthogonal and of unit length. This reproves the well-known fact that an anisotropic Gaussian is always separable along its major axes.

For implementing Eq. (4.7), however, the eigen decomposition is not very useful. Except for axis-aligned Gaussians, $\Sigma$'s eigenvectors $\mathbf{v}^i$, $i = 1, \ldots, d$, do not have any integer entries (their norms are 1). Therefore, $\mathbf{x} - \lambda \mathbf{v}$ does not lie on the sampling grid. While this is not a problem from the mathematical point of view, discretized implementations must interpolate in all $d$ dimensions during each of the $d$ integration steps. This makes the computations slow and numerical errors can occur and accumulate.

## 4.3 An optimal symmetric factorization of Σ

Following the previous train of thought on the weaknesses of an implementation of anisotropic Gaussian convolution separated along the filter's major axes leads to criteria that characterize useful symmetric factorizations of $\Sigma$. In case of the eigen decomposition, each filter direction $\mathbf{v}^i$ is a unit vector. In general, it requires $d$-dimensional interpolation when implementing discretized variants of each of the $d$ 1D-convolutions in the directions described by Corollary 1. A factorization that is optimal for discrete implementation of Eq. (4.2), on the other hand, will eliminate the need for interpolation whenever possible. Discrete implementations of the directional convolution operator $*_{\mathbf{v}}$ from Eq. 4.7 are therefore most efficient and precise if $\mathbf{v}$ fulfills the following:

- $\mathbf{v}$ should have as many zero components as possible, as each zero lowers the dimensionality of required interpolation by 1.

- $\mathbf{v}$ should have as many integer entries as possible, as each of these eliminates the need for interpolation in the corresponding dimension.

By Corollary 1, the $d$ convolution directions $\mathbf{v}^i$ are given by the columns of $V$. $V$ must have determinant 1, which restricts the possible choices. Consider the symmetric factorization

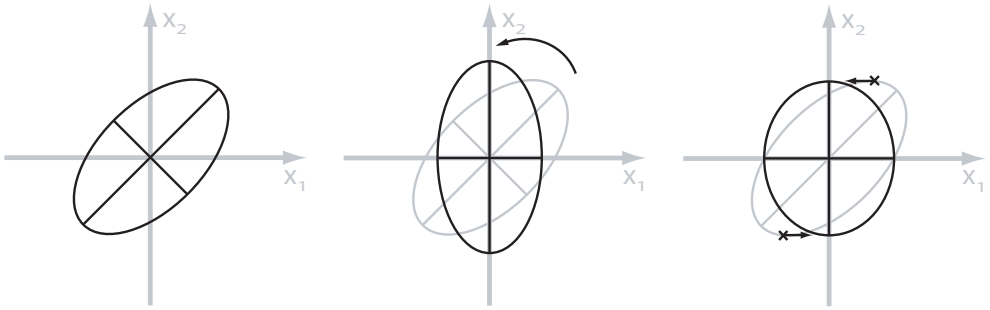
$$\Sigma = VDV^t \tag{4.8}$$
$$= \begin{pmatrix} 1 & v_{1,2} & v_{1,3} & \dots & v_{1,d} \\ & 1 & v_{2,3} & \dots & v_{2,d} \\ & & \ddots & & \vdots \\ & & & 1 & v_{d-1,d} \\ & & & & 1 \end{pmatrix} \begin{pmatrix} d_1^2 & & & & \\ & d_2^2 & & & \\ & & \ddots & & \\ & & & d_{d-1}^2 & \\ & & & & d_d^2 \end{pmatrix} \begin{pmatrix} 1 & & & & \\ v_{1,2} & 1 & & & \\ v_{1,3} & v_{2,3} & \ddots & & \\ \vdots & & & 1 & \\ v_{1,d} & v_{2,d} & \dots & v_{d-1,d} & 1 \end{pmatrix}.$$

This factorization has some similarities to the Cholesky decomposition $A = LL^t$ of a symmetric positive definite matrix $A$. There, $L$ is lower triangular and has positive diagonal entries [63]. Let $\tilde{d}_{ii}$ denote the $d$ entries along the diagonal of $L$. As all $\tilde{d}_{ii}$ are positive, the Cholesky decomposition can be further factorized into $A = LL^t = \tilde{L}\tilde{D}\tilde{L}^t$ with $\tilde{D} := \text{diag}(\tilde{d}_{11}^2, \tilde{d}_{22}^2, \dots, \tilde{d}_{dd}^2)$. As $\tilde{L}$ is again lower triangular, not upper triangular, Eq. (4.8) will be referred to as the *symmetric factorization of Cholesky type*. The existence of this factorization for $d = 2$ and $d = 3$ will follow from the results in Sec. 4.4.1. Next, the implications of the factorization of $\Sigma$ in Eq. (4.8) for $d$-dimensional anisotropic convolution filters will be discussed.

**Proposition 2 (An optimal symmetric factorization of $\Sigma$)** *For discretized implementation of separated anisotropic Gaussian filters, the symmetric factorization of Cholesky type* (4.8) *is optimal in the number of required interpolations.*

**Proof 3** Since $\Sigma$ is a $d \times d$ symmetric matrix, it has $\frac{d(d+1)}{2}$ degrees of freedom. The matrices $V$ and $D$ in any symmetric factorization $\Sigma = VDV^t$ must have at least as many degrees of freedom. As $D$ is diagonal, $\frac{d(d-1)}{2}$ degrees of freedom remain for $V$. These are exactly the $v_{i,j}$, $i \neq j$, in the upper half of $V$ in (4.8). To fulfill the condition that $|V| = 1$, the diagonal entries of $V$ are set to one, eliminating the need for interpolation in exactly one dimension per filter direction. All remaining entries of $V$ are zero. No further elements of $V$ can be restricted to be either zero or integral, as this would contradict either the fact that $V$ and $D$ together must account for the $\frac{d(d+1)}{2}$ degrees of freedom of $\Sigma$, or that $|V| = 1$. ∎

Other separations reaching this dimensionality bound are possible. However, the special form of $V$ in (4.8) has a very intuitive geometric interpretation, see below, which furthermore leads to a very efficient implementation, later in this chapter. Summarizing, the symmetric factorization of Cholesky type results in a separation of anisotropic Gaussian filters that, when discretized, is optimal in the number of required interpolations. In $\mathbb{R}^d$, the first filter operation is aligned to the $x_1$ axis and does not require any interpolation at all. The second operation filters along a line and requires 1D-interpolation. For each further filter, the dimensionality of the required interpolation grows by 1 until reaching $d - 1$-dimensional interpolation required for implementing the $d$'th discretized convolution operation. Each of these $d$ convolution steps requires fewer interpolations than any one in the eigen decomposition-based separation scheme.




(a) (b) (c)

Figure 4.1: (a) Any ellipse in the plane can be transformed into an axis-aligned ellipse by using either a (b) rotation or (c) shear. In terms of symmetric factorizations of the Gaussian function's covariance matrix $\Sigma$, these two geometric transformations correspond to an eigen decomposition and a triangular decomposition of Cholesky type thereof, respectively. (This figure first appeared in [97], ©IEEE)

### 4.3.1 Geometric Interpretation

Proposition 2 specifies $d$ filtering directions. To understand the implications of this result, it is useful to look at it from a geometric perspective. One geometric interpretation of separations of the anisotropic Gaussian convolution filter is the following. A linear transformation ($V^{-1}$) is a applied to the signal, followed by $d$ axis-parallel orthogonal filters and back-transformation of the result ($V$). Conversely, the directions of convolution in (4.6) are the $V$-transformations of the Cartesian coordinate axes. In 2D, this process can be visualized by ellipses, which correspond to level sets of Gaussians.

The matrix $V$ obtained by an eigen decomposition of $\Sigma$ is a rotation matrix that specifies the directions of the Gaussian's major axes. Therefore, implementing separated anisotropic Gaussian convolution using an eigen decomposition $\Sigma = VDV^t$ has the geometric analog of rotating an ellipse onto the coordinate axes (Fig. 4.1(b)).

In case of the symmetric factorization of Cholesky type, the matrix $V$ is a shear matrix. In 2D, the process of filtering along the directions resulting from that matrix therefore corresponds to a shear of the $x_2$ axis, while the $x_1$ axis remains fixed (Fig. 4.1(c)). In $\mathbb{R}^d$, the first direction of convolution is the $x_1$ axis (the first column of $V$ will be $(1, 0, \ldots, 0)^t$). The $i$'th convolution direction $\mathbf{v}^i$ is a sheared version of $i-1$ coordinate axes. $\mathbf{v}^i$ intersects the ellipsoid at the point where it extends farthest in the direction of the respective coordinate axis $x_i$.

## 4.4 Separable anisotropic Gaussian filters in image processing

In order to make the theoretical results of this chapter practically applicable in image processing, this section derives explicit formulas and useful parameterizations of the factorization in Eq. (4.8) for the practically most relevant cases $d = 2, 3$.

So far, it has implicitly been assumed that the triangular factorization of Cholesky type existed. This section explicitly computes the entries of $V$ and $D$. This serves two purposes. First, it proves the existence of the required factorization $\Sigma = VDV^t$ in $\mathbb{R}^2$ and $\mathbb{R}^3$. Second, the resulting formulas are useful in practice, for they allow to compute the filtering directions and the 1D-Gaussians' variances directly from a given 2D or 3D-covariance matrix $\Sigma$. For higher dimensions, this derivation could be performed in the same fashion, but that is not done in the present chapter.

Even though a Gaussian's covariance matrix encodes all information on the filter's orientation and shape, these are entangled across its entries. Therefore this section also proposes parameterizations of anisotropic Gaussian filters in terms of the direction of the filter's major axis in polar coordinates.

### 4.4.1 Triangular factorization of Cholesky type in $\mathbb{R}^2$ and $\mathbb{R}^3$

The required entries of $V$ and $D$ can be computed using elementary calculus. The basic mechanism is to compute the matrix product $VDV^t$ and to equate it with the known entries of the given matrix $\Sigma$. This leads to $\frac{d(d+1)}{2}$ equations that need to be solved for the entries of $V$ and $D$. Denote by $v_{i,j}$, $1 \leq i < j \leq d$, the $\frac{d(d-1)}{2}$ entries in the upper half of $V$, and by $d_i^2$ the positive diagonal terms of $D$, $1 \leq i \leq d$. The elements of $\Sigma$ will be denoted by $s_{i,j}$, $1 \leq i, j \leq d$.

#### Explicit formulas in $\mathbb{R}^2$

Multiplying out the two required matrices results in

$$VDV^t = \begin{pmatrix} 1 & v_{1,2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} d_1^2 & 0 \\ 0 & d_2^2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ v_{1,2} & 1 \end{pmatrix} = \begin{pmatrix} d_1^2 + d_2^2 v_{1,2}^2 & d_2^2 v_{1,2} \\ d_2^2 v_{1,2} & d_2^2 \end{pmatrix}. \tag{4.9}$$

Equating this result to the known covariance matrix $\Sigma$ leads to a system of four equations,

$$\begin{pmatrix} d_1^2 + d_2^2 v_{1,2}^2 & d_2^2 v_{1,2} \\ d_2^2 v_{1,2} & d_2^2 \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} s_{1,1} & s_{1,2} \\ s_{1,2} & s_{2,2} \end{pmatrix}, \tag{4.10}$$

which can be solved for for the 3 unknowns,

$$v_{1,2} = \frac{s_{1,2}}{s_{2,2}}, \quad d_1^2 = s_{1,1} - \frac{s_{1,2}^2}{s_{2,2}}, \text{ and } \quad d_2^2 = s_{2,2}. \tag{4.11}$$

$\Sigma$ is positive definite, such that $s_{2,2} > 0$ and $|\Sigma| = s_{1,1}s_{2,2} - s_{1,2}^2 > 0$. Hence, all expressions are well defined. Note that $d_1^2 d_2^2 = s_{1,1}s_{2,2} - s_{1,2}^2$, which is the explicit formulation in $\mathbb{R}^2$ of the fact that $|D| = |\Sigma|$. It follows that $|V| = 1$, as required by Corollary 1, proving the existence of the triangular factorization of Cholesky type of $\Sigma$ in $\mathbb{R}^2$.

#### Explicit formulas in $\mathbb{R}^3$

For the case of Gaussian filters in $\mathbb{R}^3$, the steps for deriving the six unknowns in the desired factorization of $\Sigma$ are almost identical to those in the case of $\mathbb{R}^2$, above. The formulas do get more

complicated. Therefore, this section simply states the results, and the details of the derivation are deferred to App. D.

Given a $3 \times 3$ covariance matrix, the unknowns in the factorization

$$VDV^t = \begin{pmatrix} 1 & v_{1,2} & v_{1,3} \\ 0 & 1 & v_{2,3} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} d_1^2 & 0 & 0 \\ 0 & d_2^2 & 0 \\ 0 & 0 & d_3^2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ v_{1,2} & 1 & 0 \\ v_{1,3} & v_{2,3} & 1 \end{pmatrix} \tag{4.12}$$

are equated with the corresponding elements of $\Sigma$. Again denoting these by $s_{i,j}$, the diagonal elements of $D$ result as

$$d_1^2 = s_{1,1} - \frac{(s_{1,2}s_{3,3} - s_{1,3}s_{2,3})^2}{s_{3,3}(s_{2,2}s_{3,3} - s_{2,3}^2)} - \frac{s_{1,3}^2}{s_{3,3}}, \qquad d_2^2 = s_{2,2} - \frac{s_{2,3}^2}{s_{3,3}}, \qquad d_3^2 = s_{3,3}, \tag{4.13}$$

and the elements in the upper half of $V$ are

$$v_{1,2} = \frac{s_{1,2}s_{3,3} - s_{1,3}s_{2,3}}{s_{2,2}s_{3,3} - s_{2,3}^2}, \qquad v_{1,3} = \frac{s_{1,3}}{s_{3,3}}, \qquad v_{2,3} = \frac{s_{2,3}}{s_{3,3}}. \tag{4.14}$$

From $\Sigma$ being positive definite it follows that all terms above are well defined, which is also true for the intermediate steps leading to these results, see App. D for details. By definition, the diagonal elements of $V$ are 1. It follows that $|V| = 1$ as demanded by Corollary 1, and these results therefore prove the existence of the symmetric factorization of Cholesky type of $\Sigma$ in $\mathbb{R}^3$.

### 4.4.2 Independent parameters for orientation and shape in $\mathbb{R}^2$ and $\mathbb{R}^3$

In practice, $\Sigma$ is often not given explicitly, but rather implicitly by the directions in which the Gaussian filtering should take place and by the desired kernel size in that and remaining orthogonal directions. By splitting $\Sigma$ into a rotation matrix $R$ and a diagonal matrix $S$ of variance values, these matrices independently encode filter direction and spread, respectively. This is equivalent to an eigen decomposition of $\Sigma$, cf. Sec. 4.3.1. An intuitive and compact representation of the filter's orientation is to express the filter direction in polar coordinates.

#### Parameterization in $\mathbb{R}^2$

An anisotropic Gaussian in $\mathbb{R}^2$ is uniquely determined by an angle $\alpha$ and two variances $\sigma_1^2$ and $\sigma_2^2$. $\alpha$ specifies the filtering direction corresponding to the variance $\sigma_1^2$, and $\sigma_2^2$ describes the width of the kernel in the remaining orthogonal direction. $\Sigma$'s eigen decomposition results in two matrices, one orthonormal and the other diagonal. The orthonormal one of these specifies the resulting filter's orthogonal major axes. To parameterize these directions by a polar coordinate $\alpha$, the same mechanism can be utilized to compose the required matrix. The 2D-covariance matrix is $\Sigma = R_\alpha S R_\alpha^t$ with

$$R_\alpha = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \text{ and } S = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix}. \tag{4.15}$$

Multiplication leads to a general parameterization,

$$\Sigma = \begin{pmatrix} \sigma_1^2 \cos^2\alpha + \sigma_2^2 \sin^2\alpha & (\sigma_2^2 - \sigma_1^2)\cos\alpha\sin\alpha \\ (\sigma_2^2 - \sigma_1^2)\cos\alpha\sin\alpha & \sigma_1^2 \sin^2\alpha + \sigma_2^2 \cos^2\alpha \end{pmatrix}. \tag{4.16}$$

Plugging these terms into (4.11), the symmetric factorization of Cholesky type in terms of polar coordinate $\alpha$ and variances $\sigma_1^2$ and $\sigma_2^2$ is given by

$$V = \begin{pmatrix} 1 & \frac{(\sigma_2^2-\sigma_1^2)\cos\alpha\sin\alpha}{\sigma_1^2\sin^2\alpha+\sigma_2^2\cos^2\alpha} \\ 0 & 1 \end{pmatrix}, \tag{4.17}$$

$$D = \begin{pmatrix} \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2\cos^2\alpha+\sigma_2^2\sin^2\alpha} & 0 \\ 0 & \sigma_1^2\cos^2\alpha + \sigma_2^2\sin^2\alpha \end{pmatrix}. \tag{4.18}$$

**Remark 1 (Generalization of the result by Geusebroek et al.)** This result demonstrates that the separation of 2D-anisotropic Gaussian filters proposed by Geusebroek et al. [61] is a special case of the results of the present chapter. To see this, let $\varphi$ denote the angle between $V$'s column vectors, i.e. $\varphi = \measuredangle(\mathbf{v}^1, \mathbf{v}^2)$,

$$\tan\varphi = \frac{\sigma_1^2\sin^2\alpha + \sigma_2^2\cos^2\alpha}{(\sigma_2^2 - \sigma_1^2)\cos\alpha\sin\alpha}. \tag{4.19}$$

This is exactly the direction of the second convolution operation reported in [61], with corresponding variance given by the second entry on the diagonal of $D$ in (4.18). □

### Parameterization in $\mathbb{R}^3$

To obtain a useful parameterization of the separated anisotropic Gaussian convolution filter in $\mathbb{R}^3$, the same mechanism as in the section above for $\mathbb{R}^2$ is used. That is, a triangular factorization of Cholesky type $VDV^t$ of the 3D-covariance matrix parameterized in spherical polar coordinates is derived.

In 3D, these formulas get more complicated than in 2D. To obtain results in a tolerable form, the parameterization will be derived not for the general 3D-Gaussian filter, but only for the case that is most relevant in image processing: The material in this section is limited to rotationally symmetric Gaussian filters. These can have either a prolate shape which is useful for filtering line-like image structures, or an oblate shape matching flat surfaces in 3D-images.

The polar coordinate of a point $\mathbf{x} \in \mathbb{R}^3$ is given by two angles $\theta$ and $\varphi$, which are called the colatitude and longitude, respectively [48]. The colatitude is the angle away from the $x_3$-axis, and the longitude is a rotation in the $x_1x_2$-plane, about the $x_3$-axis. To obtain the triangular factorization of $\Sigma$, the system of equations

$$R(\theta,\varphi)\begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_1^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix} R(\theta,\varphi)^t = \begin{pmatrix} 1 & v_{1,2} & v_{1,3} \\ 0 & 1 & v_{2,3} \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} d_1^2 & 0 & 0 \\ 0 & d_2^2 & 0 \\ 0 & 0 & d_3^2 \end{pmatrix}\begin{pmatrix} 1 & v_{1,2} & v_{1,3} \\ 0 & 1 & v_{2,3} \\ 0 & 0 & 1 \end{pmatrix}^t \tag{4.20}$$

needs to be solved for the unknowns $v_{1,2}$, $v_{1,3}$, $v_{2,3}$, $d_1^2$, $d_2^2$ and $d_3^2$. $\sigma_1^2$ and $\sigma_3^2$ define the variances of the resulting filter along its major axes, and $R(\theta, \varphi)$ denotes an appropriate rotation matrix. Note that in (4.20), the parameter $\sigma_3^2$ specifies the length of the axis of rotational symmetry of the resulting filter. Accordingly, an unrotated prolate filter ($\sigma_1^2 < \sigma_3^2$) is centered on the $x_3$-axis, and an unrotated oblate filter ($\sigma_1^2 > \sigma_3^2$) lies parallel to the $x_1x_2$-plane. Refer to App. D for details, where also the steps leading to the following set of solutions are given.

$$d_1^2 = \frac{\sigma_1^2\sigma_1^2\sigma_3^2}{d_2^2 d_3^2} \tag{4.21}$$

$$d_2^2 = \frac{\sigma_1^2(\sigma_3^2 \sin^2\varphi + \cos^2\varphi(\sigma_3^2\cos^2\theta + \sigma_1^2\sin^2\theta))}{\sigma_1^2\sin^2\theta + \sigma_3^2\cos^2\theta} \tag{4.22}$$

$$d_3^2 = \sigma_1^2\sin^2\theta + \sigma_3^2\cos^2\theta \tag{4.23}$$

$$v_{1,2} = \frac{2(\sigma_3^2 - \sigma_1^2)\sin 2\varphi \sin^2\theta}{\sigma_1^2 + 3\sigma_3^2 + (\sigma_1^2 - \sigma_3^2)(\cos 2\varphi - 2\cos^2\varphi\cos 2\theta)} \tag{4.24}$$

$$v_{1,3} = \frac{(\sigma_3^2 - \sigma_1^2)\cos\varphi\cos\theta\sin\theta}{\sigma_1^2\sin^2\theta + \sigma_3^2\cos^2\theta} \tag{4.25}$$

$$v_{2,3} = \frac{(\sigma_3^2 - \sigma_1^2)\cos\theta\sin\varphi\sin\theta}{\sigma_1^2\sin^2\theta + \sigma_3^2\cos^2\theta} \tag{4.26}$$

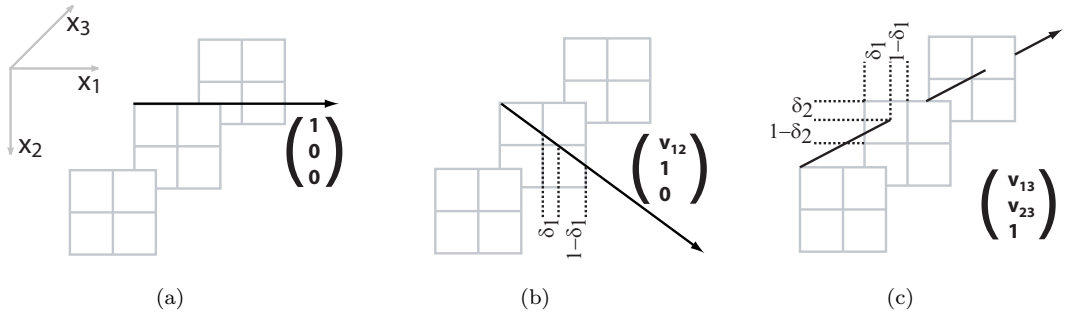
Depending on the choice of $\sigma_1^2$ and $\sigma_3^2$, these results fully specify the separated convolution integral (4.6) of a prolate, isotropic or oblate 3D-anisotropic Gaussian convolution filter. When the filter is prolate, it has exactly the form required for detecting local orientations in 3D-images of fibers in Ch. 3. The filter direction is $\mathbf{v} = (\cos\varphi\sin\theta, \sin\varphi\sin\theta, \cos\theta)^t$, the Cartesian equivalent of the polar coordinate $(\theta, \varphi)$.

## 4.5 Discrete implementations of the separated filter

The previous section derived explicit formulas for the optimal non-orthogonal separation of the Gaussian convolution integral. These can be used for 2D and 3D-image filtering. Implementations of this separated filter require discretized signals and filters, which will be investigated next. Discrete functions will be indicated by capital letters instead of small letters in the continuous case. E.g. the discrete Gaussian will be denoted by $G(\mathbf{x})$ instead of $g(\mathbf{x})$.

When implemented, the optimal triangular factorization of $\Sigma$ given by Proposition 2 results in a sequence of discrete 1D-convolutions requiring interpolation of increasing dimension. Similar to the continuous case, this can be written compactly using a discrete directional convolution operator defined as

$$(G_i *_{\mathbf{v}} F)(\mathbf{x}) = \sum_{k\in\mathbb{Z}} G_i(k)F(\mathbf{x} - k\mathbf{v}). \tag{4.27}$$




(a) (b) (c)

Figure 4.2: In 3D, implementing the proposed separation results in three directional convolution operations, requiring (a) no interpolation, (b) 1D, or (c) 2D interpolation. (This figure first appeared in [97], ©IEEE)

Here, the coefficients $G_i$ are derived from the 1D-Gaussian by sampling at the grid points. In the discrete separated Gaussian filter operation,

$$G * F = G_1 *_{\mathbf{v}^1} G_2 *_{\mathbf{v}^2} \dots G_n *_{\mathbf{v}^n} F, \tag{4.28}$$

the directions $\mathbf{v}^i$ will generally not contain only integral entries. There are a few exception, e.g. when the filter $G$ is aligned to the Cartesian coordinate grid. Therefore, $\mathbf{x} - k\mathbf{v}$ does generally not lie on the sampling grid. To obtain a value for $F$ at this position requires interpolation from neighboring sample points, see e.g. [135, Ch. 9]. To better understand this process, consider the case of $\mathbb{R}^3$. The general case of $\mathbb{R}^d$ follows the same scheme. A separated 3D-discrete convolution,

$$G * F = G_3 *_{\mathbf{v}^3} G_2 *_{\mathbf{v}^2} G_1 *_{\mathbf{v}^1} F, \tag{4.29}$$

is a sequence of three 1D-convolutions. Each step in this sequence will now be examined for required interpolations. Let $X = (X_1, X_2, X_3) \in \mathbb{Z}^3$ denote a position on the sampling grid of the image $F$. The first convolution operation,

$$\begin{aligned} F_1(X) := (G_1 *_{\mathbf{v}^1} F)(X) &= \sum_{k \in \mathbb{Z}} G_1(k) F(X - k\,(1,0,0)^t) \\ &= \sum_{k \in \mathbb{Z}} G_1(k) F(X_1 - k, X_2, X_3), \end{aligned} \tag{4.30}$$

along the direction given by the first column $\mathbf{v}^1$ of $V$, computed using the triangular factorization of Cholesky type, does not require interpolation (Fig. 4.2(a)). It convolves along lines parallel to the $x_1$-axis. If the data can be organized in memory such that these lines form compact blocks, computation of $F_1$ is both fast and accurate.

Except for axis-aligned Gaussians, the two remaining filter operations will require interpolation.

The second direction $\mathbf{v}^2 = (v_{1,2}, 1, 0)^t$, leads to the following convolution operation.

$$\begin{aligned} F_2(X) := (G_2 *_{\mathbf{v}^2} F_1)(X) &= \sum_{k\in\mathbb{Z}} G_2(k) F_1(X - k\,(v_{1,2}, 1, 0)^t) \\ &= \sum_{k\in\mathbb{Z}} G_2(k) F_1(X_1 - k v_{1,2}, X_2 - k, X_3). \end{aligned} \tag{4.31}$$

Among the variables in this function, only $v_{1,2}$ is not an integer. Consequently, this second convolution operation needs to access the discrete convolution result $F_1$ from Eq. (4.30) at non-grid positions along the first dimension. This necessitates 1D-interpolation in the direction parallel to the $x_1$-axis (Fig. 4.2(b)).

The third and last 1D-convolution step in the complete convolution operation in $\mathbb{R}^3$,

$$\begin{aligned} F_3(X) := (G_3 *_{\mathbf{v}^3} F_2)(X) &= \sum_{k\in\mathbb{Z}} G_3(k) F_2(X - k\,(v_{1,3}, v_{2,3}, 1)^t) \\ &= \sum_{k\in\mathbb{Z}} G_3(k) F_2(X_1 - k v_{1,3}, X_2 - k v_{2,3}, X_3 - k). \end{aligned} \tag{4.32}$$

operates along the third column $\mathbf{v}^3$ of the triangular matrix $V$. Here, the coefficients $v_{1,3}$ and $v_{2,3}$ will generally not be integers. Therefore, non-grid elements of the discrete convolution result $F_2$ from Eq. (4.31) need to be accessed. These accesses lie on the grid positions along the $x_3$-axis, but require two-dimensional interpolation in the $x_1x_2$-plane (Fig. 4.2(c)).

Summarizing, this closer analysis of the separated Gaussian filter procedure in $\mathbb{R}^3$ demonstrates in detail the properties of the proposed separation scheme in terms of interpolation that have already been mentioned in Sec. 4.3: General $d$-dimensional Gaussian convolution can be implemented as a sequence of $d$ one-dimensional Gaussian convolutions, where the $i$'th convolution step requires at most $i-1$-dimensional interpolation. The interpolations themselves can be implemented using nearest neighbor, linear, or higher order schemes, see e.g. [108, Ch. 8], depending on the accuracy that is required.

### 4.5.1 Finite and infinite impulse response filtering

Different variants for implementing the discrete directional convolution operator in (4.27) exist. The simplest approach is finite impulse response (FIR) filtering,

$$(G_i *_{\mathbf{v}} F)(X) = \sum_{k=-K}^{K} G_i(k) F(X - k\mathbf{v}), \tag{4.33}$$

where the sum is truncated to extend at most $K$ steps from $X$. $K$ is chosen such that only terms remain in the sum where $G_i(k)$ differs significantly from 0, usually a constant multiple of the standard deviation of the Gaussian.

Due to this inherent dependency between the filter's spread and the number of terms in (4.33), such implementations have the drawback of runtime being dependent on filter size. One well-known remedy is point-wise multiplication of filter and signal in the Fourier domain, as prescribed by the convolution theorem. However, this approach is not well-suited for the problem at hand,

as it would be more efficient to compute a $d$-dimensional Fourier transformation and to perform the convolution in one step instead of applying the proposed separation. This alternative implementation variant of Gaussian filtering will in fact be considered in the evaluations in Sec. 4.6.

A second technique that avoids this dependence on filter size is the infinite impulse response (IIR) implementation of Gaussian filters proposed by Young and van Vliet [177]. They recursively implemented a rational approximation to the 1D-Gaussian filter. This implementation consists of a pair of filters, called the forward and backward filters. It is straight-forward to extend these to directional filtering in direction $\mathbf{v}$, resulting in the following filter pair.

$$w(\mathbf{x}) = B \cdot F(\mathbf{x}) + \frac{1}{b_0} \sum_{k=1}^{3} b_k \cdot w(\mathbf{x} - k\mathbf{v}) \tag{4.34}$$

$$(G *_{\mathbf{v}} F)(\mathbf{x}) = B \cdot w(\mathbf{x}) + \frac{1}{b_0} \sum_{k=1}^{3} b_k \cdot (G *_{\mathbf{v}} F)(\mathbf{x} + k\mathbf{v}) \tag{4.35}$$

See [177, 178] for details on the filter coefficients $b_0, b_1, b_2, b_3$ and $B$ and [162] for boundary conditions which need to be applied to avoid certain errors. As $\sigma$, the standard deviation of the 1D-Gaussian filter $G$, enters only through these coefficients, the number of operations required for evaluating (4.34) and (4.35) is independent of the filter size. Note that since the number of operations in both filter steps is constant for each $\mathbf{x}$, this filter implementation satisfies the requirement of $\mathcal{O}(1)$ complexity per pixel from Sec. 4.1.1. Thus, when combined with the non-orthogonal separation scheme proposed in this chapter, it results in a $\mathcal{O}(n^d)$-implementation of the $d$-dimensional anisotropic Gaussian convolution filter.

### 4.5.2 Implementations

With different implementations of the one-dimensional convolution operator (4.27) at hand, the question remains how these should be applied. Three implementation variants of the proposed separated anisotropic Gaussian filters will be considered.

- *Naive implementation* – In this simple implementation, Eq. (4.33) is sequentially applied in $d$ directions given by the columns of $V$, at every pixel grid location $X$.

- *Line-buffer implementation* – To apply the IIR filter requires a sequence of data. Therefore, the line-buffer implementation of the proposed method extracts pixel values along parallel lines through an image, with the lines' directions given by the columns of $V$. Each 1D-filter operation results in directionally filtered values along lines that are generally not axis-aligned. This implementation therefore requires interpolation not only when extracting the line-buffers prior to IIR filtering, but also when writing the filtered values back to their respective image locations.

- *Geometric implementation* – The third variant for implementing the proposed filter follows not from the notion of the directional convolution operator, but rather from the geometric interpretation of the triangular factorization of Cholesky type, cf. Sec. 4.3.1. That is, the image is first sheared using the transformation matrix $V^{-1}$. Then, $d$ orthogonal axis-aligned

1D-Gaussian filters are applied, which do not require any interpolation. The filter operations can again be performed using the efficient IIR implementation. Finally, the filter results are transformed back to the original image coordinates using $V$. This implementation variant separates the filtering from the interpolation, as interpolation is only required when performing the shear. In $\mathbb{R}^3$, the shear operation itself can again be split into two steps, first a shear in the $x_1$, and then in the $x_2$-direction. This also implies that only one-dimensional interpolation needs to be performed. The general case of $\mathbb{R}^d$ can be treated in the same fashion.

These three alternative implementations all perform separated anisotropic Gaussian filtering. One task of the following section will be to determine which one of them is the most efficient when applied to 2D and 3D-images.

## 4.6 Results

This section presents numerical results on the speed and accuracy of the proposed separated anisotropic Gaussian filter in 2D and 3D. The three alternative implementations described in Sec. 4.5.2 will be compared in terms of their runtime. The most efficient one will then be compared against a state-of-the-art implementation based on the fast Fourier transformation (FFT). As not only speed but also accuracy are relevant, the impulse response function of the separated filter implementation will be compared to that of the true Gaussian function.

The three proposed implementations were realized in C. For the recursive IIR 1D-Gaussian filter, which two out of the three implementations use as a subroutine, cf. Sec. 4.5.2, a publicly available implementation by J.M. Geusebroek was used[1].

Furthermore, to demonstrate the impact of interpolation, two interpolation schemes were tested, linear and nearest neighbor (NN) interpolation. For the directional convolution operator in 3D, which requires 2D-interpolation in the third filter direction, a bilinear interpolation with respect to the four surrounding pixel grid position was applied.

For the control experiment comparing the efficiency of the proposed method against the FFT, the FFTW library (version 3) in single precision mode was used [54]. Single precision with real input is the fastest mode of operation of that library. The FFTW, which includes highly optimized SIMD (single instruction multiple data) assembler code, is one of the fastest implementations available. Benchmark results comparing the performance of FFTW against other implementations on different hardware are regularly published on the internet[2].

### 4.6.1 Speed

Isotropic Gaussian filters will be considered as a baseline method in the following. These are widely used, efficiently implementable, and present an upper bound to the performance that is achievable for general Gaussian filters. This reference method was implemented by applying $d$ IIR

[1] `http://www.science.uva.nl/~mark/`
[2] `http://www.fftw.org`

(a) Mean 2D-performance relative to isotropic Gaussian filtering.

| Method | Interpolation | Relative runtime | |
|---|---|---|---|
| | | float data | 8 bit data |
| isotropic | | 1.00 | |
| FFT | | $7.69 \pm 6.92$ | |
| geometric | NN | $1.98 \pm 0.55$ | $1.62 \pm 0.29$ |
| geometric | linear | $2.01 \pm 0.55$ | $1.70 \pm 0.29$ |
| line buffer | NN | $2.09 \pm 0.06$ | $2.01 \pm 0.35$ |
| line buffer | linear | $3.69 \pm 0.15$ | $2.89 \pm 0.60$ |
| Geusebroek [60] | linear | $1.26 \pm 0.07$ | $1.42 \pm 0.07$ |
| naive | | depends on $\sigma$ | *(see text)* |

(b) Mean 3D-performance relative to isotropic Gaussian filtering.

| Method | Interpolation | Relative runtime | |
|---|---|---|---|
| | | float data | 8 bit data |
| isotropic | | 1.00 | |
| FFT | | $4.30 \pm 3.99$ | |
| geometric | NN | $1.21 \pm 0.05$ | $1.01 \pm 0.07$ |
| geometric | bilinear | $1.22 \pm 0.05$ | $1.10 \pm 0.05$ |
| line buffer | NN | $2.11 \pm 0.22$ | $1.46 \pm 0.16$ |
| line buffer | bilinear | $3.73 \pm 0.44$ | $2.24 \pm 0.27$ |
| naive | | depends on $\sigma$ | *(see text)* |

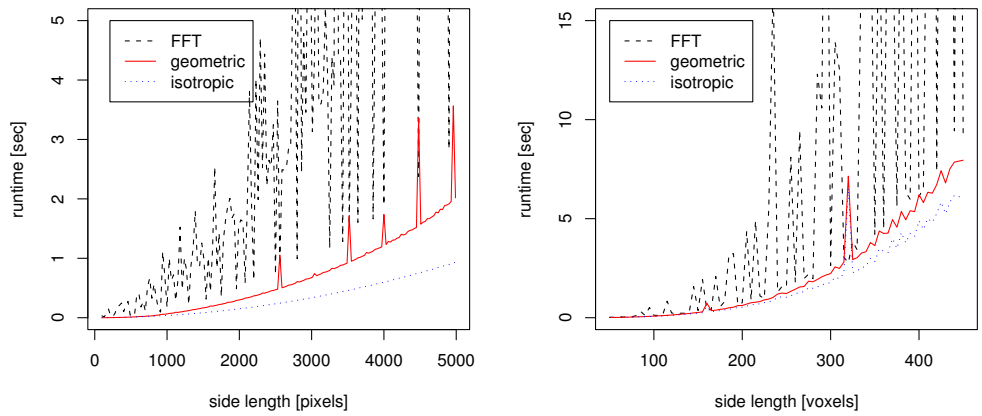
Table 4.1: Average performance of different implementations of the anisotropic Gaussian filter, measured in relative units. For each method, the ratio between its runtime and the runtime of isotropic Gaussian filtering was measured. The tables show the means and standard deviations of this ratio across different image sizes.

1D-Gaussian filters parallel to the coordinate axes, with runtime being therefore independent of the variance parameter $\sigma^2$.

In a first evaluation step, the three different separated implementations and the alternative FFT filter were applied to 2D and 3D-images with side lengths $N = 100, 130, \ldots, 4990$ and $N = 50, 55, \ldots, 450$, respectively. The goal of this first step is to identify the most efficient separated implementation scheme, while the discussion of the implementation's accuracy will be deferred to the following section. The runtime for each method was measured in 30 trials for each of these image sizes. The final performance measure is the mean runtime of these trials after excluding the top and bottom 10% as outliers.

The FFT filter timings presented here include a forward transform of the image, point-wise multiplication with the Fourier-transformed filter kernel, and an inverse transformation of the multiplication result. Note that a forward transformation step of the filter kernel is not included. This step can be omitted as it can be precomputed, either analytically or numerically.

Tab. 4.1 lists the mean results of this experiment across all tested image sizes. The runtime of the naive implementation, which applies $d$ directed FIR Gaussian filters at each image location, depends on the size of the filter kernel. It can therefore not be listed in this table, in which results are generally independent of that parameter. Furthermore, it was apparent early in the experiments that its performance was not competitive with any other implementation, and was therefore dropped.




(a) 2D (b) 3D

Figure 4.3: Performance of different methods: Geometric implementation vs. FFT-based convolution vs. isotropic filtering (float data). Runtimes were measured on a 2.2 GHz Athlon64 3200+ CPU using the Intel ICC compiler (version 9) under a 32 bit GNU/Linux operating system. (This figure first appeared in [97], ©IEEE)

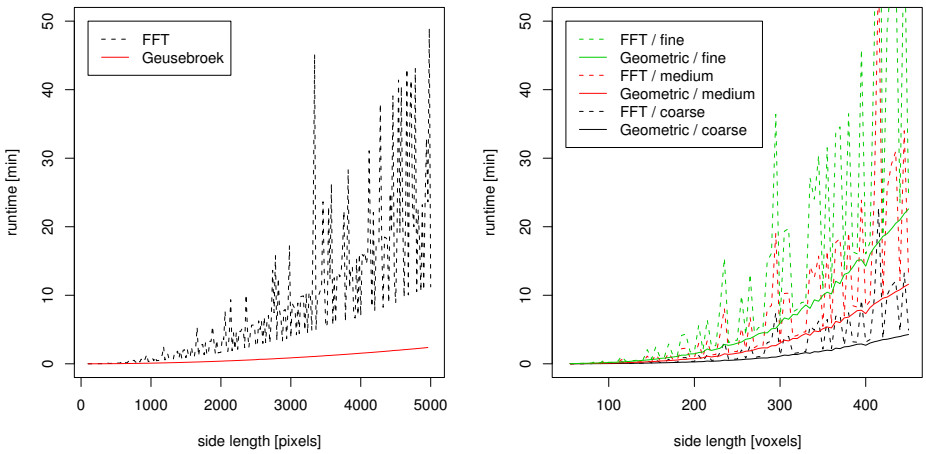
Clearly, all implementations based on the non-orthogonal separation and IIR-filter subroutine outperform the FFT implementation. Among the implementations proposed in this chapter, the geometric implementation is the fastest, both in 2D and 3D. In 2D, however, the implementation provided by J.M. Geusebroek turns out to be the best-performing one. In this implementation, interpolation and filtering are interwoven, explaining its low runtime. Geusebroek's implementation, on the other, is not able to handle integral data, which therefore had to be converted on-the-fly in order to obtain the results on 8-bit integers in Tab. 4.1. This additional data conversion step explains the lower performance of Geusebroek's implementation for integer data compared to the float case.

Regarding the differences between linear and nearest neighbor interpolation, the additional cost of linear interpolation is small, especially when using the geometric implementation.

Having identified the geometric implementation as the most efficient one among the variants described in this chapter, its performance relative to the FFT and isotropic filtering across the test image sizes is analyzed (Fig. 4.3).

The most obvious feature of the absolute runtimes in Fig. 4.3 is the smoothness of the curve corresponding to the geometric implementation compared to the FFT. For the proposed method, some image sizes with lower performance are visible ($N = 2560, 3520, 4000, 4480, 4960$ for 2D, $N = 160, 320$ for 3D). The reason for this behavior is not entirely understood, as the program follows a deterministic code path that does not differentiate in any fashion between different input data sizes. Therefore, these effects can most likely be explained by cache prefetch operations, which may fail for some disadvantageous data block sizes. These optimizations are built into the program code by modern compilers [58, 74].

So far, the performance of image filtering has been considered. One motivation for deriving a separable anisotropic Gaussian filter was its use for detecting local orientations as described in



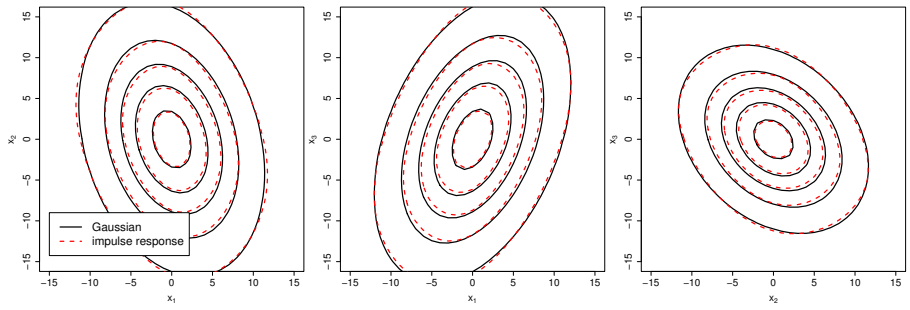

(a) 2D (b) 3D

Figure 4.4: Runtimes of computing the 2D and 3D-Gaussian orientation space, see Ch. 3. In 2D, the semi-circle was sampled in 1° steps, i.e., at 180 points. In 3D, the three curves shown for each method correspond to coarse, medium and fine sampling of the hemisphere (Sec. 3.3). These three resolution levels require 18, 50, and 98 filter operations, respectively. Runtimes were measured on a 2.33 GHz Intel Xeon 5148LV CPU using the GNU C++ compiler (version 4) under a 64 bit GNU/Linux operating system.

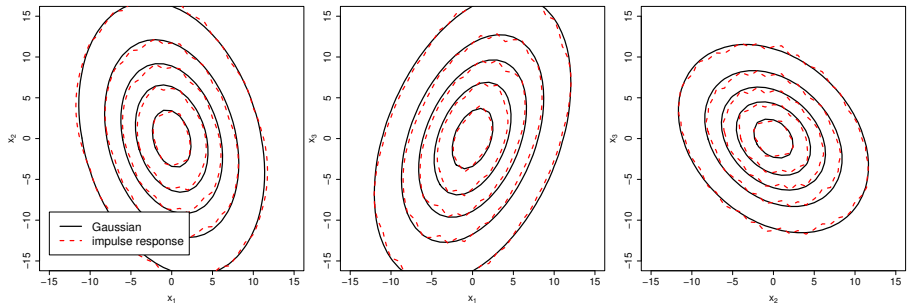
Ch. 3. Therefore, the time required for constructing the orientation space representation using either the FFT algorithm or the geometric implementation was evaluated (Fig. 4.4). Those results were measure on square or cubic images with side lengths $N = 100, 120, \ldots, 5000$ and $N = 50, 55, \ldots, 450$, respectively.

Using separable filters to construct the Gaussian orientation space is faster than using a FFT-based algorithm, both in 2D and 3D. Similar to the results for anisotropic image filtering in Fig. 4.3, the performance gain is larger in 2D than in 3D. Runtimes of FFT and separated anisotropic Gaussian filtering for constructing the Gaussian orientation space are similar at image sizes that are well suited for the FFT. For image sizes not suited for the FFT, the differences can be huge. As an example from the experiment described above, consider the computation of the finely resolved 3D-orientation space of a $300^3$ image, which required roughly 6 minutes using either implementation. When increasing the image side length to $N = 305$, the proposed geometric filter implementation used 6 minutes and 44 seconds CPU time for performing the 98 3D-filter operations and the overhead of bookkeeping necessary to obtain the best orientation in each pixel. The FFT-based implementation required 19 minutes and 20 seconds for the same operation on the same image, roughly three times the runtime of the separated implementation.

For image filtering, the FFT method requires one forward Fourier transform, multiplication of filter mask and image in Fourier space, and one inverse transformation. For computing the Gaussian orientation space, however, the image does not need to be transformed for each tested filter orientation. Rather, the Fourier transformation of the image is computed once. To obtain the filter response at any given orientation then only requires multiplication of filter and image




(a) Convolution using single precision floating point data and linear interpolation.




(b) Convolution using single precision floating point data and nearest neighbor interpolation.

Figure 4.5: Orthogonal slices through the impulse response of the geometric implementation in 3D ($\theta = 30°$, $\varphi = 60°$, $\sigma_1 = \sigma_2 = 5$ and $\sigma_3 = 10$). For comparison, the contour lines of the true Gaussians are depicted as well. (This figure first appeared in [97], ©IEEE)

in the Fourier domain, followed by one inverse Fourier transformation. This explains why the performance gains of the proposed over the FFT implementation are larger for image filtering (Fig. 4.3) than for computing the Gaussian orientation space (Fig. 4.4).

### 4.6.2 Accuracy

To judge the accuracy of separated anisotropic Gaussian filters, consider the impulse response of the geometric implementation in $\mathbb{R}^3$ in Fig. 4.5. The impulse response of the proposed implementation fits well to that of the true Gaussian function, regardless of which interpolation scheme is used. Furthermore, a comparison of Figures 4.5(a) and 4.5(b) shows that the differences due to the choice of interpolation scheme are local. This observation is crucial as interpolation errors may – in theory – be propagated through the image due to the sequential order of the $d$ IIR filters. In Sec. 4.6.1, it was shown that the additional cost of linear over nearest neighbor interpolation is small. Together with the local errors caused by nearest neighbor interpolation that are visible in Fig. 4.5(b), linear interpolation appears to be appropriate for implementing separated anisotropic Gaussian filters.

To systematically evaluate the influences of the filter parameters, the distance between the Gaussian function and the geometric implementation's impulse response was measured with respect to $\sigma_1$ and $\sigma_3$. To account for the filter orientation, the maximum across all orientations was taken. This maximum Euclidean error assumes e.g. a value of 0.0223 at $\sigma_1 = 1$ and $\sigma_3 = 3$ for

the geometric implementation with linear interpolation, but drops quickly, e.g. to a value of 0.001 at $\sigma_1 = 4$ and $\sigma_3 = 7$. Yet, this drop of the maximum Euclidean error measure is mainly caused by the lower filter values that larger filter kernels possess. Consequently, this error measure drops with the determinant of the covariance matrix, $|\Sigma|$, and is only of limited practical use. Furthermore, the accuracy of linear and nearest neighbor interpolation in terms of this maximal distance measure was observed to converge to the same value with increasing $|\Sigma|$. In summary, graphical comparison of the filter's impulse response to the Gaussian function as in Fig. 4.5 turned out to be more suitable for judging an implementation's accuracy than the Euclidean error metric.

## 4.7 Applications

The most important application of anisotropic Gaussian filters within this thesis is to construct the orientation space representation of images which was used in Ch. 3. To also demonstrate their potential as low-pass filters, this section presents some application examples for data smoothing using anisotropic Gaussian filters.

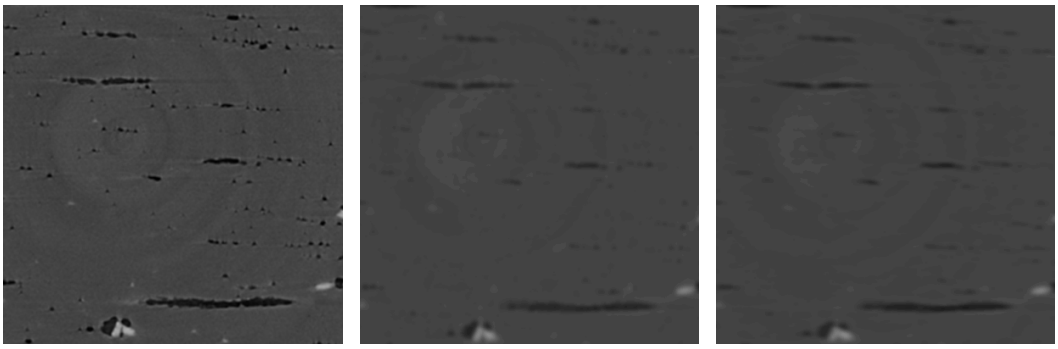
### 4.7.1 Anisotropic filtering of tomographic images

Metal foams, light-weight materials that find their applications e.g. as filters, are produced starting from a mixture of metal or alloy powder. This powder is mixed with a foaming agent and heated in a furnace. Pores start forming as this propellant releases gas. In an effort to better understand the forming process of such metal foams, current research in the field tries to better understand the mechanisms of this foaming process, see e.g. [13]. Through such findings, materials scientists hope to be able to make better predictions on the resulting foam structures.

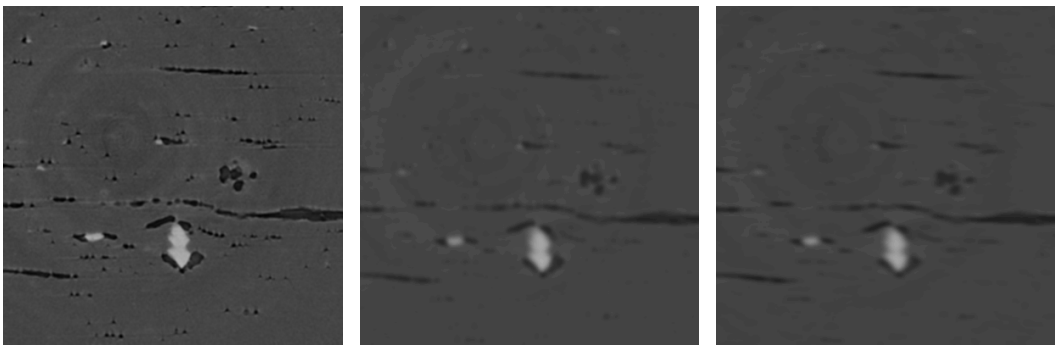
One metal foam that was recently investigated is produced from AW-6061 aluminium alloy with $TiH_2$ foaming agent [137]. There it was observed that the early forming pores had a strong spatial correlation with the original positions of $TiH_2$ particles. To analyze these correlations requires to segment the pores in $\mu$CT-images. These images with a spatial resolution of 0.7$\mu$m, show tubular pores that run in parallel through the volume data.

In this image, a vector $\mathbf{v}$ along which these tubular pores are approximately aligned was manually determined, cf. the caption of Fig. 4.6 for the corresponding polar coordinate. A prolate anisotropic filter was designed with $\sigma_1^2 = 4$ and $\sigma_3^2 = 16$, and a corresponding isotropic Gaussian filter with $\sigma^2 = 6.3$ was chosen such that $\sigma_1^2\sigma_1^2\sigma_3^2 \approx \sigma^6$. The corresponding shear parameters for the separated implementation then resulted from Equations (4.21) to (4.26). While both filter results show similar smoothing characteristics due to this choice of the variance parameters, the aligned anisotropic Gaussian filter preserves some details of the pore space, which is relevant in this application.

After zero-padding, the image was $760 \times 760 \times 760$ pixels large, requiring 55 seconds for anisotropic Gaussian filtering on a 2.4GHz Intel Xeon CPU.



(a) Original (slice #340) (b) Isotropically filtered (c) Anisotropically filtered



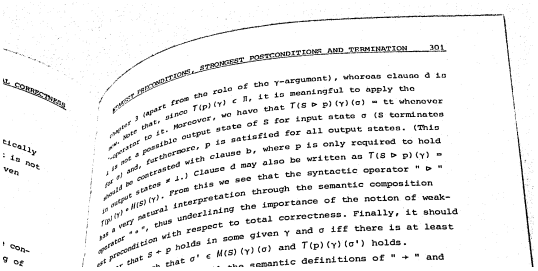
(d) Original (slice #490) (e) Isotropically filtered (f) Anisotropically filtered

Figure 4.6: Microtomographic reconstructions of an aluminium alloy (AW-6061) foam at an early foaming stage. The foaming agent ($TiH_2$) is visible in the form of bright spots. This propellant distributes through the specimen along parallel tubular pores. This 0.46mm$^3$ volume was filtered with isotropic and anisotropic Gaussian filters, with parameters set such that $|\Sigma|$ was equal in both bases. Due to the structure of parallel, tubular pores in this image, an anisotropic filter manually aligned to these pores smoothes the images while preserving more details of the pore space ($\varphi = 1.62$, $\theta = 1.50$, $\sigma_1^2 = 4$, $\sigma_3^2 = 16$, image source: ANKA Karlsruhe)

### 4.7.2 Adaptive smearing of text lines

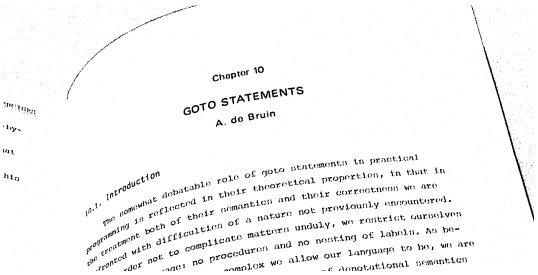
The following two application examples of anisotropic Gaussian filters demonstrate the applicability of this techniques in domains other than image processing of microstructures. In document digitization systems, which usually consist of a sequence of processing steps, overall system performance depends heavily on layout analysis, e.g. extraction of text lines [147]. This line extraction can be difficult especially for camera-captured documents, where text lines are often curled due to bent page surfaces (Fig. 4.7(a)). A recent contest at the International Workshop on Camera-Based Document Analysis and Recognition (CBDAR) objectively evaluated the performance of several algorithms for dewarping, i.e., straightening, of such images [148]. The data used here to demonstrate the applicability of anisotropic Gaussian filters to document images has been taken from the image database belonging to that contest (Fig. 4.7).

Images from the aforementioned CBDAR contest were filtered with a bank of 2D-anisotropic Gaussian filters ($\sigma_1^2 = 18$, $\sigma_2^2 = 6$) in 1° steps. At each pixel, the minimal filter response across



(a) Original image (`dsc00612`)

(b) Anisotropically filtered (`dsc00612`)




(c) Original image (`dsc00616`)

(d) Anisotropically filtered (`dsc00616`)

Figure 4.7: Camera-captured document images from the CBDAR 2007 document dewarping contest [148]. Due to the large inter-character distances and the curled text-lines, segmentation requires suitable preprocessing. The images were filtered with a bank of differently oriented anisotropic Gaussian filters. As these text lines are black on white background, taking the minimum across all filter orientations yields images in which adjacent letters have been smeared, but text lines remain separated. This property makes this method suitable for preprocessing in text line segmentation. (image source: CBDAR 2007 document dewarping contest)

these 180 orientations was recorded, and the corresponding result for two images is shown in Figures 4.7(b) and 4.7(d). For each of the $2112 \times 2816$ pixel images, only parts of which are shown here, the total processing time was about 36 seconds on a 2.4 GHz Intel Xeon CPU.

Due to the local adaptivity of this method and by choosing a filter size that allows to fill inter-character gaps, the result of this method is a filtered image in which the letters on individual text lines are merged. These results may be suitable for a subsequent text line segmentation step, e.g. by using a recently proposed modified snake model that is especially well suited for text lines in camera-captured document images [23].

### 4.7.3 Spatiotemporal smoothing of a video sequence

Low-pass filtering is a common preprocessing step for object detection in video. This can be done for each individual frame, e.g. prior to motion estimation and feature extraction. An alternative approach that better accounts for the correlation structure between subsequent video frames would be to use that information for video filtering.

This process is demonstrated for the “New Mobile and Calender” (mobcal) [67] video sequence (Fig. 4.8). It depicts a pan shot of a toy train moving from right to left in front of differently tex-


(a) Original frame (mobcal #240). (b) Anisotropically filtered.

Figure 4.8: Anisotropic filtering to selectively smoothen 2+1D video data. A strongly prolate anisotropic 3D-Gaussian is aligned with the motion of the front of the toy train. The resulting filter acts as a motion aware blurring filter, only weakly smoothing the first two cars of the train, but strongly blurring the background and other objects which show different movement patterns. (This figure first appeared in [97], ©IEEE, image source: TU München, electrical engineering department)

tured background. The motion vector of the car carrying the plastic tiger was manually extracted.

To this motion vector, a prolate anisotropic 3D-Gaussian filter was aligned and used to filter this 2+1D-video sequence around frame #240. The effect can be seen in Fig. 4.8(b): All parts of the image that have a different temporal correlation structure than this leftmost car are strongly blurred. The car carrying the plastic tiger, on the other hand, remains sharp. Here, the aligned anisotropic Gaussian filter acted mainly as a small low-pass filter, preserving most image features. Filtering the 474 MB clip (80 RGB frames in $1920 \times 1080$ resolution) took less than 1 minute on a 2.2 GHz AMD Athlon64 CPU.

## 4.8 Discussion

The main contribution of this chapter is an in-depth understanding of the mechanisms and properties of non-orthogonal factorizations of the Gaussian convolution kernel. It was shown that all symmetric factorizations of the covariance matrix $\Sigma$ of the form $\Sigma = VDV^t$ into square matrices $D$ and $V$, where $D$ is diagonal and $V$ has determinant 1, lead to a separated Gaussian convolution integral. Based on this result, a symmetric factorization of the covariance matrix, similar to a Cholesky decomposition, was identified as being optimal for discrete implementations. It is not unique in this sense. Other factorizations of the Gaussian kernel with the same or similar properties would be possible. The factorization used here has an intuitive geometric interpretation that leads to an implementation of Gaussian convolution achieved by image shearing and axis-aligned Gaussian filtering (called the "geometric" implementation). The result obtained here generalizes a previously known one concerning anisotropic Gaussian filters in $\mathbb{R}^2$ by Geusebroek et al. [61].

Factorizations of the Gaussian convolution kernel into 1D-Gaussians allow the use of fast recursive implementations of 1D-Gaussian convolutions. These implementations have $\mathcal{O}(1)$ complexity per filtered point. Combined with the proposed separation mechanism, this leads to implementations of the anisotropic $d$-dimensional convolution operation with linear complexity in the number

of pixels. Two such implementations of 2D and 3D-Gaussian filters were proposed, both of which were shown to be faster than state-of-the-art implementations based on the FFTW library. Among all tested implementations, the 2D-anisotropic Gaussian filter routine that is publicly available from J.M. Geusebroek [61], and the geometric 3D-anisotropic Gaussian filter method proposed in the present chapter are fastest for 2D and 3D-image filtering, respectively.

Another advantage of this novel non-orthogonal separation is its versatility compared to a standard implementation based on the Fourier transformation. E.g., it allows for efficient implementations of local image filters by applying a sequence of directional convolution operators, introduced in this chapter, and it even enables implementations of anisotropic Gaussian filters in fixed-point arithmetic. The feasibility of such fixed point implementations has been demonstrated in [96], where the filter described in the present chapter was implemented using integer arithmetic on an ETRAX 100LX 32 bit RISC processor. This approach should also be suitable for specialized hardware platforms such as FPGAs (field programmable gate arrays).

These efficient implementations of anisotropic Gaussian filtering are useful for speeding up computation of the Gaussian orientation space discussed in Ch. 3. Moreover, applications to images and video sequences in the present chapter demonstrate the potentials of anisotropic Gaussian filters for adaptive image smoothing.

# Chapter 5

# Discussion

From the task of determining the coordinates of cells in histological resections to the problem of computing the 3D-fiber orientation distribution in reinforced polymers, this thesis proposed novel methods for the image-based analysis of microstructures. At the outset, constraints for image analysis algorithms in these application areas were defined: Limited user interaction, avoidance of image segmentation, and applicability to large datasets. The models and algorithms proposed thereafter addressed these issues in various ways.

For planar images of non-overlapping objects such as tumor cells or metal particles, a novel statistical model comprising a spatial prior and a likelihood term for which non-parametric statistical classifiers can be used was introduced. The entire system is trainable from a given set of images and associated object coordinates. Given access to such training data, it was demonstrated in Ch. 2 that the system is applicable to different image data. As this process does not require any changes to the method itself, no user interaction other than the provision of training data is required for switching between different multiple object localization tasks. Standard image segmentation-based methods (isodata binarization, watershed transformation, nonmaximum suppression) were shown to perform poorly both on simulated data and on the H&E-stained meningioma cell images. In those cases, where noise was strong, objects overlapped or background clutter complicated the task, the trainable statistical localization method introduced in Ch. 2 produced significantly more accurate localization results than those standard methods.

The next image analysis task addressed in this thesis was the problem of determining the fiber orientation distribution in materials such as glass and carbon fiber-reinforced polymers. While micro-computed tomography is widely used in non-destructive testing of such materials, segmentation of individuals fibers is difficult. This is mostly due to the limits of spatial resolutions achievable by $\mu$CT systems (approximately $1\mu$m), which is often insufficient to separate fibers. To obtain orientation estimates in such situations nevertheless, a method based on repeated image convolutions with anisotropic Gaussian filters (2D and 3D) was introduced. This approach was shown to be more accurate and more robust to noise than an alternative, gradient-based algorithm. It is applicable to fiber-reinforced plastics, as well as to paper fibers. The only parameter of the method is the fiber diameter, which was assumed to be constant throughout the image. With this method, Ch. 3 introduced a system for fiber orientation estimation that does not require any user

interaction and which retrieves this information even when segmentation is not possible.

Repeated image filtering as it is used to detect local fiber orientations in this thesis is computationally expensive, especially for volume images. This limits the applicability of the proposed fiber orientation estimation approach to large datasets. In order to make it better applicable to large amounts of data nevertheless, factorizations of the required anisotropic Gaussian kernel were in investigated in Ch. 4. An analysis of the properties of different factorizations resulted in one that was shown to be optimal in the number of required interpolation steps when implemented. This result, which generalized a previous one reported elsewhere, showed that convolution with an anisotropic Gaussian kernel in $\mathbb{R}^d$ can be implemented efficiently by an image shear, recursive axis-aligned 1D-Gaussian filters, and an inverse image shear. Experiments with implementations of this method showed that it was faster than state-of-the-art implementations based on the fast Fourier transformation for anisotropic Gaussian filtering of 2D and 3D-image data. This improvement was demonstrated to carry over to the computation of Gaussian orientation spaces, thus making the fiber orientation estimation method proposed in this thesis applicable to large dataset.

To characterize the novelty of each contribution over previous results achieved by other authors, overviews over relevant publications were given in each chapter. Few authors have dealt with the multiple object localization problem in the form presented in this thesis. More commonly, object coordinates have been computed as a by-product of dedicated segmentation algorithms. As demonstrated in Ch. 2, it can be advantageous to treat and model these locations separately, as image segmentation can fail in many situations. Methodically, that chapter contributed a novel combination of trainable algorithms with point process statistics, resulting in the first fully trainable application of such spatial statistics models in computer vision. Comparing the proposed fiber orientation estimation algorithm to related publications, the novel contribution there should be characterized as a practical, alternative algorithm to what has been used elsewhere. Clearly, the experimental evaluations of Ch. 3 have shown that this method has some merits over an alternative approach described in the literature. In contrast to other filter types that can be used to compute orientation space representations of images, the Gaussian orientation space used in this thesis comes with the efficient implementation scheme derived in Ch. 4. The theoretical contributions of that chapter were a general analysis of separable anisotropic Gauss filters and a derivation of an optimal one for implementations. These results generalized earlier ones reported elsewhere, and opened the path to new efficient implementations of anisotropic Gaussian filters.

Summarizing the novel contributions of this thesis in all of these fields, the following improvements in the image analysis of microstructures were made. Customized image processing solutions for segmenting and thereby locating objects in images could be replaced by a trainable method, which can be trained to be applicable to a large variety of image data instead of solving only one specific problem. For 2D and 3D-images of fibrous materials in which fibers overlap due to insufficient image resolution, accurate estimation of fiber orientation distributions was enabled through the use of Gaussian orientation space. A principled mathematical treatment of factorizations of Gaussian convolution kernels lead to an optimal separated implementation of anisotropic Gaussian convolution filters with significant speedups of anisotropic 2D and 3D-image filtering and computation of Gaussian orientation spaces.

# Appendix A

# Gradient-Based Training of Convolutional Neural Networks

While the concept of weight sharing in MLPs is described e.g. in [100], the technical details on how these weights are updated and how this is implemented can rarely be found in the literature. An exception is a paper by Simard et al. [150], who provide some implementation details for convolutional neural networks. But this paper also does not formally derive the forward and backward propagation rules for this network architecture. Therefore, this appendix derives the weight update rules for gradient descent training of convolutional neural networks and lists the routines that were used for the experiments presented in Ch. 2. Due to weight sharing in this specific MLP type, the resulting update formulae are slightly different from those known for standard fully connected MLPs. The derivation below follows [20], with the main difference that one needs to deal with the indices of shared weights in the present case.

Note that for simplicity of notation, only 1D-convolutional neural networks will be considered here. Furthermore, the subsampling layers in convolutional neural networks, cf. Sec. 2.3.1, need not be treated separately, for they do not possess any trainable weights. The activations pass through the subsampling layers, and are simply added with a fixed coefficient depending on the number of units that are combined.

## A.1 Backpropagation training

Let $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ denote vectors of activations in subsequent layers of a convolutional neural network, indexed by $i$, $j$ and $k$, respectively. In forward propagation, activations are passed from layer $\mathbf{a}$ through layer $\mathbf{b}$ up to layer $\mathbf{c}$. The elements of each layer are connected by arcs, and each arc is associated with a weight $v_\alpha$ or $w_\beta$, depending on whether it connects layer $\mathbf{a}$ to $\mathbf{b}$ or layer $\mathbf{b}$ to $\mathbf{c}$. These weights are shared, i.e., the indices $\alpha, \beta \in \mathbb{Z}$ are independent of the indices of units to

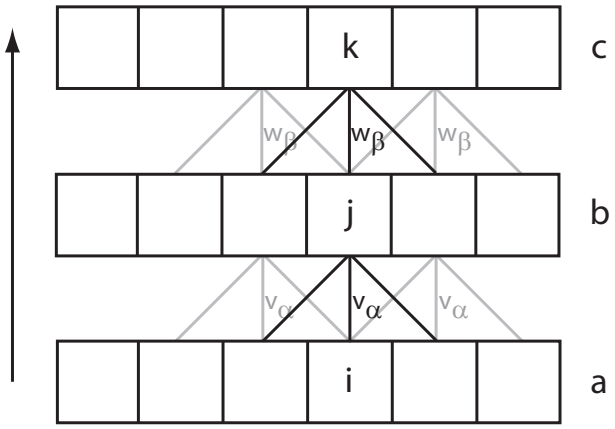



Figure A.1: Illustration of the nomenclature used to derive weight update rules for convolutional neural networks. The connections between adjacent layers **a**-**b** and **b**-**c** share weights $v_\alpha$ and $w_\beta$, respectively. The activation of unit $b_i$ is computed as a linear combination of weights $v_\alpha$ and activations of connected units in layer **a**. As other units in layer **b** use the same weights $v_\alpha$, the activations in layer **b** are the result of a finite linear convolution of **a** and weights $v_\alpha$, i.e., $\mathbf{b} = \mathbf{v} * \mathbf{a}$.

which they connect (Fig. A.1). With this notation, the activation of unit $b_j$ is computed as

$$b_j = \sum_{\alpha=-n_w}^{n_w} v_\alpha f(a_{i+\alpha}), \tag{A.1}$$

where $a_i$ is the central unit in layer **a** to which $b_j$ connects, $f$ denotes the transfer function, e.g., the sigmoid, which has the convenient property that $f'(x) = f(x)(1-f(x))$. $2n_w+1$ is the number of incoming connections in each unit.

Denote the error function measured at the output units by $E$. For the purpose of this thesis, $E$ is the sum-of-squares error function. The first derivative of the overall error with respect to one weight $v_\alpha$,

$$\frac{\partial E}{\partial v_\alpha} = \frac{\partial E}{\partial b_j} \cdot \frac{\partial b_j}{\partial v_\alpha} = \frac{\partial E}{\partial b_j} \cdot \frac{\partial}{\partial v_\alpha} \left[ \sum_{\alpha=-n_w}^{n_w} v_\alpha f(a_{i+\alpha}) \right], \tag{A.2}$$

gives the direction in which the value $v_\alpha$ should be modified in order to locally decrease the error $E$. In (A.2), the only term that cannot be computed from the state of the previous layer, **a**, is $\frac{\partial E}{\partial b_j}$. To compute it requires the following reformulation of that term, leading to the so-called backpropagation algorithm.

$$\frac{\partial E}{\partial b_j} = \sum_k \frac{\partial E}{\partial c_k} \cdot \frac{\partial c_k}{\partial b_j} = \sum_k \frac{\partial E}{\partial c_k} \cdot \frac{\partial}{\partial b_j} \left[ \sum_{\beta=-n_w}^{n_w} w_\beta f(b_{k+\beta}) \right] = f'(b_j) \cdot \sum_k \frac{\partial E}{\partial c_k} w_{j-k}, \tag{A.3}$$

where the last step follows from $k + \beta = j$, as $b_j$ and $c_k$ would not be connected if that was not true. In the partial derivative of the sum with respect to $b_j$, all terms except for the one where $\beta = j - k$ vanish. Observe here that only the term $\frac{\partial E}{\partial c_k}$ cannot be computed from the state of the current layer. It can, however, be computed from the same formula (A.3) by substituting the

appropriate term. This derivative will therefore depend again on the error in the next higher level. This process continues up to the output layer of a MLP, where it can finally be evaluated. Given the error at the last layer, these derivatives in all previous layers can be evaluated, which is called error-backpropagation.

In (A.2), the derivative of the error $E$ with respect to a weight $v_\alpha$ was split at a fixed unit $b_j$. This is a standard procedure in deriving the weight update for MLPs, as a weight can usually be assigned to exactly one unit. This is not the case when using weight sharing. There, the weight $v_\alpha$ is used by every unit $b_j$ in layer $\mathbf{b}$. Therefore, when minimizing the error $E$ with respect to $v_\alpha$, all these units must be taken into account. Using the mean gradient across all units $b_j$ leads to weight update directions $\Delta_{v_\alpha}$,

$$\Delta_{v_\alpha} = \frac{1}{n_b} \sum_{j=1}^{n_b} \frac{\partial E}{\partial b_j} \cdot \frac{\partial b_j}{\partial v_\alpha}, \tag{A.4}$$

where $n_b$ denotes the number of units in layer $\mathbf{b}$. The corresponding weight update rule

$$v_\alpha^{t+1} = v_\alpha^t - \epsilon \Delta_{v_\alpha} \tag{A.5}$$

with step size parameter $\epsilon > 0$ gives a gradient-descent learning rule for convolutional neural networks.

## A.2 Implementation

The update (A.5) rule is directly implementable. In the following C source code, all data is arranged in floating point type arrays. `delta` denotes arrays containing the values $\delta_i := \sum_j \frac{\partial E}{\partial b_j} w_{i-j}$, and `input` represents the values $f'(a_i)$ in the same form. The first step in backpropagation training is the computation of the error terms $\delta$ in each layer, starting at the last layer. This requires a call to the following routine called `backprop` for each layer.

```
// Propagate error terms delta in backward direction, using shared weights
void backprop(float *delta_left, const float *delta_right,
  const float *input, int ninput, const float *weights, int nweights)
{
  // the number of connections to the left and right of an index i
  int offset = nweights/2;

  // the following layer is smaller than the previous one
  int noutput = ninput - 2*offset;

  for (int j = 0; j < ninput; j++)
  {
    float sum = 0.0f;
```

```
    for (int beta = -offset; beta <= offset; beta++)
    {
      int k = j + beta - offset;
      if ((k >= size_out) || (k < 0)) continue;
      sum += delta_right[k] * weights[beta+offset];
    }
    delta_left[j] =  (1.0f - input[j]) * input[j] * sum;
  }
}
```

---

Given the arrays `deltas` in each layer, the weights can be updated using a second function called `update`.

---

```
// Update shared weights using gradient descent
void update(float *weights, int nweights, const float *deltas,
  const float *input, int ninput, float rate)
{
  // the number of connections to the left and right of an index i
  int offset = nweights/2;

  // number of arcs with which one shared weight is associated.
  int nlocations = ninput - 2*offset;

  for (int i = offset; i < ninput-offset; i++)
  {
    for (int j = -offset; j <= offset; j++)
    {
      // gradient descent
      weights[j+offset] += (rate / (float)nlocations) * deltas[i-offset] * input[i+j];
    }
  }
}
```

---

Result of this second function, `update`, is an updated array of weights. To simplify the code and to speed up the computation, the routines above do not perform any edge treatment. Instead, each convolutional layer is $2 \times$ `offset` units shorter than its predecessor. Note that in `backprop`, it therefore needs to be checked whether `k` remains within the bounds of `delta_right` before accessing its elements.

# Appendix B

# Source Codes of the "Isodata" and "Watershed" Control Methods

In an effort to make the results reported on this thesis fully reproducible, this appendix lists the complete source codes of the control methods used in Ch. 2. These are in the form of NIH ImageJ [138] macros. The results reported in Ch. 2 have been produced using version 1.41c of that software.

## B.1 Isodata Thresholding

---

```
macro "Batch Isodata Find Objects" {
  requires("1.41c");
  setBatchMode(true);

  // set ImageJ to assume white objects on black background
  saveSettings();
  run("Options...", "iterations=1 black count=1");

  // low-pass filter
  run("8-bit");
  run("Smooth");
  run("Smooth");
  run("Smooth");
  run("Smooth");
  run("Smooth");

  // binarize
  setAutoThreshold();
```

```
  run("Apply LUT");
  run("Options...","iterations=3 count=1")
  run("Open");

  // output center coordinate of each isolated segment
  run("Set Measurements...", "centroid decimal=0");
  run("Analyze Particles...", "size=100-Infinity circularity=0.00-1.00 show=Nothing
    clear include display");

  restoreSettings();
  close();
}
```

---

The ImageJ macro listed above smoothes a grey-valued image by applying some low-pass filters. An automatic thresholding algorithm (isodata thresholding) is applied to the smoothed image. Isolated foreground regions are then identified and the center coordinate for each such object is output, resulting in a multiple object localization method. In the last step, small regions are omitted to exclude false positives that are introduced due to noise.

## B.2 Watershed Segmentation

---

```
macro "Batch Watershed Find Objects" {
  requires("1.41c");
  setBatchMode(true);

  // set ImageJ to assume white objects on black background
  saveSettings();
  run("Options...", "iterations=1 black count=1");

  // low-pass filter
  run("8-bit");
  run("Smooth"); // smooth
  run("Smooth"); // smooth
  run("Smooth"); // smooth
  run("Smooth"); // smooth
  run("Smooth"); // smooth

  // binarize
  run("Make Binary");
```

```
  // compute distance transform, then use watershed algorithm to segment the image
  run("Watershed");
  run("Invert LUT");

  // output center coordinate of each segment
  run("Set Measurements...", "centroid decimal=0");
  run("Analyze Particles...", "size=100-Infinity circularity=0.00-1.00 show=Nothing
    clear include display");

  restoreSettings();
  close();
}
```

---

Except for the segmentation step, this ImageJ macro is identical to the Isodata one from the previous section. Instead of relying on the binarization step to yield separated objects in the image, this macro applies the watershed transformation on a distance map (these steps are subsumed in the call to ImageJ's `Watershed` method). These steps are standard in the image processing literature and allow for touching objects to to be separated, if they have circular shapes.

# Appendix C

# Points on the Upper Hemisphere

Ch. 3 presented the concept of reduced Gaussian orientation spaces, which, in 3D, requires to filter an image with $n$ anisotropic Gaussian filters oriented along uniformly distributed direction vectors on the upper hemisphere. This appendix presents a detailed description of the method that was used to generate the directions used in Ch. 3. Except for a few numbers $n$, there exists no general solution to the problem of uniformly arranging $n$ points on the sphere $S^2$ [142]. In [49] and [50], a numerical algorithm for computing uniform arrangements of $n$ points on the three-dimensional unit sphere $S^2$ has been proposed. The inverse pairwise Euclidean distance between unit vectors $\mathbf{x}_i \in \mathbb{R}^3$, $\|\mathbf{x}_i\| = 1$, $i = 1, \ldots, n$,

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|}, \tag{C.1}$$

is minimized in two stages. Initially, they apply simulated annealing to get a rough solution, then, in the second step, local Newton optimization is applied to improve upon the results from the first step. In their work, Fliege and Maier constructed these $n$ points on the sphere in order to perform accurate numerical integrations. That is, they were searching for unit vectors $\mathbf{x}_i \in \mathbb{R}^3$ and associated weights $w_i \in \mathbb{R}$ such that integrals over polynomials $f(\cdot)$ on the sphere could be computed via the cubature formula

$$\int_{S^2} f(\mathbf{x}) d\mathbf{x} = \sum_i w_i f(\mathbf{x}_i). \tag{C.2}$$

Therefore, the $w_i$ are referred to as the cubature weights. Let $m$ be the degree of the polynomial $f$. For the case of 3 dimensions, if $n = (m+1)^2$, the weights $\mathbf{w} = (w_1, w_2, \ldots, w_n)^t$ can be computed as the solution of

$$G\mathbf{w} = (1, 1, \ldots, 1)^t, \tag{C.3}$$

where $G$ is the reproducing kernel matrix,

$$G_m(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{4\pi}\left(C_m^{3/2}(\mathbf{x}_i^t\mathbf{x}_j) + C_{m-1}^{3/2}(\mathbf{x}_i^t\mathbf{x}_j)\right) \tag{C.4}$$

and $C_m^{3/2}$ denotes the *Gegenbauer polynomial* of degree $m$ with index 3/2,

$$C_m^{3/2}(\xi) = \sum_{k=0}^{[m/2]} (-1)^k \frac{\frac{3}{2}(\frac{3}{2}+1)\cdots(\frac{3}{2}+m-k-1)}{(m-2k)!k!}(2\xi)^{m-2k} \tag{C.5}$$

For large $m$, computation of the Gegenbauer polynomials via (C.5) becomes intractable due to the factorial terms. For an alternative method for computing the $C_m^{3/2}$ in such cases, and for other details, see [50]. The weights can then be computed from Eq. (C.3) via QR decomposition. For the results presented below, GNU R [136] (Version 2.4), which provides an interface to the corresponding LINPACK function, was used to compute this decomposition.

## C.1 Modifications for the hemisphere

For the purposes of Ch. 3, what is needed are uniformly distributed sampling points on the upper hemisphere. This appendix describes a modification of Fliege and Maier's approach in order to compute such points and their cubature weights, which are used here as figures of merit.

The original two-stage method for computing arrangements of $n$ points on the full sphere is reduced to a one-step simulated annealing optimization of $n/2$ points on the upper hemisphere. The second stage, a Newton optimization, was omitted since the simulated annealing alone gave satisfactory results when a very conservative annealing schedule was used (exponential cooling from $T_0 = 10^4$ to $T_1 = 10^{-7}$). To avoid degenerate solutions, and to guarantee correct edge treatment at the equator ($z = 0$), the score (C.1) is computed for the points $\mathbf{x}_i$ on the upper and their mirror points $-\mathbf{x}_i$ on the lower hemisphere, simultaneously. Due to the symmetry of the filter masks used in Ch. 3, sampling at these points does then not introduce an error at the boundary. For the reasons given in Sec. 3.3, the set of solutions is constrained to include the three unit vectors on the coordinate axes, $(1,0,0)^t$, $(0,1,0)^t$ and $(0,0,1)^t$.

Computation of weights $w_i$ is more sensitive. Of course only the solutions for $i = 1,\ldots,n/2$ on the upper hemisphere are relevant here, but to compute these weights for all $n$ points (original and mirror) need to be considered. Simply adding $-\mathbf{x}$ for each $\mathbf{x}$ as in the optimization process above is not sufficient here, as this turns the matrix $G$ from (C.4) into a singular matrix.

This is avoided by applying slight perturbations of the points on the lower hemisphere. For this, samples from the Fisher-von Mises distribution, c.f. Ch. 3, with very narrow spread ($\kappa = 10^4$) with mean directions $-\mathbf{v}_i$ were used ($i = 1,\ldots,n/2$).

| $n/2$ | $\sum_i w_i/2\pi$ | $\min\{w_i - 4\pi/n\}$ | $\max\{w_i - 4\pi/n\}$ |
|---|---|---|---|
| 18 | 1.001870 | -0.08013 | 0.08192 |
| 50 | 1.003970 | -0.04413 | 0.04718 |
| 98 | 1.000330 | -0.03424 | 0.03638 |

Table C.1: Figures of merit for the three sets of points on the upper hemisphere, analogous to [49, 50], rounded to four significant digits. The normalized sum of absolute weights should ideally be one, the minimal and maximal deviation from the theoretical weight $2\pi$ should be close to zero.

| $n/2$ | Mean [°] | Minimum [°] | Maximum [°] |
|---|---|---|---|
| 18 | 33.13 | 32.14 | 34.57 |
| 50 | 19.74 | 18.03 | 21.23 |
| 98 | 13.80 | 12.71 | 14.99 |

Table C.2: Angular distances to the nearest neighbor at the three resolution levels, computed as the arccosine of their inner products. The largest deviation is below 2°.

## C.2 Results

Due to the restrictions induced by symmetry, the proposed approach does not allow for an arbitrary number $n$ of points to be chosen. In addition to the restriction that $n = (m+1)^2$, $n$ must also be an even number. The first few $n$ that obey both conditions are $n/2 = 2, 8, 18, 32, 50, 98, 162, 200$. From these, three "resolution levels" have been selected: 18, 50, and 98. These are sufficient for handling the trade off between orientation resolution and runtime restrictions. The resulting points along with their cubature weights are listed in Tables C.3 through C.5.

The quality of the computed point arrangements on the sphere can be judged by analyzing the cubature weights $w_i$. Their absolute sum should equal the area of the hemisphere, and they should not deviate too far from their theoretical optimum, $4\pi/n$. Compared to the results presented in [50], the scaled sums of absolute weights loose several decimal places of precision, but the minimal and maximal deviations from $4\pi/n$ remain within the error bounds that have also been reported, there. This loss of accuracy is explained by the much higher number of constraints in the present optimization process (due to the fact that the points on the lower half could not move freely).

Another measure of interest is the geodesic distance to resulting points' nearest neighbors. The geodesic distance on the sphere is simply the angle between two unit vectors, cf. Tab. C.2. These distances allow for an assessment of the gain in precision when switching between one of these three resolution levels. Furthermore, the small range of angular distances to the nearest neighbor (at most 3.2°) is another indicator for the quality of the results.

| x | y | z | w | x | y | z | w |
|---|---|---|---|---|---|---|---|
| 0.0000 | 0.0000 | 1.000 | 0.3355 | -0.4067 | 0.3681 | 0.8361 | 0.3451 |
| 0.0000 | 1.000 | 0.0000 | 0.4310 | -0.7079 | -0.1383 | 0.6926 | 0.3321 |
| 1.000 | 0.0000 | 0.0000 | 0.2689 | 0.8243 | 0.4038 | 0.3969 | 0.3833 |
| 0.6882 | -0.7223 | 0.06836 | 0.3614 | -0.3536 | 0.8306 | 0.4301 | 0.2845 |
| -0.8340 | 0.3837 | 0.3965 | 0.3878 | 0.3993 | -0.4139 | 0.8180 | 0.3449 |
| 0.2682 | -0.8340 | 0.4822 | 0.3203 | -0.3993 | -0.8279 | 0.3938 | 0.3104 |
| 0.1375 | 0.6612 | 0.7375 | 0.4103 | -0.8276 | -0.5064 | 0.2419 | 0.3879 |
| -0.2538 | -0.5008 | 0.8275 | 0.3853 | 0.5262 | 0.2095 | 0.8241 | 0.3146 |
| 0.4728 | 0.8347 | 0.2826 | 0.2912 | 0.8364 | -0.2939 | 0.4626 | 0.3847 |

Table C.3: 18 approximately uniform points on the upper hemisphere along with their cubature weights.

| x | y | z | w | x | y | z | w |
|---|---|---|---|---|---|---|---|
| 0.0000 | 0.0000 | 1.000 | 0.1119 | 0.3065 | 0.1606 | 0.9382 | 0.1394 |
| 0.0000 | 1.000 | 0.0000 | 0.1208 | -0.04071 | 0.8130 | 0.5808 | 0.1006 |
| 1.000 | 0.0000 | 0.0000 | 0.1097 | 0.4786 | 0.4078 | 0.7776 | 0.08581 |
| -0.9509 | -0.3006 | 0.07362 | 0.1409 | 0.8163 | -0.4627 | 0.3458 | 0.08454 |
| 0.7522 | -0.1830 | 0.6330 | 0.09602 | 0.6441 | 0.7649 | 0.01028 | 0.1154 |
| 0.4999 | -0.1132 | 0.8587 | 0.1298 | 0.1135 | 0.4861 | 0.8665 | 0.1372 |
| -0.7268 | -0.4291 | 0.5364 | 0.1305 | 0.6003 | -0.5230 | 0.6051 | 0.1728 |
| 0.3504 | -0.4871 | 0.8000 | 0.1025 | 0.1648 | 0.9332 | 0.3193 | 0.1303 |
| -0.5508 | 0.4231 | 0.7195 | 0.1151 | -0.2319 | 0.5956 | 0.7691 | 0.1233 |
| -0.6260 | 0.7467 | 0.2248 | 0.08153 | -0.2023 | -0.9235 | 0.3261 | 0.1128 |
| 0.6810 | 0.5184 | 0.5172 | 0.1359 | -0.7970 | -0.5593 | 0.2278 | 0.09738 |
| -0.8963 | -0.1344 | 0.4225 | 0.1377 | 0.9391 | -0.1293 | 0.3183 | 0.1692 |
| -0.07356 | -0.5348 | 0.8418 | 0.1332 | 0.9086 | 0.2116 | 0.3600 | 0.09114 |
| -0.1808 | 0.2854 | 0.9412 | 0.1205 | 0.8296 | 0.5206 | 0.202 | 0.1406 |
| -0.2378 | 0.9286 | 0.2847 | 0.1442 | -0.4512 | 0.729 | 0.5148 | 0.1456 |
| 0.3961 | -0.9175 | 0.03496 | 0.1269 | -0.3654 | -0.9308 | 0.009956 | 0.1326 |
| 0.1450 | -0.2750 | 0.9504 | 0.1291 | -0.7777 | 0.4919 | 0.3914 | 0.1440 |
| -0.4510 | -0.5155 | 0.7286 | 0.1227 | 0.1807 | -0.7539 | 0.6316 | 0.1293 |
| -0.5295 | -0.7528 | 0.3911 | 0.1444 | -0.4492 | 0.09224 | 0.8887 | 0.1404 |
| 0.5013 | 0.7895 | 0.3541 | 0.1120 | 0.3007 | 0.7050 | 0.6423 | 0.1502 |
| 0.7212 | 0.1620 | 0.6735 | 0.1578 | 0.9132 | -0.4076 | 0.001086 | 0.1213 |
| -0.6430 | -0.1561 | 0.7498 | 0.1153 | 0.7095 | -0.6982 | 0.09548 | 0.1675 |
| -0.2062 | -0.7613 | 0.6147 | 0.1171 | 0.4884 | -0.7808 | 0.3896 | 0.09009 |
| 0.1636 | -0.9293 | 0.3310 | 0.1545 | -0.7626 | 0.1729 | 0.6233 | 0.1223 |
| -0.2936 | -0.2392 | 0.9255 | 0.1296 | -0.9370 | 0.1960 | 0.2892 | 0.1161 |

Table C.4: 50 approximately uniform points on the upper hemisphere along with their cubature weights.

| x | y | z | w | x | y | z | w |
|---|---|---|---|---|---|---|---|
| 0.0000 | 0.0000 | 1.000 | 0.05356 | 0.7990 | 0.3133 | 0.5132 | 0.05387 |
| 0.0000 | 1.000 | 0.0000 | 0.05561 | -0.7340 | -0.5676 | 0.3729 | 0.07807 |
| 1.000 | 0.0000 | 0.0000 | 0.07369 | -0.2617 | 0.5248 | 0.8100 | 0.07432 |
| 0.1123 | 0.2105 | 0.9711 | 0.06419 | 0.2246 | -0.2885 | 0.9308 | 0.06246 |
| 0.4640 | 0.7170 | 0.5203 | 0.07475 | 0.2230 | -0.6909 | 0.6877 | 0.08180 |
| -0.4386 | 0.3664 | 0.8206 | 0.04615 | -0.9558 | -0.04514 | 0.2905 | 0.03612 |
| 0.6042 | -0.6614 | 0.4445 | 0.09299 | 0.9608 | -0.2727 | 0.05055 | 0.05677 |
| 0.1835 | 0.8200 | 0.5421 | 0.05276 | 0.3487 | 0.3934 | 0.8507 | 0.08560 |
| -0.8416 | 0.3514 | 0.4101 | 0.06615 | 0.1053 | -0.9558 | 0.2745 | 0.07098 |
| 0.9205 | 0.1188 | 0.3723 | 0.08265 | -0.02105 | -0.7113 | 0.7026 | 0.03521 |
| 0.3976 | 0.5863 | 0.7059 | 0.04235 | 0.5157 | -0.01183 | 0.8567 | 0.05747 |
| -0.4412 | 0.8904 | 0.1119 | 0.06790 | -0.5745 | -0.5484 | 0.6077 | 0.08850 |
| 0.4022 | -0.4835 | 0.7775 | 0.06828 | -0.6623 | -0.1663 | 0.7305 | 0.07778 |
| 0.7104 | 0.6265 | 0.3207 | 0.04532 | -0.4629 | -0.08714 | 0.8821 | 0.05062 |
| -0.7290 | 0.07126 | 0.6807 | 0.03598 | 0.7631 | -0.4670 | 0.4468 | 0.04723 |
| -0.7544 | 0.5888 | 0.2902 | 0.07259 | 0.6780 | -0.1597 | 0.7175 | 0.06393 |
| -0.5773 | -0.7556 | 0.3095 | 0.07099 | -0.5033 | -0.3536 | 0.7885 | 0.07177 |
| 0.6306 | -0.4188 | 0.6535 | 0.07568 | -0.7285 | -0.6742 | 0.1213 | 0.04581 |
| 0.3253 | 0.9437 | 0.05981 | 0.06261 | -0.0825 | 0.6486 | 0.7566 | 0.05022 |
| 0.6153 | 0.7827 | 0.09348 | 0.06810 | -0.08044 | -0.8949 | 0.4390 | 0.04499 |
| -0.2080 | 0.9681 | 0.1399 | 0.06610 | 0.1608 | -0.8489 | 0.5035 | 0.07458 |
| -0.7751 | 0.6318 | 0.0002917 | 0.07112 | 0.8267 | -0.1966 | 0.5272 | 0.06607 |
| -0.3475 | -0.5671 | 0.7467 | 0.04805 | -0.4457 | -0.7350 | 0.5111 | 0.03027 |
| -0.2752 | 0.1736 | 0.9456 | 0.05140 | 0.3824 | -0.8408 | 0.3832 | 0.05841 |
| -0.8673 | 0.1268 | 0.4814 | 0.06631 | -0.07185 | 0.8220 | 0.5649 | 0.06939 |
| 0.5109 | -0.8465 | 0.1499 | 0.07899 | 0.6541 | 0.1579 | 0.7397 | 0.08107 |
| 0.6801 | -0.6961 | 0.2302 | 0.04062 | -0.8672 | -0.3559 | 0.3483 | 0.06342 |
| 0.2842 | 0.8961 | 0.3408 | 0.07022 | 0.8162 | 0.5672 | 0.1097 | 0.08290 |
| 0.6628 | 0.5296 | 0.5294 | 0.08203 | -0.3272 | -0.8899 | 0.3180 | 0.07861 |
| 0.3802 | 0.1785 | 0.9075 | 0.05094 | 0.9630 | 0.2419 | 0.1189 | 0.05983 |
| -0.2258 | -0.7664 | 0.6014 | 0.1005 | 0.1663 | 0.6637 | 0.7293 | 0.08514 |
| 0.2839 | -0.9504 | 0.1273 | 0.05653 | -0.9596 | 0.1902 | 0.2072 | 0.07666 |
| -0.1631 | -0.9726 | 0.1658 | 0.05939 | 0.8727 | 0.3922 | 0.2907 | 0.06243 |
| -0.3477 | 0.8640 | 0.3641 | 0.06000 | 0.4641 | -0.2585 | 0.8472 | 0.05850 |
| -0.1244 | -0.5157 | 0.8477 | 0.08321 | 0.2510 | -0.04831 | 0.9668 | 0.07867 |
| -0.08116 | 0.9264 | 0.3678 | 0.06516 | 0.09778 | 0.4631 | 0.8809 | 0.04617 |
| 0.4428 | -0.6666 | 0.5997 | 0.03692 | 0.1065 | 0.9722 | 0.2085 | 0.06186 |
| -0.5520 | 0.7312 | 0.4009 | 0.07292 | -0.6261 | 0.7696 | 0.1254 | 0.05341 |
| -0.9629 | -0.2334 | 0.1351 | 0.07528 | 0.9686 | -0.05805 | 0.2418 | 0.04658 |
| -0.2252 | -0.06329 | 0.9723 | 0.07986 | -0.7321 | -0.3648 | 0.5753 | 0.02988 |
| -0.2798 | -0.3154 | 0.9068 | 0.05675 | -0.8492 | -0.1375 | 0.5099 | 0.09299 |
| -0.1102 | 0.3377 | 0.9348 | 0.08216 | -0.3187 | 0.7403 | 0.5919 | 0.06328 |
| 0.5882 | 0.3877 | 0.7097 | 0.05481 | 0.8540 | -0.4903 | 0.1740 | 0.06812 |
| -0.6770 | 0.3189 | 0.6633 | 0.08162 | 0.1285 | -0.5054 | 0.8533 | 0.05953 |
| -0.6754 | 0.5194 | 0.5235 | 0.04665 | -0.4847 | 0.5791 | 0.6555 | 0.06843 |
| -0.5308 | 0.1442 | 0.8351 | 0.08507 | 0.5052 | 0.8107 | 0.2959 | 0.05914 |
| -0.4842 | -0.8698 | 0.0952 | 0.05582 | -0.02569 | -0.2709 | 0.9623 | 0.06006 |
| 0.9044 | -0.2821 | 0.3200 | 0.07742 | 0.8098 | 0.04909 | 0.5847 | 0.05192 |
| -0.8823 | -0.4613 | 0.09337 | 0.06465 | -0.8902 | 0.4334 | 0.1401 | 0.04916 |

Table C.5: 98 approximately uniform points on the upper hemisphere along with their cubature weights.

# Appendix D

# Details on the Factorization and Parameterization of $\Sigma$ in $\mathbb{R}^3$

For applications in image processing, explicit factorizations and parameterizations of the 3D-anisotropic Gaussian convolution filter were presented in Ch. 4. As the corresponding derivations are cumbersome and not very instructive, they have been deferred until the present appendix, which contains all mathematical details of the symmetric factorization of Cholesky type and its parameterization using polar spherical coordinates.

## D.1 Explicit symmetric factorization of Cholesky type in $\mathbb{R}^3$

The following steps for deriving explicit formulas for the entries of $V$ and $D$ in the factorization $\Sigma = VDV^t$ have been described in Sec. 4.4.1. As there, let $v_{i,j}$, $1 \le i < j \le d$, denote the entries in the upper half of $V$, and $d_i^2$, $i = 1, \ldots, d$, the entries on the diagonal of $D$. The elements of $\Sigma$ will be written as $s_{i,j}$ with $s_{i,j} = s_{j,i}$.

$$
\begin{aligned}
VDV^t &= \begin{pmatrix} 1 & v_{1,2} & v_{1,3} \\ 0 & 1 & v_{2,3} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} d_1^2 & 0 & 0 \\ 0 & d_2^2 & 0 \\ 0 & 0 & d_3^2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ v_{1,2} & 1 & 0 \\ v_{1,3} & v_{2,3} & 1 \end{pmatrix} \\
&= \begin{pmatrix} d_1^2 & d_2^2 v_{1,2} & d_3^2 v_{1,3} \\ 0 & d_2^2 & d_3^2 v_{2,3} \\ 0 & 0 & d_3^2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ v_{1,2} & 1 & 0 \\ v_{1,3} & v_{2,3} & 1 \end{pmatrix} \\
&= \begin{pmatrix} d_1^2 + d_2^2 v_{1,2}^2 + d_3^2 v_{1,3}^2 & d_2^2 v_{1,2} + d_3^2 v_{1,3} v_{2,3} & d_3^2 v_{1,3} \\ d_2^2 v_{1,2} + d_3^2 v_{1,3} v_{2,3} & d_2^2 + d_3^2 v_{2,3}^2 & d_3^2 v_{2,3} \\ d_3^2 v_{1,3} & d_3^2 v_{2,3} & d_3^2 \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} s_{1,1} & s_{1,2} & s_{1,3} \\ s_{1,2} & s_{2,2} & s_{2,3} \\ s_{1,3} & s_{2,3} & s_{3,3} \end{pmatrix}
\end{aligned} \tag{D.1}
$$

Solving the individual equations column-by-column from right to left, starting at the elements with the highest row index leads to the following set of solutions.

$$
\begin{aligned}
d_3^2 &= s_{3,3} \\
v_{2,3} &= \frac{s_{2,3}}{s_{3,3}} \\
v_{1,3} &= \frac{s_{1,3}}{s_{3,3}} \\
d_2^2 &= s_{2,2} - d_3^2 v_{2,3}^2 = s_{2,2} - s_{3,3}\frac{s_{2,3}^2}{s_{3,3}^2} = s_{2,2} - \frac{s_{2,3}^2}{s_{3,3}} \\
v_{1,2} &= \frac{s_{1,2} - d_3^2 v_{1,3} v_{2,3}}{d_2^2} = \frac{s_{1,2} - s_{3,3}\frac{s_{1,3}}{s_{3,3}}\frac{s_{2,3}}{s_{3,3}}}{s_{2,2} - \frac{s_{2,3}^2}{s_{3,3}}} = \frac{s_{1,2} - \frac{s_{1,3}s_{2,3}}{s_{3,3}}}{s_{2,2} - \frac{s_{2,3}^2}{s_{3,3}}} \\
&= \frac{\frac{s_{1,2}s_{3,3} - s_{1,3}s_{2,3}}{s_{3,3}}}{\frac{s_{2,2}s_{3,3} - s_{2,3}^2}{s_{3,3}}} = \frac{s_{1,2}s_{3,3} - s_{1,3}s_{2,3}}{s_{2,2}s_{3,3} - s_{2,3}^2} \\
d_1^2 &= s_{1,1} - \left(s_{2,2} - \frac{s_{2,3}^2}{s_{3,3}}\right)\frac{(s_{1,2}s_{3,3} - s_{1,3}s_{2,3})^2}{(s_{2,2}s_{3,3} - s_{2,3}^2)^2} - s_{3,3}\frac{s_{1,3}^2}{s_{3,3}^2} \\
&= s_{1,1} - \frac{(s_{1,2}s_{3,3} - s_{1,3}s_{2,3})^2}{s_{3,3}(s_{2,2}s_{3,3} - s_{2,3}^2)} - \frac{s_{1,3}^2}{s_{3,3}}
\end{aligned}
\tag{D.2}
$$

Since $\Sigma$ is positive definite, all its diagonal elements must be strictly positive. Furthermore, it also follows that the determinants of all principal submatrices of $\Sigma$ are strictly positive, especially $\left|\begin{pmatrix} s_{2,2} & s_{2,3} \\ s_{2,3} & s_{3,3} \end{pmatrix}\right| = s_{2,2}s_{3,3} - s_{2,3}^2 > 0$. For an account of these and other propeties of positive definite matrices, see e.g. [63, Ch. 4]. Therefore, all terms in the derivation above are well defined, which therefore proves the existence of the required symmetric factorization of Cholesky type for covariance matrices $\Sigma$ in $\mathbb{R}^3$.

## D.2 Parameterization of $VDV^t$ in $\mathbb{R}^3$ using polar coordinates

Let $\varphi$ and $\psi$ be rotation angles about the $x_3$-axis, and $\theta$ about the $x_1$-axis. Euler rotation reduces 3D-rotation to a sequence of three rotations about three coordinate axes. One variant of Euler rotation is a composition of rotations about the $x_3$-, $x_2$- and again the $x_3$-axes. The resulting rotation matrix $R$ in a right-handed coordinate system is composed as

$$R = R_{x_3}(\varphi)R_{x_2}(\theta)R_{x_3}(\psi) \tag{D.3}$$

$$= \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{D.4}$$

When rotating the unit vector $\mathbf{e}_3 = (0, 0, 1)^t$, the third rotation about the $x_3$-axis becomes irrelevant,

$$R\mathbf{e}_3 = R_{x_3}(\varphi)R_{x_2}(\theta)R_{x_3}(\psi)\mathbf{e}_3 = R_{x_3}(\varphi)R_{x_2}(\theta)\mathbf{e}_3 \tag{D.5}$$

$$= \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \mathbf{e}_3 = \begin{pmatrix} \cos\varphi\sin\theta \\ \sin\varphi\sin\theta \\ \cos\theta \end{pmatrix}, \tag{D.6}$$

which shows the relation between spherical polar coordinates and Euler rotations: Euler rotations transform $\mathbf{e}_3$ onto the point $(\theta, \varphi)$ in polar coordinates, where $\theta$ denots the colatitude angle away from the $x_3$-axis, and $\varphi$ the longitude angle in the $x_1x_2$-plane. This observation gives rise to a convenient parameterization of the 3D-covariance matrix $\Sigma$ in terms of polar coordinates. Let $\sigma_1^2$, $\sigma_2^2$ and $\sigma_3^2$ denote the spead parameters of an axis-aligned Gaussian kernel along its three major axes. From the practical perspective in Sec. 4.4.2, it is not necessary to consider the most general 3D-Gaussian with six degrees of freedom. Therefore, start with a Gaussian kernel that is invariant to rotations about the $x_3$-axis i.e. $\sigma_1 = \sigma_2$. Depending on whether $\sigma_3^2 > \sigma_2^2$ or $\sigma_3^2 < \sigma_2^2$, this corresponds to either a prolate or oblate filter shape, respectively. When rotating such a kernel using R from (D.4), the third rotation about the $x_3$ axis becomes once again irrelevant. Recall the eigen decomposition $\Sigma = VDV^t$, where $V$ is orthonormal and $D$ diagonal, from Sec. 3.2. By specifying $D$ as the diagonal matrix of variances and choosing a rotation matrix for $V$, $\Sigma$ can be derived as follows.

$$\Sigma = R_{x_3}(\varphi)R_{x_2}(\theta) \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_1^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix} (R_{x_3}(\varphi)R_{x_2}(\theta))^t \tag{D.7}$$

$$= \begin{pmatrix} \cos\varphi\cos\theta & -\sin\varphi & \cos\varphi\sin\theta \\ \sin\varphi\cos\theta & \cos\varphi & \sin\varphi\sin\theta \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_1^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix} \begin{pmatrix} \cos\varphi\cos\theta & \sin\varphi\cos\theta & -\sin\theta \\ -\sin\varphi & \cos\varphi & 0 \\ \cos\varphi\sin\theta & \sin\varphi\sin\theta & \cos\theta \end{pmatrix} \tag{D.8}$$

$$= \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{1,2} & a_{2,2} & a_{2,3} \\ a_{1,3} & a_{2,3} & a_{3,3} \end{pmatrix}, \tag{D.9}$$

where

$$
\begin{aligned}
a_{1,1} &= \sigma_1^2 \sin^2\varphi + \cos^2\varphi(\sigma_1^2\cos^2\theta + \sigma_3^2\sin^2\theta)\\
a_{1,2} &= (\sigma_3^2 - \sigma_1^2)\cos\varphi\sin\varphi\sin^2\theta\\
a_{1,3} &= (\sigma_3^2 - \sigma_1^2)\cos\varphi\sin\theta\cos\theta\\
a_{2,2} &= \sigma_1^2 \cos^2\varphi + \sin^2\varphi(\sigma_1^2\cos^2\theta + \sigma_3^2\sin^2\theta)\\
a_{2,3} &= (\sigma_3^2 - \sigma_1^2)\cos\theta\sin\varphi\sin\theta\\
a_{3,3} &= \sigma_1^2 \sin^2\theta + \sigma_3^2\cos^2\theta
\end{aligned}
$$

The 3D-covariance matrix in (D.9) specifies a Gaussian kernel that is aligned to the direction $(\varphi, \theta)$ in spherical polar coordinates. For $\varphi = 0$ and $\theta = 0$, all but the diagonal entries vanish, leaving a kernel that is aligned to the $x_3$-axis. Setting $\varphi = 0$ and $\theta = \pi/2$ rotates the kernel onto the $x_1$ axis, since again all off-diagonal elements vanish (each of them contains a multiplicative term $\cos\theta$) and $\sigma_3^2$ appears in the first diagonal entry of $\Sigma$. To reach the third coordinate axis $x_2$, set $\varphi = \theta = \pi/2$, thus again canceling out all off-diagonal terms and moving $\sigma_3^2$ into the central element of this 3D-covariance matrix.

To obtain a parameterization of $V$ and $D$, the two matrices representing the non-orthogonal separation of the Gaussian filter kernel proposed in Ch. 4, the elements of $\Sigma$ in (D.9) are substituted into the different terms in (D.2).

This results in six equations that define the elements of $V$ and $D$ depending on the given parameters $\sigma_1^2,\ \sigma_3^2$, the spread of the unrotated Gaussian, and $\theta,\ \varphi$, the spherical polar coordinate of the filtering direction. As these terms have rather complicated forms, they were automatically simplified using the Mathematica software package [173].

$$d_1^2 = \frac{\sigma_1^2\sigma_1^2\sigma_3^2}{d_2^2 d_3^2} \tag{D.10}$$

$$d_2^2 = \frac{\sigma_1^2(\sigma_3^2\sin^2\varphi + \cos^2\varphi(\sigma_3^2\cos^2\theta + \sigma_1^2\sin^2\theta))}{\sigma_1^2\sin^2\theta + \sigma_3^2\cos^2\theta} \tag{D.11}$$

$$d_3^2 = \sigma_1^2\sin^2\theta + \sigma_3^2\cos^2\theta \tag{D.12}$$

$$v_{1,2} = \frac{2(\sigma_3^2 - \sigma_1^2)\sin 2\varphi\sin^2\theta}{\sigma_1^2 + 3\sigma_3^2 + (\sigma_1^2 - \sigma_3^2)(\cos 2\varphi - 2\cos^2\varphi\cos 2\theta)} \tag{D.13}$$

$$v_{1,3} = \frac{(\sigma_3^2 - \sigma_1^2)\cos\varphi\cos\theta\sin\theta}{\sigma_1^2\sin^2\theta + \sigma_3^2\cos^2\theta} \tag{D.14}$$

$$v_{2,3} = \frac{(\sigma_3^2 - \sigma_1^2)\cos\theta\sin\varphi\sin\theta}{\sigma_1^2\sin^2\theta + \sigma_3^2\cos^2\theta} \tag{D.15}$$

It is again easy to check a few special cases. Whenever $\theta = 0$, $V$ reduces to a unit matrix as the terms for $v_{1,2}$, $v_{1,3}$ and $v_{2,3}$ all contain a factor $\sin\theta$ in the numerator. Furthermore, this simple, unrotated case results in $d_1^2 = \sigma_1^2$, $d_2^2 = \sigma_1^2$ and $d_3^2 = \sigma_3^2$, as required. Recall from Sec. 4.3.1 that applying the $V$-transformation to the Gaussian kernel has the geometric interpretation of a 3D-shear of an ellipsoid onto the Cartesian $x_1x_2x_3$-coordinate grid, with $x_3$ being the fixed coordinate. Therefore, $V$ will remain a unit matrix for all $\theta, \varphi = k \cdot \pi/2$, $k \in \mathbb{N}$.

# Appendix E

# Curriculum Vitae

**Personal Data**

| | |
|---|---|
| Name | Oliver Wirjadi |
| Date of Birth | October 6, 1978 |
| Place of Birth | Berlin, Germany |

**Education**

| | |
|---|---|
| 1998 | Abitur, Canisius-Kolleg Berlin |
| 2003 | Diploma, Computer Science, Technische Universität Berlin |

**Academic and Professional Experience**

| | |
|---|---|
| 1998 | Intern, Vestibular Research Lab, Freie Universität Berlin |
| 1998–2001 | Student assistant, Vision Pearls GbR, Berlin |
| 2001–2002 | Student assistant, Electrical and Computer Engineering Department, Illinois Institute of Technology, Chicago |
| 2002–2003 | Software developer, Idencom Germany GmbH, Berlin |
| 2004–2008 | Ph.D. student, Department of Computer Science, Technische Universität Kaiserslautern |
| 2004–2008 | Stipendiary, Image Processing Department, Fraunhofer ITWM, Kaiserslautern (with an interruption in 2007) |
| 2007 | Research assistant, Department of Mathematics and Sciences, Hochschule Darmstadt |

# Acknowledgements

Never had I been able to finish this thesis without the support of all the people that I worked with over the course of the past years. Thanks to everyone at the "MAB" and – just down the street! – at "IUPR".

Special thanks for help in preparing this thesis go to ...

... Prof. Thomas Breuel for supervision and counsel.

... Prof. Andreas König for volunteering as referee of this thesis.

... Ronald Rösch and Katja Schladitz for continuous support.

... all volunteer reviewers: Claudia Redenbach, Katja Schladitz, Katharina Robb, Adrian Ulges, Christoph Lampert, Yoo-Jin Kim and Steffi Peters.

... Christoph Keßler for OpenGL magic.

... Yoo-Jin Kim for providing the Meningioma data.

... Rudi Velthuis, Alexander Rack and Jürgen Goebbels for providing the glass fiber-reinforced polymer data.

... Jürgen Becker for providing the carbon paper data.

... T. Potyra, M. Steinhauser and Michael Godehardt for providing the SMC data.

... Alexander Rack and Lukas Helfen for providing the aluminium alloy data.

... Faisal Shafait for providing the camera-captured document images.

... Manuel Schöneberger for his implementation of RSA cylinder processes.

... the authors of various open source software that I used: the GNU R development team, A. Baddeley, R. Turner, B.M. Clapper, J.-M. Geusebroek, T. Joachims, W.S. Rasband, M. Frigo, S.G. Johnson.

# Bibliography


[1] M.R. Amin, M. Kurosaki, T. Watanabe, S. Tanaka, and T. Hori. A comparative study of MIB-1 staining indices of gliomas measured by NIH image analysis program and conventional manual cell counting method. *Neurological Research*, 22(5):495–500, Jul 2000.

[2] M. Andersson, J. Wiklund, and H. Knutsson. Sequential filter trees for efficient 2D 3D and 4D orientation estimation. Technical Report LiTH-ISY-R-2070, ISY, SE-581 83 Linköping, Sweden, Nov. 1998.

[3] F. Al-Awadhi, M. Hurn, and C. Jennison. Improving the acceptance rate of reversible jump MCMC proposals. *Statistics & Probability Letters*, 69(2):189–198, Aug 2004.

[4] F. Al-Awadhi, C. Jennison, and M. Hurn. Statistical image analysis for a confocal microscopy two-dimensional section of cartilage growth. *J. Royal Statistical Society: Series C (Applied Statistics)*, 53(1):31–49, Jan 2004.

[5] H. Axer, M. Leunert, M. Mürköster, D.Gräßel, L. Larsen, L.D. Griffin, and D. Keyserlingk. A 3D fiber model of the human brainstem. *Computerized Medical Imaging and Graphics*, 26(6):439–444, Dec 2002.

[6] S.R. Aylward and E. Bullit. Initialization, noise, singularities, and scale in height ridge traversal for tubular object centerline extraction. *IEEE Trans. Medical Imaging*, 21(2):61–75, Feb 2002.

[7] A. Baddeley and E.B. Vedel Jensen. *Stereology for Statisticians*. Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, FL, USA, 2004.

[8] A.J. Baddeley and M.N.M. van Lieshout. Object recognition using Markov spatial processes. In *Proc. Int. Conf. Pattern Recognition*, volume 2, pages 136–139, Aug 1992.

[9] A.J. Baddeley and R. Turner. Practical maximum pseudolikelihood for spatial point patterns. *Austr. NZ J. Statistics*, 42(3):283–322, Sep 2000.

[10] A.J. Baddeley and R. Turner. Spatstat: An R package for analyzing spatial point patterns. *J. Statistical Software*, 12(6):1–42, Jan 2005.

[11] C.E. Bakis, L.C. Bank, V.L. Brown, E. Cosenza, J.F. Davalos, J.J. Lesko, A. Machida, S.H. Rizkalla, and T.C. Triantafillou. Fiber-reinforced polymer composites for construction – state-of-the-art review. *J. Composites for Construction*, 6(2):73–87, 2002.

[12] J. Banhart, editor. *Advanced Tomographic Methods in Materials Research and Engineering*. Monographs on the Physics and Chemistry of Materials. Oxford University Press, Oxford, UK, 2008.

[13] J. Banhart, D. Bellmann, and H. Clemens. Investigation of metal foam formation by microscopy and ultra small-angle neutron scattering. *Acta Materialia*, 49(17):3409–3420, Oct 2001.

[14] J. Becker, V. Schulz, and A. Wiegmann. Numerical determination of two-phase material parameters of a gas diffusion layer using tomography images. *J. Fuel Cell Science and Technology*, 5(2):021006, May 2008.

[15] S. Becker and Y. LeCun. Improving the convergence of back-propagation learning with second-order methods. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proc. 1988 Connectionist Models Summer School*, pages 29–37, San Mateo, 1989. Morgan Kaufman.

[16] E. Bengtsson, C. Wählby, and J. Lindblad. Robust cell image segmentation methods. *Pattern Recognition and Image Analysis*, 14(2):157 – 167, 2004.

[17] H.E. Bennink, H.C. van Assen, G.J. Streekstra, R. ter Wee, J.A.E. Spaan, and B.M. ter Haar Romeny. A novel 3D multi-scale lineness filter for vessel detection. In *Proc. Medical Image Computing and Computer-Assisted Intervention*, pages 436–443, Oct 2007.

[18] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, Sep 1975.

[19] J. Besag. On the statistical analysis of dirty pictures. *J. Royal Statistical Society B*, 48(3):259–302, 1986.

[20] C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, Oxford, UK, 1995.

[21] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, NY, USA, 2006.

[22] L. Bottou and Y. LeCun. Large scale online learning. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[23] S.S. Bukhari, F. Shafait, and T.M. Breuel. Segmentation of curled text lines using active contours. In *Proc. IAPR Int. Workshop Document Analysis Systems*, Sep 2008.

[24] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, Jun 1998.

[25] W. Camacho C, C.L. Tucker III, S. Yalvaç, and R.L. McGee. Stiffness and thermal expansion predictions for hybrid short fiber composites. *Polymer Composites*, 11(4):229–239, Aug 1990.

[26] C.C. Chang and C.J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[27] S. Chaudhuri, S. Chatterjee, N. Katz, M. Nelson, and M. Goldbaum. Detection of blood vessels in retinal images using two-dimensional matched filters. *IEEE Trans. Medical Imaging*, 8(3):263–269, Sep 1989.

[28] J. Chen, Y. Sato, and S. Tamura. Orientation space filtering for multiple orientation line segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recongnition*, pages 311–317, Jun 1998.

[29] J. Chen, Y. Sato, and S. Tamura. Orientation space filtering for multiple orientation line segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(5):417–429, May 2000.

[30] X. Chen, X. Zhou, and S.T.C. Wong. Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy. *IEEE Trans. Biomedical Engineering*, 53(4):762–766, Apr 2006.

[31] S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, Nov 1995.

[32] A.R. Clarke, G. Archenhold, and N.C. Davidson. A novel technique for determining the 3D spatial distribution of glass fibres in polymer composites. *Composites Science and Technology*, 55(1):75–91, 1995.

[33] W.R. Crum, C. Tanner, and D.J. Hawkes. Anisotropic multi-scale fluid registration: evaluation in magnetic resonance breast imaging. *Physics in Medicine and Biology*, 50(21):5153–5174, Nov 2005.

[34] F. Daniels, B. ter Haar Romeny, M. Rubbens, and H. van Assen. Quantification of collagen orientation in 3D engineered tissue. In *Proc. Kuala Lumpur Int. Conf. Biomedical Engineering*, pages 282–286, Dec 2006.

[35] D. Decoste and B. Schölkopf. Training Invariant Support Vector Machines. *Machine Learning*, 46(1–3):161–190, Jan 2002.

[36] X. Descombes and J. Zerubia. Marked point process in image analysis. *IEEE Signal Processing Magazine*, 19(5):77–84, Sep 2002.

[37] P.J. Diggle, T. Fiksel, G. Grabarnik, Y. Ogata, D. Stoyan, and M. Tanemura. On parameter estimation for pairwise interaction processes. *Int. Statistical Review*, 62(1):99–117, Apr 1994.

[38] M. Donoser and H. Bischof. 3D segmentation by maximally stable volumes (MSVs). In *Proc. Int. Conf. Pattern Recognition*, pages 63–66, Aug 2006.

[39] S. Drot, H. Le Men, X. Descombes, and J. Zerubia. Object point processes for image segmentation. In *Proc. Int. Conf. Pattern Recognition*, volume 2, pages 913–916, Aug 2002.

[40] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, NY, USA, 1973.

[41] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, New York, NY, USA, 2nd edition, 2000.

[42] C. Eberhardt and A. Clarke. Fibre-orientation measurements in short-glass-fibre composites. Part I: automated, high-angular-resolution measurement by confocal microscopy. *Composites Science and Technology*, 61(10):1389–1400, Aug 2001.

[43] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. Ridges for image analysis. *J. Mathematical Imaging and Vision*, 4(4):353–373, Dec 1994.

[44] H.E. Exner. *Grundlagen von Sintervorgängen*. Materialkundlich-Technische Reihe. Gebrüder Borntraeger, Stuttgart, Germany, 1978.

[45] F.G.A. Faas and L.J. van Vliet. 3D-orientation space; filters and sampling. In *Proc. Scandinavian Conf. Image Analysis*, pages 457–466, Jun 2003.

[46] M. Felsberg. *Low-level image processing with the structure multivector*. PhD thesis, Christian-Albrechts-Universität zu Kiel, 2002.

[47] N.I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, Cambridge, UK, 1993.

[48] N.I. Fisher, T. Lewis, and B.J.J. Embleton. *Statistical Analysis of Spherical Data*. Cambridge University Press, Cambridge, UK, 1987.

[49] J. Fliege and U. Maier. A two-stage approach for computing cubature formulae for the sphere. Technical Report 139T, Fachbereich Mathematik, Universität Dortmund, 1996.

[50] J. Fliege and U. Maier. The distribution of points on the sphere and corresponding cubature formulae. *IMA J. Numerical Analysis*, 19(2):317–334, 1999.

[51] A.F. Frangi, W.J. Niessen, K.L. Vincken, and M.A. Viergever. Multiscale vessel enhancement filtering. In *Proc. Medical Image Computing and Computer-Assisted Intervention*, pages 130–137, Oct 1998.

[52] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(9):891–906, Sep 1991.

[53] W.T. Freeman, E.C. Pasztor, and O.W. Carmichael. Learning low–level vision. *Int. J. Computer Vision*, 40(1):25–47, Oct 2000.

[54] M. Frigo and S.G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, Feb 2005.

[55] S.Y. Fu and B. Lauke. Effects of fiber length and fiber orientation distributions on the tensile strength of short-fiber-reinforced polymers. *Composites Science and Technology*, 56(10):1179–1190, Oct 1996.

[56] T.J. Fuchs, T. Lange, P.J. Wild, H. Moch, and J.M. Buhmann. Weakly supervised cell nuclei detection and segmentation on tissue microarrays of renal clear cell carcinoma. In *Proc. 30th DAGM Symposium*, pages 173–182, Jun 2008.

[57] F. Gadala-Maria and F. Parsi. Measurement of fiber orientation in short-fiber composites using digital image processing. *Polymer Composites*, 14(2):126–131, 1993.

[58] GCC Team. Data prefetch support. Technical report, Free Software Foundation, Boston, MA, USA, Oct 2007.

[59] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, Nov 1984.

[60] J.M. Geusebroek, A.W.M. Smeulders, and J. van de Weijer. Fast anisotropic gauss filtering. In *Proc. Europ. Conf. Computer Vision*, volume 1, pages 99–112, May 2002.

[61] J.M. Geusebroek, A.W.M. Smeulders, and J. van de Weijer. Fast anisotropic gauss filtering. *IEEE Trans. Image Processing*, 12(8):938–943, Aug 2003.

[62] M. van Ginkel. *Image Analysis using Orientation Space based on Steerable Filters.* PhD thesis, Delft University of Technology, 2002.

[63] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, USA, 3rd edition, 1996.

[64] G.H. Granlund and H. Knutsson. *Signal processing for computer vision.* Kluwer Academic Publishers, Dordrecht, NL, 1995.

[65] P.J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, Dec 1995.

[66] M. Gupta and K.K. Wang. Fiber orientation and mechanical properties of short-fiber-reinforced injection-molded composites: Simulated and experimental results. *Polymer Composites*, 14(5):367–382, Oct 1993.

[67] L. Haglund, N. Guest, S. Einerman, H. Öster, P. Björkman, and H.Graf. Overall-quality assessment when targeting wide-XGA flat panel displays. Technical report, SVT Corporate Development Technology, Stockholm, SE, Apr 2002.

[68] T.P. Harrigan and R.W. Mann. Characterization of microstructural anisotropy in orthotropic materials using a second rank tensor. *J. Materials Science*, 19(3):761–767, Mar 1984.

[69] D. Hastie. *Towards automatic reversible jump Markov chain Monte Carlo*. PhD thesis, University of Bristol, 2005.

[70] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, New York, NY, USA, 2001.

[71] P.J. Hine and R. A. Duckett. Fiber orientation structures and mechanical properties of injection molded short glass fiber reinforced ribbed plates. *Polymer Composites*, 25(3):237–254, Jun 2004.

[72] M. Hurn, I. Steinsland, and H. Rue. Parameter estimation for a deformable template model. *Statistics and Computing*, 11(4):337–346, Oct 2001.

[73] M. Ide, M. Jimbo, M. Yamamoto, and O. Kubo. Tumor cell counting using an image analysis program for MIB-1 immunohistochemistry. *Neurologia Medico-Chirurfica (Tokyo)*, 37(2):158–162, Feb 1997.

[74] Optimizing applications with the Intel C++ and Fortran compilers. Intel Corp., Santa Clara, CA, USA, Jul 2005. White paper.

[75] B. Jähne. *Digital Image Processing*. Springer, Heidelberg, DE, 5th edition, 2002.

[76] J.L. Jensen and J. Møller. Pseudolikelihood for exponential family models of spatial point processes. *The Annals of Applied Probability*, 1(3):445–461, Aug 1991.

[77] T. Joachims. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999.

[78] M. Kawamura, S. Ikeda, S. Morita, and Y. Sanomura. Unambiguous determination of 3D fiber orientation distribution in thermoplastic composites using SAM image of elliptical mark and interference fringe. *J. Composite Materials*, 39(4):287–299, Feb 2005.

[79] Z. Khan, T. Balch, and F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(11):1805–1819, Nov 2005.

[80] M. Kiderlen. Non-parametric estimation of the directional distribution of stationary line and fibre processes. *Advances in Applied Probability*, 33(1):6–24, Mar 2001.

[81] M. Kiderlen and A. Pfrang. Algorithms to estimate the rose of directions of a spatial fibre system. *J. Microscopy*, 219(2):50–60, Aug 2005.

[82] J.K. Kim and S.H. Park. Fiber orientation and rheological properties of short fiber-reinforced plastics at higher shear rates. *J. Materials Science*, 35(5):1069–1078, Mar 2000.

[83] J.K. Kim and J.H. Song. Rheological properties and fiber orientations of short fiber-reinforced plastics. *J. Rheology*, 41(5):1061–1085, Sep 1997.

[84] Y.J. Kim, B.F. Romeike BF, J. Uszkoreit J, and W. Feiden. Automated nuclear segmentation in the determination of the Ki-67 labeling index in meningiomas. *Clinical Neuropathology*, 25(2):67–73, Mar–Apr 2006.

[85] Y.J. Kim, B.F.M. Romeike, and W. Feiden. Automated morphometric determination of the Ki-67 labelling-index in meningiomas: A validation-model for a fast and facile method. *Acta Neuropathologica*, 108(4):357, Oct 2004.

[86] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[87] D. Knossow, J. van de Weijer, R.P. Horaud, and R. Ronfard. Articulated-body tracking through anisotropic edge detection. In *Dynamical Vision*, volume 4358 of *Lecture Notes in Computer Science*, pages 86–99. Springer, 2007.

[88] D.E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, Reading, MA, USA, 1994.

[89] H. Knutsson. Representing local structure using tensors. In *Proc. Scandinavian Conf. Image Analysis*, pages 244–251, Jun 1989.

[90] K. Krissian, G. Malandain, N. Ayache, R. Vaillant, and Y. Trousset. Model-based detection of tubular structures in 3D images. *Computer Vision and Image Understanding*, 80(2):130–171, Nov 2000.

[91] C. Lacoste, X. Descombes, and J. Zerubia. Point processes for unsupervised line network extraction in remote sensing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(10):1568–1579, Oct 2005.

[92] J.D. Lafferty, A. McCallum, and F.C.N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. Int. Conf. Machine Learning*, pages 282–289, Jun 2001.

[93] V. Lakshmanan. A separable filter for directional smoothing. *IEEE Tran. Geoscience and Remote Sensing Letters*, 1(3):192–195, Jul 2004.

[94] S.Y. Lam and B.E. Shi. Recursive anisotropic 2-d gaussian filtering based on a triple-axis decomposition. *IEEE Trans. Image Processing*, 16(7):1925–1930, Jul 2007.

[95] C.H. Lampert, M.B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Proc. Int. Conf. Computer Vision and Pattern Recognition*, Jun 2008.

[96] C.H. Lampert and O. Wirjadi. Anisotropic gaussian filtering using fixed point arithmetic. In *Proc. Int. Conf. Image Processing*, pages 1565–1568, Oct 2006.

[97] C.H. Lampert and O. Wirjadi. An optimal non-orthogonal separation of the anisotropic Gaussian convolution filter. *IEEE Trans. Image Processing*, 15(11):3501–3513, Nov 2006.

[98] L. Latson, B. Sebek, and K.A. Powell KA. Automated cell nuclear segmentation in color images of hematoxylin and eosin-stained breast biopsy. *Analytical and Quantitative Cytology and Histology*, 25(6):321–331, Dec 2003.

[99] C. Lautensack, K. Schladitz, and A. Särkkä. Modeling the microstructure of sintered copper. In *Proc. Int. Conf. Stereology, Spatial Statistics and Stochastic Geometry*, Jun 2006.

[100] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[101] J.E. Lee, M.K. Chung, and A.L. Alexander. Evaluation of anisotropic filters for diffusion tensor imaging. In *Proc. 3rd IEEE Int. Symp. Biomedical Imaging*, pages 77–80, Apr 2006.

[102] K.S. Lee, S.W. Lee, K. Chung, T.J. Kang, and J.R. Youn. Measurement and numerical simulation of three-dimensional fiber orientation states in injection-molded short-fiber-reinforced plastics. *J. Applied Polymer Science*, 88(2):500–509, Apr 2003.

[103] Y.H. Lee, S.W. Lee, J.R. Youn, K. Chung, and T.J. Kang. Characterization of fiber orientation in short fiber reinforced composites with an image processing technique. *Materials Research Innovations*, 6(2):65–72, Sep 2002.

[104] P. Leopardi. A partition of the unit sphere into regions of equal area and small diameter. *Electronic Transactions on Numerical Analysis*, 25:309–327, 2006.

[105] M.N.M. van Lieshout. Stochastic annealing for nearest-neighbor point processes with application to object recognition. *Advances in Applied Probability*, 26(2):281–300, Jun 1994.

[106] M.N.M. van Lieshout. *Markov point processes and their applications*. Imperial College Press, London, UK, 2000.

[107] M.N.M. van Lieshout and R.S. Stoica. The Candy model: properties and inference. *Statistica Neerlandica*, 57(2):177–206, May 2003.

[108] J.S. Lim. *Two-Dimensional Signal and Image Processing*. Signal Processing Series. Prentice-Hall, 1990.

[109] B. Lin, X. Jin, R. Zheng, F.S. Costa, and Z. Fan. 3D Fiber Orientation Simulation for Plastic Injection Molding. In S. Ghosh, J.C. Castro, and J.K. Lee, editors, *American Institute of Physics Conference Series*, volume 712 of *American Institute of Physics Conference Series*, pages 282–287, Jun 2004.

[110] H.T. Lin, C.J. Lin, and R.C. Weng. A note on Platts probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267–276, Oct 2007.

[111] T. Lindeberg. Principles for automatic scale selection. Technical Report ISRN KTH/NA/P-98/14-SE, KTH Royal Institute of Technology, Stockholm, 1998.

[112] C. Lorenz, I.-C. Carlsen, T.M. Buzug, C. Fassnacht, and J. Weese. Multi-scale line segmentation with automatic estimation of width, contrast and tangential direction in 2D and 3D medical images. In *Proc. Joint Conf. Computer Vision, Virtual Reality and Robotics in Medicine and Medial Robotics and Computer-Assisted Surgery*, pages 233–242, Mar 1997.

[113] D.N. Louis, H. Ohgaki, O.D. Wiestler, and W.K. Cavenee, editors. *WHO classification of tumors of the central nervous system*. IARC Press, Lyon, France, 4th edition, 2007.

[114] C.G. Loukas, G.D. Wilson, B. Vojnovic, and A. Linney. An image analysis-based approach for automated counting of cancer cell nuclei in tissue sections. *Cytometry Part A*, 55A(1):30–42, 2003.

[115] N. Malpica, C.O. de Solorzano, J.J. Vaquero, Andres Santos, I. Vallcorba, J.M. Garcia-Sagredo, and F. del Pozo. Applying watershed algorithms to the segmentation of clustered nuclei. *Cytometry*, 28(4):289–297, Aug 1997.

[116] I. Markovic. *High-Performance Hybrid-Fibre Concrete: Development and Utilisation*. IOS Press, Amsterdam, NL, 2006.

[117] R.B. Martin and D.L. Boardman. The effects of collagen fiber orientation, porosity, density, and mineralization on bovine cortical bone bending properties. *J. Biomechanics*, 26(9):1047–1054, Sep 1993.

[118] R.B. Martin and J. Ishida. The relative effects of collagen fiber orientation, porosity, density, and mineralization on bone strength. *J. Biomechanics*, 22(5):419–426, 1989.

[119] J.J. McGrath and J.M. Willie. Determination of 3D fiber orientation distribution in thermoplastic injection molding. *Composites Science and Technology*, 53(2):133–143, 1995.

[120] D. Michie, D.J. Spiegelhalter, and C.C. Taylor, editors. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood Series in Artificial Intelligence. Ellis Horwood, 1994.

[121] J. Møller and R. P. Waagepetersen. *Statistical Inference and Simulation for Spatial Point Processes*. Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, FL, USA, 2004.

[122] V.J. Napadow, Q. Chen, V. Mai, P.T.C. So, and J. Gilbert R. Quantitative analysis of three-dimensional-resolved fiber architecture in heterogeneous skeletal muscle tissue using NMR and optical imaging methods. *Biophysical Journal*, 80(6):2968–2975, Jun 2001.

[123] A. Nedzved, S. Ablameyko, and I. Pita. Morphological segmentation of histology cell images. In *Proc. Int. Conf. Pattern Recognition*, volume 1, pages 500–503, Sep 2000.

[124] H. Netten, I.T. Young, L.J. van Vliet, H.J. Tanke, H. Vroljik, and W.C.R. Sloos. FISH and chips: Automation of fluorescent dot counting in interphase cell nuclei. *Cytometry*, 28(1):1–10, May 1997.

[125] M. Nöthe, K. Pischang, P. Ponizil, B. Kieback, and J. Ohser. Investigation of sintering processes by microfocus computer tomography ($\mu$-CT). In *Proc. DGZfP*, pages BB 84–CD, 2003.

[126] M. Nöthe, M. Schulze, R. Grupp, B. Kieback, A. Haibel, and J. Banhart. Analysis of particle rearrangement during sintering by micro focus computed tomography ($\mu$CT). In *Materials Science Forum*, volume 534–536, pages 493–496, 2007.

[127] J. Ohser. personal communication, 2008.

[128] M. Ortner, X. Descombes, and J. Zerubia. A marked point process of rectangles and segments for automatic analysis of digital elevation models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(1):105–119, Jan 2008.

[129] N. Otsu. A threshold selection method from grey-level histograms. *IEEE Trans. Systems, Man, and Cybernetics*, 9(1):62–66, Jan 1979.

[130] S. Perkins, K. Edlund, D. Esch-Mosher, D. Eads, N. Harvey, and S. Brumby. Genie pro: robust image classification using shape, texture, and spectral information. In *Proc. SPIE Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XI*, volume 5806, pages 139–148, Mar 2005.

[131] P. Perona. Steerable-scalable kernels for edge detection and junction analysis. *Image and Vision Computing*, 10(10):663–672, Dec 1992.

[132] S. Peters and A. König. Optimized texture operators for the automated design of image analysis systems: Non-linear and oriented kernels vs. gray value co-occurrence matrices. *Int. J. Hybrid Intelligent Systems*, 4(3):185–202, Aug 2007.

[133] J.C. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in large margin classifiers*. MIT Press, Cambridge, 2000.

[134] T. Pock, R. Beichel, and H. Bischof. A novel robust tube detection filter for 3D centerline extraction. In *Proc. Scandinavian Conf. Image Analysis*, pages 481–490, Jun 2005.

[135] J.G. Proakis and D.G. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ, USA, 3rd edition, 1996.

[136] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. ISBN 3-900051-07-0.

[137] A. Rack, L. Helfen, T. Baumbach, S. Kirste, J. Banhart, K. Schladitz, and J. Ohser. Analysis of spatial cross-correlations in multi-constituent volume data. *J. Microscopy*, 2008. to appear.

[138] W.S. Rasband. ImageJ. U. S. National Institutes of Health, Bethesda, Maryland, USA, 1997-2007. http://rsb.info.nih.gov/ij/.

[139] C. Redon, L. Chermant, J.L. Chermant, and M. Coster. Assessment of fibre orientation in reinforced concrete using fourier image tansform. *J. Microscopy*, 191(3):258–265, Sep 1998.

[140] K. Robb, O. Wirjadi, and K. Schladitz. Fiber orientation estimation from 3D image data: Practical algorithms, visualization, and interpretation. In *Proc. Int. Conf. Hybrid Intelligent Systems*, pages 320–325, Sep 2007.

[141] A.C. Ruifrok and D.A. Johnston. Quantification of histochemical staining by color deconvolution. *Analytical and Quantitative Cytology and Histology*, 23(4):291–299, Aug 2001.

[142] E.B. Saff and A.B.J. Kuijlaars. Distributing many points on a sphere. *The Mathematical Intelligencer*, 19(1):5–11, 1997.

[143] K. Sandau and J. Ohser. The chord length transform and the segmentation of crossing fibres. *J. Microscopy*, 226(1):43–53, Apr 2007.

[144] Y. Sato, S. Nakajima, H. Atsumi, T. Koller, G. Gerig, S. Yoshida, and R. Kikinis. 3D multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. In *Proc. Joint Conf. Computer Vision, Virtual Reality and Robotics in Medicine and Medial Robotics and Computer-Assisted Surgery*, pages 213–222, Mar 1997.

[145] K. Schladitz, S. Peters, D. Reinel-Bitzer, A. Wiegmann, and J. Ohser. Design of acoustic trim based on geometric modeling and flow simulation for non-woven. *Computational Materials Science*, 38(1):56–66, Nov 2006.

[146] M. Schöneberger. Entwicklung eines Hardcore Zylinderprozesses zur Modellierung von Fasersystemen in 3D. Master's thesis, Fachhochschule Kaiserslautern, 2008.

[147] F. Shafait. *Geometric Layout Analysis of Scanned Documents*. PhD thesis, Technische Universität Kaiserslautern, 2008.

[148] F. Shafait and T.M. Breuel. Document image dewarping contest. In *Proc. Int. Workshop Camera-Based Document Analysis and Recognition*, pages 181–188, Sep 2007.

[149] J. Sijbers, A.J. den Dekker, A. van der Linden, M. Verhoye, and D. Van Dyck. Adaptive anisotropic noise filtering for magnitude mr data – a general tool for early vision. *Magnetic Resonance Imaging*, 17(10):1533–1539, Dec 1999.

[150] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proc. Int. Conf. Document Analysis and Recognition*, pages 958–962, Aug 2003.

[151] P.J. Sjöström, B.R. Frydel, and L.U. Wahlberg. Artificial neural network-aided image analysis system for cell counting. *Cytometry*, 36(1):18–26, May 1999.

[152] C.O. de Solorzano, A. Santos, I. Vallcorba, J.M. Garcia-Sagredo, and F. del Pozo. Automated FISH spot counting in interphase nuclei: Statistical validation and data correction. *Cytometry*, 31(2):93–99, 1998.

[153] C. Soutis. Carbon fiber reinforced plastics in aircraft construction. *Materials Science and Engineering: A*, 412(1-2):171–176, Dec 2005.

[154] R. Stoica, X. Descombes, and J. Zerubia. A gibbs point process for road extraction from remotely sensed images. *Int. J. Computer Vision*, 57(2):121–136, May 2004.

[155] D. Stoyan, W.S. Kendall, and J. Mecke. *Stochastic Geometry and its Applications*. John Wiley & Sons, Chichester, UK, 2nd edition, 1995.

[156] J. Summerscales, editor. *Non-Destructive Testing of Fibre-Reinforced Plastics Composites*, volume 2. Elsevier Science Publishers, Barking, UK, 1990.

[157] K. Takayama, T. Igarashi, R. Haraguchi, and K. Nakazawa. A sketch-based interface for modeling myocardial fiber orientation. In *Proc. Int. Symp. Smart Graphics*, pages 1–9, Jun 2007.

[158] D.S. Taubman and M. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, Dordrecht, NL, 2002.

[159] N. Theera-Umpon and P.D. Gader. System-level training of neural networks for counting white bloodcells. *IEEE Trans. Systems, Man and Cybernetics C*, 32(1):48–53, Feb, 2002.

[160] K. Tomita, N. Sakuma, and T. Mama. Method and device for determining fiber orientation of paper, and apparatus for removing image forming substance from paper). US Patent #5729349, 1998.

[161] S. Torquato. *Random Heterogeneous Materials*. Interdisciplinary applied mathematics. Springer, New York, NY, USA, 2002.

[162] B. Triggs and M. Sdika. Boundary conditions for Young-van Vliet recursive filtering. *IEEE Trans. Signal Processing*, 54(6):2365–2367, Jun 2006.

[163] C.L. Tucker and S.G. Advani. Processing short-fiber systems. In S.G. Advani, editor, *Flow and Rheology in Polymer Composites Manufacturing*, Composite Materials. Elsevier, Amsterdam, NL, 1994.

[164] R. Waagepetersen and D. Sorensen. A tutorial on reversible jump MCMC with a view towards applications in QTL-mapping. *Int. Statistical Review*, 69(1):49–61, Apr 2001.

[165] H.M. Wallach. Conditional random fields: An introduction. Technical Report MS-CIS-04-21, Department of Computer and Information Science, University of Pennsylvania, 2004.

[166] A. Wallack and D. Manocha. Robust algorithms for object localization. *Int. J. Computer Vision*, 27(3):243–262, May 1998.

[167] Y. Watanabe and T. Fujii. Experimental study on magnetic torque measurement to estimate fiber orientation in fe-fiber-reinforced composites. *Japanese J. Applied Physics*, 42(4A):L391–L393, Apr 2003.

[168] J. Weickert. Theoretical foundations of anisotropic diffusion in image processing. *Computing. Supplementum*, 11:221–236, 1996.

[169] C.F. Westin, A. Bhalerao, R. Kikinis, and H. Knutsson. Using local 3D structure for segmentation of bone from computer tomography images. In *Proc. Int. Conf. Computer Vision and Pattern Recognition*, pages 794–800, Jun 1997.

[170] C.F. Westin, S.E. Maier, H. Mamata, A. Nabavi, F.A. Jolesz, and R. Kikinis. Processing and visualization of diffusion tensor MRI. *Medical Image Analysis*, 6(2):93–108, Jun 2002.

[171] O. Wirjadi, T.M. Breuel, W. Feiden, and Y.J. Kim. Automated feature selection for the classification of meningioma cell nuclei. In *Bildverarbeitung für die Medizin*, Informatik aktuell, pages 76–80. Springer, 2006.

[172] O. Wirjadi, A. Jablonski, K. Schladitz, and M. Nöthe. Volumetric analysis of a sinter process in time. In *Proc. 27th DAGM Symposium*, pages 409–416, Aug 2005.

[173] Wolfram Research, Inc. Mathematica, version 6.0. Champaign, IL, USA, 2007.

[174] R. Wootton, D.R. Springall, and J.M. Polak, editors. *Image Analysis in Histology: Conventional and Confocal Microscopy*. Postgraduate Medical Science. Cambridge University Press, Cambridge, UK, 1995.

[175] G.Z. Yang, P. Burger, D.N. Firmin, and S.R. Underwood. Structure adaptive anisotropic image filtering. *Image and Vision Computing*, 14(2):135–145, Mar 1996.

[176] H. Yang and W.B. Lindquist. Three-dimensional image analysis of fibrous materials. In *Proc. SPIE Applications of Digital Image Processing XXIII*, volume 4115, pages 275–282, Dec 2000.

[177] I.T. Young and L.J. van Vliet. Recursive implementation of the Gaussian filter. *Signal Processing*, 44(2):139–151, Jun 1995.

[178] I.T. Young, L.J. van Vliet, and M. van Ginkel. Recursive Gabor filtering. *IEEE Trans. Signal Processing*, 50(11):2798–2805, Nov 2002.

[179] T. Yu and Y. Wu. Collaborative tracking of multiple targets. In *Proc. Int. Conf. Computer Vision and Pattern Recognition*, volume 1, pages 834–841, Jun 2004.

[180] T. Zhao and R. Nevatia. Tracking of multiple humans in crowded environment. In *Proc. Int. Conf. Computer Vision and Pattern Recognition*, volume 2, pages 406–413, Jun 2004.