Helsinki University of Technology Laboratory of Computational Engineering Publications
Teknillisen korkeakoulun Laskennallisen tekniikan laboratorion julkaisuja
Espoo 2006 REPORT B56

STUDIES IN PROBABILISTIC METHODS FOR SCENE ANALYSIS

Timo Kostiainen

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

Helsinki University of Technology Laboratory of Computational Engineering Publications
Teknillisen korkeakoulun Laskennallisen tekniikan laboratorion julkaisuja
Espoo 2006 REPORT B56

STUDIES IN PROBABILISTIC METHODS FOR SCENE ANALYSIS

Timo Kostiainen

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Electrical and Communications Engineering, Helsinki University of Technology, for public examination and debate in Auditorium E at Helsinki University of Technology (Espoo, Finland) on the 27th of October, 2006, at 12 noon.

Helsinki University of Technology
Department of Electrical and Communications Engineering
Laboratory of Computational Engineering

Teknillinen korkeakoulu
Sähkö- ja tietoliikennetekniikan osasto
Laskennallisen tekniikan laboratorio

Distribution:
Helsinki University of Technology
Laboratory of Computational Engineering
P. O. Box 9400
FIN-02015 HUT
FINLAND
Tel. +358-9-451 4826
Fax. +358-9-451 4830
http://www.lce.hut.fi

E-mail: Timo.Kostiainen@hut.fi

ISBN-13 978-951-22-8411-5 (printed)
ISBN-10 951-22-8411-1 (printed)
ISBN-13 978-951-22-8412-2 (PDF)
ISBN-10 951-22-8412-X (PDF)
ISSN 1455-0474
Picaset Oy
Espoo 2006

HELSINKI UNIVERSITY OF TECHNOLOGY P. O. BOX 1000, FI-02015 TKK http://www.tkk.fi	ABSTRACT OF DOCTORAL DISSERTATION

Author Timo Kostiainen

Name of the dissertation

STUDIES IN PROBABILISTIC METHODS FOR SCENE ANALYSIS

Date of manuscript 15. 5. 2006	Date of the dissertation 27. 10. 2006
[X] Monograph	[ ] Article dissertation (summary + original articles)

Department	Department of Electrical and Communications Engineering
Laboratory	Laboratory of Computational Engineering
Field of research	Computational Engineering
Opponent(s)	Prof. Janne Heikkilä and Dr. Pasi Koikkalainen
Supervisor (Instructor)	Prof. Jouko Lampinen

Abstract

In this thesis, probabilistic methods are applied to a number of problems in computer vision. The goal is to provide means for a vision based system that is able to analyze and recognize scenes and objects in camera images and to use that information for autonomous navigation and machine learning. New methods are developed for different functions that are needed in such a system, including segmentation of images, model-based recognition of objects, robot navigation and model complexity control.

The approach is based on generative probability models, and Bayesian statistical inference is used to match these models with image data. Stochastic sampling methods are applied to obtain numerical results.

The self-organizing map is a neural network algorithm that has many applications in computer vision. In this thesis, the algorithm is analyzed in a probabilistic framework. A probability density model is derived and new model selection techniques are proposed, which enable complexity control for the self-organizing map.

The analysis of images is discussed from the point of view of segmentation and object recognition. Segmentation aims at dividing the image into parts of different appearance, while object recognition is meant to identify objects that fulfill given criteria. These are different goals, but they complement each other. When the recognition of all objects in an image is not possible, segmentation can provide an explanation to the rest of the image. For object recognition, different two and three dimensional object models are considered and Bayesian matching techniques are applied to them. Efficient techniques for image segmentation are proposed and results are presented.

Keywords scene analysis, MCMC methods, image segmentation, object matching, self-organizing maps

ISBN-13 (printed) 978-951-22-8411-5	ISSN (printed) 1455-0474
ISBN-10 (printed) 951-22-8411-1	ISSN (pdf)
ISBN-13 (pdf) 978-951-22-8412-2	
ISBN-10 (pdf) 951-22-8412-X	Number of pages 88

Publisher Laboratory of Computational Engineering

Print distribution Laboratory of Computational Engineering

[X] The dissertation can be read at http://lib.tkk.fi/Diss/

TEKNILLINEN KORKEAKOULU PL 1000, 02015 TKK http://www.tkk.fi	VÄITÖSKIRJAN TIIVISTELMÄ

Tekijä Timo Kostiainen

Väitöskirjan nimi
STUDIES IN PROBABILISTIC METHODS FOR SCENE ANALYSIS

Käsikirjoituksen jättämispäivämäärä 15. 5. 2006	Väitöstilaisuuden ajankohta 27. 10. 2006
[X] Monografia	[ ] Yhdistelmäväitöskirja (yhteenveto + erillisartikkelit)

Osasto	Sähkö- ja tietoliikennetekniikan osasto
Laboratorio	Laskennallisen tekniikan laboratorio
Tutkimusala	Laskennallinen tekniikka
Vastaväittäjä(t)	prof. Janne Heikkilä ja TkT Pasi Koikkalainen
Työn valvoja	prof. Jouko Lampinen
(Työn ohjaaja)	

Tiivistelmä

Tässä väitöskirjassa sovelletaan todennäköisyyslaskennan menetelmiä eräisiin tietokonenäköongelmiin. Työn tarkoituksena on tuottaa keinoja näköön perustuvaan järjestelmään, joka voi analysoida ja tunnistaa näkymiä ja kohteita kamerakuvista ja käyttää näin saatua informaatiota itsenäiseen navigointiin ja koneoppimiseen. Työssä kehitetään uusia menetelmiä järjestelmän tarvitsemiin toimintoihin kuten kuvien segmentointiin, mallipohjaiseen kohteiden tunnistukseen, robottinavigointiin ja mallien kompleksisuuden hallintaan.

Työssä käytettävä lähestymistapa perustuu generatiivisiin todennäköisyysmalleihin, ja mallit sovitetaan kuvadataan bayesiläistä tilastollista päättelyä soveltaen. Numeeristen tulosten saamiseksi käytetään stokastisia poimintamenetelmiä.

Itsejärjestyvä kartta on neuroverkkoalgoritmi, jolla on useita tietokonenäköalan sovelluksia. Tässä työssä algoritmia analysoidaan todennäköisyyspohjaisesti. Algoritmin tuottamalle mallille johdetaan todennäköisyysjakaumamalli ja sille esitetään uusia mallinvalintamenetelmiä, jotka mahdollistavat itsejärjestyvän kartan kompleksisuuden hallinnan.

Kuvien analysointia käsitellään sekä segmentoinnin että kohteiden tunnistuksen näkökulmasta. Segmentoinnissa kuva jaetaan erilaisilta näyttäviin osiin. Kohteiden tunnistus perustuu niiden ennalta tunnettuihin ominaisuuksiin. Tavoitteet ovat siten varsin erilaisia, mutta ne täydentävät toisiaan. Silloin kun vain osa kuvassa olevista kohteista pystytään tunnistamaan, segmentoinnilla voidaan saada kuvan muille osille selitys. Väitöskirjassa esitetään laskennallisesti tehokkaita menetelmiä kuvien segmentointiin. Kohteiden tunnistusta kaksi- ja kolmiulotteisten mallien avulla tarkastellaan bayesiläisiä menetelmiä käyttäen.

Asiasanat	näkymäanalyysi, MCMC, kuvasegmentointi, kohteiden sovitus, itsejärjestyvä kartta		
ISBN-13 (painettu)	978-951-22-8411-5	ISSN (painettu)	1455-0474
ISBN-10 (painettu)	951-22-8411-1	ISSN (pdf)	
ISBN-13 (pdf)	978-951-22-8412-2		
ISBN-10 (pdf)	951-22-8412-X	Sivumäärä	88
Julkaisija	Laskennallisen tekniikan laboratorio		
Painetun väitöskirjan jakelu	Laskennallisen tekniikan laboratorio		
[X] Luettavissa verkossa osoitteessa http://lib.tkk.fi/Diss/			

Preface

This thesis is the result of research work carried out in the Laboratory of Computational Engineering at Helsinki University of Technology during the years 2000–2006.

Most of all I am grateful to my supervisor prof. Jouko Lampinen for his inspiring guidance and relentless support. I would like to thank the people in my research group who I've had the pleasure of collaborating with in research projects and who have helped in many ways. Also, I would like to thank everybody in the laboratory for contributing to a cheerful and creative work environment.

Finally, I would like to thank my wife Anna-Maija for supporting me all this time.

Timo Kostiainen

Contents

Chapter 1

Introduction

Vision is the human sense that provides the highest bandwidth connection between the brain and the outside world. Vision is required in many activities performed by humans. For example, most things designed for man, from tools to built environments, rely on the ability to receive visual information. Therefore the ability to imitate human vision is required in automating a great number of routine tasks. Pictures and images are an essential method of conveying information. The amount of image material in digital form in different archives and databases is huge and constantly increasing. There is a strong need for automatic annotation and indexing of images to take full advantage of these resources.

Images contain information about objects in a very indirect form, and many noise sources are involved. The properties of objects cannot easily be defined unambiguously, and the conditions in which the images are captured are also subject to many sources of variability. The objects that automated vision systems may be required to recognize can also vary in many ways. The use of probability distributions is an effective way to handle noisy data, and the variability in objects can be modeled with probabilistic methods. Moreover, one image can only contain a limited amount of information, and there may be several possible explanations for one image. This means that the output of a vision system cannot always be correct. In order for the system to be useful, the output should be accompanied with an estimate of its confidence. Probabilistic methods are suitable for such problems.

Recent development of computers has enabled the use of many computationally demanding numerical tools such as Markov Cain Monte Carlo techniques to the inference of complicated probability models [100]. These methods have found numerous applications in various machine learning problems [1].

Scene analysis aims at producing an interpretation to the contents of a visual image. All parts of an image can be thought to represent some object, so if all the objects are recognized, the image is fully divided into separate parts, or segments.

However, it is unrealistic to assume that all objects in an image could always be recognized because of many disturbances such as partial visibility and variable illumination. In addition, a vision system should also be able to deal with unfamiliar objects that are not previously known to it. Therefore a reasonable goal for scene analysis is to recognize familiar objects and to classify the rest of the image into background. Both segmentation and object recognition are needed in that task.

The self-organizing map (SOM) is an important neural network algorithm, which has applications in many areas of information processing, including computer vision and pattern recognition. Among its most intense application areas are image and video processing and pattern recognition [50, 90]. Unlike many other models and methods in computer vision, the SOM is not defined based on probability theory. The SOM is an unsupervised learning method, which is why its performance in data processing tasks cannot be directly evaluated. This has made it difficult to compare the SOM with other types of methods. More importantly, there has been a lack of quantitative tools for the basic task of selecting suitable model parameters.

This thesis presents studies on the use of probabilistic models in selected scene analysis tasks, including the segmentation of images and the detection and matching of different types of objects. The models and methods are based on Bayesian inference. Monte Carlo sampling methods such as Markov chain Monte Carlo (MCMC) are used in numerical computations to determine model parameters from image data. In addition, a probabilistic interpretation is derived for the self-organizing map algorithm, and its use for probabilistic model selection is discussed. The work is organized as follows:

Chapter two contains a review of probabilistic inference and Bayesian methods that are used throughout this thesis. Monte Carlo sampling methods that are used to produce numerical results are also described.

In Chapter three, the segmentation of scenes is discussed. An MCMC image segmentation algorithm is presented. Its basic principle has been proposed previously. The aim is to present new sample generation techniques and demonstrate their efficiency. Different application examples are also discussed. Most of the results of this chapter have been published in publications [64, 63].

In Chapter four, the application of probabilistic sampling methods to model-based object detection and matching are studied. The goal is to explore the possibilities of Monte Carlo sampling techniques in the matching of two and three dimensional object models to image data. Most of the results of this chapter have been published in publication [59].

Chapter five presents the probability density model that can be associated with the self-organizing map algorithm. The aim is to derive the density model, discuss its properties and how it can be used for model complexity selection. The results have been published in publications [66, 60, 61, 62].

The author's major contributions to this thesis can be summarized as follows:

- new methods for generating proposal samples for Markov chain Monte Carlo (MCMC) image segmentation: a method to alter segment outlines and a method to subdivide segments; integration of these methods into the reversible jump MCMC segmentation framework and numerical simulations to demonstrate their value
- a new method for map building and navigation for a mobile robot, based on the MCMC segmentation techniques
- A new method for matching a model of two dimensional rectangles to images using MCMC techniques
- two different models for 3-D objects and their matching to images using MCMC methods
- a solution to normalize the probability density model for the self-organizing map (SOM) model
- analysis and description of properties of the SOM density model; new methods for model selection for the SOM.

Chapter 2

Probabilistic inference and Monte Carlo sampling

This chapter contains a review of the Bayesian approach to probabilistic modeling and computational methods that can be used in its practical applications. The focus is on applying the methods to computer vision problems.

2.1 Statistical models and probabilistic inference

Statistical models are basic tools in machine learning and computer vision. For example, models that describe the properties of object classes are used in object recognition, and models of image statistics are used for classification of images or parts of images [46]. These models can be based on collections of data or they can be explicitly designed. Statistical distributions are a natural way of describing many phenomena.

A generative probability model for data describes a probability distribution that is thought to be able to generate the data. Such a model may accurately describe the process that actually generated the data, but approximate models are often used. For example, the normal distribution is used to approximate many natural phenomena. Generative models are in contrast to discriminative models, where the emphasis is on the models' ability to describe differences between different classes of data samples. Discriminative models are most useful in specific classification problems that can be narrowly defined [111].

A generative probability model describes the joint distribution of observed data and all variables that are involved in the assumed generation process [8]. Some of these variables may be unobserved, and often the inference task is to estimate values for some of the unobserved variables. In computer vision, the

generative model for an image may be described by the probability distribution

$$p(I,\theta|M) = p(I|\theta,M)p(\theta|M), \tag{2.1}$$

where I is the observed image data, M is the model for the process that is assumed to explain how the image was produced, and θ represents the parameters values of the model M. In the hand right side of the equation, the distribution is represented as the product of three terms, the first of which is the likelihood of the image data given the model and its parameters. The second term is the prior distribution of the parameters θ given the model M, and the last term $p(M)$ is the probability distribution that represents all the assumptions that are included in the model M. Here θ is the unobserved part of the model. For example, if M were a model for a 3-D object that appears in the image I, the parameters θ would include the intrinsic parameters that control the shape and reflectance properties of the object, as well as parameters that determine the object's pose and illumination conditions.

A basic solution to inferring the unobserved parameters is the maximum likelihood (ML) principle, where the likelihood of observed data $p(I|\theta,M)$ is maximized with respect to θ [104]. If different parameter values are not considered to be equally probable *a priori*, or before observing the data, this is expressed in the prior distribution $p(\theta|M)$. Then, instead of the likelihood function, one maximizes the product of the likelihood and the prior

$$p(I|\theta,M)p(\theta|M). \tag{2.2}$$

This is the *maximum a posteriori* (MAP) estimate of θ. Thus if the prior distribution distribution $p(\theta|M)$ is uniform, the MAP estimate is reduced to the maximum likelihood estimate. These methods result in point estimates of θ, that is, the solution is a single parameter vector that maximizes the criterion.

2.1.1 Bayesian inference

Bayesian inference is based on defining the joint probability space for all variables, both observed and unobserved, and inferring the conditional distribution of the variables of interest, given the observations [8]. The aim is to estimate the entire conditional distributions of the variables of interest instead of point estimates. Various statistics, such as the mean and confidence intervals, can be obtained from these distributions [34].

Continuing from the object recognition example in the previous section, the quantity of interest is $p(\theta|I,M)$, the posterior distribution of process parameters given the observed image and model assumptions. In other words, we want to find out what can be said about θ based on the observation I. This quantity is given by Bayes' rule as follows:

$$p(\theta|I,M) = \frac{p(I,\theta|M)}{p(I|M)} = \frac{p(I|\theta,M)p(\theta|M)}{\int p(I,\theta|M)d\theta}, \tag{2.3}$$

where the denominator is the total probability of the observed data I given the model assumptions M, marginalized over θ.

The Bayesian framework enables inference in complicated model structures, an important example of which is a hierarchical model of the form

$$p(I,\theta,\alpha|M) = p(I|\theta,\alpha,M)p(\theta|\alpha,M)p(\alpha|M). \tag{2.4}$$

This model differs from the model (2.1) in that it includes one additional unknown parameter α. The parameter α is a hyperparameter that controls the parameter θ, giving rise to a hierarchical model structure [38]. Bayesian inference of θ from this model involves integration of the model over α to obtain the marginal posterior distribution $p(\theta|I,M)$.

2.1.2 Model comparison

The above discussion is about fitting the parameters of a given model to data. Another essential part of statistical inference is to select the right model for each problem. This subject has been discussed by many authors in statistical literature [10, 8]. The problem of model selection is involved in the choice of model hyperparameters that control the complexity of the model, such as regularization parameters for neural network models. Another case is the choice between different types of models, for example whether to fit a polynomial or exponential function to a given dataset.

In likelihood based methods the basic approach to the comparison of models is based on measuring the fit of an independent test data set to each model after fitting it to training data. The purpose is to find a model that, based on the available data sample, will best generalize to the population that the data sample represents. Methods such as cross-validation and bootstrapping have been developed to make efficient use of the data without having to rely on just one division of samples for training and testing [25].

In contrast to likelihood methods, the Bayesian framework automatically encompasses mechanisms for model selection [76]. The concept of estimating posterior distributions instead of point estimates leads to model averaging. This means that all possible outcomes are taken into account instead of using just the one that is most likely. An example is the posterior distribution $p(\theta|I,M)$ computed from (2.4) as the integral

$$p(\theta|I,M) = \int p(\theta|I,\alpha,M)p(\alpha|I,M)d\alpha, \tag{2.5}$$

where $p(\theta|I,\alpha,M)$ is the posterior conditional on α, and $p(\alpha|I,M)$ represents the posterior probabilities of different values of α.

When considering a number of competing models $M_i (i = 1, 2, \ldots)$, it is often convenient to choose just one of them instead of averaging. Models can be compared by the evidence framework [76]. The evidence of model M_i of the form (2.1), given the data I, is defined as

$$p(I|M_i) = \int p(I|\theta, M_i) p(\theta|M_i) d\theta \ , \tag{2.6}$$

where the parameters θ are integrated out or marginalized over. In the case of a hierarchical model (2.4), the evidence in marginalized over the hyperparameters, too. The evidence is the same as the normalization constant in the denominator of equation (2.3). This means that although the posterior probabilities can be compared as a function of different parameter values, different models can only be compared by computing this integral.

2.1.3 Bayesian methods in practice

Above we have reviewed the theory of Bayesian inference. The challenging part of Bayesian methods is their application to practice. Apart from textbook examples, it is rarely possible to compute the posterior distributions analytically. Practical solutions are based on numerical approximation methods [3, 87]. These methods are often computationally expensive, and this is why they have only become realistic recently. Nonetheless, the comparison of all possible models and combinations of parameter values is an extremely ambitious goal, and many shortcuts must be taken.

A problem that is typically arises is the need to evaluate high-dimensional integrals. Computation of marginal distributions over parameters and hyperparameters is one such situation. Also, Bayesian analysis produces estimates of the posterior distributions of target quantities. To compute their expectations, one has to integrate over these distributions. For example, the posterior mean of the parameter θ is computed as the integral

$$E[\theta|I, M] = \int \theta p(\theta|I, M) d\theta \ . \tag{2.7}$$

In high-dimensional problems, these integrals are almost always analytically intractable. One possible solution is to approximate the distributions by simple functions that can be treated analytically, such as Gaussian functions for Laplace approximations, but this may not be sufficiently accurate if the distributions are multimodal or otherwise complicated [53]. In variational Bayes methods, the posterior distributions are approximated by the product of data-dependent and parameter dependent terms. Such methods have been developed for several types of problems [3, 37]. Another solution, which has wider applicability, is the numerical approximation of the integrals based on Monte Carlo integration [38]. In this thesis, the focus is on Monte Carlo methods.

2.2 Monte Carlo sampling

Monte Carlo integration is a stochastic technique for the approximation of integrals by drawing random samples from a distribution and evaluating the integrand in those sample points. The expected value of the function $f(\theta)$ of the random variable θ, which follows the distribution $\pi(\theta)$, can be approximated as

$$E[f(\theta)] = \int f(\theta)\pi(\theta)d\theta \approx \frac{1}{n}\sum_{i}^{n} f(\theta_i) \tag{2.8}$$

using n independent random samples θ_i from the distribution $\pi(\theta)$ [87].

In Bayesian data analysis, the target distribution is the posterior distribution of the quantity that is being estimated. In the Monte Carlo approach, that distribution is represented as a collection of samples. The problem is to find methods to generate of a representative set of samples in a computationally efficient manner.

Rejection sampling

Rejection sampling is a basic solution to produce samples from the distribution $\pi(\theta)$. It is based on using a proposal distribution $q(\theta) \geq \pi(\theta)$, which can be directly sampled from and which does not need to be normalized. The proposal samples θ_i that follow $q(\theta)$ are accepted with the probability $\pi(\theta_i)/q(\theta_i)$. Sampling is efficient when only a small proportion of samples are rejected, which means that $q(\theta)$ should resemble $\pi(\theta)$ as closely as possible [38].

Importance sampling

Importance sampling is an alternative sampling method, where samples are associated with weights. The only restriction for the proposal distribution $q(\theta)$ is that it has to be positive whenever $\pi(\theta)$ is positive. The parameter space to explore may be sparse such that most of it is occupied by very small probabilities. If the rare areas of high probabilities can be identified, the proposal distribution can be chosen to concentrate on them, which can greatly increase the efficiency of sampling. In the sampling procedure, all samples θ_i from the proposal distribution are accepted and are they are assigned with weights according to the ratio $w_i = \pi(\theta_i)/q(\theta_i)$.

2.2.1 Markov chain Monte Carlo

In high dimensional and complex problems, it can be very difficult to find proposal distributions that are suitable for importance or rejection sampling methods. In Markov chain Monte Carlo (MCMC) sampling, this problem is circumvented

by designing a sampling chain that converges to the target distribution. The individual samples can then be generated from simpler proposal distributions.

In MCMC sampling, Markov chains are used to simulate random walk in the space that converges to the target distribution $\pi(\theta)$. In a Markov chain, each state θ^t depends only on the previous state, and the consecutive states are related by a transition (or proposal) distribution $j_t(\theta^t|\theta^{t-1})$. Markov chain simulation provides efficient means of producing samples from arbitrary distributions. The MCMC concept has inspired a great amount of research and numerous algorithms and their variates have been proposed. Below, we review MCMC algorithms that have relevance in later chapters of this thesis. More thorough treatments of the subject can be found for example in books by Gilks et al. [38] and Liu [71].

Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm [42] produces a random sequence which converges to the target distribution $\pi(\theta)$. One step in the algorithm consists of sampling a point θ^* from a proposal distribution $j_t(\theta^*|\theta^t)$, and accepting θ^* in the chain as a new point with probability

$$A(\theta^t \rightarrow \theta^*) = \min\left\{1, \frac{\pi(\theta^*)\, j(\theta^t|\theta^*)}{\pi(\theta^t)\, j(\theta^*|\theta^t)}\right\}. \tag{2.9}$$

The proposal distribution j_t has to be able to eventually reach all states with a finite probability [34]. The form of the proposal distribution is essential for fast convergence of the algorithm. Finding good proposal distributions can be very difficult in practice. In the basic form of the algorithm, the proposal distribution is not adapted to the target distribution, which means that the proposed samples are very much random. This may lead to very slow convergence, in many applications, especially in high dimensions.

Other MCMC algorithms

Many advances have been made to improve the slow convergence of the Metropolis-Hastings algorithm by developing more complicated sampling schemes.

The *Gibbs sampler* picks samples from conditional distributions of the target distribution $\pi(\theta)$. Each iteration cycles through all the components of θ in random order. At each step in the cycle, the new state θ^t is the previous state θ^{t-1} with the component $\theta^{t-1}_{\{j\}}$ substituted with a sample from the conditional distribution $\pi(\theta_{\{j\}}|\theta^{t-1}_{\{1\ldots d\setminus j\}})$. So the components of θ are updated one at a time. Since the samples are taken directly from the conditional posterior density, all samples are always accepted. In order for Gibbs sampling to work, sampling from all conditional distributions of $\pi(\theta)$ must be possible. Then it is not necessary to know

the target distribution explicitly. The fact that the conditional distributions are univariate can be used for efficient sampling.

Hybrid Monte Carlo [23] and *slice sampling* [88] are other important MCMC sampling methods. They are both based on using auxiliary variables to enable sampling one variable at a time from a multivariate distribution, and their purpose is to facilitate adaptation of the sampling chain to the target distribution. Hybrid Monte Carlo uses information of the gradient of the target distribution. Slice sampling is a special strategy to sample from a univariate distribution, which enables choosing the step size of the sample chain adaptively.

Reversible jump MCMC

The sampling techniques discussed above for estimating one probability distribution, such as the posterior distribution of model parameters (2.3). To compare different models defined in different parameter spaces, the same analysis must be carried out for each different model, and after that the models can be compared, for example using the evidence method. If there are many models to compare, the computational cost may often become prohibitively large.

Green [39] has introduced a strategy that enables simultaneous sampling in different parameter spaces in some special situations. In other words, the posterior distribution over the parameters of many models can be computed using a single MCMC sampling chain. This is meaningful in cases where some of the parameters are shared, or have the same meaning, in all parameter spaces considered. The advantage is that the shared subspaces only need to be estimated once. An example of this is image segmentation, where the number of segments is not known. Different solutions for some part of the image can be tested while keeping other parts constant. Estimating the number of mixture components in a mixture density model [98] or the number of hidden units in a radial basis function network [44] are other examples of applications for the reversible jump MCMC algorithm.

In the algorithm, the communicating spaces need to be extended to enforce a match between the dimensions. Consider a move from the state of the Markov chain (k, θ^k) of dimension n_k to a new state $(k', \theta^{k'})$ that has dimension $n_{k'}$. The matching can be done by introducing auxiliary variables $\mathbf{u}$ and $\mathbf{u}'$ with dimensions $|\mathbf{u}|$ and $|\mathbf{u}'|$ and an invertible function f, such that $(\theta^{k'}, \mathbf{u}') = f(\theta^k, \mathbf{u})$ and $n_k + |\mathbf{u}| = n_{k'} + |\mathbf{u}'|$. The acceptance probability for the move is given by

$$A(k \to k') = \frac{\pi(\theta^{k'})}{\pi(\theta^k)} \frac{j(k' \to k)}{j(k \to k')} \frac{q'(\mathbf{u})}{q(\mathbf{u}')} \left| \frac{\partial(\theta^{k'}, \mathbf{u}')}{\partial(\theta^k, \mathbf{u})} \right|, \tag{2.10}$$

where j represents the proposal probabilities of the opposite moves and the auxiliary variables are generated from the distributions q and q'. The last product term is the Jacobian of the variable exchange function f.

Data driven proposal distributions

MCMC sampling is a top-down process. Samples that are generated are evaluated by computing their posterior probabilities. The computational complexity is proportional to the number of samples needed to sufficiently represent the target distribution. In general, the number of samples is exponentially related to the dimensionality of the problem's parameter space – this is known as the curse of dimensionality, and the the sampling techniques described above cannot escape it. In high-dimensional problems, it is essential to bring as much information into the sampling process as possible to speed it up. An example of this is the use of gradient information with hybrid Monte Carlo. In many complex problems, including image analysis, careful adaptation of the sampling techniques using domain specific knowledge and heuristics can produce crucial advantages [1].

In MCMC methods, a key issue is efficient sample generation. High dimensional distributions rarely consist of one smooth concentration in the space. Instead, they can often be characterized by narrow peaks that are broadly scattered. Basic MCMC algorithms, such Metropolis-Hastings with a symmetrical proposal distribution, search for these peaks by trial and error without direction, and all the information is taken from the target function. Data driven techniques are a special way of taking advantage of domain specific information by adapting proposal distributions. The aim is to concentrate sampling on interesting parts of the parameter space by using special cues in the design of the proposal distributions. The use of these cues in sample generation can be seen as bottom-up information processing, in contrast to the evaluation of the samples' posterior probabilities based on the model. Data driven MCMC methods have been successfully applied to image segmentation [17, 108] and in statistics to sampling from Dirichlet mixture models [47].

Chapter 3

Scene segmentation

In this chapter, we discuss the segmentation of scenes into different parts. We review previous research on segmentation techniques and the analysis of texture features, which are the basis of segmentation. We discuss an MCMC image segmentation method in detail and present new sample generation techniques for it. We also present example results in different applications.

3.1 Introduction

A visual scene consists of objects. The visual input captured by a camera or the retina of the eye is either directly emitted by objects or reflected from them. Depending on the purpose of scene analysis, not all the objects are equally interesting. Often it is convenient to divide the scene into foreground objects and a less interesting background.

By image segmentation we mean the division of an image into segments that are visually different. This is obviously a vague description, but there is no theory for segmentation and no mathematically accurate definition can be given. For the purposes of scene analysis, an ideal segment division would separate different objects from the scene. This is also the way that human intuition treats segmentation. Unfortunately, the current status of research in this area indicates that human-like segmentation results cannot be achieved without the recognition of objects and understanding of the 3-D structure of the scene. However, segmentation is necessary for image understanding, and it is possible to get results where most objects are correctly segmented without knowledge of what the objects are. The lack of a theoretic basis also makes it difficult to quantify and compare the results of segmentation.

A traditional view of scene analysis presented by Marr regards the problem as a sequential bottom-up task where detail and accuracy increase at each level of processing from the extraction of elementary features toward object recognition

and interpretation of the scene [79]. According to this view, segmentation of the image will be among the first low-level tasks. More recently, the relevance of feedback mechanisms has been recognized. Most authors currently believe that it is important to combine both simple visual cues on the low-level and the feedback from complex models on a higher level [55, 118]. This means that segmentation and the recognition of objects should be joined into a concurrent process. In any case, image analysis requires some type of a starting point, and that is what coarse segmentation of the image provides.

3.2 Related work

Image segmentation has been studied for a long time and a wide variety of different techniques have been published [92]. Various nonparametric segmentation techniques have been developed. Clustering and thresholding of the intensity histogram [70, 18] is a basic solution. Inclusion of connectivity constraints for the segments [93] is essential for many applications. The algorithm based on mean shift filtering [19] falls in the same category. The normalized cuts algorithm [101] optimizes a graph theoretic criterion to find an optimal partitioning of the image into segments.

Apart from homogeneity, an important property of segments are their continuous boundaries, which appear as edges in the images. Many publications have focused on using the edge information. Active contours [52] have been used for separating a single object from its background. To determine the optimal contour, variational minimization of the Mumford-Shah energy functional [85] may be used. This approach has been applied especially in the segmentation of graylevel medical images such as X-ray and magnetic resonance images. Ma and Manjunath have proposed a method to recover boundaries of both color and texture regions [75].

3.2.1 Bayesian image segmentation

Bayesian probabilistic methods have been applied in many authors to image segmentation. The problem is one of finding an optimal decision between two opposite objectives: homogeneous segments and a simple segmentation [92]. It is obvious that the segments should be homogeneous, but in order for the segmentation to be informative, it must consist of a relatively small number of segments. Bayesian methods are suitable for such decision tasks. Many authors have used Bayesian methods aimed at MAP estimates for segmentation models based on Markov random fields [35, 13].

Bayesian MCMC methods have been applied to image segmentation since Green introduced the reversible jump scheme and showed a simple example appli-

cation to segmentation [39]. Since then, other authors MCMC segmentation studies have been published, most of them based on MRF models [6, 54]. Typically estimation of the distribution is done in a top-down manner: random samples are picked and their fit to the data is evaluated. Segmentation is a high-dimensional problem, and therefore a large number of samples are required and convergence takes a long time. Clark and Quinn [17] have presented an MCMC image segmentation algorithm where the sampling efficiency is improved by using image information to produce data driven proposal samples. In their approach, image segments are modeled by Gaussian intensity distributions, with a Markov random field model for the prior probabilities of segment labels.

Tu and Zhu [108] have proposed another data driven MCMC approach to image segmentation, in which various types of cues from image data are used to drive the MCMC algorithm, that is, to produce good proposition samples using bottom-up information processing. The probability model for the observed image is based on different texture models for the image segments.

Lee [68] has presented a model designed to explain how texture segmentation could be performed by the human visual cortex. The following priors for the statistics of surfaces and boundaries are used in the model: spatial homogeneity within segments; the systematic variation of texture due to perspective and surface shape variations (illumination is not mentioned but a contrast normalization mechanism is included in the processing); abrupt changes in statistics imply segment boundaries, the number of segments and lengths of segments are limited.

3.2.2 Evaluation and comparison of segmentation results

The evaluation of segmentation results and the comparison of different methods is very problematic. There is no standard dataset, and each author typically presents results on a few example images. The types of images can be very different, from aerial images to mosaics of synthetic textures. The goals of segmentation can be very different, too. In some publications, the number of segments is restricted to a pre-defined value, which can often be as low as two.

There is no standard method to compare different segmentations of the same image. Since there is no theoretical framework for segmentation and no definite ground truth for non-synthetic images, it seems unlikely that a universally accepted criterion could be established. Nonetheless, a number of methods and criteria for the quantitative evaluation of segmentation quality have been proposed [119]. For example, Levine and Nazif [69] have defined criteria for uniformity of regions and edges. This type of approach is most suitable in a restricted application, where the quality criteria are easy to define and to weight. It is unlikely that the same set of goodness criteria could be used to evaluate arbitrary types of images. Everingham et al. have proposed applying the Pareto front to using a number of different segmentation quality measures together [29]. The

choice of good measures still remains open.

If the correct segmentation is known, it can be used directly to evaluate segmentation results. The correct segmentation can be defined without ambiguity in the case of synthetic images or a restricted application (such as one object against a uniform background). In general, definition of the correct segmentation can be very subjective. A simple way to compare segmentation results with the ground truth is to consider the proportion of pixels misclassified [117] or the probability of misclassification [67]. These types of measures require that the segments be identified using prior information, so they are best applicable in supervised segmentation when the number of segments in the image is low. Martin et al. suggest comparing segmentation results to ground truths produced by human subjects [80]. They propose an error measure that is tolerant to different levels of refinement, such that no penalty is incurred if the same segment is subdivided into more than parts in another segmentation. The number of segments must be approximately equal, however.

3.3 Features for image segmentation

Image segmentation is based on the assumption that different segments have different appearance. In computer vision, the appearance is characterized in terms of variations in the colors and textures of the image. Segmentation algorithms either attempt to detect edges between different areas or to model regions that can be described by different values of color or texture features.

3.3.1 Color and gray values

The most basic choice of features are the three spectral channels of the colorspace used to represent the color of each pixel, for example RGB used in display devices or YCbCr [40] which is used in many image compression schemes. In many colorspaces, brightness information is separated from color information, as in YCbCr, where the Y channel is the brightness component and the two chromacity channels carry color information. This is a way to minimize correlation between the three channels, which is essential for effective compression techniques. Another interesting colorspace is called CIELAB, in which color differences are designed to resemble human vision, such that visual perceptions of how close or far apart colors seem should correspond to Euclidian distances measured in this colorspace [46].

Pixel brightness values are the most important single feature in image processing. In many computer vision publications, color information is not used at all. Filters are often only applied to the monochrome intensity image. Monochrome information is adequate for human vision. The effect of surface shape and illumination conditions appear as brightness differences. Strong brightness gradients on

objects are common in natural images, and this causes spatial inhomogeneities in brightness value distributions. A single dominant light source can often be associated with a smooth brightness gradient, while edges of objects typically appear as abrupt changes. This makes the brightness gradient a powerful tool in many image processing tasks.

3.3.2 Texture features

The concept of texture in computer vision is difficult to describe mathematically, and many different definitions have been proposed [109]. Here we consider the use of texture features to discriminate different segments in images and texture is understood as the variation of the color or grayscale values within a segment. In many contexts, textures are regarded as homogeneous repetitive structures, but in practice many objects have irregular and variable textures.

Early approaches to texture description have concentrated on single pixels and their neighbourhood relationships. An example are features computed from the co-occurrence matrix of intensity values of pixels at different spacings [41]. These types of features are sensitive to scale and contrast of the image. Markov random fields (MRF) [35] have been used extensively to model the spatial distribution of pixels values. Different methods have been reviewed by Tuceryan and Jain [109].

More recently, the analysis of spatial frequencies has become a prominent approach. Several studies make use of the Brodatz set of images [14]. These images are samples of different textures, which are mostly homogeneous and characterized by high spatial frequencies. Various wavelet representations are reported to produce good results in texture classification and retrieval experiments [30]. One possibility for the basis of a wavelet representation is the Gabor filter. The Gabor filter has many advantageous properties [77], and cells that perform a similar operation are found in the mammalian visual cortex. The descriptor for homogeneous texture that is included in the MPEG-7 multimedia content description standard is based on Gabor filters [78].

The classification and retrieval of textures is a prominent area of texture research. In that context, the division of images into different textures is not necessarily considered. Often, images are treated as being composed of one single texture [78]. This means that the images are assumed to be homogeneous throughout and free of occlusions. Synthetic textures are also often used. In such cases, highly specific descriptors are needed, which means that there is a large number of free parameters. Extending results from such studies to the discrimination of textures in natural images can be difficult.

When the goal is to segment an image into different objects and textures, the boundaries of segments have a special significance. The transition between two different textures appears as some kind of an edge, which is a high frequency event, so the frequency distribution near the edge is quite different from the fre-

quencies of either of the two textures. Moreover, many textures that appear in natural images are too irregular from the point of view of many texture analysis methods such as structural models based on texture primitives [109]. The discernibility of texture is highly dependent on scale. The texture segments may be too small to constitute a repetitive structure that could be meaningfully described by many texture modeling methods.

3.3.3 Distance measures for textures

Values of different feature measures for texture segments or textured images tend to vary over the area, so instead of single values, it is usually more appropriate to compare distributions of features, such as pixel intensities or outputs of different filters. Histograms are typically used to estimate these distributions. Many different methods can be used to compare texture histograms. These include nonparametric statistical tests, information theoretical measures and various heuristic measures such as intersection [103] and norms of vectorized histogram differences.

A comprehensive evaluation of different texture measures is difficult to perform because of the great number of different texture features and dissimilarity measures that have been proposed. In addition, there are different goals for texture analysis, so the criteria for the most suitable texture measure may depend on the application. Ojala et al. [91] have compared some different filter based measures in the classification of Brodatz textures. They found simple gray-level differences to produce best results. Puzicha et al. [96] report results from tests between many different measures of texture similarity on both Brodatz textures and natural images. As features, they used histograms of Gabor filter responses for Brodatz images and histograms of CIELAB color channels for the color photos, which were treated as one single texture. The results are somewhat inconclusive, as the ranking of different measures depends on the goal of the analysis, size of the texture sample and the number of different classes, among other things. For segmentation, the χ^2-statistic and the $\mathcal{L}_1$ norm of multivariate histograms determined by a clustering procedure were found to yield best results.

3.3.4 Comparison of different texture measures for segmentation

We used a large dataset of human segmented natural images [80] to evaluate the ability of different texture measures to separate any object from its background. We included 300 different images in the database, ranging from landscape images to pictures of animals and people.

For each image, we examined all human segmented objects to see how clearly they stand out from their environment with respect to different criteria. The environment of an object was taken to be the 10 pixels wide area that surrounds it.

Table 3.1: Comparison of different texture distance measures for the YCbCr colorspace.

	Y	Cb	Cr
KL distance	0.822	0.815	0.825
histogram distance	0.795	0.789	0.796
cumulative histogram	0.827	0.807	0.812

The objects were divided into two simply by drawing a vertical line in the middle. This produced four areas with different textures: the left and right parts of the object T_l and T_r, the whole object T_{l+r}, and the background area T_b. The textures were represented using competing texture descriptors. We compared two distances between the textures: left side–right side $D(T_l||T_r)$ and foreground–background $D(T_{l+r}||T_b)$. The criterion is such that both the texture descriptors and the distance measure are satisfactory for a given object if $D(T_{l+r}||T_b) > D(T_l||T_r)$. The quantity that was used to compare the different measures is γ_{sep}, the average percentage of segments for which the above condition holds.

A 100 % correct result cannot be expected because of the nature of the database. Some segments do not stand out from the background, and some objects' left and right side happen to be very different by appearance.

We compare the following measures: 1) Kullback–Leibler distance, 2) histogram distance, 3) cumulative histogram distance. The KL distance is a standard information theoretical measure to compare the difference between two distributions. Distance between histograms is the most direct way to compare two distributions. We use the L_2-norm to do this. There is some reason to assume that the difference of cumulative histograms should be more robust against changes in illumination, as they appear as shifts in the histogram [102]. The textures distributions are represented as 35 bin histograms.

The values of γ_{sep} for different distance measures and different color spaces are shown in Tables 3.1 and 3.2. From the results it appears that there is no significant difference between YCbCr and CIELAB colorspaces. L_2 norm between histograms is inferior to the other two distance measures. It is notable that the chromacity channels are not significantly weaker features than the brightness channels (the leftmost columns).

Another factor to consider is the correlation between different channels' performances. If the correlation is strong, then the inclusion of all channels will not add much information compared to using only one of them. The correlation coefficients are shown in Tables 3.3 and 3.4. The column labels indicate which channels are being compared. For our test data the values are somewhat smaller in the case of the CIELAB colorspace. This suggests that there may be a slight advantage in choosing the CIELAB colorspace, when all the channels are included

Table 3.2: Comparison of different texture distance measures for the CIELAB colorspace.

	L*	a*	b*
KL distance	0.826	0.813	0.815
histogram distance	0.792	0.791	0.805
cumulative histograms	0.824	0.805	0.820

Table 3.3: Correlation coefficients between segment discriminability of color channels using different distance measures; YCbCr colorspace.

	Y–Cb	Y–Cr	Cb–Cr
KL distance	0.387	0.382	0.411
histogram distance	0.423	0.350	0.362
cumulative histograms	0.404	0.314	0.392

in the model. An advantage of YCbCr is that it is used in many common image storage formats, so some savings in computation time can be made by avoiding the colorspace conversions. As for the distance measures, no significant differences can be observed between them.

The Sobel mask is a simple gradient operator which is used in image processing as an edge extraction tool. The operator can be vertical or horizontal. We carried out experiments similar to those described above using histograms of responses of both horizontal and vertical Sobel filters. Using one of the filter responses yields a poor 50 % accuracy (on average 50 % of the segments are separated from their background). When the combination of horizontal and vertical edge histograms is used, the value is 76 %, which is worse than any of the individual color channels. We assume that this is because a continuous line or edge structure is rare and strongest edges appear at boundaries, where their effect extends both inside and outside the segments.

3.4 A method for MCMC image segmentation

In this section, we present an MCMC method for the unsupervised segmentation of natural images. The method is based on defining a likelihood function and a prior probability distribution for different segmentations of an image. The probabilistic framework of the algorithm is similar to the approach of Tu and Zhu [108], but the sampling method is different [63]. The goal is a rough segmentation of different objects into different segments. Our emphasis is on minimizing computational complexity without allowing any significant degradation in the quality of results.

Table 3.4: Correlation coefficients between segment discriminability of color channels using different distance measures; CIELAB colorspace.

	L*–a*	L*–b*	a*–b*
KL distance	0.271	0.330	0.404
histogram distance	0.218	0.364	0.402
cumulative histograms	0.212	0.304	0.422

We first describe the overall probability model and discuss details of the prior distributions and texture distribution model selection.

3.4.1 Probability model for image segmentation

The segment division $\mathbf{S} : \{S_1, ..., S_K\}$ divides the image I into K different segments, such that each image pixel is assigned with a segment label. A texture model t_r, controlled by parameters θ_r, is associated with each segment S_r. The probability model for the likelihood of the segment S_r is defined in terms of the texture model as $p(I_r|\mathbf{S}, t_r, \theta_r, M)$, I_r is the part of the image that belongs to segment S_r. We treat the segments as independent of each other, so the likelihood of the image I with segmentation $\mathbf{S}$ is the product of segment likelihoods

$$p(I|\mathbf{S}, \mathbf{t}, \theta, M) = \prod_r^K p(I_r|\mathbf{S}, t_r, \theta_r, M). \tag{3.1}$$

The M reminds that the probabilities are conditional to constraints that are due to this using this kind of a model.

Using Bayes' theorem, the posterior probability of the segment division $\mathbf{S}$ and model parameters $\mathbf{t}$ and θ, given the observed image I, can be written as follows:

$$p(\mathbf{S}, \mathbf{t}, \theta|I, M) = \frac{p(I|\mathbf{S}, \mathbf{t}, \theta, M)p(\mathbf{S}, \mathbf{t}, \theta|M)}{p(I|M)}. \tag{3.2}$$

The term $p(\mathbf{S}, \mathbf{t}, \theta|M)$ is the prior distribution of the segment division and segment texture model parameters. We are interested in the segment division of the image, not the texture models. Therefore, we choose the prior distribution of the segment division $p(\mathbf{S}|M)$ to be independent of $\mathbf{t}$ and θ, such that

$$p(\mathbf{S}, \mathbf{t}, \theta|M) = p(\mathbf{S}|M)p(\mathbf{t}, \theta|M), \tag{3.3}$$

and integrate out the parameters $\mathbf{t}$ and θ to obtain the posterior distribution of the segment division:

$$p(\mathbf{S}|I, M) \propto \int\int p(I|\mathbf{S}, \mathbf{t}, \theta, M)p(\mathbf{t}, \theta|M)d\mathbf{t}d\theta p(\mathbf{S}|M). \tag{3.4}$$

For computational efficiency, we approximate the likelihood function for each segment S_r as a function of the parameters t_r and θ_r with a delta function at the maximum of the these parameters' conditional posterior distribution:

$$p(I_r|\mathbf{S},t_r,\theta_r,M) \approx p(I_r|\mathbf{S},t_r^{MAP},\theta_r^{MAP},M)\delta(t_r-t_r^{MAP},\theta_r-\theta_r^{MAP}), \tag{3.5}$$

where

$$\{t_r^{MAP},\theta_r^{MAP}\} = \max p(t_r,\theta_r|I,S,M). \tag{3.6}$$

This approximation leads to an empirical Bayes method and in theory it has the effect of over-fitting the rest of the parameters, but the following arguments support this choice:

- We choose simple texture models which have unimodal likelihood functions (MAP estimates for parameters θ_r^{MAP} can be computed in closed form)[1]
- The texture models have only a few degrees of freedom, and a validation technique is used to prevent over-fitting
- The likelihood functions are expected to be highly peaked as functions of texture model parameters, and our experiments with images indicate that the difference between the likelihood of the point estimate and the expected likelihood obtained by taking samples from the conditional posterior is negligible
- Savings in the computational cost are substantial.

Computation of segment likelihoods

The probability model for image segmentation (3.1) is based on the assumption that different segments are independent and have different appearance. The likelihood of a given image segment

$$L_r(t_r,\theta_r) = p(I_r|\mathbf{S},t_r,\theta_r,M) \tag{3.7}$$

is a function of the segment's texture model t_r and the parameters of that model θ_r.

Ideally, the segments would have consistent, distinctive pixel patterns or textures, but in real images strong variability within segments are common. Since there is no theoretically correct way to define the general segmentation of natural images in terms of texture models, the choice of texture models will have to be heuristic and its goodness can only be measured by subjective evaluation of practical results. Our goal is not to produce an accurate model that is able to generate

[1]The concept of modeling images as independent segments is a strong simplifi cation of reality, so there is no point in using overly complex texture models.

a realistic-looking reconstruction of the image in terms of the segment division and the texture models. Instead, we seek a rough segmentation where different-looking objects should be classified into different segments. For this purpose we find that the texture models for the segments should have low complexity. Moreover, we prefer texture descriptors that are reasonably quick to compute, because the search involves performing a comparison for each possible segment division.

We define a number of alternative segment texture models. To choose one of these models for a segment I_r, we randomly divide the pixels in the segment into two sets: a training set for fitting the models and a test set for evaluating model fit. The best model t_r is that which gives the highest likelihood value on the test pixels. Parameter values θ_r for the models are estimated by maximizing their posterior probabilities conditional on the segment division and the training pixels according to (3.6), and one of the alternative models is selected based on the predictive likelihood of test pixels. This procedure corresponds to the comparison of partial predictive likelihoods [89] of the texture models, which we use as submodels of the full probability model. The competing texture models have equal prior probabilities.

We define four different models that describe the distribution of pixel color values. We use the *YCbCr* color space, which consists of one luminance channel I^1 and two chrominance channels I^2 and I^3. The channels are modeled independently, so the likelihood (3.7) is computed as the product of individual channels' likelihood terms

$$L_r(t_r, \theta_r) = \prod_{c=1}^{3} L_r^c(t_r^c, \theta_r^c). \tag{3.8}$$

The distribution models are Gaussian, Laplacian, multinomial, or – for the luminance channel only – a linear spatial model with additive Gaussian noise. The multinomial model has a large number of degrees of freedom, and to avoid overfitting we apply a Dirichlet distribution as its conjugate prior. For the rest of the models, which have no more than a few free parameters, we use non-informative uniform priors.

The *Gaussian* model

$$L_r(\theta_r^c; t_r^c = g) = \prod_{i \in S_r} \frac{1}{\sqrt{2\pi}\sigma} e^{-(I_i^c - \mu)^2/(2\sigma^2)}, \;\; \theta_r^c = (\mu, \sigma) \tag{3.9}$$

is suitable when there is little structure in the distribution of the spectral channel in the segment. Parameters of the model θ_r^c are noise variance σ and the mean value μ. The index i runs through the list of pixels that belong to the segment S_r, which defines the image region I_r.

The *Laplace distribution* model

$$L_r(\theta_r^c; t_r^c = l) = \prod_{i \in S_r} \frac{1}{2\beta} e^{-|I_i^c - \mu|/\beta}, \;\; \theta_r^c = (\mu, \beta) \tag{3.10}$$

has heavier tails than the Gaussian distribution, and it is appropriate for modeling a cluttered background, for example.

The *multinomial* model expresses the distribution of the pixels in terms of a non-parametric histogram. The histogram $\mathbf{h}$ for the intensity or chrominance channel c is computed from discrete (8 bit) values I_r^c of the channel. The likelihood of the channel c in segment S_r can be written as

$$L_r(\theta_r^c; t_r^c = m) = \prod_k (h_k)^{n_k}, \quad \theta_r^c = \mathbf{h}, \tag{3.11}$$

where h_k denotes the histogram density of (discrete) pixel values within the segment, and n_k is the number of pixels with intensity value k. Model parameters are values of the histogram h_s^c, so the number of degrees of freedom is high compared to the number of samples (number of pixels in a segment). Therefore the use of an appropriate prior is essential. We apply a Dirichlet prior

$$p(\mathbf{h}) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k (h_k)^{\alpha_k - 1}, \tag{3.12}$$

where the parameters $\alpha_k = \alpha$ are equal, to get the MAP estimate for $\mathbf{h}$. The value $\alpha = 4$ has been used in simulations. We have experimentally evaluated the error in the likelihood of the MAP estimate and the expected likelihood obtained taking random samples from the conditional posterior distribution of $\mathbf{h}$. We found this error to be negligible for typical size segments. Other texture models that we use have very few parameters compared to the multinomial model.

The *brightness gradient* model is only enabled for the luminance channel. Nearly constant luminance gradients caused by uneven lighting are common in natural images. This is a spatial linear regression model, written as

$$L_r(\theta_r^c; t_r^c = r) = \prod_{x_i \in S_r} N\left(I^c(x_i) | \mathbf{w} \begin{bmatrix} X(x_i) \\ Y(x_i) \\ 1 \end{bmatrix}, \sigma \right), \quad \theta_r^c = (\mathbf{w}, \sigma), \tag{3.13}$$

where $X(x_i)$ and $Y(x_i)$ are the image coordinates of the pixel x_i, and $\mathbf{w}$ is the vector of regression coefficients. $N(x|\mu,\sigma)$ represents the Gaussian residual noise distribution.

Prior probability

In this section we describe the prior probability of the segment division $p(\mathbf{S}|M)$, which is required in equation (3.4). The prior controls certain properties of the segments.

In a problem as complex as model based image analysis, there will necessarily be many implicit restrictions incurred by the structural properties of the models

that are chosen. There are also adjustable parameters that cannot be practically resolved in a purely Bayesian framework. In a specialized application, the prior probabilities can describe known statistical properties of the types of objects or image areas that are expected to be found. Because we want our methods to be applicable to a broad variety of different images, our choices for the priors are quite general and they are not tuned for any particular type of scene.

One property of segment boundaries that is generally applied in computer vision, is their smoothness (see e.g. the discussion on representing boundaries by *snakes* in ref. [46]). The smoothness prior is effective especially for man-made objects. We evaluate the smoothness of the segment outline by comparing the boundary to its low-pass filtered version. The boundary of segment S_r is represented as a sequence of n connected points $B_i^r, i = 1, \ldots, n$, which are vectors in image coordinates. The number of points n is determined by the pixel resolution. A moving average filter is used to compute a smoothed version of the boundary:

$$\hat{B}_i^r = \frac{1}{m} \sum_{j=i-m/2}^{i+m/2} B_j^r. \tag{3.14}$$

Values around 50 have been used for the length of the filter m in most simulations. A Gaussian prior is applied to the distance between the boundary and its low-pass filtered version, as follows

$$p_{\text{smoothness}}(B^r) \propto e^{-\alpha_s \frac{1}{n} \sum_i^n \left(|B_i^r - \hat{B}_i^r|^2 \right)}, \tag{3.15}$$

where α_s is a constant.

We use a Poisson distribution for the number of segments

$$p_K(K) = \frac{e^{-\lambda} \lambda^K}{K!}, \tag{3.16}$$

where the expected value is $\lambda = 7$. following Tu and Zhu [108], the prior for the sizes of the segments is given by

$$p_{\text{size}}(A_r) \propto A_r^{-\kappa}, \tag{3.17}$$

where A_r is the number of pixels in the segment S_r and $\kappa = 0.9$ is a constant. These two priors control the detail level of the segment division.

The prior probability of the segment division is computed as the product of the above components:

$$p(\mathbf{S}|M) \propto p_K(K) \prod_r^K p_{\text{size}}(A_r) p_{\text{smoothness}}(B^r). \tag{3.18}$$

3.4.2 MCMC segmentation process

The problem is to estimate the posterior probability distribution of segment divisions for the given image (3.4). We apply the reversible jump MCMC technique [39] to obtain samples of the posterior distribution. The samples are different segment divisions of the image. Proposal samples are generated using three different methods. The proposal samples are accepted with a probability that maintains the balance of the MCMC chain, according to equation (2.10).

Ehlers and Brooks [26] have shown that if in the RJMCMC jump from dimension k to k', the variables specific to k' are generated from their conditional posterior distributions, conditioned on the variables common to k and k', then the generated variables do not contribute to the acceptance probability of the jump. We use this result and generate the parameters of the texture models for segments created in the MCMC transitions by approximating their posterior distributions.

Data driven techniques are used to to generate the proposal samples. Three different transition steps are defined: diffusion, split and merge. Diffusion does not affect the dimension of the parameter space, while split and merge form a pair of opposite transitions between different dimensions. Together, they enable reversible jumps between the segmentation states of different dimensionalities. Each proposal sample is generated using one of these three mechanisms, which are equally probable.

The segmentation is initialized by classifying the whole image into one single segment. An overview of the whole process is described in Algorithm 3.1.

Diffusion

The diffusion routine draws the proposal samples directly from the segment likelihood functions (3.7). We randomly select one of the segments, S_g, for region growing. All segments have an equal probability of being selected. We choose a random radius value ρ for a circular diffusion kernel. We dilate the selected segment by moving the kernel along its boundary and denote the resulting segment S'_g. For each pixel on the diffusion area, $\{x_i|(x_i \in S'_g, x_i \notin S_g)\}$, we compute two likelihood values: the likelihood $p(x_i|t_o,\theta_o)$ for the model (t_o,θ_o) of the segment that the x_i currently belongs to and $p(x_i|t_g,\theta_g)$ for the model (t_g,θ_g) of the growing segment S_g. The likelihood ratio $\gamma(x_i) = p(x_i|t_g\theta_g)/p(x_i|t_o,\theta_o)$ determines which pixels should be transferred to segment S_g. We use a spatial low-pass filter F with a radius proportional to ρ to smooth the values of $\gamma(x_i)$ between neighboring pixels in order to keep the segments contiguous. Those pixels x_j for which the filtered values of the likelihood ratio is $\tilde{\gamma}(x_j) = F[\gamma(x_j)] > 1$ are proposed to be transferred to the segment S_g. We have found this procedure to maintain the contiguity and smoothness of segment boundaries relatively well. The diffusion process is illustrated in Fig. 3.1. It is possible that the balance of the MCMC chain is not strictly

Compute the posterior probability of the initial segment division $\mathbf{S}^0$;
repeat Generate new sample $\mathbf{S}^{i+1}$
  Select transition step at random (diffusion, split or merge) ;
  Generate new segmentation state $\mathbf{S}'$ according to selected transition ;
  foreach *segment S_r that changes in the transition* $\mathbf{S}^i \rightarrow \mathbf{S}'$, **do**
    Estimate texture models t_r and their parameters θ_r ;
    Compute new likelihood value $L_r(t_r, \theta_r)$ from (3.8);
  end
  Compute the change in the prior probability $\frac{p(\mathbf{S}'|M)}{p(\mathbf{S}^i|M)}$ from (3.18) ;
  Compute the change in the marginal posterior probability
  $p(\mathbf{S}'|I,M)/p(\mathbf{S}^i|I,M)$ from (3.4) ;
  Compute the acceptance probability $A(\mathbf{S}^i \rightarrow \mathbf{S}')$ using (2.10) or (2.9) ;
  Generate a random number $q \sim \text{Uniform}[0,1]$;
  if $A(\mathbf{S}^i \rightarrow \mathbf{S}') > q$ **then**
    $\mathbf{S}^{i+1} \leftarrow \mathbf{S}'$
  else
    $\mathbf{S}^{i+1} \leftarrow \mathbf{S}^i$
  end
until *sampling chain is sufficiently converged*;

Algorithm 3.1: Overview of the RJMCMC segmentation process.

preserved in the diffusion step, but we have not encountered any problems due to this in our experiments.

Split

The split transition attempts to divide a segment into two new segments along apparent edges. First, one of the segments is selected for splitting, and a starting point is picked within that segment. These choices are based on using the likelihood values of segments and pixels, such that poorly fitting areas and segments are likely to be targeted. Finally, an edge tracing method is applied to produce a division of the segment.

The edges are extracted using Gabor filters [77]. The type of filter is not a critical choice; other types of edge-sensitive filters would probably apply as well. We use six different filter orientations. Responses of Gabor filters are complex, but we only use their absolute values.

Poorly fitting segments are assigned with high probabilities of being selected for splitting. We have two alternative ways of determining the selection probabilities. The first way is to use the likelihood values of the segments (3.7) directly (sums of pixel log-likelihoods divided by the number of pixels in the segment; these have been computed previously). The probability of a segment to be se-

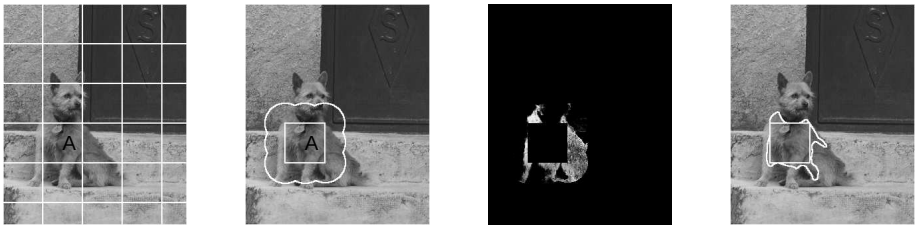


Figure 3.1: Diffusion of segment boundaries. *a)* The initial segment division. The segment marked with *A* is selected for growing. *b)* The segment *A* is dilated by moving a circular kernel along the boundary of the segment. *c)* The pixel-wise likelihood ratio of competing models in the diffusion area (see text for explanation). *d)* Old boundary of segment *A* and the new boundary, determined by low-pass-filtering and thresholding the likelihood ratio map. The neighbouring segments (not shown) are modified to accommodate for the change.

lected for splitting is proportional to the inverse of the geometric average of the likelihoods of pixels in that segment. A problem with this is that a large segment that contains small, poorly fitting areas may have a very small probability of being selected. Such areas within segments would be good candidates for new segments. To find such areas we use an alternative segment selection rule: we examine single-pixel likelihood values smoothed by a spatial median filter to find contiguous poorly fitting areas. The starting point for edge tracing will be on such an area, and the segment containing this point gets selected. In the case of the former segment selection method, the starting point is chosen randomly such that the probability is proportional to the strength of the edge filter response.

Edge tracing proceeds in two opposite directions, following continuous edges in the edge map with a random step size. We use a weighting mask to emphasize those edges that lie ahead in the direction of the tracing path. This has the effect that a sharp change of direction requires strong support from the edge map. The split is complete either when the tracing path reaches segment boundaries at both ends or when the tracing path crosses itself.

The resulting split curve is a function of the selected starting point (which indicates the selected segment, too) and the selected step size for edge tracing. In mathematical terms, the split move proposes a transition from the segmentation state $\mathbf{S}$ to the state $\mathbf{S}'$. The selected segment $S_k \in \mathbf{S}$ is divided into $S_i, S_j \in \mathbf{S}'$, so only the parameters of these segments are affected, as follows:

$$S_k : \{\Omega_k\} \rightarrow S_i : \{\Omega_k, \gamma_{ij}\}, S_j : \{\Omega_k, \gamma_{ij}\}, \tag{3.19}$$

where Ω_k is the boundary of segment S_k, and γ_{ij} is the new proposed split curve,

which together with Ω_k defines boundaries of S_i and S_j. An auxiliary random variable u is needed to define a bijection from the parameters of $\mathbf{S}$ to the parameters of $\mathbf{S}'$. The acceptance probability is given by

$$A(\mathbf{S} \to \mathbf{S}') = \min\left\{1, \frac{p(S_i|I)p(S_j|I)}{p(S_k|I)} \frac{j_{Mij}(\mathbf{S}')}{j_{Sk}(\mathbf{S})} \frac{q'(\gamma_{ij})}{q(u)} \left|\frac{\partial(\gamma_{ij})}{\partial(u)}\right|\right\}, \tag{3.20}$$

where $j_{Sk}(\mathbf{S})$ is the probability of proposing to split the segment S_k and $j_{Mij}(\mathbf{S}')$ is the probability of proposing the opposite move, that is, to merge S_i and S_j. Let u be equivalent to γ_{ij}, so the Jacobian term becomes $|\partial(\gamma_{ij})/\partial(u)| = 1$. The distribution $q(u)$ is the proposal distribution for the split curve, and we compute it based on the probabilities of selecting different starting points and step sizes for the edge tracing. The opposite probability equals $q'(\gamma_{ij}) = 1$, because there is only one way to merge two regions.

Merge

The merge transition is for joining two randomly selected adjacent segments. We choose the first segment with equal probabilities for all segments, but for the other one we use probabilities proportional to the inverse of the Kullback-Leibler distance between the probability models of the first segment and its neighbors. The Kullback-Leibler distance is a standard measure for comparing the similarity of probability density functions, and for discretized density functions h_1 and h_2 it is computed as follows:

$$D_{KL}(h_1\|h_2) = \sum_j h_1^j \log \frac{h_1^j}{h_2^j} \tag{3.21}$$

The density histograms h_1 and h_2 are computed by discretizing the probability models for the segment spectral channels. We compute the sum of the KL distances over the three channels in each of the competing neighbor segments, and select one of these for merging with a probability that is inversely proportional to this sum. The acceptance probability is the reciprocal of (3.20).

3.4.3 Analysis of the posterior distribution

The result of MCMC sampling is a chain of samples from the posterior distribution. In the beginning of the sampling chain, there is a transient effect known as the burn-in period, where the chain moves from the improbable initial value (just one segment) to states that have much higher probabilities. We determine the end of the burn-in period by monitoring the stabilization of number of segments K. The burn-in samples are discarded.

In the context of image segmentation, it is not obvious how to combine the samples in a way that is easy to interpret. The result is a set of N sample segmentations $\mathcal{S} = \{\mathbf{S}^1, \ldots, \mathbf{S}^N\}$, and each segmentation $\mathbf{S}^i = \{S_1^i, \ldots, S_{M_i}^i\}$ consists

of M_i segments. Thus we can identify $M = \max(M_i)$ distinct segment labels $s_m : m = 1, ..., M$. To compute an average segment label for each pixel, we suggest the following scheme, which is based on the co-occurrence of pixels in the same segment:

1. Choose a pivot pixel $x_p(s_m)$ for each segment label s_m. $x_p(s_m)$ is any of those pixels that most often get classified to the segment labeled by s_m. Discard those segment labels that get winning support in no pixel.

2. For each pixel x, compute the number of samples in which the pixel gets the same label as pivot pixel $x_p(s_m)$, that is,

$$\eta(x, s_m) = |\left\{ \mathbf{S}^k \subset \mathcal{S} : \{S_j^k \in \mathbf{S}^k : x, x_p(s_m) \in S_j^k\} \right\}|, \tag{3.22}$$

   where $|\cdot|$ denotes set cardinality.

3. The expected label $\hat{s}(x)$ of the pixel x is the value s_m that maximizes $\eta(x, s_m)$.

4. The value $c(x) = \max_n\{\eta(x, s_m)\} / \sum_m \eta(x, s_m)$ can be used as a measure of confidence in the classification of pixel x.

3.4.4 Comparison with a related approach

This segmentation method is in many respects similar to the one proposed by Tu and Zhu [108]. Both methods share the same basic generative probability model that defines the image in terms of independent segments that follow different stochastic texture models, and both methods use data-driven proposal distributions in the reversible jump MCMC algorithm to estimate the posterior distribution of the segment division. Here we explain the key differences between these two approaches.

Tu and Zhu define MCMC transition steps *model adaptation*, which means altering segment model parameters, and *switching image models*, which means changing the texture model of a segment. Model fitting is done by steepest ascent toward the maximum of the segment likelihood function, however. We integrate the estimation of segment models into the transition steps that change the segment division (diffusion, split and merge).

The data-driven proposal mechanisms are different. For diffusion, Tu and Zhu apply a curve based region competition method, whereas our approach is based on contiguous areas. For segment splitting proposals, clustering and edge tracing are used in both approaches. Tu and Zhu do clustering based on current texture models, while we do clustering based on the likelihoods of the models. They trace all potential edges beforehand; we trace edges online.

3.5 Experimental results

The segmentation method has been tested on 300 images of various subjects in the Berkeley segmentation dataset [80]. Examples of results are shown in Fig. 3.2 [2]. In Fig. 3.3, two examples are shown, where an artificial mosaic of five different textures is segmented. All segment boundaries are accurately identified, but some of the segments are subdivided into more than one parts.

Figure 3.2: Examples of segmentation results.

[2] More images are available at the web address http://www.lce.hut.fi /research/compinf/segment/

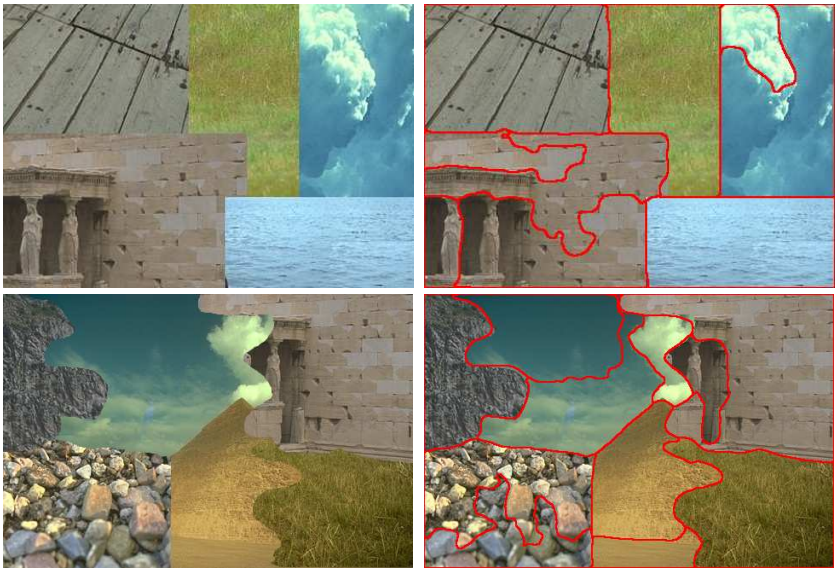


Figure 3.3: Segmentation results for two artificial images made up of five (upper image) or six (lower image) different textures.

In Fig. 3.4, three more segmentation examples are shown, including images synthesized from the estimated texture models.

Due to efficient generation of proposal samples, especially the diffusion algorithm, most of the computation time is spent evaluating the segment likelihoods. With 300×400 pixel images, running Matlab on a 1.8 GHz processor, 5 – 10 MCMC samples per second can be processed. Upward of one hundred samples are required for practically sufficient convergence, so running times per image range between 20 – 60 seconds. Tu and Zhu report processing times of 10 – 50 minutes for their data driven MCMC algorithm in a similar setting [108].

The evaluation of segmentation results is difficult because there is no ground truth and there is no generally accepted method for quantitative analysis. However, we have computed scores according to the method proposed by Martin et al., which is based on comparing human segmentations of the images in the Berkeley dataset [80]. The average F-measure as defined by Martin et al. was 0.55 for the 100 test images. We also compared the probabilities of hand segmented solutions computed from our model (3.4) with posterior probabilities of the results given by the proposed segmentation method for 200 images. In that group, there were only two images in which some hand segmentations had higher posterior probabilities than our segmentations. This shows that the probability model is not good at representing the properties of segments that are important to humans. On the other hand, this result suggests that the MCMC algorithm is quite reliable in finding good solutions conditional on the model.

Figure 3.4: Segmentation results with texture model synthesis. Left: Original images with segment boundaries. Middle: Images generated from the segment texture models and their parameters. Right: Texture models of the brightness channel color-coded: black – Gaussian, white – Laplace, dark gray – Multinomial, light gray – Brightness gradient.

In order to evaluate the advantage of the data driven proposal distributions, in Fig. 3.5 we compare the performance of the proposed algorithm to that of a sampling chain in which the proposal samples are random and do not depend on the data. The difference in the rate of convergence is very significant.

In Fig. 3.6 we compare our method to two segmentation algorithms based on color clustering. A simple segmentation method to produce spatially confined segments can be obtained by including image coordinates as input variables to the clustering algorithm in addition to the three color channels. We applied the fuzzy c-means algorithm directly to do the segmentation on the five dimensional input data. We scaled the variance of the coordinate values to twice the variance of the RGB channels. The result appears in Fig. 3.6*d*. This algorithm has a convergence time that is approximately equal to that of our MCMC method for 0.1 – 1 megapixel image sizes. The result that the MCMC method gives in an equal CPU time is shown in Fig. 3.6*b*. Both were implemented in Matlab. The algorithm presented in ref. [70] (Fig. 3.6*c*) analyses color channel histograms to determine thresholds for segment classification, and applies fuzzy c-means clustering to produce a vector quantization in color space. We applied a C implementation that

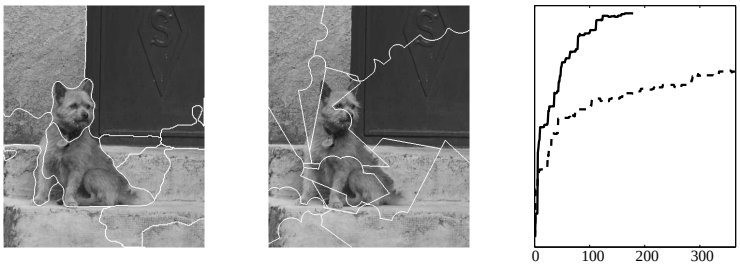


Figure 3.5: Data driven vs. random proposals. *Left*: result of data driven segmentation after 179 samples. *Center*: result of segmentation using random proposal distributions in the same amount of CPU time (365 samples). Metropolis-Hastings acceptance probabilities were 27 % and 20 %, respectively. *Right*: evolution of log posterior probabilities $\log p(\mathbf{S}|I,M)$ in the MCMC chains with data driven (*continuous*) and random (*dashed*) proposal distributions.

runs roughly twice as fast as the other two algorithms.

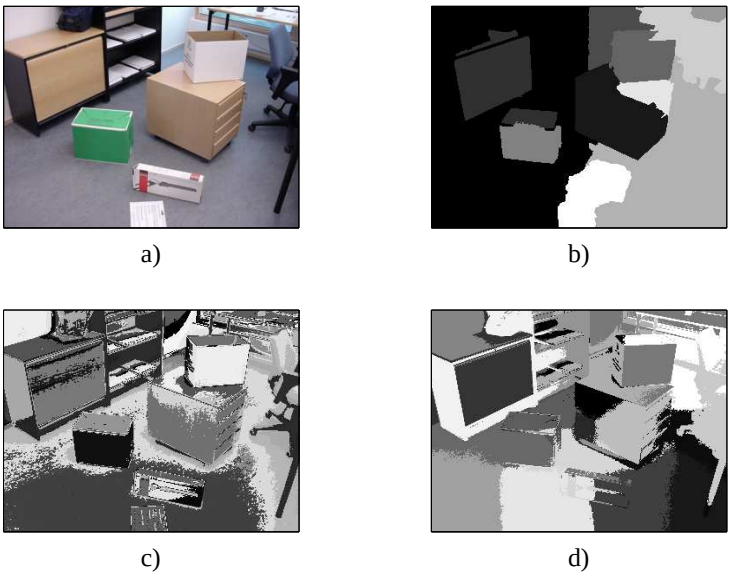
The example shows that MCMC techniques, despite their reputation as being frustratingly slow, can be quite competitive against classical heuristic techniques if they are designed to combine different types of information effectively. The use of bottom-up information in the generation of MCMC samples yields substantial savings in computation time.

3.6 Application in mobile robot navigation

In this section, we describe an application of the segmentation method to map building and navigation for a mobile robot. The task is to use a single camera to build a map of obstacle free area to enable navigation for a mobile robot. A standard solution to this problem involves the use of active sensors, such as laser or infrared. Instead, we attempt to accomplish this with a robot that is equipped with a single inexpensive camera.

3.6.1 Simultaneous localization and mapping

One interesting application for computer vision and scene analysis is in the control of autonomous mobile robots. On the other hand, a mobile robot can be used to explore an area and vision is essential for classification and recognition of the area. Active sensors such as active laser and sonar devices are commonly used in robot navigation. Their advantage when compared to vision is the accuracy and certainty of depth information, and the fact that the sensor data do not require much processing. The main limitation is the inability to distinguish between



a) b)

c) d)

Figure 3.6: Comparison with segmentation methods based on clustering. Segments are displayed using different grayscales in no particular order. *a)* Original target image. *b)* Segmentation produced by the MCMC based method with a running time equal to that in *d)*. *c)* Results of the FCM method [70] *d)* Fuzzy c-means clustering of color channels and spatial coordinates.

different objects. Using both vision and active sensors in combination should produce the best results. The main reason for the limited use of vision is the lack of reliable methods for analysing the images and the high computational cost.

The skill to simultaneously locate a robot and to generate a map of its surroundings is essential for enabling robots to function autonomously. Most approaches are based on defining landmark locations according to which the robot's pose is estimated [106]. Typically, the landmarks are point-like features which are identified in images. As the robot moves, methods such as extended Kalman filter are used to track the landmarks. Bayesian Monte Carlo sampling methods have been applied to the simultaneous mapping and localization problem by using particle filters as an alternative to the extended Kalman filter [82].

Point-like landmarks have many advantages: they may be easy to detect and three measurements in a coordinate system are enough for accurate representation of their location. On the other hand, if the scene consists of large objects or surfaces, detection of uniquely identifiable landmarks in them may be difficult. If the landmarks are not unique enough to be reliably identified from each other by appearance, data association becomes a major problem in tracking. Indoor environments mostly consist of regularly shaped surfaces and objects. To represent each object as a collection of multiple independent landmarks is redundant

and overly complex compared to a wireframe model, for example. Some authors have studied such representations [72], but they have been much less popular than 2-D point landmarks. A mapping system that identifies surfaces such as floors, walls and doors might be more robust than a system that is based on independent landmark points, but reliable recognition of such objects is difficult.

3.6.2 Solution based on scene segmentation

Color segmentation techniques applied in the map building problem typically classify image pixels using one or more predefined color models. This is a valid approach when the colors of different object classes are distinctive and do not overlap. However, in many real-world situations objects of interest have complex textures, similar colors appear on different surfaces, and shadows and reflections add to the complexity of the problem.

In our approach, we divide each image into two segments: the foreground that represents an obstacle-free ground area, and the background, which includes everything else and is treated as obstacles. The foreground is contiguous, since we only consider areas that the robot has access to. An initial division of the first image in a sequence is required to get an initial model for both foreground and background textures. [64]

Similar approaches to the robot navigation task have been proposed previously. For example, Thorpe et al. [105] have presented a system for outdoor navigation which divides pixels to road and non-road classes based on color and texture models. The color model is adaptive to a certain extent.

We apply the probabilistic image segmentation algorithm described earlier in this chapter. The method should be suitable for the real-time navigation task, so a computationally efficient solution is needed. In this application, the number of segments can be constant. Addition of a new segment to the model requires some external information. For example, if the surface material changes, bumpers or short range infra-red sensors can be used to determine that it is not an obstacle.

To demonstrate the advantage of the approach based on contiguous segments, we compare it to some more simple strategies of using the color models. The comparison is illustrated in Fig. 3.7, where the initial division between the foreground (floor) and the background can be seen in the image *a)*. Color distribution models are learned for both segments and the result of applying the models directly to classify each pixel is shown in 3.7*b)*. Pixel classification in a single pass is very quick, but the result is significantly inferior to our MCMC method (3.7*c*), and it cannot be relied on for map building. It is apparent here how the reflective glass walls make the task difficult. If those parts that are not connected with the initial floor segment are left out, the result is still very blotchy (3.7*d*). Finally, we show the result of region growing using the same model adaptation as in the proposed method but without the MCMC dynamics (3.7*e*, 3.7*f*). Segment texture models

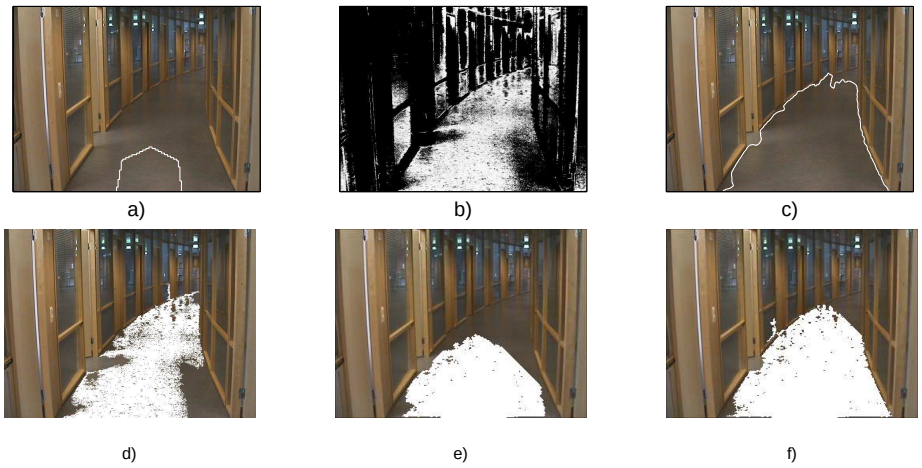


Figure 3.7: Segmentation vs. pixelwise classification. *a)* Initial segment division. *b)* Pixelwise classification result (probabilities of pixels belonging to the floor segment). *c)* Result of probabilistic segmentation. *d)* Parts of result (b) that are not connected with the initial segment have been left out. *e)* Sequential region growing with model adaptation over 4-connected neighbour pixels; equal CPU time as in (c). *f)* Same as (e) but twice the CPU time.

are updated after two cycles of adding those neighbouring pixels to the floor segment that are classified to belong to it. After a sufficient number of iterations, the result is not much different from 3.7*c)*, but some type of post-processing would be needed to resolve the numerous holes in the floor segment to enable navigation.

3.6.3 Results

We have tested the concept with a mobile robot that is equipped with a camera that is supported at the height of 1.6 m and can be rotated about both vertical and horizontal axes. Viewed from above, the robot has a round shape with a diameter of 0.5 m, and it is able to move at the speed of 0.5 m/s.

Given a target point, the robot's task is to build a map between its present location and the target and then to use that map in navigating to the target. The robot has knowledge of the location of a small patch that contains obstacle-free ground area. The robot aims its camera such that the patch is within the field of view and grabs an image. The MCMC diffusion algorithm is applied to grow the homogeneous ground area up to a boundary that naturally separates it from the rest of the scene. With knowledge of the position of the camera, its focal length and calibration parameters to account for lens distortion, and assuming the ground planar, image coordinates can be mapped to intersections of the ground plane and

3-D ray vectors originating from the camera. The boundary on the image is thus projected onto an occupancy grid [27] of the area. What is within the boundary is regarded as safe ground.

Based on the target point and the map, the robot checks whether the map to the target is complete, and if not, chooses a point on the map that is preferably in the direction of the target but not too close to the edge of the safe ground area. The map is filtered by a circular mask that is the size of the robot and transformed into the form of a graph by skeletonization, so that a graph search algorithm can be used to plan a path to the selected point. The robot follows the path and acquires a new image such that the field of view includes part of the mapped area. Now the patch of mapped area is used as the initial state for the segmentation algorithm, after projecting it onto the new image. Odometer measurements are used to determine the robot's current position.

Processing typically takes around one second per image in a Matlab implementation on a 1.8 GHz processor. The robot autonomously decides when it needs a new image for further path planning. Thus the rate at which new images are required varies according to visibility (obstacles, corners, etc.). The robot travels at the speed of 0.5 m / s, and image processing rarely causes any delays. In our experiments the navigation system has proved relatively robust against variations in floor appearance caused by illumination effects. The system works on different floor materials without any modification. An example case is shown in Fig 3.8.

The results show that this method enables collision-free navigation on the office floor. The robot odometer measurements are subject to small inaccuracies, which are the source of cumulative error in the maps. Thus the maps can only be used locally. An accurate mapping would require the compensation of this error by comparing the scenes of consecutive images.

3.7 Interactive segmentation

An important problem in photo editing is the separation of objects from arbitrary backgrounds. It is obvious that any unsupervised segmentation algorithm cannot match human abilities in such a task. In order to get results of acceptable quality, external information is needed. In the context of image editing, user interaction is a natural way to bring in the information. Some solutions to this problem have been proposed based on boundary detection. One of the best known methods is called intelligent scissors, proposed by Mortensen and Barrett [84], where graph search is used to find an optimal contour for an object through manually selected control points along the boundary. The authors show that the method is an effective tool for object extraction. The user typically has to point and click dozens of points right on the boundary to get the desired result.

The segmentation algorithm discussed in this thesis can be directly applied

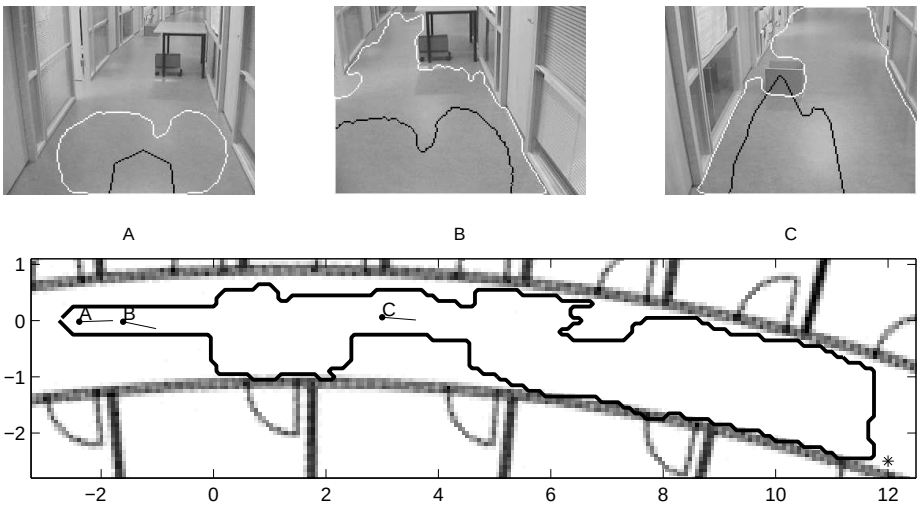


Figure 3.8: Robot navigates to target point along a curved corridor using image segmentation. Five images were required to build a map from the starting point to given target point $(12m, -2.5m)$. *Top*: Three of the five images with initial values (*black*) and segmentation results (*white*). *Bottom*: Final map based on the images. The vectors starting from labeled points indicate camera orientation during acquisition of the shown images. The target point is indicated with an asterisk. The robot started from the origin and moved backward to the point *A*. The track of this movement was used as the initial value for floor appearance. For the rest of the images, the segmentation result of the previous image was projected back on the current one and used as the initial value. At points *B* and *C*, two images were acquired using different camera orientations. The first image of each pair is shown. The approximate location of the true map is shown in the background.

to the same problem, enabling a different means of user interaction: the user can define regions by selecting smaller areas within them. This can be quicker and easier than selecting boundary points, because the selection normally does not need to follow the segment boundary. The selected areas are used as initial values for segmentation and they are labeled permanently as belonging to the named segments. Both the target object and the background can be divided to multiple segments if they are not homogeneous. The user has control over the number of segments, so splitting and merging are not used. This limits the solution space and makes the process fast enough for interactive use.

Examples are shown in Fig. 3.9. In the first example, the colors are distinctive, but the objects consist of differently colored regions. In the second image, the object blends well with its surroundings and there is no clearly visible edge. In

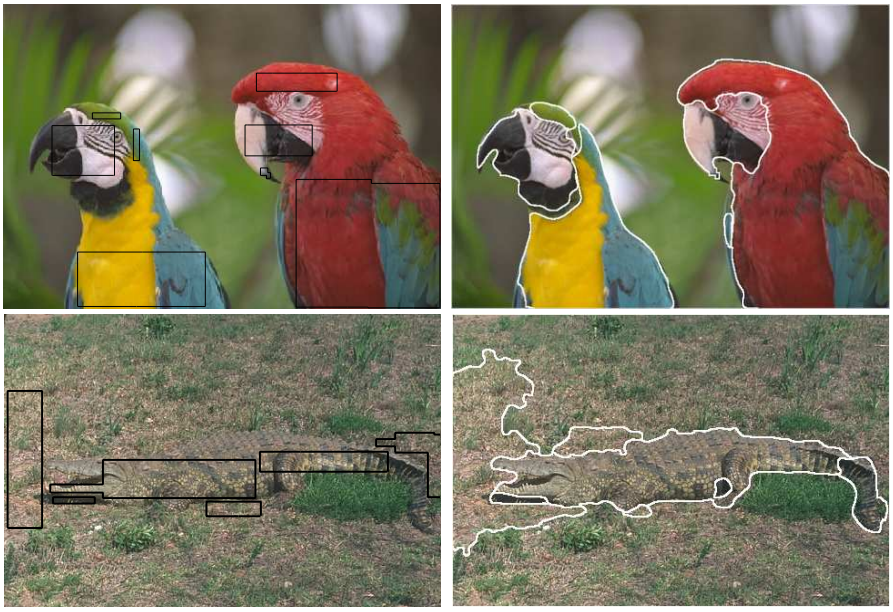


Figure 3.9: Example of interactive segmentation. Left: the user selects rectangular areas to control the segmentation. The process is interactive and the segmentation is corrected by adding new areas to foreground or background segments. Right: result based on the rectangles shown on the left.

both cases, nine rectangular areas were enough to produce a reasonably accurate segmentation result.

Previously proposed methods for this object selection tasks are based on specifying the boundary of the object. The advantage of this segment based method is that less accuracy is required from in mouse pointing by the user. In the example tests, only rectangular area selections were used. The use of free-form areas would make the process more efficient. The segment prior includes a boundary smoothness constraint (3.15), which could be relaxed for fine-tuning the result.

3.8 Discussion

In this section we have discussed a method for Markov chain Monte Carlo image segmentation and presented example results in different situations. The principle of MCMC segmentation has been previously proposed, but new techniques for efficient generation of samples are presented. A problem that is often associated with Monte Carlo sampling methods in image processing is their slow convergence. The results indicate that careful design of sample generation mechanisms can help to reduce computation times significantly, though for general images the

processing rates are still far from real-time.

All image segmentation methods are based on very simple models of reality, and that is why results on general images cannot be accurate. A semantically correct segmentation would requires recognition of objects and understanding the 3-D structure of the scene, which is beyond what is meant by segmentation in computer vision literature. It is questionable whether using richer or more complicated texture models in the 2-D segment framework would offer any advantage, because the segments still would not be modeled realistically but further increased model complexity would cause additional difficulty and lead to much slower computation times. We expect that an increase in model complexity would need to be balanced by bringing in new information.

The application examples show that the situation is quite different when the problem is restricted, and the methods presented in this section can be directly applied in practical problems. When the application is constrained, the use of more specialized models becomes feasible.

Chapter 4

Model based object matching

In this chapter, we discuss the matching of different two and three dimensional object models to images by MCMC sampling.

4.1 Object representation

Choosing an effective scheme for representing three dimensional objects is a difficult task. There are many possible ways to model objects, but a great number of parameters are needed for accurate descriptions. An important distinction in approaches is between 3-D models, which focus on determining the volume occupied by the object, and representations based on 2-D views, which describe the projected appearance of the object from one or more views. Though information processing in digital computers is very different from biological systems, it is interesting to note that brain research has produced evidence of both view based 3-D model based representations [15] being used by humans.

In computer graphics, efficient methods have been developed for handling various types of objects using three-dimensional models, and the rendering of images from these models can be performed very quickly on modern computers. The structure of a scene specified as 3-D models fully explains the scene and can be used in applications that require metric information. Geometric models used in computer aided design (CAD) are a good example. These are a very effective way of describing objects that have simple geometric shapes. Complicated surface shapes that cannot be conveniently modelled by analytical geometry can be approximated by dense wireframe representations. For object recognition, the parametrization of these models is a difficult problem. The number of parameters needed for a sufficiently accurate representation varies widely between different types of objects, so a general parametrization suitable for any object shape does not seem practical. Instead, objects must be treated in terms of specialized object classes. Geometrically simple shapes may be treated directly as CAD models,

and and occasionally their special properties can be used to determine robust features [97]. For more complex shapes, a regular grid representation is generally used. The number of reference points in this type of a model is very large, so dimensionality reduction techniques are needed to produce a tractable parametric representation for object recognition. The *eigenshape* representation that is created by principal component analysis is a popular method. It has been successfully applied in the modeling of human faces [2, 12]. The appearance of an object depends not only on the 3-D shape but also on the surface texture, and these two cannot be estimated separately. The acquisition of 3-D models is an important question. CAD models can be designed manually, but to specify a statistical model for an object class of variable shape, a large amount of measurement data is needed. Laser, infrared or MRI scanners can be used to obtain surface depth information. Another alternative is stereo imaging.

Representations based on two dimensional views of the objects are an important alternative to 3-D models especially for the purposes of object recognition. The appearance of many objects depends more on the surface texture than the 3-D shape. The 2-D view representations can be treated directly as image templates, or they can be based on various types of image features. The features can be either holistic or point-like, and they can have various advantageous properties. For example, features have been proposed that are invariant or insensitive to rotations and scale differences [73]. The view based approach can be effective in object recognition, especially in standard views, but handling the correspondence between 2-D and 3-D representations is difficult. Treating objects as collections of separate views easily leads to redundancies and does not advance actual understanding of the structures of scenes. Methods for view interpolation [110, 86] and clustering [94] have been proposed to improve the trade-off between storage space and accuracy. Models that describe the mutual positions of landmark points, which can be recognized based on their features, add flexibility to the 2-D view representation and some degree of view interpolation can possibly be combined to such models [65].

Both view based and 3-D representation schemes lead to models that are highly specific to the target objects, so object recognition requires either systematic matching of all candidate object classes or another level of model hierarchy. The most suitable representation depends on the target object and the application. For example, 3-D models can be preferred for sculptures, which are seen in arbitrary poses and their appearance is dominated by the shape. For the recognition of faces from passport photographs, view-based approaches are likely to be more effective.

4.2 Detection and matching of object models

Object detection and matching are very broad subjects. In the model-based approach the problem is one of matching model-based descriptions of objects to image data, which is the result of light reflected from or emitted by the objects and projected onto the imaging device. Most solutions to object matching problems concentrate on highly specific applications and object types. Robust detection and matching require accurate model descriptions.

Approaches based on feature matching attempt to find a correspondence for individual feature points. Such features may be included in an object model. For example, the model may describe the positions of such feature points in the object. The matching is based on an error measure or a cost function that quantifies the match between the object and the image data. In the Bayesian approach, the target of the analysis is the probability of object model parameter values given the image data. This probability can be related with a cost function. It may be difficult to determine the correct probability model or cost function accurately. Sometimes simplified models are used due to this and also for computational efficiency.

Global approaches seek a transformation of the model such that the model best fits with the image. Effect of the background of the object on the image must be taken into account. If the object can be segmented from the background, the problem is greatly simplified. This is especially important if the model relies on the shape or appearance of the object's outline. Object detection techniques based on matching templates, grouped feature points or parametric models to the image typically concentrate on fitting the object, without much emphasis on the background. However, the environment of the object also affects the probability of object classification. For example, let us consider using a crescent-shaped model for what the moon looks like in a certain phase. If the model does not include any assumptions of the background, it would fit just as well into an image of a full moon. The rest of the moon would be left in the image without any explanation, so the probability of this interpretation for the image should be very low.

All the data in a digital image is in the pixel values but that is a very high-dimensional and low-level representation. Images can be processed in many ways to produce compact representations of attributes that can be more directly linked with properties of objects. Various methods have been developed for the extraction of robust features from an image, such as edges, homogeneous regions and points that match specific descriptions. Clearly multiple different types of features should be used together, but combining them is quite difficult. If different features are used for object matching sequentially, the results will depend on the weakest link in the chain. In theory, different information sources can be combined using Bayesian probability theory, but this requires knowledge of the conditional dependences between the information sources, which are difficult to estimate. The method called boosting to combine different classification techniques

has produced good results in object recognition [32, 114], but each classifier is used separately and the results are only combined afterwards. For optimal performance, different types of information should be exploited simultaneously, but there is currently no general solution to this.

4.3 MCMC sampling for 2-D rectangles

Right angles and straight lines are very typical features of man-made objects and environments. Rectangular structures of all scales tend to dominate indoor and outdoor environments created by man. Walls, doors and windows are most often rectangles. Importantly, these are very significant objects for navigation and map building. If all the rectangles in an indoor scene can be correctly detected, the problem of forming an interpretation of the area is greatly simplified.

In this section we propose an approach to finding rectangular objects in natural images. Relatively few studies on this subject have been published, although rectangular structures occur frequently in images. Rectangle detection has mostly been applied in the analysis of aerial images, where the targets of interest include objects such as houses and vehicles. In such cases, perspective transformations can be ignored. Our point of view is from scene analysis for robot navigation, so many assumptions cannot be made about the sizes of rectangles and perspective transformations need to be taken into account.

Hough transform is a traditional method for detecting objects with simple parametric descriptions such as straight lines [46]. Jung and Schramm [49] have proposed a method to detect rectangles by analyzing Hough transform spaces for lines in local image regions. Sizes of rectangles are limited by the fact that the method can only detect one rectangle per image region. A method to detect rectangular areas using a edge operator based rectangle model has been proposed by Moon et al. [83]. Their results showed relatively robust performance, but the method requires an exhaustive search over different rectangle sizes and orientations as well as perspective effects, which adds up to a heavy computational cost. The model does not take into account perspective transformations.

In our approach, we concentrate on vertically aligned rectangles, so two of the vertices of any rectangle an image are approximately vertical. This is used for initialization of the search, as follows. We use the Canny edge detector [16] to detect and trace edges in the image. From these we select those that are vertical by applying a vertically oriented filter to the edge map. Finally, we eliminate edges that are two short and parametrize the remaining edges by fitting a straight line through each of them. The process is illustrated in Fig. 4.1.

The Metropolis-Hastings algorithm is used to locate the rectangles. Two sampling chains are associated with each vertical line: the lines are used as both left and right vertical edges of potential rectangles. As the horizontal position of one

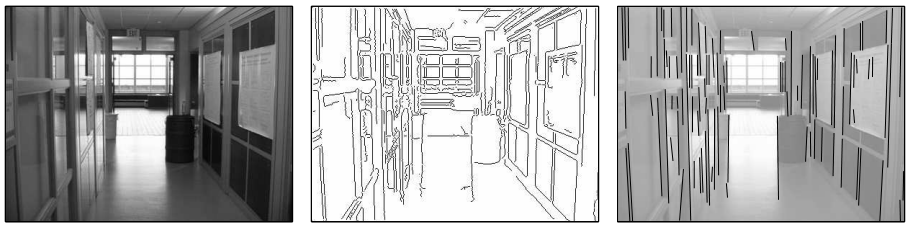


Figure 4.1: Example of the detection of vertical edges. Left to right: Original image, Canny edges and the final edge map.

edge is fixed, five additional parameters are needed to fully represent a vertical rectangle including perspective deformations. There are many alternative ways to choose those parameters; our choice is the following: the distance between the vertical edges w, vertical coordinates of the endpoints of the first edge relative to the initial estimates o_1^y and o_2^y, differences of the vertical coordinates of the other two corners from the first ones o_3^y and o_4^y.

Importance sampling is applied to the width parameter w, which determines the location of one vertical edge of the rectangle. The edge is assumed to appear in the vertical gradient image, so the marginal gradient histogram of that image is used to form the proposal distribution. The marginal histogram is computed in the image strip that extends from slightly below the bottom to above the top of the first edge. The proposal distribution goes to zero very near the first edge as well as near the edge of the image; this is a natural way to enforce these bounds. The importance distribution causes the samples to concentrate at strong vertical edge responses in the image and it significantly speeds up the search. The same strategy is applied to the endpoints of the first edge, o_1^y and o_2^y, based on the horizontal gradient image.

The likelihood function is based on the assumption that the rectangle forms a segment with a texture that is different from the texture of the segment's background. Thus we compare the texture of the rectangle and its surroundings. Three texture samples are used in the comparison. One of the samples is a strip next to the first edge, T_m, the other two are the entire rectangle and the 8 pixels wide area that surrounds the rectangle, T_f and T_b. The error function (negative log-likelihood) is computed by comparing the Kullback-Leibler distances between these areas, as follows:

$$l(r|I) \propto [D(T_m|T_f) - D(T_b|T_f)]^2. \tag{4.1}$$

Prior distributions for the parameters above are all normal distributions. The prior for the rectangle width w has a wide variance. The vertical offsets have zero mean, and standard deviations 10 % and 25 % of the initial length of the first edge for endpoints of the first edge and the opposite edge, respectively. In addition,

there are priors that control the size and shape of the rectangles and measure edge support in the image. The size prior is a broad log-normal distribution, and the shape prior is a Gaussian distribution on the ratio of lengths of the vertical edges with mean 1.0 and standard deviation 0.1. The edge prior measures the average edge response along the outline of the rectangle.

Sampling chains and results after convergence are shown in Fig. 4.2. The

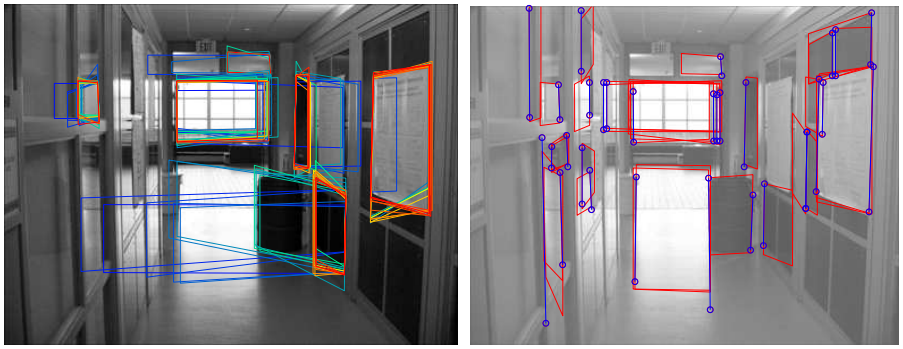


Figure 4.2: Left: Sampling chains for six different rectangles. The colors run from blue to red as sampling progresses Right: Final samples of 30 chains with highest posterior probabilities. The blue line segments indicate the corresponding edges used for initialization.

use of edge information in the proposal probabilities of the opposite vertical edge makes the sampling efficient. Many of the matches coincide with actual rectangular objects in the image. There is one solution for each of the vertical edge proposals, and not all of these can be correct. Thus the false positive rate is high unless some post-processing criteria are applied. The posterior probabilities of the objects are correlated with the correctness of the rectangles. More results are shown in Fig. 4.3.

4.4 MCMC sampling for 3-D objects

In this section we explore the possibilities of MCMC sampling for object detection and matching with different three dimensional object models. We present results of estimating model parameters based on the difference between the target image and the projection of the model. A model of a rectangular solid and a more general spline based volume model are considered.

4.4.1 Matching a rectangular solid using edge information

In this example we explore the use of edge information in matching a 3-D model with an image. The estimation is based on information about the location of one

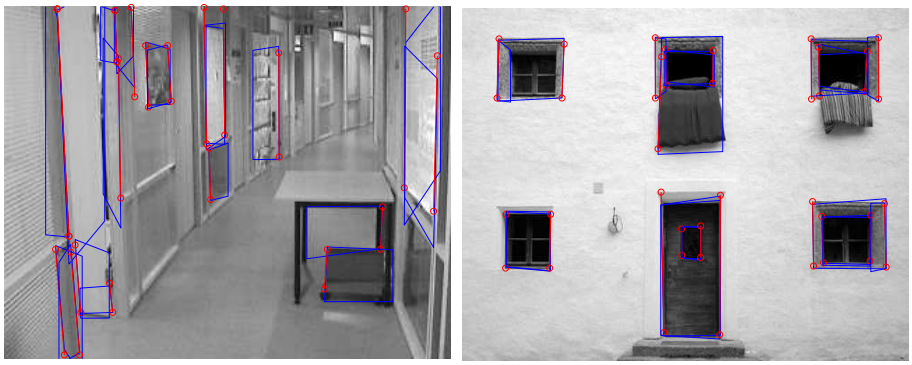


Figure 4.3: Example results of rectangle search.

of the edges. A different solution to the same problem has been proposed by Rao [97].

If the locations of some likely edges in an image are known, there is no effective way of finding out the parameter distribution of an object, given the information that those edges are part of the object. The difficulty is in the complicated relationship between the model parameters and the corresponding edge locations. This causes the parameters to have strong mutual nonlinear correlations. The 3-D model produces the locations of edges from given model parameter values, but the inverse relationship is intractable in anything but the simplest cases. Even for very simple models the possibility of rotation ensures that each edge location can be dependent on every single model parameter. That is why in the general case, the conditional model parameter distribution given the edge locations cannot be computed or sampled from. This means that Gibbs sampling, where samples are drawn from the conditional distributions, is not suitable for this case. Specific strategies for computing the conditional distributions can be designed for some simple object models, but they cannot be generalized to other models.

On the other hand, knowledge of the location of one model point in the image can easily be exploited in the fitting procedure by making sure that the same point in the model is projected to that location, for example using that information to define a tight prior distribution for the coordinates of the point in the Bayesian model fitting approach. Then the distribution of the rest of the parameters will be conditioned on that information. That is easy because only the translation parameters are involved. An example is shown in Fig. 4.4*d*. To include another parameter in the model is much more difficult because rotational degrees of freedom causes a dependence between all parameters.

We consider a wire-frame model of a rectangular solid that has seven free parameters: edge lengths, coordinates of 2-D projection and camera angles. The Metropolis-Hastings algorithm is used to estimate the position of such an object

in an image using edge information.

The likelihood function for a given position is computed as the product of pixelwise Gabor filter responses along the projection of the wire-frame onto the image. Filters of eight different orientations are used and the nearest orientation is chosen for each segment in the wireframe. A gamma distribution is used as the prior for the length of one edge to ensure that all edge lengths are non-zero. The camera angles have uniform prior distributions, and the rest of the parameters have Gaussian priors.

Results obtained with a test image of a relatively cluttered scene are illustrated in Fig. 4.4. Different initializations were used for the Metropolis chains. Two positions where most of the chains converged with small error are shown in image *a)*. The sketches represent medians of the parameter distributions obtained from two sample chains. Variance of the distributions was very small in this case. The cardboard box cannot be matched more accurately because the wire-frame model does not take perspective deformation into account. Images *b)* and *c)* display typical values from chains which failed to converge in the course of a few hundred samples.

In image *d)*, the location of the strongest edge observation was used as prior information of the location of one of the edges of the object. This was implemented by assigning the edge center a Gaussian prior distribution, centered on where the strongest edge filter response was obtained. In this case all sampling chains converged to the correct result. Our experiments indicate that use of such prior information normally yields savings in computation time, even if there are a number of different candidates for the edge location.

4.4.2 Estimation of shape using the generalized cylinder model

In this section we discuss the MCMC estimation of the shape of a model that describes generalized cylinders. In the model, the median axis is a linear combination of spline curves and the cross section is a circle or ellipse. The linear basis for the median axis is created from B-spline approximations to random curves. The random curves are generated from a Markov model, such that each increment to the curve is correlated with the previous increment. The median axis is a parametric curve in the sense that $x = x(z)$ and $y = y(z)$. The x and y curves share the same basis but have different parameters. One of the camera angle parameters becomes redundant because the object can be rotated by changing the scaling of the x and y curves. The radius can share the same model with x and y curves, although it is favorable that the radius is always positive. This model lends itself well to the representation of various man-made objects, especially if the definition of the cross-section is changed to allow rectangular shape.

In the estimation of 3-D shape we use a likelihood function which includes

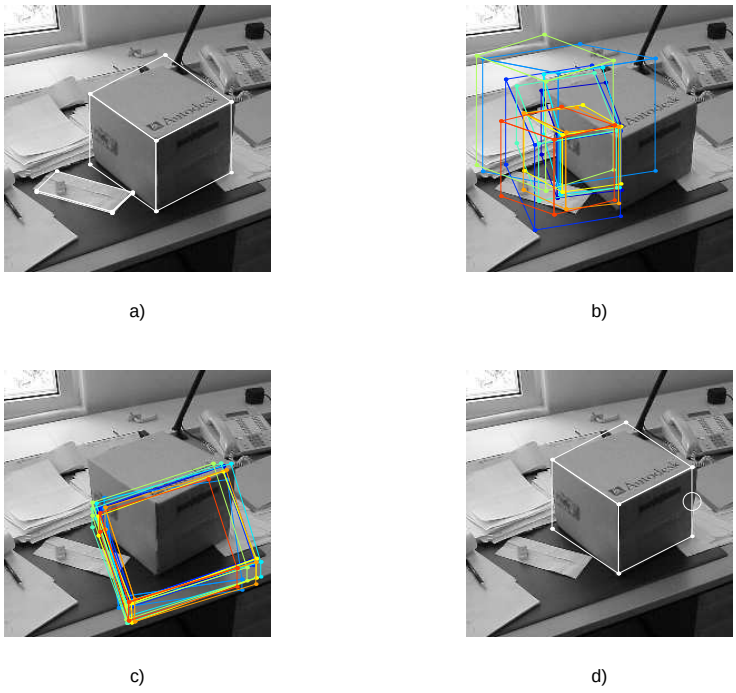


Figure 4.4: Estimating parameters of a rectangular solid based on edge information. *a)* Two modes of the distribution with small error. *b)* and *c)* Typical samples of MCMC chains which did not converge well. *d)* Median of distribution obtained using the assumption that the middle of one edge is located near the area marked by the circle.

two different error measures:

$$P(I|\theta) = \exp\left(-\frac{1}{\sigma^2}\left(\beta E_{edge} + \sum \|I_{ref} - I_m(\theta)\|^2\right)\right). \tag{4.2}$$

E_{edge} is a measure of edge match and the other error term is the pixel-wise intensity difference between the target image I_{ref} and the rendered model I_m. σ denotes the error variance and β is a constant. Since at this stage we do not attempt to estimate the texture of the object, the edge error term is allowed to dominate and the intensity error is basically only used to separate the object from the background. We compute the edge error from smoothed edge filter responses at vertices which lie on the edge of the rendered model. Whether the end points of the cylindrical object are visible or not can be computed using knowledge of the tangent vector of the model axis at the end points and the location of the camera. The edge vertices on the side of the model will be connected to triangles which are approximately

perpendicular to the camera angle. In the example illustrated in Fig. 4.5, we used Metropolis sampling to estimate 11 parameters: eight for the model, three for the camera angle and scale and two for the alignment of the target image with the image of the rendered model (translation). There were no local minima because the target object was well separated from the background. Since we did not use a prior which would favor symmetric or linear objects the distribution of the side axis profile is wider than that of the front profile. This means that there is more ambiguity in the side view of the object than the front view.

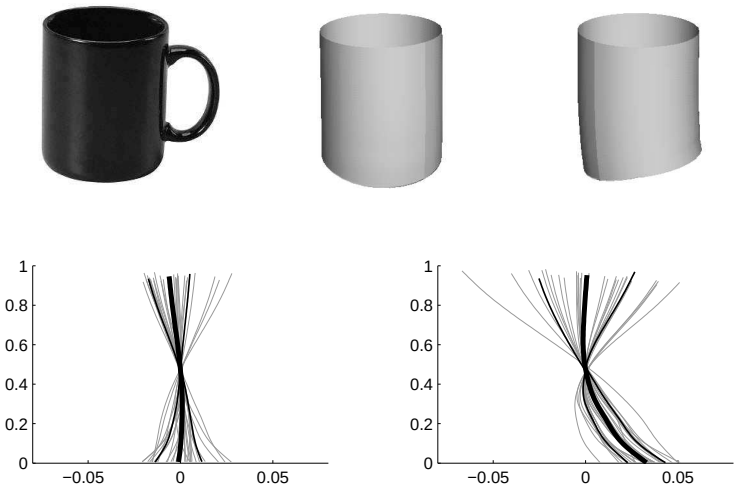


Figure 4.5: Results from fitting the generalized cylinder model to an image of a mug. Top left: target image. Middle: model which corresponds to the mean of estimated parameter distribution. Right: Side view of the same model. Neither direction of light nor texture was estimated in this example. The distribution of the estimated axis profile is shown in the bottom row; front view on the left and side view on the right. The black curves depict the mean and standard deviation of the distributions and the gray curves are a randomly chosen subset of individual samples. Note the stretched aspect ratio which makes the samples seem less linear than they actually are.

4.4.3 Texture estimation

Texture has a decisive effect on the appearance of the 3-D object. Texture accounts for details that are not explained by the model geometry and for changes in the surface color. Ideally the estimation of texture should be done jointly with the rest of the model parameters. This requires a relatively constrained model for the texture (or the object) because, as an extreme example, any scene could be explained

as a flat object where the texture accounts for every detail, like a photograph. Actually this is a good explanation and it is one mode of the posterior distribution. A realistic model for texture is difficult to define and therefore we have kept texture separate from the shape estimation. We first estimate the position, shape and orientation of the object and then, keeping those parameters constant and assuming a suitable lighting model, estimate the texture of the object.

Texture coordinates which establish the correspondence between the model geometry and the texture map are computed using projective texturing. The same 3-D-to-2D projective transformation that transforms the 3-D coordinates of the model to 2-D points on the screen is used to compute 2-D texture coordinates associated with each 3-D triangle in the model. Projective texture mapping can be performed efficiently using current graphics hardware. The approach is similar to [22], but only a single texture is used on the object.

Lighting is computed using a simplified Phong lighting model [31] with four parameters, including k_a, k_d, k_s which control the contribution from ambient, diffuse and specular terms and ν which controls the size of specular highlights. The color value of a triangle after lighting calculation is determined by

$$C_L = k_a + k_d(\vec{N}\cdot\vec{L}) + k_s(\vec{R}\cdot\vec{V})^\nu, \tag{4.3}$$

where $\vec{N}$ is the surface normal, $\vec{L}$ is direction of illumination, $\vec{R} = 2\vec{N}(\vec{N}\cdot\vec{L}) - \vec{L}$ is the direction of reflection, and $\vec{V}$ is the view direction.

The color value of a pixel is generated as a product of the lighting and texture values, $C = C_L C_T$. Texturing is implemented in OpenGL graphics library using the GL_MODULATE texturing function, and the multiplications are done rapidly with the graphics hardware. Given the perceived image I and estimates of the light direction and model geometry, the texture C_T that is required to produce image I is computed as

$$C_T = \frac{C}{C_L} = \frac{I}{k_a + k_d(\vec{N}\cdot\vec{L}) + k_s(\vec{R}\cdot\vec{V})^\nu}. \tag{4.4}$$

A non-zero ambient term k_a ensures that the texture estimate remains finite everywhere. An example of the texture estimation procedure is shown in Fig. 4.6. The shape of the object has been estimated using a generalized cylinder model and Metropolis sampling as in section 4.4.2. The illumination-corrected texture, estimated using eq. (4.4), is shown in Fig 4.6*b*. Note how the texture remains uniformly white regardless of the shading present in the target image. The black stripe on the rightmost edge of the mug appears because the shape estimate was not quite accurate. The form of the texture is completely unconstrained, and thus any texture that produces, together with the shape, an image similar to the target image is considered plausible.

a) b) c) d)

Figure 4.6: Estimation of shape and texture. a) Target image. b) Estimated, illumination-corrected texture. c) Texture mapped shape estimate. The dark artifacts near the edge of the mug appear because of inaccuracies in the shape estimation stage. d) New view and illumination.

4.5 Discussion

In this chapter Markov chain Monte Carlo sampling methods were applied to different object matching tasks. Parametric models of certain object types were matched to images to detect and identify objects.

First, the detection of rectangular objects under perspective transformations was studied. The results indicate that the approach has good potential for applications in this type of a problem. The search for such broadly defined object shapes in cluttered images is a computationally challenging problem, and the method is too slow to consider for real-time use, unless the problem were restricted to locating one rectangle per image.

The results of experiments in matching parametric 3-D models to images suggest that MCMC sampling techniques are a feasible solution in such tasks. The wireframe model for a rectangular solid is sufficiently specific to be feasibly applied in a cluttered environment. The 3-D volume model, on the other hand, requires that the target objects be distinctly visible. The main advantage of the Bayesian model-based approach is that models can be defined freely and they can be directly controlled using priors.

A negative aspect of the model based approach is that in order to be efficient, the the sampling techniques need to be specifically adapted for each model, and they cannot be easily extended or generalized for different object types. Instead, specialized models and sampling methods need to be crafted for each object type separately.

Chapter 5

Probability model for the self-organizing map

The topic of this chapter is the neural network model called the self-organizing map. The SOM is traditionally seen as a computational method for mapping high dimensional data vectors to a lower dimensional space. In contrast to this view, here we present the SOM as a generative probability density model in the maximum likelihood framework. We first give a review of the SOM learning algorithm, its properties and its applications. Next we derive the probability density model that can be associated with the SOM. We present the theoretical and practical properties of the density model and its application to model selection and conclude with a discussion of its significance.

The self-organizing map [58] is an unsupervised learning algorithm which seeks to maintain a certain topology among prototype vectors arranged on a regular grid while letting them adapt to input data. In the input space, the model produces a vector quantization of the data, and a mapping between the data samples and the discrete map is established. Topology preservation means that data samples close in the input space are mapped onto nearby units in the SOM lattice and vice versa [113].

There are several benefits in associating a probability density, or a generative model, with a mapping method (see the discussion by Tipping and Bishop about a generative model for principal component analysis [107]):

- The density model enables computation of the likelihood of any data sample (training data or test data), facilitating statistical testing and comparison of the SOM with other probabilistic techniques.

- Parameters of the SOM that control the model's complexity or stiffness can be optimized to ensure best fit to data using standard statistical methods, such as cross-validation.

- The density model facilitates quantitative analysis of the model, for example, by computing conditional densities to test hypotheses found by visual data analysis.
- The probabilistic interpretation offers a new view into the SOM algorithm, which may aid in understanding its behaviour, its capabilities and its limitations.
- In principle, the probability model enables the use of Bayesian methods for model complexity control and model comparison. However, as shown later, the normalization of the probability density in the original SOM requires a numerical procedure that seems to render the Bayesian approach impractical.

5.1 SOM algorithm

The self-organizing map consists of a regular grid of map units that are tied together by their neighbourhood relationship. In the input space, each map unit is represented by a prototype vector. During SOM training, the prototype vectors are adapted to the data. The part of the input space that is mapped into a map unit is called the receptive field of that unit.

5.1.1 Sequential algorithm

In the basic SOM algorithm [58], input data samples are presented sequentially. The updating step begins by finding the map prototype vector m_b that is nearest to the presented input vector x_n – its *best matching unit* (BMU). The matching criterion is Euclidean distance. The BMU is determined by the condition

$$\|x - m_b\| = \min_k \{\|x - m_k\|\}. \tag{5.1}$$

The prototype vectors are displaced toward the data sample a distance that depends on the topological relationship between them and the BMU. The update equation is written as

$$m_k(t+1) = m_k(t) + \alpha H_{bk}(t)[x(t) - m_k(t)]. \tag{5.2}$$

The parameter α is called the learning rate factor and it usually decreases monotonically during training. The neighborhood function H_{bk}, defined in the map lattice, is responsible for the topology preservation. The principle is that units close to the BMU in the map lattice are updated along with the BMU. The radius or width of the neighborhood function is also meant to reduce as training

progresses, enabling increasingly local adaptation. The most typical choice of the neighborhood function is the Gaussian function

$$H_{bk} \propto \exp\left(-\frac{\|r_b - r_k\|^2}{2\sigma^2}\right), \tag{5.3}$$

where r_b and r_k are the coordinates of the best matching unit and the unit k on the map lattice, respectively. The parameter σ controls the width of the neighbourhood. Reducing the size of the neighborhood is not related to convergence of the algorithm; the SOM algorithm will converge at any neighbourhood size. Gradual reducing of the neighbourhood improves chances of good topological ordering in the map.

5.1.2 Batch algorithm

Another method of training the SOM is known as the batch map algorithm [58]. It consists of the following steps:

1. Assign initial values to the prototype vectors.
2. Let the new value of each prototype vector be the mean of the input samples weighted by the neighborhood function

$$m_k(t+1) = \sum_n H_{bk}(t) x_n, \tag{5.4}$$

   with the BMU taken to be the BMU of x_n. H_{bk} must be normalized such that $\sum_n H_{bk} = 1$.
3. Repeat from step 2 until convergence.

The size of the neighborhood is reduced in the course of training as in the sequential algorithm. The batch algorithm is simpler than sequential training because there is no learning rate parameter.

5.2 Error functions

The SOM algorithm is not defined in terms of an error function but directly via the training rule, and the training rule is not a gradient of any global error function [28]. This makes the mathematical analysis of the SOM algorithm fairly difficult. However, the converged state of the SOM is a local minimum of the error function which is given by [99]

$$E(X) = \sum_{n=1}^{N} \sum_{j=1}^{M} H_{bj} \|x_n - m_j\|^2, \tag{5.5}$$

where $X = \{x_n\}, n = 1, \ldots, N$ is the discrete data sample, j is the index (or position) of a unit in the SOM, with prototype vector m_j, and $H_{bj} = H(b(x) - j)$ is the neighborhood function, $b(x)$ being the index of the best matching unit for x.

When a sample is near a boundary of two or more receptive fields, a small change in the position of one prototype vector can change the best matching unit of that sample. Thus the gradient of the error function with respect to the prototype vectors is infinite. The error function is not defined at the boundaries of the receptive fields, or Voronoi cells [58], so the function does not exist in continuous input space. In the context of a discrete data set, the probability of any sample lying at any boundary is zero (tie rules can be applied), so in the error function can always be computed for any real data set. The error function changes if any sample changes its best matching unit. That is why the error function is only consistent with the SOM training rule when the algorithm has converged. In practical data analysis the data set is always discrete and the algorithm is allowed to converge, so analysis of the error function is thereby justified.

The error function (5.5) can be thought of as the sum of the *distortion function*

$$D(x) = \sum_j H_{bj} \|x - m_j\|^2, \tag{5.6}$$

computed over the data set. This is the sum of distances of a data point to all prototype vectors that are in its topological neighborhood, weighted by the values of the neighborhood function. The distortion function is also discontinuous at the Voronoi cell boundaries. The discontinuity is a result of the winner selection rule of the training algorithm. Luttrell [74] has shown that exact minimization of equation (5.5) leads to an approximation to the SOM training rule, where, instead of the nearest neighbor winner rule, the best matching unit is taken to be the one that minimizes the value of the distortion function (5.6), as follows:

$$b(x) = \operatorname*{argmin}_i \sum_j H_{ij} \|x - m_j\|^2. \tag{5.7}$$

The minimum distortion rule avoids many theoretical problems associated with the original rule, without compromising any desirable properties of the SOM except for an increase in the computational burden [43]. The gradient of the error function becomes continuous at the boundaries of receptive fields (which are no longer the same as the Voronoi tessellation). The distortion function with the modified winner selection rule also becomes continuous across the unit boundaries.

5.3 Relation to constrained mixture density models

The main aspects of the SOM algorithm that make its analysis difficult are 1) the hard assignment of the input samples to the nearest units, which makes the

receptive fields irregularly shaped according to the Voronoi tessellation, and 2) the regularizing effect defined in terms of updating the prototype vectors of the neighboring units towards each input, instead of regularization applied directly to the positions of the prototype vectors.

The first issue is due to the shortcut algorithm [57] devised to speed up the computation and to enhance the organization of the map from a completely random initial state, and it is not a characteristic of the assumed model for the self-organization in biological networks. The original on-center off-surround lateral feedback mechanism in [57] produces a possibly multimodal pattern of activity on the map, which is in the SOM model approximated by a single activity bubble around the best-matching unit, shaped by the neighborhood function. An equal Hebbian learning rule in each unit produces the concerted updating towards the input in the neighborhood of the BMU[58]. An interpretation of the mapping of an input data point to the SOM lattice that is consistent with the biological model would be to map the input point to an activity bubble H_{ib} around the BMU b instead of just one single unit. This is the same as using the minimum distortion rule (5.7) instead of the shortest distance (5.1).

By ignoring the winner-take-all mechanism, the SOM can be approximated by a kernel density estimator, where the activity of a unit is only dependent on the match of the data point and the unit prototype vector. This is often called a soft assignment of data samples to the map units, in contrast to the hard assignment of a data point to one single unit.

The second characteristic of SOM training, the way of constraining the unit positions, is dictated by the biological origin of the method. A regularizer directly on the prototype vector positions would require a means for the neurons to update their weights towards the weights of the neighboring neurons, while in the SOM rule all learning is towards the input data. The biological plausibility is obviously non-relevant in data analysis applications, even though it may have a role in building a larger neural system with the SOM as a building block.

Utsugi has proposed a Bayesian model in which small approximations are made to render the SOM more easily analyzable: the winner-take-all rule is replaced by a soft assignment, and the neighborhood effect is approximated by a smoothing prior directly on the prototype vector positions [112]. The model is thus a Gaussian mixture model with kernels constrained by the smoothing prior. This approach yields a very efficient way to set the hyperparameters of the model, that is, the widths of the kernels and the weighting coefficient of the smoothing prior, by an empirical Bayesian approach. For any values of the hyperparameters, the evidence, or conditional marginal probability of the values given the data and prior distributions, can be computed by integrating over the posterior probability of the model parameters (kernel positions). The values with maximum evidence are then chosen as the most likely values.

Another model close to the SOM is the Generative Topographic Mapping pro-

posed by Bishop et al. [9]. In that approach, the Gaussian mixture density model is constrained by a nonlinear mapping from a regularly organized distribution in a latent space to the component centroids in data space. Hyperparameters of the model, which control noise variance, stiffness of the nonlinear mapping and the prior distribution of mapping parameters, can be optimized using Bayesian evidence approximation [11], similar to the one used by Utsugi.

5.4 Assessing the quality of maps

Factors that affect the training process are the topology of the map, initial values of prototype vectors and training parameters, the most important of which is the size of the neighbourhood. The choice of topology depends on the application, and initial values are not a proper way to control the final state of a learning algorithm. The width of the neighbourhood controls the degree of localization of the map units and determines the amount of interaction between map units. In the beginning of training, the neighbourhood is wide and prototype vectors are affected by data points that belong to the receptive fields of topologically remote units, hiding local variations – the model is underfitted to data. As the neighbourhood reduces, regularization of the model is diminished and at some point, depending on noise and the abundance of samples, some prototype vectors may be determined by too few samples, in which case the model becomes overfitted. To enable the generalization of the model to new data, an optimum should be found between underfitting and overfitting.

The SOM is seen as producing a mapping from high-dimensional data to a low-dimensional manifold. The topology of that manifold must be specified before SOM training (one or two dimensional topologies are common). The topology should be suitable for the distribution of the data, and it should be ensured that the map is organized according to the structure of the data.

Research on quantifing the adaptation of the SOM to data is mostly based on measuring how well the topology is preserved in the map [7, 113]. Kaski and Lagus [51] have proposed a heuristic measure of map goodness that is based on the topological relationship between the two prototype vectors that are nearest to each data point. None of these measures have gained much popularity among publications on practical applications of the SOM.

5.5 Applications of the SOM

Since the introduction of the self-organizing map algorithm in 1981, it has found applications in many areas ranging from engineering sciences to medicine, biology and economics. Image and video processing and pattern recognition have been among the most popular research topics [50, 90].

5.5.1 Exploratory data analysis

Many applications are based on using the SOM for exploratory data analysis, where the ambitious goal is to search for non-linear statistical dependences between multiple variables. The clustering property of the SOM together with the topological ordering provide unique possibilities for the visualization and interpretation of high-dimensional data distributions [20].

A basic task in practical data analysis problems is to search for dependences between variables. A statistical dependence means that the conditional probability distribution of one variable is dependent on the values of other (explanatory) variables. Thus the analysis of dependences is closely related to the estimation of variables' joint probability density and conditional probability densities.

A common procedure for using the SOM in exploratory data analysis involves fitting a two-dimensional map to data of higher dimensionality. A component level representation is used for visual inspection of the map. The projections of all SOM units on each data variable are displayed as cell patterns, with each cell colored to reflect the variable's value in the map unit associated with that cell. See Fig. 5.1 for an example. In a topologically ordered map, the values of all components will change smoothly with respect to map coordinates. One searches for regions on the map where there are constant values in two or more variables. Such a region is interpreted as a hypothesis that the variables are dependent such that, for example, a low value of one variable indicates a low value for the other variable, assuming that the rest of the variables are close to the corresponding values in the prototype vectors.

A caveat in the approach is that what is seen in the component level display is the model of the data, not the data itself. If the model produces a poor description of the data, then any interpretation made based on the display will be subject to error. Overfitting is a key source of generalization error in statistical models. In the case of the SOM, overfitting means that prototype vectors of some units have been determined by too few data points, and therefore they are not representative of the underlying probability density. Conclusions based on such units are not likely to generalize to new data. Of course, this only applies when analyzing a finite sample from a population.

In the literature on the SOM, very little has been written about the selection of model complexity. The main problem in visual inspection of the SOM is that in general the lack of dependence between variables is difficult to observe visually. That is, even if two variables, say, x^1 and x^2 both have high values at map unit m_{ij}, that alone is not enough to show that the variables have any mutual dependence. As a simple example, consider a two-dimensional uniform distribution $x_1, x_2 \sim U(-1,1)$. A 2×2 SOM with zero neighborhood would have component planes

(in any order of the columns and rows)

$$\mathbf{m}^1 = \begin{bmatrix} -0.5 & -0.5 \\ 0.5 & 0.5 \end{bmatrix} \qquad \mathbf{m}^2 = \begin{bmatrix} -0.5 & 0.5 \\ -0.5 & 0.5 \end{bmatrix}$$

The coincidence of high values in unit m_{22} and low values in m_{11} are only a result of the vector quantization. To see that high values in m_{22} do not indicate dependence between x^1 and x^2, one must observe that a high value for x^1 also occurs in m_{12} with low value for x^2 (i.e., tallied over the map, high values for x^1 co-occur with both high and low values for x^2).

In a high dimensional space, visual inspection of the dependences becomes more difficult, as the map folds into the data space, and the range of values for each variable is distributed around the map. Fig. 5.1 illustrates this for random data with no dependences between the variables; the SOM was trained on three random variables, $x \sim N(0,1)$. In Fig. 5.2, an example of a real data analysis case is shown, where all hypotheses made from SOM maps were later rejected in statistical testing.

5.5.2 Computer vision and robot navigation

Image processing has been one of the most active application areas of the SOM algorithm. Most often the SOM is used for classification of pre-processed image features [116, 4].

In landmark-based navigation tasks, the detection and identification of landmarks is among the key problems. The SOM has been used as a basic pattern recognition tool for this [45, 48]. Gerecke et al. [36] have proposed a method to combine an ensemble of self-organizing maps for the landmark localization task.

A very prominent feature of the SOM is the topological ordering. It seems natural to try to use this property in map building for navigation, such that spatial neighbourhood relations would be encoded as connections between neighbouring map units. A problem is the fact that the topology of the SOM is not adaptive but it must be specified before training. Some variants of the SOM algorithm have been introduced to overcome that limitation. Fritzke [33] has proposed a growing self-organizing network structure based on the neural gas network by Martinetz and Schulten [81], which appears to produce better results [5]. When compared to alternative localization methods, the advantage of the neural gas is a very low computational cost, but it is not as good in terms of accuracy [24].

5.6 Probability density model for the SOM

In this section we derive the probability density model for the original SOM in the maximum likelihood framework, without any approximations to the neighborhood regularization mechanism.

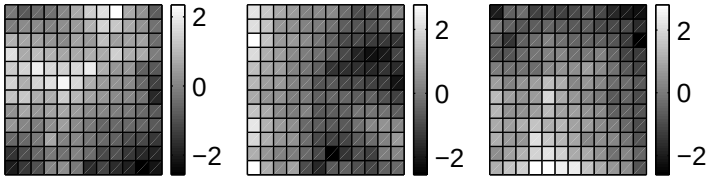


Figure 5.1: Example of a SOM trained on purely random data, with zero neighbourhood. Values of the three components of reference vectors are shown side by side. Independence of the variables is not trivial to observe in the component level display. For example, one might erroneously conclude that high values of the rightmost variable would indicate low values of the variable in the middle.

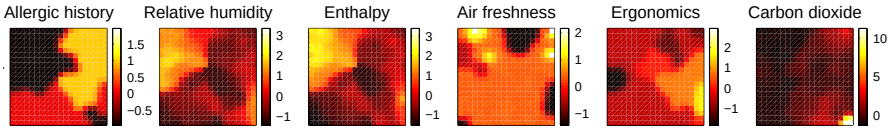


Figure 5.2: Example of real data analysis. In the case study, the dependence of *Air freshness* on the other variables was investigated with the help of SOM maps. In the final analysis, all hypotheses obtained using the SOM were rejected using methods like RBF models, Bayesian neural networks, and hierarchical generalized linear models using Bayesian inference, etc. [115]. One evident conclusion, from the lower right corner of the map, is that high value for variable *Ergonomics* appears only with low value for *Air freshness*, but careful analysis showed that this was just an effect of vector quantization.

The probability model is based on the mean-of-squares type error function (5.5), discussed in section 5.2. The error function is specific to the given neighborhood parameters, so it cannot be directly used to compare maps that have different neighborhoods. The maximum likelihood (ML) framework is based on maximizing the likelihood of data given the model. We want to find a likelihood function that is consistent with the error function. This can be achieved by making the error function proportional to the negative logarithm of the likelihood of data. Assuming the training samples x^n independent, the likelihood of the training set $X = \{x^n\}, n = 1, \ldots, N$ is the product of probabilities

$$p(X|\mathbf{m}, H) = \prod_n p(x^n|\mathbf{m}, H), \tag{5.8}$$

where $\mathbf{m}$ denotes the codebook (set of prototype vectors) and H is the neighborhood. The negative log-likelihood is $L = -\log p(X|\mathbf{m}, H)$ and setting it propor-

tional to Eq. 5.5 yields

$$p(X|\mathbf{m},H) = Z' \exp(-\beta E) = Z' \exp(-\beta \sum_n \sum_j H_{bj} \|x^n - m^j\|^2). \tag{5.9}$$

Here we have introduced two constants, Z' and β, which are not needed in the ML estimate of the codebook m but which are necessary for the complete density model. The probability density function of the input space that is equivalent with (5.9) is given by

$$p(x|\mathbf{m},H) = Z \exp(-\beta \sum_j H_{bj} \|x - m^j\|^2), \tag{5.10}$$

which is a product of Gaussian densities centered at m_j, with variances inversely proportional to the neighbourhood function values H_{bj}. Note that the discontinuity of the density is due to the discontinuity of the best-matching unit index b for the input x.

Within the receptive field, or Voronoi cell, of unit m_b, the density function has Gaussian form:

$$p(x|x \in V_b) = Z e^{-\beta W_b} \exp\left(-\frac{1}{2s_b^2}\|x - \mu_b\|^2\right), \tag{5.11}$$

where V_b denotes the Voronoi cell around the unit m_b. The mean and variance of the Gaussian kernel are denoted by μ_b and s_b^2, respectively, and W_b is a weighting coefficient. The values of these parameters are

$$\mu_b = \frac{\sum_j H_{bj} m_j}{\sum_j H_{bj}} \tag{5.12}$$

$$s_b^2 = 1/(2\beta \sum_j H_{bj}) \tag{5.13}$$

$$W_b = \sum_j H_{bj} \|m_j - \mu_b\|^2. \tag{5.14}$$

Thus the density model consists of spatially truncated Gaussian kernels, which are centered at the neighborhood-weighted means of the prototype vectors and clipped by the Voronoi cell boundaries. The parameter W_b controls the height of the kernel; it depends on the density of the neighboring prototype vectors near the centroid μ_b. The density function is discontinuous at the boundaries of the Voronoi cells. See Figs. 5.3, 5.4 and 5.6 for examples of the density models.

The variances of the kernels depend on the parameter β and they are equal if the neighborhood is normalized (as will be explained in section 5.6.1). In the standard SOM formulation, the border units with incomplete neighborhood have larger variances, as can be seen in Figs. 5.5 and 5.6, allowing the map to shrink into the middle of the training data distribution.

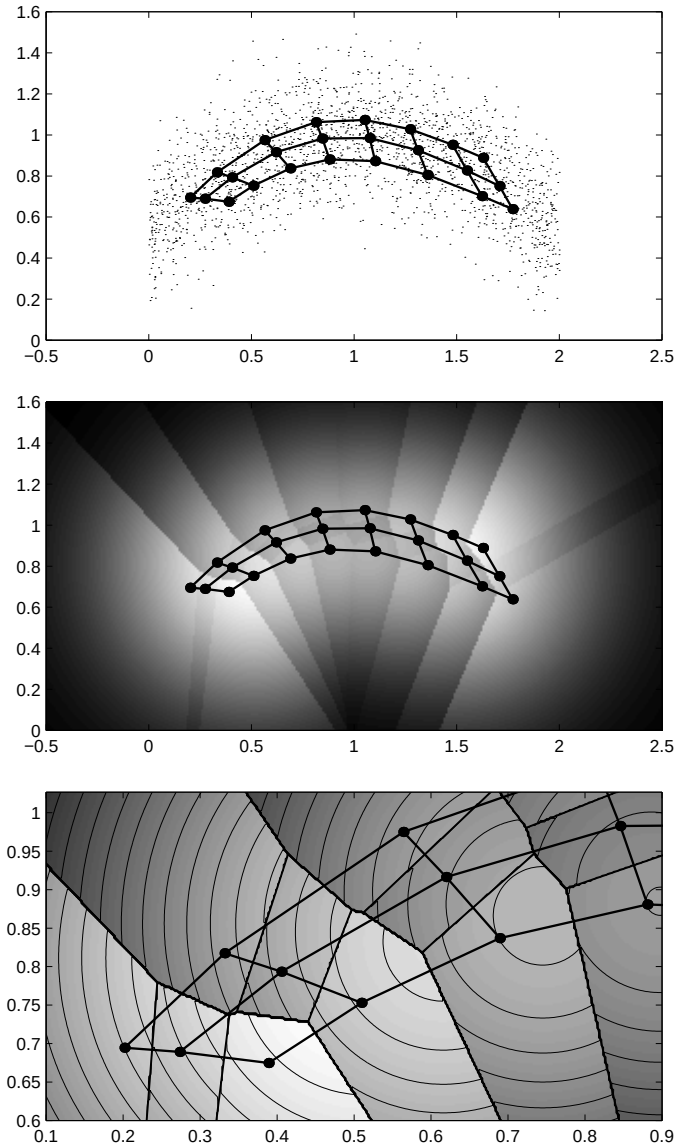


Figure 5.3: Example of the density model in a 3×8 SOM. Top: training data and the resulting SOM lattice using Gaussian neighborhood with $\sigma = 1$. Middle: the density model of the SOM. Bottom: a detail of the density model above, with contours and Voronoi cell boundaries added. Here it can be seen that the Gaussian kernel centers do not coincide with the prototype vectors.

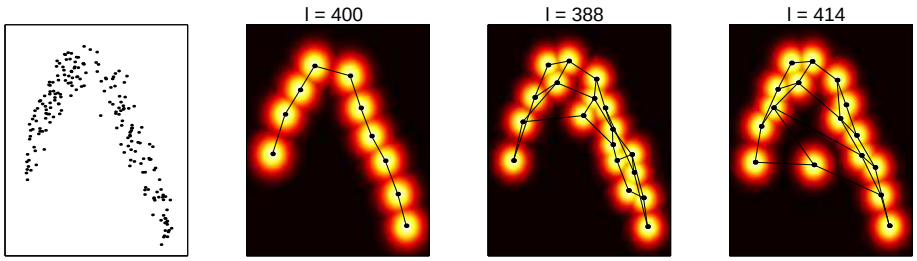


Figure 5.4: Training data and density estimates due to different SOM topologies (1×10, 3×6 and 4×4). $l = -logp(X|\mathbf{m}, H)$ denotes the negative log-likelihood of test data. The optimal value for the parameter β (which controls the widths of the density components) is so small that the model is close to an additive Gaussian mixture density. Among these alternatives, the 3×6 topology (middle) produces the best model according to the ML criterion.

The normalizing constant Z and the noise variance parameter β are bound together by the constraint that the integral of the probability density over the data space must equal unity. The constraint can be written as

$$\int p(x|\mathbf{m}, H)\mathrm{d}x = Z \sum_r e^{-\beta W_b} \int_{x \in V_r} \exp(-\frac{1}{2s_r^2}\|x - \mu_r\|^2)\mathrm{d}x = 1, \tag{5.15}$$

where the integration over the data space is decomposed to the sum of integrals over each Voronoi cell V_r. From this equation, Z can be solved for a given value of β. The integrals cannot be computed in closed form, but they can be approximated numerically using Monte Carlo rejection sampling. A simple solution is the following algorithm:

For each cell r,

1. draw K samples from the normal distribution $N(\mu_r, s_r)$
2. For each sample, test if it is inside the cell r.
3. Compute the acceptance ratio $q_r = K_r/K$, the fraction of the samples that are inside the cell r.
4. The integral over V_r in (5.15) equals $q_r(2\pi s_r^2)^{d/2}$, where d is the dimension of the data space.

For a map that contains M units, this algorithm requires the computation of distances between $M \times K$ samples and M prototype vectors. Thus if M is large the computational cost of the normalization procedure exceeds that of the training algorithm itself.

In an efficient implementation the number of samples K should be chosen according to the desired accuracy. The acceptance ratio q_r varies in a large range according to the neighborhood size. When the neighborhood is small, the neighborhood-weighted center μ_r is close to the prototype vector m_r and s_r is likely to be small, so q_r is high. When the neighborhood is large, the situation is opposite and K will have to be much greater to achieve an equivalent accuracy. A detailed analysis of the dependence of the accuracy on K is presented in publication [62].

The maximum likelihood estimate for β can only be found by numerical optimization, for example by maximizing the likelihood of a validation data set. The normalization procedure described above must be carried out for each candidate value of β, so the computational cost is quite significant. If an optimization method such as golden section search [95] is applied, savings can be made by allowing the accuracy to vary. Initial estimates can be very coarse, corresponding to a small number of MC samples K, if the accuracy gradually increases towards the convergence of the search. The final accuracy should reflect the size of the validation data sample.

5.6.1 Border effects

In typical implementations of the SOM, the neighborhood function is the same for each map unit. This causes problems near the borders of the map, where the neighborhood is truncated on one side. The effect is that data samples in the receptive fields of border units are given less significance than those that belong to more central units [56]. As can be seen from equation (5.13), the density components associated with units close to the border have larger variance and data points are allowed to reside farther away from the map units. Consequently, the border units are pulled towards the center of the map, and the map does not extend close to the edges of the input distribution until the neighborhood is relatively small and the regularization is loose. This leads to a decrease in the likelihood of a map with a large neighborhood (or an increase in the quantization error), biasing the optimal width of the neighborhood toward smaller values.

This bias can be reduced by normalizing the neighborhood function at the edges of the map [56]. In the case of the sequential algorithm, it suffices to normalize the neighborhood function in (5.2) such that its sum is the same for all best matching units:

$$\sum_k H_{bk}(t) = H_s, \forall b. \tag{5.16}$$

For the batch algorithm, the portion of the neighborhood function that gets clipped off due to the finite size of the map lattice can be transferred to nearest edge units. Normalization of the neighborhood function is of particular importance, if the minimum distortion rule (5.7) is applied to winner selection. As can be

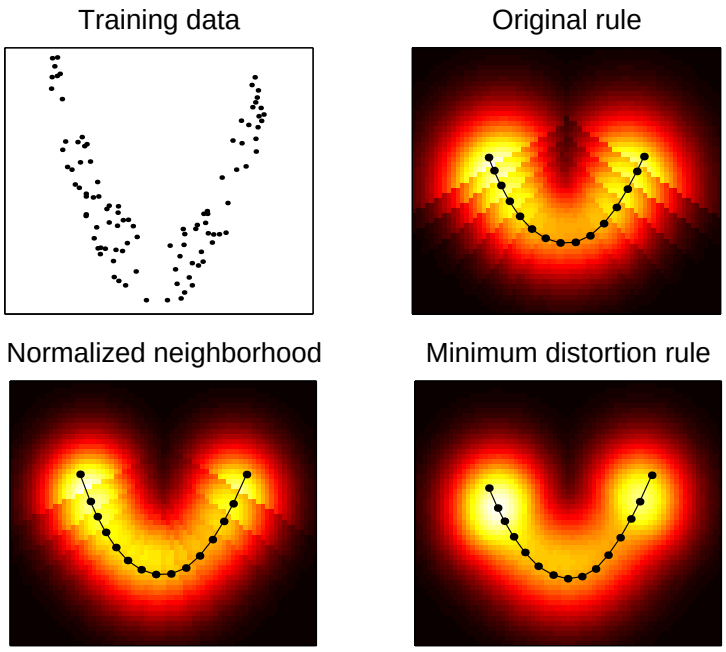


Figure 5.5: Example of the effect of the minimum distortion training rule on the density model. The neighborhood width is the same $\sigma = 2.8$ for all the maps. (This value is greater than the optimal value, as the purpose of this example is to highlight differences between these cases.) Note that the density kernel centers do not coincide with the prototype vector positions. Especially in the middle of the maps the kernels are shifted upwards from the unit positions.

seen from equation (5.13), when the sum of the neighborhood function is constant throughout the map, all cells have equal noise variance.

In practice we find that the minimal distortion rule will produce very similar results as the original rule in terms of model selection. The equalization of the neighborhood volume near map borders smoothes out the density model somewhat, due to a better fit of the data in a map with a larger neighborhood. The differences are illustrated in Fig. 5.5. The continuous density model, stemming from the minimum distortion rule, has many favourable qualities [43].

5.7 Model selection

The SOM algorithm produces a model of the input data. The complexity of this model is determined by the number of units and the width of the neighborhood,

which has a regularizing effect on the model. When the input data is a sample from a larger population, the objective is to choose the complexity such that the model generalizes as well as possible to new samples from that population. The maximum likelihood framework provides a consistent way to compare the fit of different models. In this section we discuss how it can be applied to the self-organizing map. The examples in this section have been computed using the standard SOM definition, including the nearest neighbor winner rule and the boundary bias.

Let us first regard the number of units as given, so the neighbourhood width σ is the only control parameter. The probability model allows us to select an optimal value for the neighborhood width σ by maximizing the likelihood of data $p(X|\mathbf{m},H)$. In the course of SOM training, σ gradually decreases in some prespecified manner, such that $\sigma = \sigma(t), t = 1,\ldots,K; \sigma(t+1) < \sigma(t)$. We assume that the training algorithm will find an ML estimate for the map codebook $\mathbf{m}(t)$ for each value $\sigma(t)$, if it is allowed to converge at every step. To construct the density model for each of these K candidate maps, we numerically optimize $\beta(t)$ for each map. This yields K different density models to compare. To choose between these we compute the likelihood values $p(X_V|\mathbf{m}(t),\sigma(t),\beta^{\mathrm{ML}}(t))$ for validation data X_V, by substituting into (5.10). Obviously, X_V should be different from the data that is used to select $\beta^{\mathrm{ML}}(t)$. Other statistical methods such as cross-validation can also be applied. An example of model selection is shown in Fig. 5.6. The map with neighbourhood $\sigma = 1.00$ maximizes the likelihood of validation data. This approach extends directly to the comparison of different map sizes and topologies (see Fig. 5.4). The computational cost is linearly related to the number of units on the map; in the case of a large map, it may be advisable to first find the correct σ for a smaller map and then to scale it up in proportion with the map dimensions. (For example, if $\sigma = \sigma_K$ is the optimal neighborhood width for a $K \times K$ map, then $\sigma = 5\sigma_K$ is likely to be a reasonable value for a $5K \times 5K$ map.)

Because the exact value of the density function cannot be directly evaluated, it is difficult to apply methods such as Bayesian evidence to parameter selection. The numerical normalization procedure causes such methods to be computationally too expensive to be practical.

A common application of the SOM is to look for dependences between variables by visual inspection. In that context, the density model can be used to select the complexity of the model, but it also enables quantitative analysis. Regression or conditional expectations can be computed directly from the joint density (5.10) by numerical integration. For example, the conditional distribution for variable x_j equals

$$p(x_j|x_{\setminus j},m,H) = \frac{p(x|\mathbf{m},H)}{\int p(x|\mathbf{m},H)\mathrm{d}x_j}, \tag{5.17}$$

where $x_{\setminus j}$ denotes the vector x with dimension j excluded. Likewise, the regres-

sion of x_j on other variables can be computed as the conditional mean $E[x_j|x_{\setminus j}, \mathbf{m}, H]$. It should be noted that the SOM density model may not give the best possible description of the input distribution. The purpose of this discussion is to illustrate the significance of model selection in data analysis.

An important special case of the density model is when the variance parameter is reduced to zero, $\beta \to \infty$. This causes the conditional density of the "output" variables x_j to be sharply peaked at the best matching unit for the "input" variables $x_{\setminus j}$. The conditional mean $E[x_j|x_{\setminus j}]$ then gives the same value as nearest neighbor (NN) regression [21] with the neighborhood-weighted SOM prototype vectors (5.12) as output values, producing a piecewise constant estimate. Comparison with the NN rule is interesting, because it is a close quantitative counterpart of visual analysis of the SOM.

Fig. 5.7 illustrates the difference between computing the conditional mean from the density model and using the nearest neighbor rule. A random data set of three normally distributed variables ($x_1, x_2, x_3 \sim N(0,1)$) is analyzed by a 6×6 SOM. We attempt to infer $E(x_2|x_1, x_3 = 0)$, the expected value of the variable x_2 given x_1, with x_3 equal to zero. As the variables are truly independent, the result ought to be $E(x_2|x_1, x_3 = 0) = E(x_2) = 0$. The optimal width of the Gaussian neighborhood function is $\sigma = 4.2$, which is a relatively large value, suggesting that the distribution has a simple form. At zero neighborhood, the model is severely overfitted. Clearly, neglecting to select the correct model complexity would give unreliable results. When the complexity is right, the nearest neighbor rule can produce a good approximation of the conditional mean, though the lack of confidence intervals limits the reliability of analysis.

An example of using the conditional distributions is shown in Fig. 5.8. The neighborhood width was chosen based on the maximum likelihood of test data. The data set is three dimensional. There is a dependence between two of the variables x_1 and x_2, as follows:

$$x_2 = B\sin(\omega x_1) + \varepsilon, \tag{5.18}$$

where $B = 1.5$ and normally distributed noise $\varepsilon \sim N(0, 0.2)$. The third variable is independent from the others ($x_3 \sim N(0,1)$). This kind of a distribution can be easily modeled by means of a two dimensional SOM. That is why no severe overfitting is observed and a small neighborhood width gives the best fit to test data. Yet it is not easy to observe the dependence from the component level display. The conditional densities, on the other hand, are easy to interpret. Nearest neighbor regression also gives relatively accurate results, because the model complexity is correct.

By visual inspection of the map it is difficult to perceive the mean or the shape of the conditional distributions. Therefore results may be very unreliable. Choosing parameter values to optimize the density estimate does not necessarily

result in a mapping that is optimal for visual display. However, the examples in Figures 5.6 and 5.7 indicate that the results of ML model selection tend to agree with intuition. In any case, the results of visual inspection should be validated by other, more reliable techniques.

5.8 Discussion

In this chapter, we have presented the probability density model that is associated with the self-organizing map model. We have also discussed certain difficulties that arise in the application of the SOM to statistical data analysis and shown how the density model can be used to alleviate them.

The determination of correct model complexity is an essential part of statistical modeling. In the analysis of a noisy data sample, the self-organizing map is used as a statistical model, so the same applies to it, too. Previously, no principled method of model selection has been proposed for the original SOM model. The probability density model enables model selection for the SOM in the maximum likelihood framework and thus fills that need. The parameter search involves a considerable increase in computational cost, but that is the case in all statistical modeling.

It should be stressed that although the density model is based on an error function that is not defined in all cases, in practice the only restriction for the application of the density function to data analysis is that the algorithm should be allowed to converge at each step. Unfortunately, maximizing the likelihood of data is not directly related with the accuracy of the visual representation of the SOM. Especially when the data dimension is high, the units that encode the co-occurrences of all correlated variables cannot be grouped together on the map, and thus the conditional densities become distributed among different map units. Such effects cannot be observed by visual inspection of the component levels, no matter how the model hyperparameters are set. Examination of the conditional densities, or the conditional means and the confidence intervals may reveal them to some extent.

The association of a generative probability density model with the SOM enables the comparison of the SOM with other similar methods, like Utsugi's SOM model [112] and the Generative Topographic Mapping [9]. If the minimum distortion rule for winner selection is adopted, thus avoiding many theoretical difficulties of the SOM, the main difference that remains between these is the hard vs. soft assignment of data to the units. The hard assignments of the SOM are perhaps easier to interpret and visualize. In the SOM the activation of the units (that is, the posterior probability of the kernels given one data point) is always one for the winning unit and zero for the others, or a unimodal activation bubble in the shape of the neighborhood around the winning unit, depending on the interpreta-

tion. With soft assignments, the posterior distribution may be multi-modal (when two distant regions in the latent space are folded close to each other in the input space), and thus the activation is more difficult to visualize. Note, however, that this multi-modal response gives visual indication of the folding, which may also be valuable. Ultimately, the choice of methods will depend on the application and its objectives.

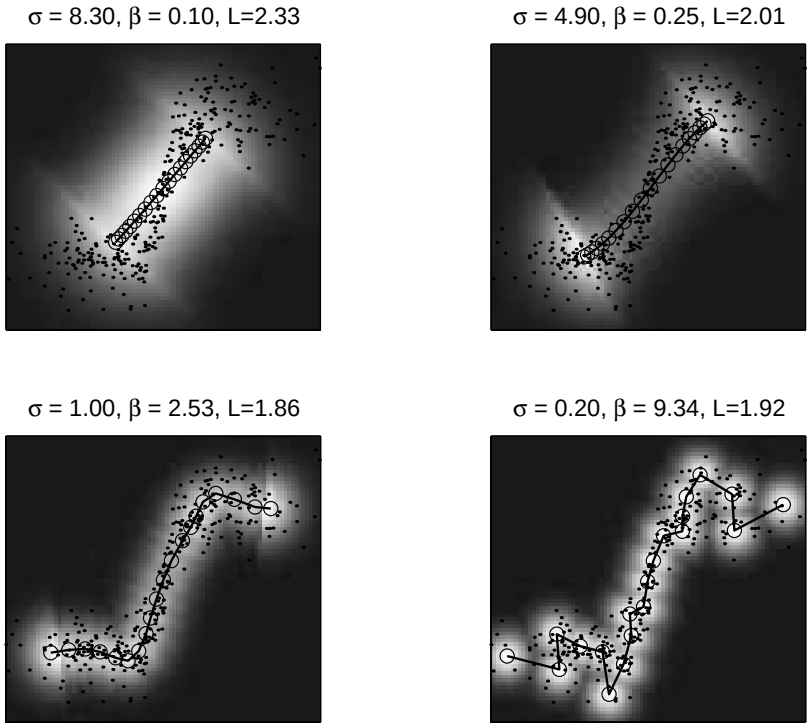


Figure 5.6: SOM density models for different widths σ of the Gaussian neighborhood. From the total likelihood of validation data the optimal neighborhood can be chosen to avoid overfitting. L denotes the negative log-likelihood of validation data (per sample), so the third model $\sigma = 1.00$ is the best of these. With greater values of σ (a and b), we observe underfitting, where adaptation to data is limited, and for smaller σ (d), overfitting has occurred and the effect of noise can be seen in the SOM model.

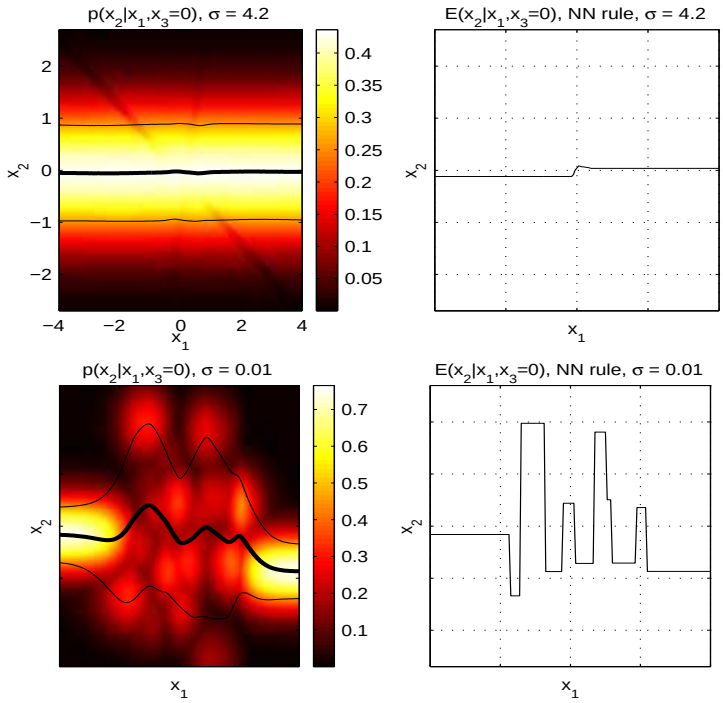


Figure 5.7: Conditional densities from a SOM trained on random independent data. The coordinates are the same in all graphs. Upper row: the conditional density and the nearest neighbor prediction for the optimal neighborhood $\sigma = 4.2$. Lower row: conditional density and the nearest neighbor prediction for a small neighborhood $\sigma = 0.01$. The curves represent the means and standard deviations computed from the density models.

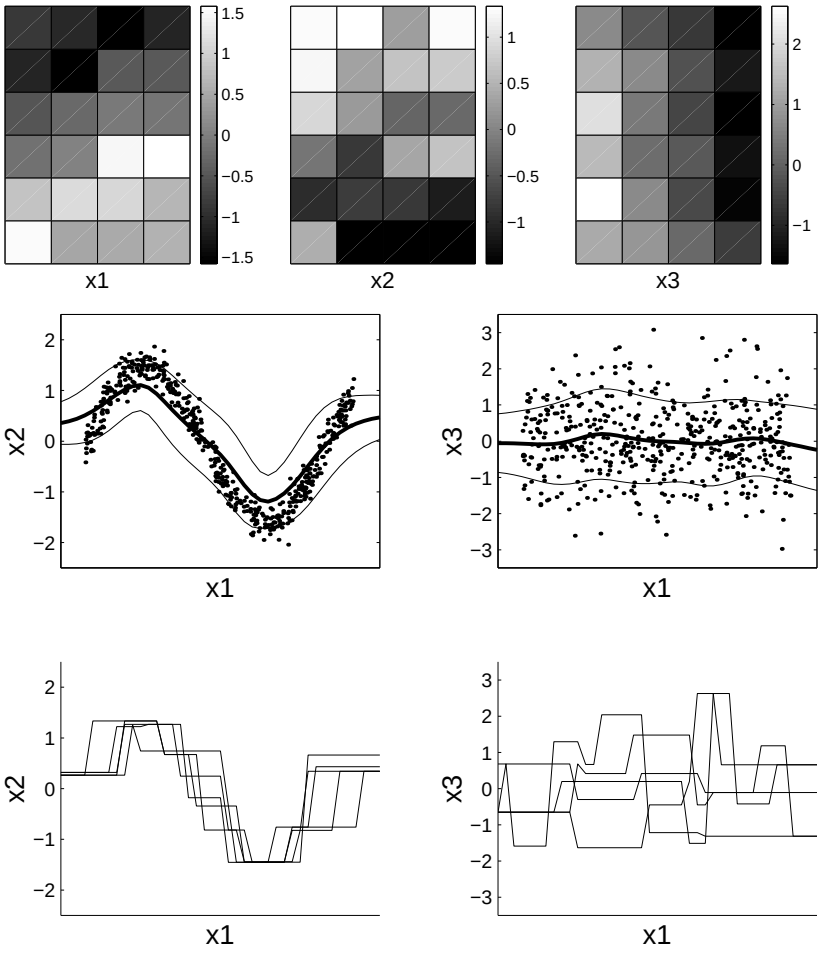


Figure 5.8: Example of the use of the SOM for data analysis. Top: all three component levels of a SOM trained down to the optimal neighborhood width $\sigma = 0.2$. Middle row: Two different views of the training data and the means and standard deviations of the respective conditional densities $p(x_2|x_1)$ and $p(x_3|x_1)$ computed from the SOM model, integrated over x_3 and x_2, respectively. Bottom: nearest neighbor estimates based on the best matching units using five different values for x_3 and x_2, respectively.

Chapter 6

Conclusion

This thesis has described some probabilistic methods and examples of their applications in computer vision problems. New methods were proposed for image segmentation and the model-based detection and matching of both two and three dimensional objects. The self-organizing map method was analyzed from a probabilistic viewpoint.

Different computer vision problems were studied from the point of view of generative probability models. In this approach, object models and the image production process are described by parametric models. If these parameters can be estimated, the results directly describe the target of the image. The alternative approach is to rely on indirect features and their statistical co-occurrence with objects in the image. The latter approach is the only choice, if the model is too complicated to be solved. In practice, fully accurate description of the image formation process is not realistic due to the great number of factors involved. Many approximations and restrictions are needed, and results can be improved by including indirect features in the model. In this thesis, examples were presented of cases where the model-based approach proved feasible. Advantages of the model-based approach include adaptability of object models and conditions and robustness against such variations that can be included in the model.

Markov chain Monte Carlo sampling methods were applied to estimating model parameters in different scene analysis problems. The results indicate that they are a viable solution to these high-dimensional problems. In principle, probabilistic sampling methods are better at avoiding local minima of cost functions than gradient based optimization methods. However, many high-dimensional and sparse parameter spaces, which are typical of computer vision problems, cannot be fully explored within a reasonable computation time. MCMC methods are one possible solution, and they have many benefits. One example is the quite unique reversible jump technique. The results in this thesis indicate that it is very suitable for certain types of problems.

In this thesis, both the segmentation of images and the detection and matching of objects have been discussed. The integration of these tasks into a simultaneous process is an important challenge for future research in computer vision. For object detection, a segmentation that finds the boundary between the object and its background and provides an explanation to the background is essential information. On the other hand, complex objects cannot be accurately segmented without a detailed model of the object.

In many computer vision problems, theories that solutions are based on are very strong simplifications of reality, due to the highly complex dependence between an image and the objects that it represents. This means that results depend very much on many details of the implementation of the solution. Having a good optimization procedure is important, but it is crucial to be able to define suitable target functions. Apparently, many implementation details and tricks are quite as important as the basic principles and modeling framework.

The self-organizing map algorithm was analyzed from a probabilistic viewpoint. This is a novel approach. A number of authors have proposed other models and algorithms that are similar to the SOM but that are based on probabilistic principles. With the help of the probability model for the SOM, these models can be directly compared, and standard model selection techniques can be applied to choose the model that is most suitable for the purpose.

The density model clearly reveals some important properties that are inherent to the SOM. In practical data analysis, abrupt discontinuities in the density can only be viewed as undesirable artifacts in the model, and their effect on results should be estimated and taken into account. Model selection for the SOM can be done, but it is difficult compared to mixture density models, because the likelihood function cannot be computed without numerical approximations. These are reasons why for each application, it should be considered whether related latent variable methods that are based on mixture density models could be used instead of the SOM. The possibility to visually display the component levels is an advantage of the SOM, but its use requires careful reliability analysis.

References

[1] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

[2] Joseph J. Atick, Paul A. Griffin, and A. Norman Redlich. Statistical approach to shape from shading: Reconstruction of three-dimensional face surfaces from single two-dimensional images. *Neural Computation*, 8(6):1321–1340, 1996.

[3] H. Attias. A variational bayesian framework for graphical models. *Advances in Neural Information Processing Systems (NIPS)*, 12, 2000. MIT Press, Cambridge, MA.

[4] Claus Bahlmann, Gunther Heidemann, and Helge Ritter. Artificial neural networks for automated quality control of textile seams. *Pattern Recognition*, 32(6):1049–1060, 1999.

[5] Paola Baldassarri, Paolo Puliti, Anna Montesanto, and Guido Tascini. Self-organizing maps versus growing neural gas in a robotic application. In *Artificial Neural Nets Problem Solving Methods: 7th International Work-Conference on Artificial and Natural Neural Networks, IWANN2003, Part II*, Lecture Notes in Computer Science, pages 201–208, 2003.

[6] S. A. Barker and P. J. W. Rayner. Unsupervised image segmentation using markov random field models. *Pattern Recognition*, 33(4):587–602, 2000.

[7] H.-U Bauer and K.R. Pawelzik. Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Transactions on Neural Networks*, 3(4):570–579, July 1992.

[8] José M. Bernardo and Adrian F. M. Smith. *Bayesian Theory*. John Wiley & Sons, 1994.

[9] C. M. Bishop, M. Svensen, and C. K. I. Williams. GTM: a principled alternative to the self-organizing map. In C. von der Malsburg, W. von Seelen,

J. C. Vorbruggen, and B. Sendhoff, editors, *Artificial Neural Networks—ICANN 96. 1996 International Conference Proceedings*, pages 165–70. Springer-Verlag, Berlin, Germany, 1996.

[10] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[11] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[12] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 187–194, Los Angeles, 1999. Addison Wesley Longman.

[13] C. A. Bouman and M. Shapiro. A multiscale random field model for bayesian image segmentation. *IEEE Trans. Image Processing*, 3:162–177, 1994.

[14] P. Brodatz. *Textures: A Photographic Album for Artists and Designers*. Dover, New York, 1996.

[15] H. H. Bülthoff, S. Y. Edelmann, and M. S. Tarr. How are three-dimensional objects represented in the brain? *Cerebral Cortex*, 5(3):247–260, 1995.

[16] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 1986.

[17] E. Clark and A. Quinn. A data-driven Bayesian sampling scheme for unsupervised image segmentation. In *International Conference on Acoustic, Speech and Signal Processing*, 1999.

[18] G. B. Coleman and H. C. Andrews. Image segmentation by clustering. *Proceedings of the IEEE*, 67(5), 1979.

[19] Dorin Comaniciu and Peter Meer. Mean shift analysis and applications. In *ICCV (2)*, pages 1197–1203, 1999.

[20] M. Cottrell, P. Gaubert, P. Letremy, and P. Rousset. Analyzing and representing multidimensional quantitative and qualitative data : Demographic study of the Rhone valley. the domestic consumption of the canadian families. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 1–14. Elsevier, Amsterdam, 1999.

[21] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inform. Theory IT-13*, pages 21–27, 1967.

[22] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Computer Graphics*, 30(Annual Conference Series):11–20, 1996.

[23] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

[24] Tom Duckett and Ulrich Nehmzow. Performance comparison of landmark recognition systems for navigating mobile robots. In *AAAI/IAAI*, pages 826–831, 2000.

[25] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Applied Probability. Chapman & Hall, 1993.

[26] R.S Ehlers and S.P. Brooks. Constructing general efficient proposals for reversible jump MCMC. Available online at http://www.est.ufpr.br/rt/ehlb03.htm, July 2003.

[27] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22:46–57, June 1989.

[28] Ed Erwin, Klaus Obermayer, and Klaus Schulten. Self-organizing maps: Ordering, convergence properties and energy functions. *Biol. Cyb.*, 67(1):47–55, 1992.

[29] Mark Everingham, Henk Muller, and Barry Thomas. Evaluating image segmentation algorithms using the pareto front. In M. Nielsen A. Heyden, G. Sparr and P. Johansen, editors, *Proc. the 7th European Conference on Computer Vision, Part IV*, pages 34–48, 2002.

[30] Guoliang Fan and Xiang-Gen Xia. Wavelet-based texture analysis and synthesis using hidden Markov models. *IEEE Transactions on Circuits and Systems Part I*, 50(1):106 – 120, January 2003.

[31] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, MA, 1990. Second edition.

[32] Y. Freund and R. Schapire. A decision theoretic generalisation of online learning. *Computer and System Sciences*, 1(55), 1997.

[33] B. Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge MA, 1995.

[34] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald R. Rubin. *Bayesian Data Analysis*. Texts in Statistical Science. Chapman & Hall, 1995.

[35] S. Geman and D. Geman. Stochastic relaxation, gibbs distribution, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[36] U. Gerecke, N.E. Sharkey, and A.J.C. Sharkey. Common evidence vectors for self-organized ensemble localization. *Neurocomputing Journal*, 55(3-4):499–519, 2003.

[37] Z. Ghahramani and M. J. Beal. Variational inference for bayesian mixture of factor analysers. In *Advances in neural information processing systems, 12*, pages 449–455, Cambridge, MA, 2000. MIT Press.

[38] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1996.

[39] P. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrica*, 82(4):711–732, 1995.

[40] E. Hamilton. JPEG file interchange format (version 1.02). Technical report, C-Cube Microsystems, September 1992.

[41] R.M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, 3, 1973.

[42] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 1970.

[43] Tom Heskes. Energy functions for self-organizing maps. In Erkki Oja and Samuel Kaski, editors, *Kohonen maps*, pages 303–315. Elsevier, 1999.

[44] C. C. Holmes and B. K. Mallick. Bayesian radial basis functions of variable dimension. *Neural Computation*, 10:1217–1234, 1998.

[45] H. Hu and D. Gu. Landmark-based navigation of industrial mobile robots. *Industrial Robot*, 27, 2000.

[46] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall Information and System Sciences Series. Prentice Hall, 1989.

[47] S. Jain and R. M. Neal. A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13:152–182, 2004.

[48] J. A. Janet, R. Gutierrez-Osuna, T. A. Chase, M. White, and R. C. Luo. Global self localization for autonomous mobile robots using self-organizing kohonen neural networks. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 504–509, 1995.

[49] C.R. Jung and R. Schramm. Rectangle detection based on a windowed hough transform. In *Proc. 17th Brazilian Symposium on Computer Graphics and Image Processing*, Curitiba, Brazil, 2004.

[50] S. Kaski, J. Kangas, and T. Kohonen. Bibliography of self-organizing map (SOM) papers: 1981–1997. *Neural Computing Surveys*, 1:102–350, 1998.

[51] S. Kaski and K. Lagus. Comparing self-organizing maps. In *Proc. ICANN'96, International Conference on Artificial Neural Networks*, volume 1112, pages 809–814. Springer, Berlin, 1996.

[52] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[53] R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90:773–795, 1995.

[54] Zoltan Kato. Segmentation of color images via reversible jump MCMC sampling. *Image and Vision Computing*, 2006. To appear.

[55] D. C. Knill and W. Richards, editors. *Perception as Bayesian Inference*. Cambridge University Press, 1996.

[56] T. Kohonen. Things you haven't heard about the self-organizing map. In *Proc. of 1993 IEEE International Conference on Neural Networks*, pages pp. 1147–1156, San Francisco, USA, 1993.

[57] Teuvo Kohonen. Self-organizing formation of topologically correct feature maps. *Biol. Cyb.*, 43(1):59–69, 1982.

[58] Teuvo Kohonen. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).

[59] Timo Kostiainen, Ilkka Kalliomäki, Toni Tamminen, and Jouko Lampinen. 3D object recognition based on hierarchical eigen-shapes and Bayesian inference. In Casasent and Hall, editors, *Proceedings of SPIE*, volume 4572, pages 165–173, Newton, USA, 2001.

[60] Timo Kostiainen and Jouko Lampinen. Maximum likelihood optimization of self-organizing map parameters. In *Proceedings of SCI'2000, 4th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, USA, July 2000.

[61] Timo Kostiainen and Jouko Lampinen. Self-organizing map as a probability density model. In *Proceedings of IJCNN 2001*, Washington, D.C, USA, July 2001.

[62] Timo Kostiainen and Jouko Lampinen. On the generative probability density model in the Self-Organizing Map. *Neurocomputing*, 48:217–228, October 2002.

[63] Timo Kostiainen and Jouko Lampinen. Efficient proposal distributions for MCMC image segmentation. In *IEEE International Conference on Image Processing*, Singapore, October 2004.

[64] Timo Kostiainen and Jouko Lampinen. Probabilistic image segmentation for low-level map building in robot navigation. In *Workshop on Processing Sensory Information for Proactive Systems (PSIPS 2004)*, Oulu, Finland, June 2004.

[65] Martin Lades, Jan C. Vorbrüggen, Joachim Buhmann, J. Lange, Christoph von der Malsburg, Rolf P. Würtz, and Wolfgang Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42:300–311, 1993.

[66] Jouko Lampinen and Timo Kostiainen. Self-Organizing Map in data-analysis - notes on overfitting and overinterpretation. In Michel Verleysen, editor, *Proc. ESANN'2000*, pages 239–244, Bruges, Belgium, April 2000. D-Facto.

[67] S. U. Lee, S. Y. Chung, and R. H. Park. A comparative performance study of several global thresholding techniques for segmentation. *Computer Vision, Graphics, and Image Processing*, 52, 1990.

[68] Tai Sing Lee. A bayesian framework for understanding texture segmentation in the primary visual cortex. *Vision Research*, 35(18):2643–2657, 1995.

[69] M. D. Levine and A. M. Nazif. Dynamic measurement of computer generated image segmentations. *IEEE PAMI*, 7:155–164, 1985.

[70] Young Won Lim and Sang Uk Lee. On the color image segmentation algorithm based on the thresholding and the fuzzy c-means techniques. *Pattern Recognition*, 23(9):935–952, 1990.

[71] Jun S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.

[72] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun. Using EM to learn 3D models with mobile robots. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

[73] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[74] Stephen P. Luttrell. Code vector density in topographic mappings: scalar case. *IEEE Trans. on Neural Networks*, 2(4):427–436, July 1991.

[75] W. Y. Ma and B. S. Manjunath. Edge flow: A framework of boundary detection and image segmentation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 744–749, 1997.

[76] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[77] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, August 1996.

[78] B. S. Manjunath, J.-R. Ohm, V. V. Vasudevan, and A. Yamada. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):703 – 715, 2001.

[79] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman, San Francisco, 1982.

[80] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[81] T. Martinetz and K. Schulten. A "neural gas" network learns topologies. In T. Kohonen et al., editor, *Artificial Neural Networks, Vol. I*, pages 397–402. North-Holland, Amsterdam, 1991.

[82] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[83] H. Moon, R. Chellappa, and A. Rosenfeld. Performance analysis of a simple vehicle detection algorithm. *Image and Vision Computing*, 20, 2002.

[84] Eric N. Mortensen and William A. Barrett. Interactive segmentation with intelligent scissors. *Graph. Models Image Process.*, 60(5):349–384, 1998.

[85] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. Pure Applied Math.*, 42:577–684, 1989.

[86] H. Murase and S. K. Nayar. Visual learning and recognition of 3D objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[87] Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, 1996.

[88] Radford M. Neal. Slice sampling. *Annals of Statistics*, 31(3):705–767, 2003.

[89] A. O'Hagan. Fractional Bayes factors for model comparison (with discussion). *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):99–138, 1995.

[90] Merja Oja, Samuel Kaski, and Teuvo Kohonen. Bibliography of self-organizing map (SOM) papers: 1998-2001 addendum. *Neural Computing Surveys*, 3:1–156, 2003.

[91] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[92] N. R. Pal and S. K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1227–1294, 1993.

[93] T. N. Pappas. An adaptive clustering algorithm for image segmentation. *IEEE Transactions on Signal Processing*, 40(4), 1992.

[94] Arthur R. Pope and David G. Lowe. Probabilistic models of appearance for 3-D object recognition. *International Journal of Computer Vision*, 40(2):149–167, 2000.

[95] William H. Press, Saul A. Teukolsky, William T. Wtterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1995.

[96] Jan Puzicha, Yossi Rubner, Carlo Tomasi, and Joachim M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. In *ICCV (2)*, pages 1165–1172, 1999.

[97] K. Rao. A computer vision system to detect 3-D rectangular solids. In *Applications of Computer Vision, WACV '96, Proceedings 3rd IEEE Workshop on*, Sarasota, USA, 1996.

[98] S. Richardson and P. Green. On bayesian analysis of mixtures with an unknown number of components (with discussion). *J. Royal Statist. Soc. Series B*, 59:731–792, 1997.

[99] H. Ritter and K. Schulten. Kohonen self-organizing maps: exploring their computational capabilities. In *Proc. ICNN'88 Int. Conf. on Neural Networks*, volume I, pages 109–116, Piscataway, NJ, 1988. IEEE Service Center.

[100] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, second edition, 2004.

[101] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[102] Markus A. Stricker and Markus Orengo. Similarity of color images. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 381–392, 1995.

[103] M. Swain and D. Ballard. Color indexing. *Int. Journal of Computer Vision*, 7(1):11–32, 1991.

[104] Martin A. Tanner. *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions*. Springer Series in Statistics. Springer-Verlag, third edition, 1996.

[105] Charles Thorpe, Martial Hebert, Takeo Kanade, and Steven Shafer. Vision and navigation for the Carnegie-Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362 – 373, May 1988.

[106] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[107] Michael E. Tipping and Christopher M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[108] Zhuowen Tu and Song-Chun Zhu. Image segmentation by data-driven Markov chain Monte Carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):657–673, 2002.

[109] M. Tuceryan and A. K. Jain. *The Handbook of Pattern Recognition and Computer Vision (2nd Edition)*, chapter Texture Analysis. World Scientific Publishing Co., 1999.

[110] S. Ullman and R. Basri. Recognition by linear combination of models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(10):992–1006, 1991.

[111] I. Ulusoy and C. M. Bishop. Generative versus discriminative methods for object recognition. In *Proc. Computer Vision and Pattern Recognition*, volume 2, pages 258–265, 2005.

[112] Akio Utsugi. Hyperparameter selection for self-organizing maps. *Neural Computation*, 9(3):623–635, 1997.

[113] T. Villmann, R. Der, M. Herrmann, and T. Martinetz. Topology preservation in self-organizing feature maps: exact definition and measurement. *IEEE Transactions on Neural Networks*, 8(2):256–266, 1997.

[114] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. Conference on Computer Vision and Pattern Recognition*, 2001.

[115] Irma Welling, Erkki Kähkönen, Marjaana Lahtinen, Kari Salmi, Jouko Lampinen, and Timo Kostiainen. Modelling of occupants' subjective responses and indoor air quality in office buildings. In *Proceedings of the Ventilation 2000, 6th International Symposium on Ventilation for Contaminant Control*, volume 2, pages 45–49, Helsinki, Finland, June 2000.

[116] K.C. Yao, M. Mignotte, C. Collet, P. Galerne, and G. Burel. Unsupervised segmentation using a self-organizing map and a noise model estimation in sonar imagery. *Pattern Recognition*, 33(9):1575–1584, 2000.

[117] W.A. Yasnoff, J. K. Mui, and J. W. Bacus. Error measures for scene segmentation. *Pattern Recognition*, 9:217–231, 1977.

[118] A.L. Yuille and James Coughlan. High-level and generic models for visual search: When does high level knowledge help? In *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[119] Y. J. Zhang. A survey on evaluation methods for image segmentation. *Pattern Recognition*, 29(8):1335–1346, 1996.

ISBN-13 978-951-22-8411-5 (printed)
ISBN-10 951-22-8411-1 (printed)
ISBN-13 978-951-22-8412-2 (PDF)
ISBN-10 951-22-8412-X (PDF)
ISSN 1455-0474