Segmentation of Laser Range Radar Images using Hidden Markov Field Models

LINKÖPING STUDIES IN SCIENCE AND TECHNOLOGY

Thesis no: 403
Predrag Pucar

LIU-TEK-LIC-1993:45
ISBN 91-7871-184-3
ISSN 0280-7971

Linköping, October 1993
Segmentation of Laser Range Radar Images using Hidden Markov Field Models

Predrag Pucar

Department of Electrical Engineering
Linköping University
S-581 83 Linköping, Sweden
Abstract

Segmentation of images in the context of model based stochastic techniques is connected with high, very often unpracticle computational complexity. The objective with this thesis is to take the models used in model based image processing, simplify and use them in suboptimal, but not computationally demanding algorithms. Algorithms that are essentially one-dimensional, and their extensions to two dimensions are given.

The model used in this thesis is the well known hidden Markov model. Estimation of the number of hidden states from observed data, is a problem that is addressed. The state order estimation problem is of general interest and is not specifically connected to image processing. An investigation of three state order estimation techniques for hidden Markov models is given.
Acknowledgments

First of all, I would like to thank my supervisor Professor Lennart Ljung for creating an excellent and inspiring atmosphere to work in. Of course, the rest of the people in the Automatic Control group should not be omitted when speaking about the satisfaction it is to work in this group.

Secondly, I am very grateful to Dr. Mille Millnert, my thesis advisor, for introducing me to the topic and giving me support throughout the work.

I am indebted to Professor Lennart Ljung, Dr. Mille Millnert and Mr. Josip Blažić for proof-reading and giving numerous valuable comments on the manuscript.

I would also like to thank Dr. Fredrik Gustafsson, whose work has been a source of inspiration. Our discussions have always been valuable and opened my eyes to new perspectives on the topic.

I thank Dr. Ingmar Renhorn and Dr. Dietmar Letalick at the National Defense Research Establishment, for their excellent work with the laser range radar system, and for providing data used for testing algorithms.

Finally, I thank my considerate and beloved wife Mirjana, and my wonderful children Sandra and Sanja, for their support and the joy they give me. I thank also those other people being dear to me: in particular my parents Aurelio and Dragica, as well as my brother Nenad, his wife Diana, and their son Dejan.

Predrag Pucar
## Abbreviations

<table>
<thead>
<tr>
<th>Vignette</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviations</td>
<td>3</td>
</tr>
</tbody>
</table>

## 1 Introduction

<table>
<thead>
<tr>
<th>Vignette</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Background</td>
<td>5</td>
</tr>
<tr>
<td>1.2 Classification vs. Segmentation</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Notation</td>
<td>7</td>
</tr>
<tr>
<td>1.3.1 Stochastic Processes</td>
<td>7</td>
</tr>
<tr>
<td>1.3.2 The Normal Distribution</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Outline</td>
<td>8</td>
</tr>
</tbody>
</table>

## 2 The Laser Radar

<table>
<thead>
<tr>
<th>Vignette</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Description of the Laser System</td>
<td>12</td>
</tr>
<tr>
<td>2.2 Noise in the Laser System</td>
<td>13</td>
</tr>
<tr>
<td>2.3 Presentation of Test Images</td>
<td>14</td>
</tr>
</tbody>
</table>

## 3 Hidden Markov Models

<table>
<thead>
<tr>
<th>Vignette</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Markov Chains</td>
<td>20</td>
</tr>
<tr>
<td>3.2 The Observed Process</td>
<td>22</td>
</tr>
<tr>
<td>3.2.1 Linear Filtering</td>
<td>22</td>
</tr>
<tr>
<td>3.3 Estimation of HMMs</td>
<td>24</td>
</tr>
<tr>
<td>3.3.1 Baum’s Algorithm</td>
<td>25</td>
</tr>
<tr>
<td>3.3.2 Bayesian Methods</td>
<td>29</td>
</tr>
<tr>
<td>3.3.3 AFMM</td>
<td>33</td>
</tr>
<tr>
<td>3.4 Summary</td>
<td>34</td>
</tr>
</tbody>
</table>

## 4 Review of Models used in Image Segmentation

<table>
<thead>
<tr>
<th>Vignette</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Markov Random Fields</td>
<td>38</td>
</tr>
<tr>
<td>4.1.1 Simulated Annealing</td>
<td>42</td>
</tr>
<tr>
<td>4.1.2 Graduated Non-convexity Algorithms</td>
<td>44</td>
</tr>
<tr>
<td>4.2 Markov Mesh Random Fields</td>
<td>46</td>
</tr>
<tr>
<td>4.3 NSHP Markov Chain</td>
<td>47</td>
</tr>
</tbody>
</table>
## 5 Row by Row Segmentation of Laser Range Images

### 5.1 The Baum-Welch Algorithm

- 5.1.1 The Terrain Model
- 5.1.2 Resulting Segmentation

### 5.2 Bayesian Methods

- 5.2.1 The Priors
- 5.2.2 Results
- 5.2.3 Resulting Segmentation

### 5.3 Summary

## 6 Extension of 1-D Techniques to 2-D

### 6.1 Extension of the Baum-Welch algorithm to Two Dimensions

- 6.1.1 The Algorithm
- 6.1.2 Results

### 6.2 Bayesian Methods in Two Dimensions

- 6.2.1 NSHP Markov Field
- 6.2.2 The Influence of the Scanning Pattern
- 6.2.3 The Algorithm
- 6.2.4 Results

### 6.3 Summary

## 7 State Order Estimation for HMMs

### 7.1 Information Theoretic Techniques

### 7.2 Predictive Least Squares

- 7.2.1 Using PLS for Order Estimation of HMMs

### 7.3 MAP Estimation of the Model Order

### 7.4 Summary

## 8 Conclusions and Open Problems

## A Derivation of \( a \) priori Probabilities

### A.1 Derivation of the \( a \) priori Probability in the Case of a Flat Prior on \( \lambda \)

### A.2 Derivation of the \( a \) priori Probability in the Case of Inverse Wishart Prior \( \lambda \)

## Bibliography
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFMM</td>
<td>adaptive forgetting through multiple models</td>
</tr>
<tr>
<td>AIC</td>
<td>A-information criterion</td>
</tr>
<tr>
<td>AR</td>
<td>autoregressive</td>
</tr>
<tr>
<td>BIC</td>
<td>B-information criterion</td>
</tr>
<tr>
<td>CNR</td>
<td>carrier-to-noise ratio</td>
</tr>
<tr>
<td>DEA</td>
<td>detection-estimation algorithm</td>
</tr>
<tr>
<td>EM</td>
<td>expectation-maximization</td>
</tr>
<tr>
<td>GNC</td>
<td>graduated non-convexity</td>
</tr>
<tr>
<td>GPBA</td>
<td>generalized pseudo-Bayes algorithm</td>
</tr>
<tr>
<td>HMF</td>
<td>hidden Markov field</td>
</tr>
<tr>
<td>HMM</td>
<td>hidden Markov model</td>
</tr>
<tr>
<td>ICM</td>
<td>iterated conditional mode</td>
</tr>
<tr>
<td>IMM</td>
<td>interacting multiple models</td>
</tr>
<tr>
<td>LO</td>
<td>local-oscillator</td>
</tr>
<tr>
<td>MAP</td>
<td>maximum a posteriori</td>
</tr>
<tr>
<td>MDL</td>
<td>minimum description length</td>
</tr>
<tr>
<td>MMRF</td>
<td>Markov mesh random field</td>
</tr>
<tr>
<td>MRF</td>
<td>Markov random field</td>
</tr>
<tr>
<td>NSHP</td>
<td>non-symmetric half-plane</td>
</tr>
<tr>
<td>PMDL</td>
<td>predictive minimum description length</td>
</tr>
<tr>
<td>PLS</td>
<td>predictive least squares</td>
</tr>
<tr>
<td>RLS</td>
<td>recursive least squares</td>
</tr>
<tr>
<td>RSA</td>
<td>random sampling algorithm</td>
</tr>
</tbody>
</table>
Introduction

1.1 Background

The field of image segmentation is very broad. There are numerous different branches in the field and the work presented in this thesis could be classified under "statistical methods in image processing". One of the models used in statistical image processing is the hidden Markov field (HMF) model. The earliest references using HMFs in the field are from 1925. However, an increasing interest in HMFs can be noted at the end of 1960's. A renewed interest was sparked in the middle of the 1980's.

The reason for using the HMF in this thesis is because it is a natural model for images used in our work. The images used are obtained by a laser range radar system. The laser scans a part of the terrain and delivers an image where every pixel has two attributes; range and intensity. It is natural to assume that pixels in small neighborhoods are likely to have similar properties. Objects in the terrain, man-made or natural, and the background usually are connected areas that can be viewed as clusters of pixels with similar properties. Hence, HMFs are very natural.

One drawback of HMFs is that the computational complexity in connection with estimation of HMFs is very high. The complexity, for example, completely excludes their use in real-time systems. There is an obvious need for simpler methods, with relatively low computational complexity, that can be used in image segmentation. The desire of keeping the natural image model and finding estimation methods with low complexity led to this thesis. Our approach is to design less complex algorithms by using modified, simpler HMF models and new scanning schemes.

The laser range images have been used as a source of inspiration since they include a lot of interesting principal problems, of which only a few are addressed here.

1.2 Classification vs. Segmentation

There are two important concepts used when describing the process from collecting image data to recognizing objects in the image, and we would like to make a distinction between the two. The concepts are segmentation of the image and classification of the segments. Of the two, segmentation is the more low level operation and it is also a prerequisite for
Segmentation of an image can be described as partitioning the image into areas with different properties. Assume for a moment that we are trying to find objects on a table. It could look like in Figure 1.1.

![Figure 1.1: Table with four objects on it. The objects have different properties, in this case shape and color.](image)

Segmentation of the image consisting of the table with objects on it can be performed in at least two ways. One way is the one depicted in Figure 1.1, namely simply take as many segments as there are objects and denote the different areas by 1 through 5. When applying that segmentation philosophy in practice one is confronted with the problem of complexity. As will be mentioned in subsequent chapters, the computational complexity grows rapidly as the number of segments used increases.

Another way to go is to only use two segments. In the example of the table and objects on it, this would mean that the table was denoted by 1, and all the areas that could be recognized as non-table were denoted by 2. Now the segmentation task is accomplished. In the segmentation process only a few of the object properties might be used, for example, in this case the color of the objects. Obviously all the objects differ from the table by their color.

In the classification stage, which takes over after the segmentation, the goal is to use the results of the segmentation to put labels on all of the segments using some a priori knowledge. In the example with the table when all the items on it were denoted by 2, it is possible to use a bank of shapes to classify the triangle as a triangle, and the ellipse as an ellipse.

When segmenting laser range radar images in subsequent chapters, often the images are segmented in groups of rows (or columns). It follows from the discussion above that it is enough to use a few segments, and then in the classification stage use information as range, range variance, shape etc, to distinguish between different objects. In this thesis we will solely concentrate on the segmentation problem.
1.3 Notation

In this section the notational conventions used throughout this thesis will be introduced. The notation deals with stochastic processes and the Normal distribution.

1.3.1 Stochastic Processes

Let \( X(t) \) be a stochastic process and \( t \in \{0, 1, \ldots\} \) denote time. A sequence of the process in the time interval \( \{t_1, \ldots, t_2\} \) will be represented by

\[
X_{t_1}^{t_2} = \{X_{t_1}, \ldots, X_{t_2}\}
\]

and the realization of the process will be denoted using lower case letters (generally lower case letters will denote the outcome of a random variable)

\[
x_{t_1}^{t_2} = \{x_{t_1}, \ldots, x_{t_2}\}.
\]

The abbreviated notation \( X^t \) will be used to refer to the stochastic process \( X^t = \{X_1, \ldots, X_t\} \), and similar notation will apply to the realization as well, only lower case letters will be used.

Assume two stochastic processes are given, one of which is not directly observed. We will denote the unobserved process by \( Z \) and the observed process by \( Y \). Often the observed process depends in some way on the unobserved, and to highlight the dependence we write the unobserved variable in parenthesis, e.g. \( Y_t(z_t) \).

1.3.2 The Normal Distribution

In this thesis we will mostly use the Normal (also called Gaussian) probability distribution, whose notation will now be introduced. A random variable \( X \) is said to have a Normal distribution with mean \( \mu (-\infty < \mu < +\infty) \) and variance \( \sigma^2 > 0 \) if it has the density function

\[
f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x-\mu}{\sigma}\right)^2}, \quad -\infty < x < +\infty.
\]

Shorthand notation will be used to indicate that a random variable is normally distributed.

\[
X \sim N(\mu, \sigma^2)
\]

denotes a normally distributed random variable with mean \( \mu \) and variance \( \sigma^2 \). The density function, mean \( 0 \) and variance 1, is depicted in Figure 1.2.

A \( n \)-dimensional random vector \( x \) whose distribution is Gaussian, is characterized by its mean vector and covariance matrix

\[
\mu = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix}, \quad C = \begin{bmatrix} C(1,1) & \cdots & C(1,n) \\ \vdots & \ddots & \vdots \\ C(n,1) & \cdots & C(n,n) \end{bmatrix},
\]

where \( C \) denotes the covariance matrix whose elements are defined by

\[
C(j,k) = E \{(x_j - E(x_j))(x_k - E(x_k))\}.
\]

The density function for the \( n \)-dimensional random vector is

\[
f(x) = \frac{1}{(2\pi)^{n/2} \sqrt{\det C}} e^{-\frac{1}{2}(x-\mu)^T C^{-1} (x-\mu)}.
\]
Chapter 1. Introduction

Figure 1.2: Density function of a normally distributed random variable, with mean 0 and variance 1.

1.4 Outline

This thesis is organized in the following way. In Chapter 2 the laser range radar system is described. The presentation in Section 2.1 goes into some details and can be skipped by readers not interested in the hardware and details about the laser system. In Section 2.2 different noise sources acting on the laser system are described. The last section of Chapter 2 contains a presentation of the available, and in this thesis used, range images with different scenes.

In Chapter 3 the concept of hidden Markov chains is introduced. The reader familiar with the concept can skim through the chapter. Also a short review of Kalman filtering is included. In Section 3.3 estimation in the context of hidden Markov chains is addressed. The last section in Chapter 3 is devoted to the pruning routine used in the subsequent chapters.

Chapter 4 contains a review of the most frequently used models with Markov properties and related estimation techniques. The models mentioned are:

- non-causal HMFs
- Markov mesh models
- non-symmetrical half-plane Markov chains.

The material presented in Chapter 5 is an application, with some modifications, of one-dimensional segmentation techniques to laser range images. In Section 5.1 the Baum-Welch algorithm is used and in Section 5.2 a method that can be labeled as Bayesian, is used for row-by-row segmentation.

The one-dimensional techniques presented in Chapter 5, are extended to two dimensions in Chapter 6. Essentially we keep the good properties of the one-dimensional algorithms in Chapter 5 and extend the hidden Markov chain to a hidden Markov field. The scanning pattern is modified to better suit the two-dimensional model.
In Chapter 7 the model order estimation issue is discussed. Three approaches are treated:

- information-theoretic techniques
- predictive least-squares
- maximum a posteriori estimation.

The thesis is summarized in Chapter 8 and some concluding remarks and future research problems are stated.
The Laser Radar

The research presented in this thesis is the result of work inspired by the laser range images, produced by a laser radar system. The laser system delivering the test images in this thesis is located at the Department of Information Technology, National Defense Research Establishment in Linköping, Sweden. A block diagram of the laser system is found in Figure 2.1. The diagram is copied from [46], where more details on the laser system can be found.

Two more references on the topic of the laser system used in this thesis are [45] and [47]. In this chapter the properties of the laser system will be briefly covered. The text in this chapter is not meant to be complete, but only a summary of above mentioned references is given.

The laser radar has the potential to become a key sensor in various fields of application. Among the more conspicuous advantages of using a laser radar system as a sensor are:

- High angle resolution.
- Narrow lobe width, which admits precision tracking.

Figure 2.1: Block diagram of the laser system.
Chapter 2. The Laser Radar

- High Doppler resolution.
- High range resolution.
- Reflectance information.
- Insensitivity to changes in the environment surrounding the system.

The coherent laser technique has shown to be technically realizable and reliable in a rough environment.

In Section 2.1 a description of the laser system is given, in Section 2.2 the different noise sources and their characteristics are presented, and finally in Section 2.3 the family of available test images, which are used when testing different algorithms throughout this thesis, is presented.

2.1 Description of the Laser System

The imaging coherent laser radar system is based on a CO₂ waveguide laser with programmable pulse profile. The imaging laser system is a flexible one and a multisensor setup is possible, e.g., the system can be optimized for Doppler or range measurements. It also has a bore-sighted TV camera with the same field of view as the laser radar and it is possible to obtain reflectance information, allowing for multi-sensor coordination on pixel level. In this thesis we have only used the range measuring mode.

The transmitter laser, a CO₂ waveguide laser emitting at a wavelength of 10.6 μm, has been designed to allow for a programmable pulse profile. Different pulse forms can be chosen, e.g., continuous, Q-switched or Q-switched cavity-dumped. When the Q-switched cavity-dumped pulse profile is chosen, the peak power is 1 kW and the pulse length is 13 ns. The measurements typically have been obtained with a 50 ns pulse length and a peak power of 500 W. The range is measured by using the time difference between an envelope-detected start pulse derived from the transmitted pulse on the reference detector, and similarly, on the signal detector envelope-detected, received signal from the target on the signal detector. Constant fraction discriminators detect the pulse position and generate start and stop signals for a time-interval counter. The counter resolution is 2.5 ns, and the measurement range is 30 ns - 163 μs. With a 50 ns pulse length, the range resolution (standard deviation) is typically 3 m.

The local-oscillator (LO) waveguide laser has a tunability of 300 MHz. The tunability is used for tuning an appropriate frequency difference between the transmitter laser and the local oscillator laser. The power of the LO laser is 200 mW.

The galvanometer scanners are located near the entrance pupil of the trailing afocal telescope. The scanner mirrors are placed in a x-y configuration where the x-mirror is the slow mirror. The control of the scanners is performed by the computer, via a servo amplifier. The angles of the scanner mirrors are controlled with optical angle sensors, composed of laser diodes and position sensitive detectors. The angle of the scanner mirrors can thus be measured with high accuracy. General scanning patterns can easily be generated, although in the this thesis we use simple raster scanning only. Continuous operation of the laser radar system at a pulse repetition frequency of 20 kHz results in two frames per second, where each frame comprises 100 × 100 spatially uncorrelated target returns.
The telescope is afocal and consists of a concave-concave paraboloid system with an expansion factor of 4.5. The entrance pupil is 145 mm and the diffraction limited field of view is 24 mrad. The transmitted laser beam has a radius of 34.6 mm, which means that the divergence for a collimated beam is approximately 0.15 mrad.

With the pointing mirror the field of view of the laser radar can be moved in a sector approximately 50° in azimuth and 40° in elevation. The mounted TV camera follows that motion.

The signal detector has a bandwidth of 100 MHz. A pre-amplifier on 60 dB is integrated with the detector. At the signal detector the received radiation is mixed with the (LO) laser beam.

At the reference detector the beams from the LO laser and the transmitter laser are mixed to obtain the frequency difference.

2.2 Noise in the Laser System

There are several noise sources in a laser radar system. In this section some of the noise sources will be mentioned.

The limitation in pointing accuracy can be attributed to four sources:

- Platform instability and instability of the laser radar system relative to this platform.

- The pointing accuracy of the laser radar beam is determined by the measurement of the scanner mirror angles. The scanning angle is measured with an accuracy of approximately 10 μrad.

- The atmospheric turbulence can cause both beam spread and beam wander. The beam wander will give a small deviation from the geometric direction. At strong turbulence this error can be of order 30 μrad.

- The pixelation gives a maximum error of the order of a half the pixelation angle. For uncorrelated pixels the angular separation between measurements is 100 μrad.

The conclusion is that the dominant error is the maximum pixelation error. The pointing error starts to become significant when the error is of the same magnitude as the error in the range measurements. A pointing error of 100 μrad corresponding to a dimension error of 0.2 m at a range of 2 km. Since the range uncertainty is 3 m, the pointing error is negligible.

Another type of noise is the occurrence of speckle effects. The speckle effects are due to the non-returning of the laser beam. That results in no information for some pixels, also called drop-outs. Two different events can result in a drop-out. A drop-out will appear if the laser is pointing into non-target areas (e.g. the sky). The other cause for drop-outs is that the laser beam may scatter from a diffuse target and the returning power is too low to be detected. If the light pulses sent out do not return within a preset time interval (or the power is too low to be detected), the pixel values are set to a predetermined value, in our case -1. Even for target returns with a large carrier-to-noise ratio (CNR), the drop-out probability is high. Rough-surfaced targets exhibit Rayleigh distributed signal amplitude returns, see [45, 46, 62]. The signal amplitude threshold is set to a value that gives an anomalous range probability of less than $10^{-5}$.
Chapter 2. The Laser Radar

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelength</td>
<td>10.6 μm</td>
</tr>
<tr>
<td>Pupil diameter</td>
<td>0.15 m</td>
</tr>
<tr>
<td>Beam divergence</td>
<td>0.15 mrad</td>
</tr>
<tr>
<td>Image frequency (100 x 100 pixels)</td>
<td>2 Hz</td>
</tr>
<tr>
<td>Doppler shift</td>
<td>200 kHz/(m/s)</td>
</tr>
<tr>
<td>Mean power</td>
<td>2.2 W</td>
</tr>
<tr>
<td>Pulse length (Q-switched)</td>
<td>180 ns</td>
</tr>
<tr>
<td>Peak power (Q-switched)</td>
<td>300 W</td>
</tr>
<tr>
<td>Pulse length (cavity dumped)</td>
<td>13 ns</td>
</tr>
<tr>
<td>Peak power (cavity dumped)</td>
<td>1 kW</td>
</tr>
<tr>
<td>Pulse rep. freq.</td>
<td>0-70 kHz</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>150 MHz</td>
</tr>
<tr>
<td>Maximum range (C/N=25 dB)</td>
<td>3 km</td>
</tr>
</tbody>
</table>

Table 2.1: Data of the laser radar range system used in the examples.

Anomalous range probability, also called the false alarm probability, is the probability of a signal return when there is no target, i.e., there should not be any signal return. The effect is due to noise in the system, and if the threshold for signal detection is set to a lower value, the noise will more often be interpreted as a signal return. The detection probability is expressed by the following equation

\[ P_D = P_f^{1+\text{CNR}} \]

where \( P_f \) is the probability of anomalous range returns. Assume, for example, that \( P_f = 10^{-6} \) and \( \text{CNR} = 45 \) (16.5 dB), then the detection probability is \( P_D = 74\% \). This means that diffuse targets will always have a large drop-out probability even at large CNR.

In Table 2.1 a summary of technical data of the coherent laser radar system is given. Different combinations of parameter values can be designed in the system depending on the application for the system in question.

2.3 Presentation of Test Images

In this section the test images used in this thesis will be presented. The images have been obtained by the above mentioned laser range radar system.

Test image #1 is an image obtained by the laser system with technical data found in Table 2.1. There is a shield in the center of the image. Recall that the z-axis is distance to the object. If the distance is constant, i.e., there is a vertical object in the field of view of the laser, the result will be an horizontal area in the laser range image. Further, there is one more object in test image #1, namely bushes in the upper right corner of the image.

In Figure 2.3 the drop-outs existing in test image #1 are highlighted. The drop-outs are marked with dots in the figure.
2.3 Presentation of Test Images

Test image #1 contains a shield in a terrain. In the image to the right, the drop-outs have been removed by replacing the drop-out pixel values with the median of the values of the immediately surrounding pixels which are not drop-outs.

![Test image #1 and Test image #1 without drop-outs](image)

Figure 2.2: Test image #1 - shield in terrain. In the image to the right the drop-outs have been removed by replacing the drop-out pixel values with the median of the values of the immediately surrounding pixels which are not drop-outs.

Test image #2 contains two objects, one man-made and one natural. The scenario is that a small shield is put beside a pine tree. There are no other objects in the image. The image is obtained with an updated version of the laser system. The variance of the range measurements is brought down to 0.3 m. Test image #2 is found in Figure 2.4.

The drop-outs in test image #2 are depicted in Figure 2.5.

![Drop-outs in test image #1](image)

Figure 2.3: Drop-outs in test image #1.
Chapter 2. The Laser Radar

Figure 2.4: Test image #2 - A shield besides a pine tree. In the image to the right the drop-outs have been removed by replacing the drop-out pixel value with the median of the values of the surrounding pixels which are not drop-outs.

Figure 2.5: Drop-outs in test image #2.
2.3 Presentation of Test Images

Test image #3 is similar to test image #1, with the difference that the shield's normal is not pointing in the direction of the laser. The shield is turned relatively to the laser and the angle between the shield's normal and the direction of the laser is 45°.

![Figure 2.6: Test image #3 - turned shield in terrain. In the image to the right the drop-outs have been removed by replacing the drop-out pixel value with the median of the values of the surrounding pixels which are not drop-outs.](image)

In Figure 2.7 the drop-outs in test image #3 are shown. They are marked with black dots in the image.

![Figure 2.7: Drop-outs in test image #3.](image)

The last test image to be presented is test image #4. In test image #4 two shields are overlapping. The rear shield is placed at a distance of 2 m behind the front shield. We will, in Chapter 6, examine if the presented algorithms can distinguish between the two shields. Test image #4, with and without drop-outs is shown in Figure 2.8.
Figure 2.8: Test image #4 - two shields overlapping. In the image to the right the drop-outs have been removed by replacing the drop-out pixel value with the median of the values of the surrounding pixels which are not drop-outs.

The corresponding image containing the positions of the drop-outs can be found in Figure 2.9.

Figure 2.9: Drop-outs in test image #4.
In this chapter we will introduce the concept of hidden Markov models (HMM). We begin with a formal definition.

**Definition 1** A HMM is a doubly stochastic process with one underlying process that is not observable, but can only be observed through another set of output processes that produce the observed data. The underlying hidden process is a Markov chain.

A simple example of a doubly stochastic process is the following experiment.

**Example 3.0.1** Assume we are given a sequence of drawn marbles. The marbles are either blue or white. Further assume that the sequence has been generated in the following way: One person is put in a room with 2 urns. Both urns contain blue and white marbles, but with different proportions between the blue and white marbles. In urn 1 we have 25% blue marbles, and in urn 2 we have 35% blue marbles. We then give the person two unfair coins. The probabilities for heads or tails of the both coins are given below:

\[
\begin{align*}
P(\text{head}_1) &= 0.45 \\
P(\text{tail}_1) &= 0.55 \\
P(\text{head}_2) &= 0.40 \\
P(\text{tail}_2) &= 0.60,
\end{align*}
\]

where subscript 1 denotes coin 1 and similarly for subscript 2. Assume the following scenario: the person start tossing coin one, if heads comes up he draws a marble from urn 1, reports the color and returns the marble. When tails comes up, he switches coins and starts to draw marbles from urn two. Every time tails comes up he switches coin and urns. All this is hidden from us, the only information we get is the sequence of colors of the drawn marbles.

The example above, although it is simple, leads to a number of questions. We will now list some interesting questions concerning the experiment. Given the reported sequence of colors is it possible to, in some sense, optimally estimate:

1. What is the mix of marbles in each urn?
2. What are the probabilities of the "false" coins, i.e., $P(\text{head}_1)$ etc?
3. In what order did the person toss the coins, i.e., which coin sequence gave the resulting sequence of marbles?

4. How many urns are there?

In this chapter answers to the first three questions will be given. The fourth question regarding Example 3.0.1 is how to estimate the number of urns from the given drawn colors. This problem is referred to as the Markov chain state order estimation problem. The objective is to estimate the number of Markov chain states from the observed output process. In Chapter 7 three solutions to this problem are suggested.

It turns out that there are two main approaches leading to answers for the three previously posed questions. Both approaches will be covered in this chapter. The answers will be given for a special class of HMMs, i.e., a special class of output processes will be used. As mentioned, the hidden process in a HMM, corresponding to the coin tossing in the example, is a Markov chain. There is more freedom in choosing the observed process, corresponding to the colored marbles, and it can be chosen to be a discrete process, e.g., colors, with given discrete probability distributions. Another possibility is to attach continuous probability density functions to the states of the Markov chain. The output process will then take real values at discrete time instants, which corresponds to sampled real systems, e.g., measurements of range, flow, thickness, etc. We choose the latter and assume that the observed process is generated by a dynamic linear system driven by white Gaussian noise. The parameters in the system correspond to the "mix in the urns" in the example, and are to be estimated. We would like to stress that although very simple models are used when testing the algorithms derived in this thesis, the theory used and developed can be applied to more complex systems. However, one of the more important points in this thesis is to show how good results are possible to obtain despite the use of simple models.

The HMMs have been used extensively in a variety of areas, e.g., ecology, cryptoanalysis, quality monitoring and a variety of speech applications. A selection of references in the mentioned areas are [4, 10, 39, 71].

Before the answers to the listed questions are given, we introduce both the processes and related formal notation. In Sections 3.1 and 3.2 the two parts of the HMM are introduced. Section 3.3 gives two strategies in estimation of interesting quantities in HMMs. In Section 3.4 a summary of the chapter is given.

### 3.1 Markov Chains

We will start the introduction of the Markov chain by defining a more general process – a Markov process. A Markov chain is a Markov process with some additional properties. A Markov process is a stochastic process whose past has no influence on its future if its present is specified. The discrete time, discrete state Markov process has the following properties:

\[ f(z_t|z_t^{t-1}) = f(z_t|z_{t-1}), \]

where \( f \) is a probability density function and \( z_t \) is the Markov process. Repeated use of the relation above, which also is called the Markov property, gives

\[ f(z_t^i) = f(z_t|z_{t-1})f(z_{t-1}|z_{t-2})\cdots f(z_2|z_1)f(z_1). \]
Another consequence of (3.1) is the following relation for the conditional expectation

\[ E\{Z_t|Z_{t-1}\} = E\{Z_t|Z_{t-1}\}. \]

1) If the time is reversed a Markov process keeps its Markov property (3.1), i.e.,

\[ f(z_t|z_{t+1}) = f(z_t|z_{t-1}). \]

2) If the present is specified, then the past is independent of the future in the following sense: If \( k < m < n \), then

\[ f(z_n, z_k|z_m) = f(z_n|z_m)f(z_k|z_m). \]

A discrete time Markov chain is a Markov process taking values in a finite set \( \mathcal{M} = \{1, \ldots, M\} \). The underlying, non-observable, process in the case of HMMs is a finite state Markov chain with stationary, \( M \times M \), transition matrix. A Markov chain is specified in terms of its stationary distribution

\[ q_i = P(Z_t = i), \]

where \( Z_t \) denotes the state of the Markov chain at time instant \( t \), and the transition probabilities

\[ Q = [q_{ij}], \]

where

\[ q_{ij} = P(Z_{t+1} = j|Z_t = i) \quad i, j \in \mathcal{M}. \]

Further we assume that \( q_{ij} > 0 \). It can readily be shown that

\[ \sum_j q_{ij} = 1 \quad \text{and} \quad \sum_i q_{ij} = q_j. \quad (3.2) \]

The transition matrix \( Q \) is shown schematically in Figure 3.1. The circles represent the states of the process and the arrows have labels which represent the transition probabilities.

![Figure 3.1: Illustration of a two-state Markov chain with transition matrix \( Q \).](image)

We see that the second expression in equation (3.2) can be written in vector form as

\[ q^T Q = q^T, \]

which implies that the state probability vector \( q \) of a stationary Markov chain is an eigenvector of its transition matrix \( Q^T \). The corresponding eigenvalue equals 1.
3.2 The Observed Process

The second part of a HMM is the observed process. Loosely speaking, the observed process is a sequence of random variables generated by a family of probability density functions, e.g., the output of a dynamic linear system driven by white noise. Which probability density function generates an observed random variable at a specific time instant depends on the state of the hidden Markov chain. Possible models for the observed sequence are random variables, stochastic processes, etc. The distribution of the observed process can be discrete or continuous. In this thesis we will assume that the observed sequence is generated by a dynamic linear system driven by white Gaussian noise. We concentrate on systems that can be rewritten as regression models, i.e.,

\[ y_t = \varphi^T \theta + e_t, \quad (3.3) \]

where \( e_t \) is Gaussian white noise with variance \( \Lambda \). The vector \( \theta \) contains the unknown parameters and \( \varphi \), the regression vector, contains information about past behavior, e.g., old measurements or more generally, known data. With this assumption we cover the often used autoregressive (AR) models and a variety of models of which some are mentioned below. In the case of AR models the regression vector contains old measurements. Other examples of models are, for example,

\[ y_t = -\theta_1 y_{t-1} + \theta_0 + e_t, \]

which implies the following regression and parameter vectors

\[ \varphi_t = \begin{bmatrix} -y_{t-1} \\ 1 \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_0 \end{bmatrix}, \]

which is an AR model of a signal assuming there is an unknown constant level in the signal. Another example of possible regression models is

\[ y_t = \theta_0 + \theta_1 \cdot t + e_t, \]

with corresponding regression and parameter vector

\[ \varphi_t = \begin{bmatrix} 1 \\ t \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}. \quad (3.4) \]

The model above is a slope embedded in white Gaussian noise. The extension to general state space models is straightforward, but because of the notational convenience and computational simplicity we choose to work with the simpler regression models.

Before addressing the problems of estimation of parameters in the HMMs, listed in the introduction of this chapter, a short review of linear filtering will be given in the next subsection.

### 3.2.1 Linear Filtering

The material in this subsection is mainly based on the material in books [49] and [50]. The model assumed is a linear, finite-dimensional system that can be written in the following state space form

\[ x_{t+1} = F_t x_t + v_t \quad v_t \in N(0, R_t), \quad x_0 \in N(x(0), P_0) \]

\[ y_t = H_t x_t + e_t \quad e_t \in N(0, \Lambda_t). \quad (3.5) \]
3.2 The Observed Process

where \( v_t \) and \( e_t \) are mutually independent, white Gaussian noises with covariance matrices \( R_t \) and \( \Lambda_t \). Matrices \( F \) and \( H \) describes the behavior of the states and determining the linear combination of states producing the output, respectively. The initial state \( x_0 \) is assumed to have Gaussian distribution with mean \( \pi(0) \) and covariance \( P_0 \).

**Example 3.2.1** Assume that the regression model in equation (3.4) is given. The state space form of the model is the following

\[
\begin{bmatrix}
    x_1(t+1) \\
    x_2(t+1)
\end{bmatrix} = \begin{bmatrix}
    1 & 0 \\
    0 & 1
\end{bmatrix} \begin{bmatrix}
    x_1(t) \\
    x_2(t)
\end{bmatrix} + v(t) \\
y(t) = \begin{bmatrix}
    1 & 1
\end{bmatrix} \begin{bmatrix}
    x_1(t) \\
    x_2(t)
\end{bmatrix} + e(t),
\]

and

\[ x_1(0) = \theta_1(0), \quad x_2(0) = \theta_2(0). \]

In this case \( x_1 = \theta_0 \) and \( x_2 = \theta_1 \). Note that in this example we have assumed that the initial values are known. To adjust that to the formulation in equation (3.5) we assume that \( P_0 = 0 \). We have also assumed that the parameters may not be constant. The drift in the parameters is modeled by the noise \( v_t \).

The optimal estimate of the state vector \( x_t \), in the sense that the mean squared prediction error \( E[|x_t - \hat{x}_t|^2] \) is minimized, is given by the following Kalman filter equations:

\[
\begin{aligned}
\hat{x}_{1|0} &= x_0 \\
P_{1|0} &= P_0 \\
\varepsilon_t &= y_t - H_t \hat{x}_{t|t-1} \\
S_t &= H_t P_{t|t-1} H_t^T + \Lambda_t \\
K_t &= F_t P_{t|t-1} H_t^T S_t^{-1} \\
\hat{x}_{t+1|t} &= F_t \hat{x}_{t|t-1} + K_t \varepsilon_t \\
P_{t+1|t} &= F_t P_{t|t-1} - P_{t|t-1} H_t^T S_t^{-1} H_t P_{t|t-1} F_t + R_t.
\end{aligned}
\]

It should be stressed that the initial value of the state vector \( x \) is not known, but rather is considered to have a prior distribution. The a priori distribution of the initial state vector is Gaussian with mean \( \pi(0) \) and variance \( P_0 \).

The Kalman filter does not only give the state vector estimate but also the conditional probability distribution of the state vector given the measurements

\[ x_t|y^{t-1} \in N(\hat{x}_{t|t-1}, P_{t|t-1}), \]

and the conditional probability distribution of the predicted measurement given old measurements

\[ y_t|y^{t-1} \in N(H_t \hat{x}_{t|t-1}, H_t P_{t|t-1} H_t^T + \Lambda_t). \]

Since our approach in this thesis is to model the data in the images as simply as possible, so as to reduce the computational complexity of the proposed algorithms, a special case of the general system (3.5) is used. The first simplification is to use models that can be written as regression models, as in equation (3.3). The other simplification is that we assume that the parameters in the regression model are constant, i.e., if the regression model (3.3) is written in state space form (3.5) it is assumed that no noise is
present in the first equation. The model in state space form, after simplifications, is the following

\[
\begin{align*}
\theta_{t+1} &= \theta_t \\
y_t &= \varphi_t^T \theta_t + e_t.
\end{align*}
\] (3.7)

To obtain equation (3.7) from equation (3.5) the state vector \(x\) is replaced by \(\theta\) and the following substitutions are made

\[
\begin{align*}
F_t &= I \\
v_t &= 0 \\
H_t &= \varphi_t^T,
\end{align*}
\]

where \(I\) is the identity matrix.

The first equation in (3.7) is the difference equation for a constant, which is the assumption on the parameter vector, and the second is the measurement equation with white Gaussian measurement noise. If the above stated Kalman filter is applied to the model (3.7) the following parameter estimator is obtained

\[
\begin{align*}
\hat{\theta}_{t+1|t} &= \hat{\theta}_{t|t-1} + K_t \varepsilon_t \\
\varepsilon_t &= y_t - \hat{\theta}_{t|t-1} \\
K_t &= P_{t|t-1} S_t^{-1} \\
S_t &= P_{t|t-1} + \Lambda_t \\
P_{t+1|t} &= P_{t|t-1} - P_{t|t-1} S_t^{-1} P_{t|t-1} \\
\hat{\theta}_{1|0} &= \theta_0 \\
P_{1|0} &= P_0,
\end{align*}
\] (3.8)

which is the well known recursive least squares algorithm (RLS) without forgetting factor, see [50].

3.3 Estimation of HMMs

In this section, answers to the questions listed in the introduction will be given. We introduce the compact notion \(\mathcal{H} = (\theta, Q, q)\) to represent a HMM, where \(\theta\) is the unknown parameter vector in the model of the output process, \(Q\) the transition matrix of the Markov chain and \(q\) the initial state probability distribution. The problems of interest are

i) How to, in a suitable sense, optimally estimate the state sequence \(z^N\).

ii) How to estimate the model parameters, or some of the model parameters, in the sense of choosing parameters that maximize \(P(y^N|\mathcal{H})\), where \(P(y^N|\mathcal{H})\) denotes the (conditional) probability of the measured data sequence \(y^N\) given the model – the parameters in the probability distributions of the observed process, the transition probabilities of the hidden Markov chain and the initial state probability distribution of the hidden Markov chain.
3.3 Estimation of HMMs

Problem i) corresponds to the third question 3 in the list at the beginning of this chapter. Two criteria for estimating the sequence $z^N$ will be covered, the local and global maximum a posteriori estimate. Problem ii) includes both question 1 and 2. The model parameters are the transition probabilities and the parameters in the observed process.

There are two main approaches to the solution of the two problems. The first is the "numerical" solution as in [11, 12] and the second is the Bayesian approach. For more information about hidden Markov models and estimation in that framework we refer to the comprehensive papers [39, 48]. For an overview see [31].

3.3.1 Baum's Algorithm

In the introduction to this section two problems, denoted by i) and ii), were mentioned, both formulated as maximum likelihood estimation problems. Practical implementation of the maximum likelihood requires a computationally efficient method for locating the global maximum of the likelihood of observed sequences of the output process $y^N$. Baum's algorithm is an example of a general technique for likelihood function maximization in the presence of incomplete observations. This technique is known in the statistical literature as the EM algorithm [22]. An efficient implementation of such a procedure, which is also known as the "forward-backward" algorithm, is reported in [12] and will be repeated here for the reader's convenience. In [12] it is proved that an iterative procedure produces a sequence of iterates that converge to a local maximum of the likelihood function. In the proof it is assumed that the observed process is generated by discrete probability distributions, i.e., the observed process can only take a finite number of values. It can be shown that Baum's algorithm is a fixed-point iteration, hence the usage of the adjective "numerical" when mentioning the algorithm in Section 3.3.

Before the forward-backward procedure is presented four conditional probabilities are introduced:

\[
\begin{align*}
\alpha_{t+1}(i) & = P(y^t, z_t = i | \mathcal{H}) \\
\beta_t(i) & = P(y^N_{t+1} | z_t = i, \mathcal{H}) \\
\gamma_t(i) & = P(z_t = i | y^N, \mathcal{H}) \\
\xi_t(i,j) & = P(z_t = i, z_{t+1} = j | y^N, \mathcal{H}).
\end{align*}
\]

Let us for a moment assume that $\alpha$ and $\beta$ are given. The remaining conditional probabilities, $\gamma_t(i)$ and $\xi_t(i,j)$, can then be expressed in $\alpha$ and $\beta$. The following calculations give the expression for $\gamma_t(i)$:

\[
\gamma_t(i) = \frac{P(z_t = i | y^N, \mathcal{H})}{P(y^N | \mathcal{H})} = \frac{P(y^N_{t+1} | z_t = i, y^t, \mathcal{H})P(z_t = i, y^t | \mathcal{H})}{P(y^N | \mathcal{H})} = \frac{\alpha_t(i)\beta_t(i)}{N},
\]

where the last equality follows from the independence of past and future data given the present state of the Markov chain. In other words, $\alpha_t(i)$ accounts for $y^t$ and state $z_t = i$ at time instant $t$ and $\beta_t(i)$ accounts for $y^N_{t+1}$ given state $z_t = i$ at time instant $t$. The
normalization factor $A'$, here equal to $P(y^N | \mathcal{H})$, makes $\gamma_t(i)$ a conditional probability, so that $\sum_{i=1}^N \gamma_t(i) = 1$. The probability denoted by $\gamma_t(i)$ is the probability that the state sequence takes state $i$ at time instant $t$, given all measurements.

In a similar way $\xi_t(i, j)$ can be derived from $\alpha$ and $\beta$. The expression for $\xi_t(i, j)$ is the following

$$\xi_t(i, j) = \frac{\alpha_t(i) q_{ij} f_j(y_{t+1}) \beta_{t+1}(j)}{N},$$

where $f_j(y_{t+1})$ is the probability of $y_{t+1}$ according to the $j$th density function, i.e., the probability of $y_{t+1}$ given that the hidden Markov chain is in state $j$. The interpretation of $\xi_t(i, j)$ is that it is the probability of a transition from state $i$ assumed at time instant $t$ to state $j$ at the next time instant $t + 1$, given all measurements. In the light of that interpretation, relation (3.13) can be intuitively understood. It is simply the joint probability of $y^t$ and being in state $i$ at time instant $t$, probability of the transition to state $j$ (from $i$), probability of $y_{t+1}$ given the Markov chain is in state $j$ and, finally, the probability of the rest of the sequence $y^N_{t+2}$ given the Markov chain is in state $j$ at time instant $t + 1$.

Using $\gamma_t$, the most likely state at time instant $t$, if the rest of the sequence is not taken into consideration, is

$$\hat{z}_t = \arg \max_{i=1, \ldots, M} \{\gamma_t(i)\} \quad 1 \leq t \leq N.$$

This estimate of $z_t$ is also called the local maximum likelihood estimate. There may be some problems with the above estimate since it is possible that the it takes a value that is impossible if we take into consideration the neighboring values of the state sequence. The estimate is simply the most likely state at each time instant with no regard to the complete sequence $z^N$.

Example 3.3.1 Assume a data sequence is given and that the transition matrix of the hidden Markov chain contains a zero, e.g., assume that $q_{12} = 0$. The local maximum likelihood estimate chooses the value of the state which gives the largest probability $P(z_t = i | y^N)$. This means that it is possible that $\hat{z}_t = 1$ and $\hat{z}_t = 2$ even though that combination is impossible since $P(z_{t+1} = 2 | z_t = 1) = 0$. This will be clearer when the recursive calculation of $\alpha$ and $\beta$ is presented below.

The local maximum likelihood estimate is still useful in practice and is often successfully used. The single best state sequence, i.e., the sequence which maximizes $P(y^N, z^N | \mathcal{H})$, is found using the Viterbi algorithm [67]. Note that the model $\mathcal{H}$ must be known. The formal steps in the Viterbi algorithm can be found in Box 1:

**Box 1: Viterbi algorithm.**

1. Initialization — The $M$ initial values of the Viterbi algorithm is simply the initial probabilities of the data sequence for $M$ different states of the hidden Markov chain. The quantities $\delta$ and $\Psi$ are explained in the next step.

$$\delta_1(i) = f_1(y_1) q_i, \quad 1 \leq i \leq M$$

$$\Psi_1(i) = 0$$
2. Recursion — The two quantities $\delta$ and $\Psi$ can be written as matrices with $N$ rows and $M$ columns. Let us take a closer look at time instant $t$. The algorithm goes through the $M$ possible states and calculates the probability of the data sequence assuming that the previous state was the one which was the most probable. The probabilities are stored in row $t$ of the $\delta$-matrix. In row $t$ of the $\Psi$-matrix the previous state which was the most probable is stored. One could say that the quantity $\Psi$ "remembers" which way the algorithm took to the present point.

$$
\delta_t(j) = \max_{i=1,...,M} [\delta_{t-1}(i)q_{ij}] f_j(y_t)
$$

$$
\Psi_t(j) = \arg\max_{i=1,...,M} [\delta_{t-1}(i)q_{ij}],
$$
for $t = 2,\ldots,N$, $j = 1,\ldots,M$

3. Termination — $\hat{z}_N$ is chosen to the value leading to the highest probability.

$$
\hat{z}_N = \arg\max_{i=1,...,M} [\delta_N(i)]
$$

4. State sequence backtracking — The backtracking step can be explained as follows. Assume the algorithm have just finished step 3. Further assume that the most probable state is $2$, in $\Psi_N(\hat{z}_N)$ the most probable way that led to state $2$ at time instant $N$ is found. That state is the estimate $\hat{z}_{N-1}$.

$$
\hat{z}_t = \Psi_t^{-1}(\hat{z}_{t+1}), \text{ where } t = N - 1,\ldots,1.
$$

Note that although the previously defined, see Equations (3.9) and (3.10), quantities $\alpha$ and $\beta$ are not found in the Viterbi algorithm, they are used in the calculation of $\hat{q}_{ij}$, $\hat{q}_i$ and parameters in $f$. The relations for $\hat{q}_{ij}$ etc, is shown in later in this subsection, see Equations (3.14)-(3.16).

The main point is that $\alpha$ and $\beta$ can be easily calculated. "Easily" here refers to low computational burden. The recursive calculations are of low complexity and can be transformed into vector multiplication, which decreases the computation time when programs designed for matrix calculation, e.g., MATLAB - see [35], are used. The recursive formulas for $\alpha$ and $\beta$ are the following:

$\alpha$:  
- a) $\alpha_1(i) = f_i(y_1)q_i$ for $i = 1,\ldots,N$.
- b) For $t = 1,\ldots,T - 1$ and $j = 1,\ldots,M$,

$$
\alpha_{t+1}(j) = \left[\sum_{i=1}^{M} \alpha_t(i)q_{ij}\right] f_j(y_{t+1}).
$$

- c) $P(y^N|\mathcal{H}) = \sum_{i=1}^{N} \alpha_N(i)$.

$\beta$:  
- a) $\beta_T(i) = 1$, for $i = 1,\ldots,M$.
- b) For $t = T - 1,\ldots,1$ and $i = 1,\ldots,M$,

$$
\beta_t(i) = \sum_{j=1}^{M} q_{ij} f_j(y_{t+1}(j)).
$$
The second problem still remains, namely to adjust the model parameters, i.e., \( \hat{q}_{ij} \), \( \hat{q}_i \), and \( \hat{\theta}(i) \), to maximize the probability of the observation sequence given the model. There is no known analytical solution to this problem, so an iterative procedure such as the Baum-Welch algorithm, or gradient techniques for optimization, must be used.

**Baum-Welch's Algorithm**

The idea behind Baum-Welch's algorithm is to iterate between two assumptions. That leads to the steps presented in this section. One "mode" of the algorithm is the calculation of the conditional probabilities (\( \alpha, \beta, \gamma \) and \( \xi \)) given the model \( \mathcal{H} \) and the measurements, and the estimation of the state sequence \( z_N \) using the above presented Viterbi algorithm, see Box 1. The second "mode" is to assume that the related conditional probabilities are given and the objective is the estimation of the model itself. Of course, neither the model \( \mathcal{H} \) nor the conditional probabilities are known. Instead the estimates of the conditional probabilities and the model are used, alternately, as if they were the true values.

The second mode, i.e., the re-estimation formulas for the model parameters \( q, Q \) and \( \theta \) are given below:

1. \( \bar{q}_i = \gamma_1(i) \quad \text{where} \quad i = 1, \ldots, M. \) \hspace{1cm} (3.14)

2. \( \bar{q}_{ij} = \frac{\sum_{t=1}^{N-1} \xi_t(i,j)}{\sum_{t=1}^{N} \gamma_t(i)}. \) \hspace{1cm} (3.15)

3. The expression for the estimate of \( \theta \) depends on the model for the observed process. When the regression model is used the following estimate for \( \theta \) is obtained:

\[
\hat{\theta}(i) = \left( \sum_{t=1}^{N} \varphi_t \varphi_t^T \gamma_t(i) \right)^{-1} \left( \sum_{t=1}^{T} \varphi_t y_t \gamma_t(i) \right). \hspace{1cm} (3.16)
\]

The whole algorithm, step by step, is summarized below.

**Box 2: Baum-Welch's Algorithm**

The Baum-Welch algorithm is given step-by-step. The variables \( \alpha, \beta, \gamma \) and \( \xi \) are defined in Equations (3.9)-(3.12). The Viterbi algorithm has been presented in Box 1.

1. The measurements \( y^N \) are given.
2. Initial values of the model \( \mathcal{H} \) are chosen.
3. Calculation of \( \alpha \) and \( \beta \) using the measurements and model \( \mathcal{H} \).
4. Calculation of \( \gamma \) and \( \xi \) using \( \alpha, \beta \), the model and measurements.
5. Estimation of the state sequence using the Viterbi algorithm.
6. Estimation of the model \( \mathcal{H} \) using measurements, \( \gamma \) and \( \xi \).
7. Go to 3 using the new model.
3.3 Estimation of HMMs

3.3.2 Bayesian Methods

The research on Bayesian methods for state estimation in linear discrete-time systems operating in Markov dependent switching environments was made possible by the work of Kalman and Bucy [40, 41]. Before we go further in the survey of existing Bayesian methods, the problem will be defined and the steps contained in almost all Bayesian methods will be enumerated.

In this section, RLS is used for estimating the parameters in the regression models (3.3). When merging the Markov chain and the regression model into a HMM we have, in fact, ended up with a bank of regression models and one Markov chain governing the switching between the different regression models

\[ y_t = \phi^T \theta(z_t) + e_t(z_t), \]  

(3.17)

where \( z_t \) is a Markov chain with properties as in Section 3.1, \( \phi \) and \( \theta \) were described in Section 3.2 and \( e_t \) is Gaussian white noise with different variances depending on the state of the Markov chain.

Equation (3.17) can be interpreted as \( M \) models running in parallel with the variable \( z_t \) deciding which model generates the output.

Example 3.3.2 Let us assume that \( M = 3 \). That implies that the measured data, at every time instant, is generated by one of the following models

\[ y_t = \phi^T \theta(1) + e_t(1) \]
\[ y_t = \phi^T \theta(2) + e_t(2) \]
\[ y_t = \phi^T \theta(3) + e_t(3). \]

Assume that the state sequence \( z^N \) is known and the task is to estimate the \( M \) parameter vectors \( \theta \). Nothing essentially changes in comparison with the linear filtering case and the solution is to run as many Kalman filters in parallel as there are states (\( M \)). We label the filters with 1 through \( M \) and update only one filter at each sampling instant, namely the one with the same label as the state the variable \( z_t \) is assuming. We end up with \( M \) estimates of the parameter vectors \( \theta(i) \) where \( i = 1, \ldots, M \), one for each state. For the system in Example 3.3.2 that would imply using three Kalman filters, \( K(1) \), \( K(2) \) and \( K(3) \). Since we know which model generated data at every time instant we simply update filter \( K(z_t) \) at time instant \( t \) and leave the other two resting.

If the state sequence is not known, the problem becomes more difficult and if the optimal solution is desired, we are inevitably faced with an exponential growth of computational complexity. Now, both the state sequence and the parameter vector have to be estimated. The strategy is the following, for each possible state sequence \( M \) Kalman filters are run and updated exactly as in the case when the sequence is known. The procedure results in a parameter estimate for every possible state sequence. This estimate is used for calculation of the probability for the sequences. The sequence with the highest probability, and the corresponding \( M \) parameter estimates, are chosen as estimates for the sequence and parameter vectors. The problem is that the number of possible sequences is high.

As mentioned previously, the RLS algorithm, without forgetting factor, will be used to identify the parameters in the regression models (3.3). The reason for using the RLS
algorithm without forgetting factor is the fact that we model the distance as being a constant within every segment, see Equation 3.7. The state space model used in the RLS algorithm is the one in Equation (3.7).

The review of Bayesian methods is continued with a list of the common steps in almost all Bayesian methods. The common steps are listed in Box 3. In Box 3, the measured variable is denoted by \( y \). In our application \( y \) is the measured distance. The hidden regime variable is denoted by \( z \) and the parameters in the model of the observed output process are put in a common vector denoted by \( \theta \).

**Box 3: Common steps in Bayesian methods**

1. \( P(z_{t-1}|y^{t-1}) \rightarrow P(z_t|y^{t-1}) \).
   
   The relation above is easily established
   \[
   P(z_t = i|y^{t-1}) = \sum_{j=1}^{M} q_{ij} P(z_{t-1} = j|y^{t-1}).
   \]

2. \( P(\theta_{t-1}|z_{t-1}, y^{t-1}) \rightarrow P(\theta_{t-1}|z_t, y^{t-1}) \).
   
   To show how 2) is performed the following expression will be used in the calculations
   \[
   P(z_{t-1} = j|z_t = i, y^{t-1}) = \frac{q_{ij} P(z_{t-1} = j|y^{t-1})}{P(z_t = i|y^{t-1})}.
   \] (3.18)
   
   Now, the relationship can be derived as follows
   \[
   P(\theta_{t-1}|z_t = i, y^{t-1}) = \sum_{j=1}^{M} P(\theta_{t-1}|z_{t-1} = j, z_t = i, y^{t-1})P(z_{t-1} = j|z_t = i, y^{t-1}),
   \]
   and substituting (3.18) in the equation above we obtain the following expression
   \[
   P(\theta_{t-1}|z_t = i, y^{t-1}) = \sum_{j=1}^{M} P(\theta_{t-1}|z_{t-1} = j, z_t = i, y^{t-1})q_{ji} \frac{P(z_{t-1}|y^{t-1})}{P(z_t = i|y^{t-1})} = \sum_{j=1}^{M} q_{ji} \frac{P(z_{t-1} = j|y^{t-1})P(\theta_{t-1}|z_{t-1} = j, y^{t-1})}{P(z_t = i|y^{t-1})}.
   \] (3.19)

3. \( P(\theta_{t-1}|z_t, y^{t-1}) \rightarrow P(\theta_t|z_t, y^{t-1}) \).
   
   The distribution for \( \theta_t \), given \( z_t \) and \( y^{t-1} \), is given directly by the first equation in (3.7) and the properties of the Kalman filter estimating \( \theta_t \). Recall that we are using the Kalman filter for estimation of \( \theta_t \). That implies that we have assumed \( \theta_t \) to be a random variable with a distribution given by the Kalman filter. The parameter vector \( \theta_t \) is Gaussian distributed with mean \( \hat{\theta}_{t-1} \) and variance \( P_{t-1} \).
3.3 Estimation of HMMs

4. $P(z_t|y^{t-1}) \rightarrow P(z_t|y^t)$.

To derive the relationship 4) we proceed as follows

$$P(z_t = i|y^t) = \frac{P(z_t = i, y^t)}{P(y^t)} = \frac{P(y_t|z_t = i, y^{t-1})P(z_t = i|y^{t-1})P(y^{t-1})}{P(y_t|y^{t-1})P(y^{t-1})} = \frac{P(y_t|z_t = i, y^{t-1})P(z_t = i|y^{t-1})}{\sum_{j=1}^{M} P(y_t|z_t = j, y^{t-1})P(z_t = j|y^{t-1})},$$

where the distribution of $y_t$ given old measurements and $z_t$, is given by the Kalman filter.

5. $P(\theta_t|z_t, y^{t-1}) \rightarrow P(\theta_t|z_t, y^t)$.

The probability distribution $P(\theta_t|z_t, y^t)$ is given by the Kalman filter. When the new measurement $y_t$ becomes available the Kalman filter produces the estimate $\hat{\theta}_t$ and $P_t$. This is the mean and variance in the normal distribution of the parameter $\theta_t$, given all measurements and states up to, and included, time instant $t$.

In the literature the first three steps are often called time update equations and the fourth and fifth are called the measurement update equations. The troublesome step is the second one. In step two we calculate the probability distribution for the parameter vector $\theta$, given old measurements and states, recursively. The number of such probabilities is $M^t$ since the possible paths, or the number of possible state sequences, grows in exactly this way. The effect of this is that in every new time instant we have to introduce $M$ new Kalman filters for every sequence existing so far. We will illustrate step two with an example.

Example 3.3.3 Assume that $M = 2$ and that $t = 3$. The number of possible sequences so far is $2^3$, and they are

$$z^{(1)} = 111, \quad z^{(5)} = 211,$$
$$z^{(2)} = 112, \quad z^{(6)} = 212,$$
$$z^{(3)} = 121, \quad z^{(7)} = 221,$$
$$z^{(4)} = 122, \quad z^{(8)} = 222.$$

In step 2) we calculate the probability of the parameter vector $\theta_3$ given $z_4$ from the probability of $\theta_2$ given $z_3$, and in step 3) we calculate the probability of $\theta_4$ given $z_4$. In step 2) we, in fact, split every sequence into two sequences. For example, $z^{(1)}$ will now be split into

$$z^{(10)} = 1111 \quad \text{and} \quad z^{(15)} = 1112.$$

Since all calculations are recursive, we can in step three use the Kalman filters based on the sequence $z^{(1)}$ to calculate the distribution (in fact the estimate of $\theta$) for $\theta$ when $z^{(10)}$ is given. We can, however, also transfer the information of the filters based on $z^{(1)}$ to two new filters used for calculation of the distribution of $\theta$ given the extended sequence $z^{(15)}$. The example shows how the splitting of sequence $z^{(1)}$ results in the introduction of $M = 2$ new filters.
Obviously the optimal estimator is not implementable even for a moderate number of states in the Markov chain and/or a moderate number of measurements. As we will see, the different methods in the literature all consist of either merging different state sequences and approximating the mixture of $M'$ normal distributions (see step 3) with $S$ normal distributions, or cutting off sequences with low probability to reduce the computational complexity.

Ackerson and Fu [2] appear to have been the first to treat the problem of state estimation in a switching environment. They considered the model where jumps are confined to means and covariances of the input and measurement noises, and they proposed a suboptimal algorithm assuming that the \textit{a posteriori} probability density of the system parameter given the past observations is Gaussian. As we have shown in step 3), it is a mixed Gaussian sum.

A combined detection-estimation scheme has also been employed in [66]. It is based on using a predetermined number $S$ of Kalman filters and keeping only $S$ most probable sequences "alive". The \textit{a posteriori} probability for the parameter is here assumed to be a mixed sum of $S$ Gaussian distributions. Merging "almost equal" sequences was also proposed. This approach is closely related to the methods used extensively for tracking targets in a dense multi-target environment [7].

So far, all the algorithms reviewed here deal with the parameter estimation problem only. Such is the case in general in literature before 1982, with exception of [32]. The problem of parameter estimation and state sequence detection is considered in [65], where an overview of existing methods is given. Three different classes of methods were recognized: i) generalized pseudo-Bayes algorithm (GPBA), ii) detection-estimation algorithm (DEA) and iii) random sampling algorithm (RSA).

The state sequence detection problem deserves additional treatment. We choose to summarize how the state sequence detection problem is treated in DEA. The criterion to be maximized is

$$\hat{z}_t = \arg \max_{z_t \in M} P(z_t | y^t).$$

In the DEA all possible extensions of the state sequence at time $t$ is considered. Assume an upper limit $S$ on the number of Kalman filters in parallel, set by the user, is given. Consider now the situation when the measurement $y_{t+1}$ is collected. Assume the algorithm uses $S$ Kalman filters at time instant $t$. We do not bother about how the state reduction has been performed so far. At time $t+1$ there are $M \cdot S$ branches. The branches are ordered by their \textit{a posteriori} probabilities, given by the Kalman filters, with the $S$ most probable branches being kept and the rest cut off. The state estimate is given by

$$\hat{\theta}_{t+1} = \sum_{i=1}^{S} \tilde{\theta}_{t}(i) P(z_t(i) | y^t).$$

The difference between the DEA and the GPBA is that in the DEA the state estimate is "clean" in the sense that it consists of a weighted sum of $S$ estimates having their origins in $S$ different state sequences. In the GPBA, on the other hand, the state estimate is based on merged sequences, so, in some sense, all sequences "survive" and influence the state estimate.

The random sampling algorithm (RSA) [3] is based on the Monte Carlo method. Since the idea behind RSA is principally different from the other mentioned methods and not used further in this thesis, we choose to refer the interested reader to [3].
There are some variations within the different classes of methods. In [19] the interacting multiple models algorithm (IMM) is introduced. The IMM algorithm is closely related to the GPBA. the difference is the timing of merging. In the GPBA the merging of different branches is performed after step 3). The IMM algorithm is derived by performing merging after step 2).

In [5] a new search strategy, called adaptive forgetting through multiple models (AFMM), was introduced in the context of recursive identification of the following class of “jumping” models

\[
\begin{align*}
\theta_{t+1} &= \theta_t + u_t \\
y_t &= \varphi_t^T \theta_t + e_t,
\end{align*}
\]  

(3.20)

where \( u_t \) is modeled as

\[
\begin{align*}
u_t = \begin{cases} 
  v & \text{w.p. } p \\
  0 & \text{w.p. } 1 - p,
\end{cases}
\end{align*}
\]  

(3.21)

where \( v \sim N(0, \delta^2) \). The model used in [5] is obviously of a different character than the models used in this thesis. The occurrence of jumps is modeled by a two-valued random variable. The main difference is however the modeling of the jumps in the parameter vector. When a jump has occurred, the parameter vector takes a new value which is \( \theta_t + v \), where \( v \) is a Normally distributed random variable with mean 0 and a given variance.

A detailed treatment of AFMM is given in the next subsection.

### 3.3.3 AFMM

In [5] AFMM denotes the complete identification algorithm, i.e., both the assumed models, (3.20) and (3.21), and the search strategy. We will in this thesis use the search strategy introduced in [5] and refer to the strategy as AFMM.

AFMM is closely related to the previously mentioned DEA. The strategy is to cut off branches with low probability, thus limiting the number of needed Kalman filters. The splitting of branches in AFMM is different than in the DEA. In the DEA all branches are allowed to split, while in AFMM we use the prior knowledge that the jumps from one segment to another have a low probability. With that assumption in mind, it is reasonable that only the most probable branch is allowed to split. AFMM is depicted in Figure 3.2, where the most probable sequence at every time instant has a filled dot.

In Figure 3.2 it is assumed that the hidden Markov chain has 3 states and the upper limit on the number of Kalman filters is \( S = 5 \). In the AFMM algorithm the number of Kalman filters (in our case, due to the specific model of the signal, we use RLS algorithms) is limited to \( S \). At each pixel step the \textit{a posteriori} probabilities for the \( S \) branches, i.e., different paths of \( z_1^T \), is produced by the \( S \) Kalman filters. The most probable branch is allowed to split, and the \( S - M \) branches with the lowest probabilities are cut off (forgotten).

A modification of AFMM is used in this thesis. The modification is that branches are not allowed to be cut off if they are younger than a specified age, i.e., it is not allowed to cut off a branch until it has “lived” a given number of samples. This additional parameter is called \textit{life length}, and its objective is to assure that branches live long enough so a change in variance can be detected. For that at least four or five data points are needed.
Figure 3.2: State sequence evolution when applying AFMM as search scheme. The number of states is 3 and the maximum number of filters is restricted to 5. The most probable sequence is denoted with black dots, and is the only one allowed to split. At every time instant the remaining sequences are either cut off, if their probability is low, or kept in the same state as previous time instant.

The main difference in comparison with the DEA is that only the most probable branch is allowed to split. Given the prior knowledge that the probability of switching a segment is low, we do not waste the limited computing capacity on splitting branches with low probability into branches with even lower.

3.4 Summary

In this chapter two main strategies related to estimation of state sequences and parameters in the framework of hidden Markov models were presented. The two strategies are the "numerical" Baum-Welch algorithm and the family of methods named Bayesian methods.

The questions answered in the chapter are:

- How to estimate the parameters in a hidden Markov model, i.e., how to estimate \( \theta(i), q \) and \( Q \), see Box 1.
- How to estimate the hidden state sequence \( z^N \), see Box 1 and Box 2.

Regarding the state sequence estimate, two criteria have been mentioned; the local maximum likelihood and the global maximum likelihood. The drawback of the local
maximum likelihood criteria is that not always a possible sequence is obtained as an estimate. The drawback of the global maximum likelihood is the higher computational complexity.

The Baum-Welch method is an iterative method. The two steps in every iteration are:
i) estimation of the state sequence and some related conditional probabilities given the model, i.e., $\theta(i)$, $q$ and $Q$, and
ii) estimation of model parameters given the conditional probabilities ($\alpha$ and $\beta$). Since neither the parameters nor the conditional probabilities are known, estimates from the previous iteration are used as if they were the true values. It is shown in the literature that the procedure converges to a local maximum of the likelihood function.

The Bayesian methods are a combination of linear filtering and different strategies of pruning and merging of possible state sequences in order to keep the computational complexity down. If computation power allowed it, the global maximum likelihood would be possible to obtain (which is not the case with the Baum-Welch method). In practice there is no guarantee that the most probable sequence is not cut off by the pruning algorithm. Several merging/pruning algorithms have been mentioned and a more detailed treatment of the AFMM algorithm has been performed.
Review of Models used in Image Segmentation

Loosely speaking, image segmentation can be described as an attempt to partition an image into areas with significantly different properties, relative to some measure of significance defined by the user. The segmentation is meant to be the starting point from which a further examination of the image departs. For more discussion on the topic of segmentation see Chapter 1.

In the area of image segmentation that could be called “model-based” image segmentation, one of the mostly used models is the class of stochastic processes called Markov random fields (MRF). The concept of a MRF derived from attempts to put into a general probabilistic setting a very specific model called after the German physicist Ernst Ising, see [36]. Using this model, Ising tried to explain certain empirically observed facts about ferromagnetic materials.

To use neighborhoods to model correlation of pixel intensities, labels etc., is of course natural. Using different prior probabilities for the possible neighborhood configurations is one tool for modeling clustering of pixels with similar properties. The result is a variety of Markov-type models related closely to MRFs. In our review of existing models, the Markov-type models will have special names, and when referring to MRF we will mean the model defined in Section 4.1.

There are other ways to extract information from images, and determine areas with similar properties. Edge detection using linear two-dimensional filters is a classical approach. Derivative operators are often used to emphasize the changes in the measured quantity of the image, e.g. gray-levels, distances etc. It also turns out that not only edges are important structures in an image, but also their orientation. The orientation is a significant property not only of edges but also of any pattern. The local orientation of a pattern is the property that leads to another large field in image processing, namely image segmentation by textures. For an overview of techniques in that field see [53], and for a more detailed investigation of the different techniques, such as feature-based methods, model-based methods and structural methods see [15, 17, 20, 73].

Recently the idea of representing the image on a multigrid or multiscale grid has drawn much attention. The fine structures which are contained in an image are represented on a fine grid, whereas the large scales can be represented on a much coarser grid. A new research area, multiresolutional image processing, has been established based on these
4.1 Markov Random Fields

The same notational conventions as in Chapter 3 will be used in this chapter. The field will be denoted by \( Z \) and the measured pixel values, i.e., intensity, range, or really any pixel attribute by \( Y \). As mentioned previously, the concept of neighbors is used when describing the correlation between adjacent pixels.

Let \( Z_{N_x \times N_y} = \{(i, j) | 1 \leq i \leq N_x, 1 \leq j \leq N_y \} \) denote the \( N_x \times N_y \) integer lattice. At the junctions of the lattice an integer value is attached. The integer values can be one of a set \( M = \{1, \ldots, A/\} \) and the junctions of the lattice are called sites. If this notion is compared with the one-dimensional case, the \( Z \) corresponds to the hidden Markov chain (in this case field) taking values at the different time instants (in this case sites). In the two-dimensional case it may be useful to see the sites as the coordinate system, and the integer values as an integer valued function with the sites as support. Then \( Y_{N_x \times N_y} \) denotes the measured pixel attributes of the original image. For simplicity, also assume that the pixel attribute is modeled by

\[
y_{i,j} = \theta(z_{i,j}) + \epsilon_{i,j},
\]

where \( \theta(z_{i,j}) \) is the distance associated with state \( z_{i,j} \), and \( \epsilon \) is Gaussian, zero mean with the property

\[
E\{\epsilon_{i,j}\epsilon_{l,k}\} = 0 \text{ if } i \neq l \text{ or } j \neq k.
\]

**Definition 2** Let \( S \) be the set of sites in the previously defined rectangular lattice

\[
\mathcal{S} = \{s_1, \ldots, s_{N_x \times N_y}\}.
\]

A neighborhood system \( \mathcal{G} \) on the lattice \( S \) is defined as

\[
\mathcal{G} = \{\mathcal{G}_s : s \in \mathcal{S}\},
\]

such that for any \( s \in \mathcal{S} \)

\[
s \not\in \mathcal{G}_s,
\]

and

\[
s \in \mathcal{G}_r \iff r \in \mathcal{G}_s.
\]

The second condition in the definition of a neighborhood system states that there has to be symmetry when naming neighbors, i.e., if \( j \) is a neighbor of \( i \) then \( i \) must be a neighbor of \( j \). There are some additional terms in connection with neighborhood systems. We say that the neighborhood system is of order one if only the four nearest sites are included in the neighborhood system. A neighborhood system of order two includes the diagonal sites in the neighborhood system too, see Example 4.1.1.

**Definition 3** Given a finite lattice \( S \) and a neighborhood system \( \mathcal{G} \) on \( S \), a random field \( Z = \{Z_s, s \in \mathcal{S}\} \) is a Markov random field (MRF) if and only if

\[
P(Z_s = z_s | Z_r = z_r, r \in \mathcal{S} - \{s\}) = P(Z_s = z_s | Z_r = z_r, r \in \mathcal{G}_s).
\]  
(4.1)
Definition 4 A subset $C \subseteq S$ is a clique if every pair of distinct sites in $C$ are neighbors, see Definition 2. $C$ denotes the set of cliques.

Let us take a closer look at one site $s$ in the image and assume that a neighborhood system, say a second order system, is given. Every subset of the neighborhood $s \cup G_s$ (where $G_s$ is all surrounding sites in this case) where $s$ is an element, that has the additional property that all the sites in the subset also are mutual neighbors (according to the neighborhood system), is called a clique.

Example 4.1.1 An example of first order MRFs and second order MRFs, with related cliques will be shown in this example. The black dot in Figures 4.1 (first order MRF) and 4.2 (second order MRF) is the location for the pixel whose probability given the rest of the image is sought. The related set of cliques $C$ is shown in the box to the right in the both figures.

![First order neighborhood system](image1)

Figure 4.1: First order neighborhood system. The white dots are neighbors to the black dot. To the right the set of cliques is shown.

![Second order neighborhood system](image2)

Figure 4.2: Second order neighborhood system. The white dots are neighbors to the black dot. To the right the set of cliques is shown.

Obviously the number of clique types grows rapidly as the order of the neighborhood system increases. Very complex dependencies can be modeled by choosing suitably large neighborhood systems. However, relatively simple neighborhood systems are adequate
in modeling most scenes of interest. Most often, first and second order neighborhood systems $Q^1$ and $Q^2$ are used. Notice that the sites at the boundaries will have fewer neighbors. In image processing it is natural to use the so-called “free-boundary” convention, i.e., certain adjustments are made for sites at the boundaries. The adjustments is performed in the following way. The neighborhood systems of the sites at the boundaries of the image are modified to only include sites in the image. The sites which would naturally fall outside the image are simply deleted. So, e.g., the neighbors of the site in the upper left corner of an image when a first order neighborhood system is used, see Figure 4.1, are the sites located one step below and one step to the right. Other conventions are periodic boundaries and toroidal lattices.

Definition 5 ([28]) A Gibbs distribution relative to \( \{S, G\} \) is a probability measure \( P \) on $Z^{N_x \times N_y}$ with the following representation

$$P(z^{N_x \times N_y}) = \frac{1}{\mathcal{N}} e^{-U(z^{N_x \times N_y})}, \quad (4.2)$$

where \( \mathcal{N} \) is a normalization constant and \( U \), called the energy function, is of the form

$$U(z^{N_x \times N_y}) = \sum_{C \in \mathcal{C}} V_C(z^{N_x \times N_y}),$$

where \( V_C \) is called the neighborhood system Gibbs potential. The normalizing constant \( \mathcal{N} \) is defined by

$$\mathcal{N} = \sum_{z^{N_x \times N_y}} e^{-U(z^{N_x \times N_y})},$$

and is called the partition function.

An useful way of thinking of this measure is the following. The potential \( V_C \) is a way to assign a number to every sub-configuration \( z_C \) of a configuration \( z^{N_x \times N_y} \). In practice, every clique in, e.g. Figure 4.1, is assigned a number, i.e., indirectly a potential, depending on the field state at the sites. Assume the states one and two are possible, and further assume that the probability of two neighbors being in the same state is higher than them being in different state, i.e., we assume there are clusters of pixels in the same state in the image. One should then assign a lower potential to the cases when the second clique in Figure 4.1 has the values 1-1 and 2-2, than when it has the values 1-2 and 2-1. Now, for a fixed realization \( z^{N_x \times N_y} \), at every site a sum of all clique-potentials is taken, and that sum is \( \sum_{C \in \mathcal{C}} V_C(i,j) \). This sum, of course, depends on the actual realization of the field, and the a priori given potentials for the different combinations of the states in the cliques. This is repeated for all sites \( (i,j) \) and is summed to \( U(z^{N_x \times N_y}) = \sum_{C \in \mathcal{C}} V_C(z^{N_x \times N_y}) \) which is the energy of the field. The probability of the specific realization is then given by equation (4.2).

A very important theorem is the one which states the equivalence of Gibbs distribution and MRFs with strictly positive probabilities.

Theorem 1 Let $G$ be a neighborhood system. Then $Z$ is a MRF with respect to $G$ if and only if $P(z^{N_x \times N_y})$ is a Gibbs distribution with respect to $G$. 
4.1 Markov Random Fields

Proof [29, 30]. Let \( P(z^{N_x \times N_y}) \) be a probability measure on \( Z^{N_x \times N_y} \) defined by a nearest neighbor Gibbs potential \( V \),

\[
P(z^{N_x \times N_y}) = \frac{1}{Z} e^{-\sum_{c} V_C(z^{N_x \times N_y})},
\]

where the summation is over all cliques on the lattice. Then

\[
P(z_{i,j}|z_{S-(i,j)}) = \frac{P(z^{N_x \times N_y})}{\sum_{\tilde{z}^{N_x \times N_y}} P(\tilde{z}^{N_x \times N_y})},
\]

where \( \tilde{z} \) is any configuration which agrees with \( z \) at all sites except possibly \((i,j)\).

Thus,

\[
P(z_{i,j}|z_{S-(i,j)}) = \frac{e^{-\sum_{c} V_C(z^{N_x \times N_y})}}{\sum_{\tilde{z}^{N_x \times N_y}} e^{-\sum_{c} V_C(\tilde{z}^{N_x \times N_y})}}.
\]

For any clique that does not contain \((i,j)\), \( V_C(z) = V_C(\tilde{z}) \). Thus the terms that corresponding to cliques not containing \((i,j)\) will cancel. The result is that the probability depends only on the values at \((i,j)\) and its neighbors, which implies that \( P \) is a MRF.

The proof that a MRF determines a nearest neighbor Gibbs measure is more complicated and we refer the reader to [29, 30] for more details.

\[
\]

The probabilities on the right hand side of equation (4.1) are called the local characteristics of the field and what is shown in Theorem 1 is that for a random field, these local characteristics are uniquely determined by choosing potentials in the corresponding Gibbs distribution. The main difficulty with the MRF formulation is that the joint distribution of \( Z \) is not apparent, and that it is extremely difficult to spot local characteristics, i.e., to determine when a given set of functions \( \Pi(z_s|z_r, r \neq s) \) and \( s \in S \) are conditional probabilities for some unique joint distribution on \( Z \). The difficulty which has been extensively discussed in the literature, see [8, 9, 14, 15, 21, 26, 64, 72], is that there are two main definitions of nearest-neighbor models which might be applied to describe the interaction between the field at different sites. One definition is in terms of a joint probability distribution of \( Z \) and the form required is a product of the form

\[
\prod_{i,j} \psi(z_{i,j}, z_{i-1,j}, z_{i+1,j}, z_{i,j-1}, z_{i,j+1}).
\]

The other approach is through conditional probabilities

\[
P(z_{i,j}|z^{N_x \times N_y} - z_{i,j}) = P(z_{i,j}|z_{i-1,j}, z_{i+1,j}, z_{i,j-1}, z_{i,j+1}),
\]

i.e., the probability distribution of the field at one specific pixel, conditioned on the whole image, is dependent on the nearest neighbors only. The latter approach is more intuitively appealing, but there are difficulties associated with it. One is the lack of a direct method of evaluating the joint probability distribution on the lattice and the functional form of the conditional probability is subject to severe consistency conditions. The consequence of this is that the conditional probability formulation is degenerate with respect to the joint probability formulation.

The problem of expressing the joint probability of the hidden field, however, is solved by Theorem 1 by using the Gibbs distribution - MRF equivalence. In fact, with some
experience, one can choose potentials in accordance with the desired local behavior. The problem of degeneracy of the conditional probability formulation has not proved to be a significant problem in practice. The class of distributions described by the conditional probability formulation, has shown itself to be sufficiently rich to justify the now intensive research on MRFs described through the Gibbs distribution. The new wave of research on the mentioned field was sparked by the publication of [28].

The Gibbs distribution is essentially an exponential distribution. However, by choosing the clique potential function $V_C$ properly, a wide variety of distributions both for discrete and continuous random fields can be formulated as Gibbs distributions. Some examples are binary, $m$-valued (also called autologistic), binomial, Poisson and Gaussian random fields. For a more detailed overview of various types of Gibbs distributions defined in terms of corresponding potential functions, see [15].

We have so far not addressed another problem associated with MRF modeling of images, namely the computational complexity associated with finding maximum a posteriori estimates of the field. The criterion which is maximized is the following

$$P(Z^{N_x,N_y}|Y^{N_x,N_y}) = \frac{P(Y^{N_x,N_y}|Z^{N_x,N_y})P(Z^{N_x,N_y})}{P(Y^{N_x,N_y})},$$

where $P(Y^{N_x,N_y}|Z^{N_x,N_y})$ is given by the model used for modeling the image attribute (intensity, range, etc.) and $P(Z^{N_x,N_y})$ is given by

$$P(Z^{N_x,N_y} = z^{N_x,N_y}) = \frac{1}{N}e^{-U(z^{N_x,N_y})}.$$

It is obvious that the maximum a posteriori estimation presents a formidable computational problem. The number of possible field configurations is $M^{N_x \times N_y}$, where $M$ is the number of states of $Z_{ij}$. Even for images with only two states the number of combinations is so huge, e.g., for an image of the size $64 \times 64$ pixels the number of combinations is $2^{4096}$, that it rules-out any direct search. Two approaches for tackling the complexity problem will be presented in next two subsections. One approach is to use simulated annealing and the other is based on graduated non-convexity.

### 4.1.1 Simulated Annealing

The complexity of a direct search for maximum a posteriori estimates of the field configuration of the image forces us to use alternative methods. One alternative is to use the conditional probabilities at every site to change the values of the field at that site according to the conditional probability. Doing so, we hope to arrive to the maximum of $P(Z^{N_x,N_y}|Y^{N_x,N_y})$. There are two different approaches. We could either change the field values by drawing a new value according to the conditional probability distribution, which implies that we allow the new value to be less probable than the old, or always set the value of the field to the one with highest probability. The two approaches are called stochastic relaxation and deterministic relaxation.

#### Stochastic Relaxation

The idea behind stochastic relaxation involves introducing an auxiliary distribution of the following type

$$P_T(Z^{N_x,N_y}|Y^{N_x,N_y}) \sim \left[ P(Y^{N_x,N_y}|Z^{N_x,N_y})P(Z^{N_x,N_y}) \right]^{1/T}, \quad (4.3)$$
4.1 Markov Random Fields

where \( T \) is an additional parameter called "temperature of the system". Note that when \( T \to \infty \) the distribution (4.3) tends to an uniform distribution, while if \( T \to 0 \) the distribution (4.3) is concentrated on the maximum a posteriori estimate.

The idea is to create, in a stochastic way, a set of images by making random runs following a Gibbs distribution \( P(Z_{N_x,N_y}|Y_{N_x,N_y}) \). Starting from an initial field configuration \( Z(0) \), the change from \( Z(n) \) to \( Z(n + 1) \) only concerns one pixel, at most, and the change of the pixel in question is done randomly according to the local conditional distributions. When \( n \to \infty \), \( Z(n) \) resembles a random configuration with distribution \( P(Z_{N_x,N_y}|Y_{N_x,N_y}) \). When this technique is used with a distribution like the one defined in (4.3) and when the temperature \( T \) is brought very slowly to zero, convergence takes place towards the configuration with the maximum a posteriori probability. The conditions to be fulfilled by the random set of images \( Z(0), Z(1), \ldots \), in order to achieve convergence are very flexible and do not depend on the order in which the pixels in the image are explored. The only requirement is that all pixels are visited.

In [28] stochastic relaxation was introduced for solving the field estimation problem. In addition a schedule for lowering the temperature which also guarantees convergence to the global maximum of the posterior distribution is given. The major weakness is that the prescribed schedule cannot be followed in practice, since the time required would be unacceptable. Instead, a faster schedule is used and the result, this time without guarantee of obtaining the global maximum, is obtained after 300-1000 sweeps of the image. Another novelty introduced in [28] is the "hierarchical" stochastic model. In [28] it is assumed that the observed data can be modeled by a MRF. The field consists of two "dimensions": the usual field (the observed data) and a hidden field of edge elements located between the pixel sites, called the line process.

Deterministic Relaxation

The concept of deterministic relaxation will be introduced through a discussion of a specific model.

Compound Gauss-Markov random fields are introduced in [38]. The model used is similar to the one in [28], but with the difference that the observed data are continuous grey levels modeled by several linear-shift invariant models

\[
y_{i,j} = \sum_{k,l \in G_{i,j}} c_{k,l} s_{i-k,j-l} + w_{i,j},
\]

where \( G_{i,j} \) is the coefficient support region and \( w_{i,j} \) is a Gaussian random field satisfying the following covariance constraint

\[
E\{w_{i,j}w_{k,l}\} = \begin{cases} 
\sigma_w^2 & \text{if } (i,j) = (k,l) \\
-\sigma_{i-k,j-l}\sigma_w^2 & \text{if } (i-k,j-l) \in G_{i,j} \\
0 & \text{otherwise}
\end{cases}
\]

The estimation of the configuration of the hidden MRF is performed using an algorithm based on deterministic relaxation. An extension of the iterated conditional mode algorithm (ICM), presented in [13], is used. In the ICM the estimation of the MRF is performed by maximum marginal probabilities, i.e., the argument maximizing the following probability

\[
P(z_{i,j}|Z_{N_x,N_y} - z_{i,j}, Y_{N_x,N_y}),
\]
is chosen as the field estimate. When applied to each pixel in turn, this procedure defines a single cycle of an iterative algorithm for estimating $\hat{Z}^{N_x \times N_y}$. The algorithm is applied for a fixed number of cycles or until convergence is achieved, to produce the final estimate of $Z^{N_x \times N_y}$. Note that there is nothing random in this case. The algorithm always takes a step towards higher probability. In the stochastic relaxation (simulated annealing) there always is a possibility that the drawn value of a field would result in a field configuration with lower probability.

The novelty in [38] is the usage of a line process similar to the one in [28] in combination with the extension of the ICM to continuous valued image data.

The advantage of deterministic relaxation methods is the fast convergence if compared to simulated annealing. The disadvantage is the local nature of the optimum achieved, i.e., convergence to a global optimum cannot be guaranteed.

4.1.2 Graduated Non-convexity Algorithms

In image segmentation a frequently-used criterion for estimation of hidden fields is the maximum a posteriori criterion. Finding the maximum of the a posteriori distribution is an intricate problem. The form of the a posteriori distribution, when Gaussian noise is used as model for the uncertainty in the image that cannot be explained by the image model itself, is the following

$$
\log P(Z|Y) \sim \log P(Y|Z) + \log P(Z). (4.4)
$$

The first term on the right hand side of equation (4.4) is essentially a sum of quadratic terms (prediction errors)

$$
D = \sum_{i,j} (y_{i,j} - \hat{y}_{i,j})^2.
$$

The other term contains the interaction between the different sites of the field. If the Gibbs distribution is used, the form of the last term is usually

$$
S = \sum S(z_{i,j}, z_{\neq i,j}) = \sum \sum V_C
$$

which is simply a sum of potentials (chosen by us). The sum is a function of discrete variables, i.e., the values of the hidden field at the site visited and the sites in the neighborhood of the site visited.

We thus have ended up with a function $F = D + S$ that is to be maximized to obtain the maximum a posteriori estimate. The function $F$ lacks the mathematical property of "concavity", meaning that the field $Z$ may have numerous configurations corresponding to a local maximum of $F$. Obviously a method of maximizing $F$ is needed which avoids the pitfall of getting stuck at local maxima. In the graduated non-convexity (GNC) method the cost function $F$ is first approximated by a new function $F^*$ which is concave and hence can only have one maximum, the global one. Ascent on $F^*$ (ascending, that is, in the $N_x \times N_y$ dimensional space of $Z$) must end up at that maximum. If the maximum of the function $F^*$ coincides with the maximum of $F$, the objective is accomplished. If not, a more general strategy must be used. An entire sequence of cost functions $F^{(p)}$, $0 \leq p \leq 1$, is used. These functions are chosen such that $F^{(0)} = F$, the true cost function, and $F^{(1)} = F^*$, the concave approximation. In between, $F^{(p)}$ changes in a continuous
4.1 Markov Random Fields

A figure is shown illustrating the maximum of a non-concave function $F$ may be found by maximizing a concave approximation $F^*$ (a). If that does not work (b), the maximum may still be found by the GNC algorithm, which runs uphill on each of a sequence of functions (c), to reach the global maximum. The figure is taken from [18].

An intricate problem is how to choose the new approximating cost function. The procedure is depicted in Figure 4.3.

To conclude, the GNC algorithm contains two parts. The first part is the iterative translation from $F$ to $F^*$. Within every, previously mentioned, translation, the problem of maximizing $F^{(p)}$ remains. For this, any numerical method can be used. Numerical methods for maximizing functions are also iterative procedures, so the result is an iterative procedure where every iteration consists of an iterative numerical algorithm.

Some recent papers using GNC are [27, 43, 51]. In [43] GNC is used as a tool for finding the "simplest" description of an image. The idea is to translate the image partitioning problem into a problem of finding a complete description of an image, in terms of a specified descriptive language, that is the simplest in the sense of being shortest. A solution is to model the image as the corruption of an underlying piecewise smooth image. As criterion adopted for the purpose of determining an unique description, is the minimum description length criterion. The obtained criterion has many local minima, due to various possibilities for image partitioning, and GNC is used to find the global minimum.
4.2 Markov Mesh Random Fields

Estimation in the framework of MRFs is highly computationally complex, and thus very time consuming. One way to circumvent the computational complexity problem is to introduce simplifications in the assumed model. This leads to a new class of models - causal MRFs.

The causal Markov mesh random fields (MMRF) were first proposed in [1]. We begin with the definition of a MMRF.

**Definition 6** A third order Markov mesh random field is defined by the property

\[ P(z_{i,j}\mid z_{k,l}, k < i \text{ or } l < j) = P(z_{i,j}\mid z_{i-1,j-1}, z_{i-1,j}, z_{i,j-1}), \]

for any \( 1 < i \leq N_x \) and \( 1 < j \leq N_y \). Along the first row and first column of the lattice, we have the boundary conditions

\[ P(z_{1,j}\mid z_{i-1,j-1}, z_{i-1,j}, z_{i,j-1}) = P(z_{1,j}) \quad \text{if } i = j = 1, \]

\[ = P(z_{i,j}\mid z_{i,j-1}) \quad \text{if } n > m = 1, \]

\[ = P(z_{i,j}\mid z_{i-1,j}) \quad \text{if } m > n = 1. \]

The support of the MMRF is depicted in Figure 4.4.

![Figure 4.4: Set of sites associated with a third order MMRF.](image-url)

One useful property of a MMRF is the possibility of calculating the joint probability of the whole field recursively. That can be written formally in the following way. For any \((i,j) \in S\), where \(S\) is the set of sites, the probability for the field at \((i,j)\) is given by

\[ P(Z_{i,j}) = \prod_{k=1}^{i} \prod_{l=1}^{j} P(z_{k,l}\mid z_{k-1,l-1}, z_{k-1,l}, z_{k,l-1}). \]

The approach used in [24, 25] is to take advantage of the causality of the relation above. As an estimate of the field a smoothed, using the analogy of the term “smoothed”
4.3 NSHP Markov Chain

in filtering, local maximum a priori estimate is used. Formally the estimate can be written as follows:

\[ \hat{z}_{ij} = \arg \max_m P(z_{ij} = m|Y^{i-1,j+1}). \]

Obtaining a recursive relation for the expression above demands derivation of recurrence relations for the following probabilities:

- \( P(z_{ij}, z_{i-1,j}, z_{i,j-1}, z_{i-1,j-1}|Y_{i,j}) \)
- \( P(z_{i-1,j}, z_{i,j-1}, z_{i-1,j-1}|Y_{i,j} - Y_{i,j}). \)

Derivation of recursive expressions for the relations above, together with a simplification of the Baum-Welch re-estimation algorithm for estimation of model parameters, i.e., the conditional probabilities in the MMRF—both the conditional probabilities in the body of the image and the conditional probabilities of the pixels at the borders with reduced support, can be found in [24, 25].

4.3 NSHP Markov Chain

The non-symmetric half-plane (NSHP) Markov chain, also called the unilateral (one-sided) Markov random field, is another simplification of the general MRF. The NSHP Markov chain will be used as a model in segmentation algorithms later in this thesis. The NSHP Markov chain is defined in the following way.

**Definition 7** Consider \( Z \) to be a random field defined over an \( N_x \times N_y \) rectangular lattice \( S \). \( Z \) is a NSHP Markov chain (or an unilateral MRF) if and only if

\[ P(z_{ij}|z_{k,l}, l < j \text{ or } (l \leq j \text{ and } k < i)) = P(z_{ij}|z_{i-1,j}, z_{i+1,j-1}, z_{i,j-1}, z_{i-1,j-1}). \] (4.5)

Near the boundary of the image, natural "shortening" of the support is applied. The NSHP Markov chain has similar properties to the Markov mesh models regarding calculation of the field probability \( P(z_{N_x,N_y}) \), i.e., the probability \( P(z_{N_x,N_y}) \) can be calculated recursively. It is shown in [37] that MMRFs form a subset of NSHP Markov chains and that NSHP Markov chains have better properties in the sense that the correlation between pixels induced by using NSHP Markov chains, is better suited for segmentation of images containing objects whose edges lie diagonally in the image, i.e., form an angle of 45° with a horizontal line.

A drawback of the NSHP Markov chain in comparison with the MMRF is the increase of computational complexity of the estimation algorithms. This is due to the bigger support of the NSHP Markov chain, i.e., the conditional probability of a pixel depends on a bigger neighborhood. This is, however, negligible compared to the time consuming algorithms for non-causal MRFs.

The support of the conditional probability (4.5) is depicted in Figure 4.5. Figure 4.5 suggests that the NSHP Markov chain model is a more natural assumption if the scanning pattern is assumed to be of the type "row-by-row". This is especially important when using the algorithms in real-time applications like target tracking *etc.*
4.4 Summary

In this chapter some of the existing Markov-type models often used in image segmentation have been briefly reviewed. Related estimation methods and problems associated with them have been mentioned. The following models were briefly presented:

- Markov random field (with non-causal support)
- Markov mesh model
- NSHP Markov chain.

Two aspects of modeling images with MRFs were treated. One is the properties of the model itself and its usefulness when modeling images, taking into account the scanning pattern and real-time issues. Among the causal models the NSHP Markov chain appears to be the most useful. The model fits well to the scanning order and is natural to choose.

The other aspect is the complexity of estimation/segmentation algorithms induced by the models. Especially the complexity of estimation algorithms used in connection with non-causal MRFs was highlighted. As examples of standard routines for field estimation of non-causal MRFs stochastic relaxation, deterministic relaxation and the graduated non-convexity algorithm were mentioned.

The subject of MRFs is old and the amount of literature on the subject is large. We have throughout the presentation included pointers to the most often cited literature in the subject.
Row by Row Segmentation of Laser Range Images

In this chapter we present an approach to the problem of laser radar range image segmentation. The approach is based on row-by-row segmentation of the given images. The algorithms in this chapter make almost no use of the knowledge that in images there exist a correlation in two dimensions between pixels. The general idea is modelling horizontal scans of the terrain as piecewise constant or piecewise linear signals in random noise. Segmentation with such prerequisites has been reported previously in [44], but with the difference that hidden Markov models were not used to model the switching between the different models. One approach [44] is that the transitions from one segment to the next is controlled by a two-valued stochastic process. The model used is similar to the model used in [5] and can be found in Section 3.3.2. Equation (3.20) and (3.21). The two values of the stochastic process determine if a jump has occurred at a specific pixel, or if the pixel belongs to the same segment as the previous one. Different segments have nothing in common, since when a jump is detected the algorithm is “restarted” and old information is forgotten. As mentioned, in this section we use hidden Markov models to model the switching between segments. This infers that different parts of a signal (in our case a row from the laser range image) can belong to one specific segment (state) and hence can segments be used in a future classification of the image, e.g., background terrain vs. object etc. The search scheme used in [44] is the same as in this thesis – AFMM.

The main advantage of the one-dimensional algorithms is the relatively low computational complexity, which is important in real time applications. The drawback is, of course, that it is an unnatural model for an image, and as will be shown in Chapter 6, better results are obtained with two-dimensional methods.

Two methods will be presented. In Section 5.1, the Baum-Welch algorithm will be used. In Section 5.2 we will, inspired by the work in [31], use an one-dimensional segmentation method for segmentation with respect to both the range and variance. Finally in Section 5.3 a summary of the chapter will be given.
5.1 The Baum-Welch Algorithm

In this section the method presented in Chapter 3, the Baum-Welch algorithm, will be used for segmentation of laser range images. The segmentation will result in finding the estimates of the state sequence and the parameters in the terrain models. Since the theory used in this chapter has already been presented in Chapter 3, first the terrain models used will be introduced in the following subsection. We will proceed to present results of the application of the algorithm to real data from the laser system.

5.1.1 The Terrain Model

We remind the reader that the data we are working with is data collected by a laser radar range system. The data is thus an image with pixel values corresponding to the distance between the terrain and laser looking in that specific direction. The objective is to find objects in the image, and one way to do that is to first segment the image, i.e., partition the image in areas with different properties, and then look closer at the different image segments and try to locate the objects looked for. Since one of the prerequisites for fast algorithms is simple models, we have chosen to model the range variation in a simple fashion. The terrain model is the same as in [44] and it is assumed that a row/column can be decomposed into segments with constant range. This model will be used throughout the thesis when testing different algorithms. The model can be written formally as

\[ y_t = \theta(z_t) + e_t(z_t), \]  

(5.1)

where \( y_t \) is the measured range, \( \theta \) is the constant (given the hidden state) with which we model the “true” range (there are \( M \) different ranges since a Markov chain with \( M \) states is used and hence one of \( M \) models generate the output at every time instant) and \( e_t \) is white Gaussian noise. The Markov chain state is denoted by \( z_t \). This model can readily be extended to a general regression model, e.g., for modeling the terrain with piecewise-linear models we introduce the following parameter and regression vector

\[ \phi^T = [\theta_1 \theta_2] \]
\[ \phi^T = [1 \ t]. \]

See Section 3.2 for more comments on possible models for the observed output process.

The result is a bank of \( M \) models each describing a segment in the horizontal scan with different range and/or variance of the measurement noise. In other words, the local variations within every segment, i.e., small variations in the terrain, are modeled with white Gaussian noise. The level of the variance is reflected on the variance of the white noise. “Noisy” objects, such as bushes, crown of trees etc, will have larger variance than, e.g., man made objects. The modeling of essentially different parts, in our case different distances, of the horizontal scan is handled by using different linear models, i.e., models for the range in different segments. Note that here “segment” means the set of data generated by the same model (one of \( M \)), and furthermore the segments do not have to be connected, i.e., data in the first part of the scan and in the last part can belong to the same segment despite belonging to different parts of the row. That fact can be easily be exemplified with test image #1, see Figure 2.2. If two states are used in the segmentation process the shield in the middle of test image #1 will be assigned one state and the background the other. The segment denoting the background is not connected
(the shield divides it into two parts), but still the same model is used for data to the left and to the right of the shield.

The algorithms presented in this chapter are essentially, but not completely, one-dimensional. They are one-dimensional in the sense that the model used is one-dimensional, but some information is passed from one row to the next. The algorithms (both the Baum-Welch algorithm and the Bayesian methods) have to be restarted at every row, and then initial values have to be chosen. In the case of Baum-Welch, e.g. the parameters of the $M$ different models (in our case $M$ different distances) have to be chosen. The initial values are used for inducing some dependency from row to row. A reasonable assumption is that the distance does not change drastically from row to row, and therefore when the algorithm is restarted at every row, the estimated values from the previous row are used as initial values. The estimated values are the different ranges and the estimate of the transition matrix of the Markov chain.

### 5.1.2 Resulting Segmentation

Test image #1, without drop-outs, will be used in this example. Two cases are tested, first using two states and then three states in the hidden Markov chain. A segmentation of test image #2 and #4 will also be performed, this time with only two and three states respectively. The initial value of the transition matrix $Q$ is set to

$$Q = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}. $$

A similar, non-informative initial value is chosen when three states are used. The initial value of $\theta$, when a two-state Markov chain is used, is set to the maximum distance in the first row for $\theta(1)$ and minimum value for $\theta(2)$. If three or more states are used, the interval between the maximum and minimum distance in the first row, is divided equidistantly. The values are then chosen as initial values. Remember that this procedure of initial value determination is needed only when the segmentation of the first row is to begin. Later on the initial values are chosen as the final, estimated values of the previous row. The result is the estimated state sequences, row-by-row.

The images are post-processed after the segmentation routine, to remove the obvious wrongly segmented pixels. The post-processing is possible since we know what we are looking for, namely relatively large objects in the image. We are not interested in very small objects (size of 2-3 pixels), so for the sake of clarity of the segmented image we remove occasional single-pixel segments. This is performed by going through the segmented image pixel by pixel and setting the value at every pixel to the median of the pixel in question and all immediately surrounding pixels. The result using two states is shown below in Figure 5.1 and Figure 5.2.
Figure 5.1: Resulting segmentation of test image #1, see Figure 2.2, using the one-dimensional Baum-Welch algorithm. Since a hidden Markov chain with two states is used, the image is segmented into two segments: 1 and 2.

In Figure 5.2 the shield can be easily recognized. The area labeled by 2 in the middle of the image represents the shield. Also the bushes in the upper right corner are found by the segmentation algorithm.

Figure 5.2: The segmented image in Figure 5.1 is post-processed. At every site the field value is set to the median of all surrounding pixels.
5.1 The Baum-Welch Algorithm

The result of the segmentation using a three-state Markov chain is shown in Figure 5.3 and 5.4. The result using three states is similar to the one using two states.

Figure 5.3: Resulting segmentation of test image #1 using the one-dimensional Baum-Welch algorithm. Here the segmentation is performed using three states.

Figure 5.4: The segmented image in Figure 5.3 is post-processed. At every site the field value is set to the median of all surrounding pixels.
The resulting segmentation of test image #2 and the result after post-processing is shown in Figure 5.5.

**Figure 5.5:** Resulting segmentation of test image #2 using the one-dimensional Baum-Welch algorithm. Here the segmentation is performed using two states.

The pine tree and the shield are distinguished from the background. Both are, however, labeled equally. Since only two states are used, one state labels the background terrain and the other everything else. Recall the discussion in Chapter 1 about classification versus segmentation!
As a last example in this section the resulting segmentation of the test image in Figure 2.8 is shown. In Figure 5.7 and 5.8 the resulting segmentation is shown, and the conclusion is that the algorithm does not distinguish the two shields. In the middle of the segmented image, the two shields are both labeled with state 2.

**Figure 5.7:** Resulting segmentation of test image #4 using the one-dimensional Baum-Welch algorithm. Here the segmentation is performed using three states.

**Figure 5.8:** Resulting segmentation of test image #4 after post-processing of the image in Figure 5.7.
Chapter 5. Row by Row Segmentation of Laser Range Images

5.2 Bayesian Methods

In this section we present an one-dimensional Bayesian method for image segmentation. A novelty in this section is the use of not only changes in distance, but also changes in variance of the distance for segmentation, i.e., changes in variance of the stochastic process $e_t$ in Equation 5.1. Variance is used indirectly by noting that if the assumption that the variance is known is replaced by assuming a less informative prior distribution on the variance, the \textit{a posteriori} probability of the state sequence is changed. This new \textit{a posteriori} probability is used when searching for the estimate of the sequence. Taking variance in consideration when performing segmentation is very useful since man-made objects differ from the background in the terrain by their smoothness. A natural extension of our approach is of course to use vertical scans and then combine the information into a final segmentation of the image. This is however not treated here.

Calculation of \textit{a posteriori} probabilities of the Markov states given past measurements, i.e.,

$$P(z^N | y^N)$$

is performed for three different assumptions on the measurement noise $e_t$, namely

- constant known variance
- constant but unknown variance
- different and unknown variance in the different segments.

As an introduction we will first deal with the simplest case, constant known variance, and the role of priors will be discussed. In the case of unknown measurement variance the variance is considered to be a stochastic variable with a prior. Two cases of priors on the variance of the measurement noise is discussed: the flat prior, i.e., noninformative, and the case when the variance is assumed to be inverse Wishart distributed. The latter will be only briefly investigated although it might be the most useful of the two in practice. The extension to the case of inverse Wishart prior is straightforward given the discussion assuming a flat prior. This treatment of the variance results in a modified \textit{a posteriori} distribution for the states given the past measurements. For a detailed and excellent overview of segmentation see [31], from where many ideas in this section originated.

The approach of considering the noise variance as a stochastic variable with a prior, and using it in combination with hidden Markov models for segmentation of laser radar range images with respect to range and range variance simultaneously, is a new one to our knowledge. A further advantage is the low computational complexity, especially when compared to finding maximum a posteriori (MAP) estimates for Markov random fields with simulated annealing [28] and graduated non-convexity type algorithms [43, 51].

The notation used in this chapter is consistent with the notation introduced in Chapter 1. The priors used are discussed in Section 5.2.1. Our main result is formulated in Section 5.2.2 and is experimentally verified on simulated and real data in Section 5.2.3.

5.2.1 The Priors

Assume that the variance of the measurement noise $R_t$ and the variance of the parameter prior $P_0$ is incorrectly chosen in such a way that the true value differs with a scaling $\lambda$. 
We denote the true values of $R_t$ and $P_0$ with $\tilde{R}_t$ and $\tilde{P}_0$.

\[
\begin{align*}
\tilde{P}_0 &= \lambda P_0 \\
\tilde{R}_t &= \lambda R_t,
\end{align*}
\]

(5.2)

where $\lambda$ is the previously mentioned, unknown scaling. The effect of this scaling on the estimated variance for the parameter $P_{t+1|t}$ is the following

\[
\tilde{P}_{t+1|t} = \lambda P_{t+1|t}.
\]

The value of the estimated parameter $\tilde{\theta}$ is, however, still the same. For pure filtering the actual level of the variances is not important. This is easily checked by substitution of (5.2) into (3.6). An important effect of the scaling, which will be used in Section 5.2.2, is that the \textit{a posteriori} density function of $z^N$ is dependent on $\lambda$.

The motivation for calculating the \textit{a posteriori} density function of $z^N$ given data is to find a value of $z^N$ which maximizes the density function, and then pick that as an estimate (which is the definition of MAP estimate of the sequence $z^N$). In Bayesian statistics the available information we have about the sequence or a parameter is used in the estimation. Prior information has the form of a probability density function of the random sequence or random variable. In our case we will have two priors. One is the prior on the state sequence, which is given by the Markov transition matrix and the initial probabilities of the Markov chain. The second is the prior on the parameter $\lambda$ which we will choose as flat or inverse Wishart. For the derivation of the \textit{a posteriori} density function of $z^N$ we first need the distribution of data given past data and states. The Kalman filter theory gives the distribution for the measurement prediction as

\[
y_t|y^{t-1} \sim N(\tilde{\theta}_{t|t-1}, S_t),
\]

where $N(\mu, \sigma^2)$ denotes the Gaussian distribution with mean value $\mu$ and variance $\sigma^2$.

The density function for the complete data sequence given the states is

\[
P(y^N|z^N) = (2\pi)^{-N/2} \left( \prod_{t=1}^{N} \det S_t(z_t) \right)^{-1/2} e^{-\frac{1}{2} \sum_{t=1}^{N} e_t^T(z_t) S_t^{-1}(z_t) e_t(z_t)}.
\]

(5.3)

We have used the fact that the data are independent if conditioned on the states. Bayes' law together with (5.3) gives the \textit{a posteriori} distribution for the state sequence

\[
P(z^N|y^N) = (2\pi)^{-N/2} \left( \prod_{t=1}^{N} \det S_t(z_t) \right)^{-1/2} e^{-\frac{1}{2} \sum_{t=1}^{N} e_t^T(z_t) S_t^{-1}(z_t) e_t(z_t)} \cdot \frac{P(z^N)}{P(y^N)}.
\]

This expression is valid if the measurement noise variance is known or, in other words, if $\lambda = 1$. In terms of priors we could interpret $\lambda$ as a random variable with a one-point distribution. We can, thus, conclude that using the following density function (prior) is equivalent with saying that $\lambda$ is known

\[
P(\lambda) = \delta(\lambda - 1),
\]

where $\delta(x) = 1$ if $x = 0$ and zero elsewhere. The extensions of the reasoning above is of course to assume other priors on the scaling $\lambda$ and this will be discussed in Section 5.2.2.
5.2.2 Results

Constant Known Variance

Of the three cases mentioned in this section, the case of constant known noise variance \( R_t \) is the simplest. We begin here and concentrate on issues other the segmentation with respect to variance, which we leave for a subsequent section. Our intention is to first introduce how our estimator works and then extend it, quite easily, to the case of varying variance.

If we knew the value of the sequence \( z^N \), where \( N \) is the length of the horizontal scan, the Kalman filter would give us the optimal estimate of the range in the different segments. The difficulty is that we do not know \( z^N \). So, at least in the optimal case, we will have to run \( M \) Kalman filters for every possible state sequence of \( z^N \) and calculate the optimal estimates of the different ranges \( \theta(z) \) and corresponding covariance matrices \( P(z) \). These are later used in the calculation of the likelihoods

\[
P(y^N | z^N) = (2\pi)^{-N/2} \prod_{t=1}^{N} \det S_t(z_t))^{-1/2} e^{-\frac{1}{2} \sum_{t=1}^{N} \epsilon_t^T(z_t)S_t^{-1}(z_t)\epsilon_t(z_t)}. \tag{5.4}
\]

Using Bayes' rule we easily obtain the a posteriori density which we maximize. The state sequence with the highest probability is then chosen as the estimate. The estimate of \( \theta \) and \( P \) following the chosen state sequence is our estimate of the range and its variance. To summarize the discussion on how to treat the case of known variance we here give an expression for the a posteriori probability of the sequence given data

\[
P(z^N | y^N) = \frac{P(y^N | z^N)P(z^N)}{P(y^N)},
\]

where \( P(y^N | z^N) \) is given by (5.4). \( P(z^N) \) is given by the state transition matrix and the actual sequence \( z^N \)

\[
P(z^N) = \prod_{i=2}^{N} P(z_i | z_{i-1}) \cdot P(z_1),
\]

where the \( P(z_1) \) is the initial probability of the Markov chain.

If an image containing 128 x 128 pixels with range data is given and assume that one wants to segment it into three segments one would have to run \( 3^{128} \) Kalman filters. Clearly this is impossible and a suboptimal search method has to be used. The search for the best sequence is performed with AFMM which was described in Section 3.3.3.

Constant Unknown Variance

In this subsection we go one step further and assume that the variance of \( e_z \) is constant, and the same over the different segments, but we do not know its value. We continue our discussion about priors in Section 5.2.1 here. In this subsection we look closer at the situation when the level of the variance is unknown, i.e., we assume that the variance is \( \lambda R_0 \), but we do not know the value of \( \lambda \). If we consider \( \lambda \) to be a stochastic variable, the choice of \( P(\lambda) = \delta(\lambda - 1) \) as prior distribution for \( \lambda \) is equivalent to say that \( \lambda \) is known. We, however, assume that we do not know the value of \( \lambda \), and a more adequate prior for \( \lambda \) should be chosen. The goal is to inflict as little prejudice as possible with the prior density and the prior should reflect our true knowledge about the random variable
5.2 Bayesian Methods

in question. When the random variable, here the level of the variance, is completely unknown, the best choice is a flat prior. We use the prior to modify the \textit{a posteriori} distribution of $z^N$. When we write down the left hand side of the expression (5.3) we have implicitly assumed a prior on $\lambda$. It really should stay

$$P(y^N | z^N) = \int_{-\infty}^{+\infty} P(y^N | z^N, \lambda) \cdot P(\lambda) d\lambda. \quad (5.5)$$

The correct expression for $P(y^N | z^N, \lambda)$ is the following

$$P(y^N | z^N, \lambda) = (2\pi)^{-N/2} (\prod_{t=1}^{N} \det S_t)^{-1/2} \lambda^{-N/2} e^{-\frac{V_N}{N}}, \quad (5.6)$$

where $V_N = \sum_{t=1}^{N} \varepsilon_t^T (z_t) S_t^{-1} (z_t) \varepsilon_t (z_t)$. If we assume the variance to be known or, essentially, assume the prior on $\lambda$ as $P(\lambda) = \delta(\lambda - 1)$ and insert that together with (5.6) into (5.5) we obtain the expression (5.3). If we instead use the more fair prior $P(\lambda) = 1$ on $\lambda$, i.e., we assume all values of $\lambda$ to be equally probable, we obtain the following \textit{a posteriori} likelihood function

$$P(z^N | y^N) = \frac{\Gamma(N-2)}{2\pi^{N/2} \prod_{t=1}^{N} \det S_t} \left( \frac{P(z^N)}{P(y^N)} \right). \quad (5.7)$$

Notice that the dependence of $z^N$ has been suppressed in the expression (5.7). The derivation of Equation (5.7) is given in Appendix A.1. $\Gamma$ is the gamma-function and is defined as follows

$$\Gamma(a + 1) = \int_0^\infty x^a e^{-x} dx \quad a > -1.$$

Unknown Variance Varying over the Different Segments

The next step is to assume unknown variance but allow different variances in the different segments. This assumption on the variance is the most interesting of the three mentioned. It is this \textit{a posteriori} likelihood function we try to maximize when the image in Example 5.2.3 is segmented with respect to variance. If we assume that the noise variance in the different segments is $\lambda(i) \cdot R_t$, where $R_t$ is known and $\lambda(i)$ is unknown but considered a random variable with a flat prior, i.e., $P(\lambda) = 1$, and $i = 1 \ldots M$, then the expression for the \textit{a posteriori} likelihood is the following

$$P(z^N | y^N) = \frac{1}{\pi^{N/2} M} \prod_{k=1}^{M} \frac{\Gamma(N(k)-2)}{(D(k) V(k) (N(k)-2)^{1/2} P(z^N)} \quad (5.8)$$

where

$$D(k) = \prod_{t \in T_k} \det S_t.$$ 

In words the expression above means taking the product of $\det S_t$ over the data points $t$ which belong to segment $k \in M$. The number of data points summed over the segments $k$ is of course $N$, i.e., $\sum_{k=1}^{M} N(k) = N$. We will not present the calculations leading to expression (5.8); they are similar to those for expression (5.7).
Inverse Wishart Prior

As already mentioned in Section 5.2 three cases of the degree of knowledge about the variance are treated in this section: known, unknown but constant over the segments and, finally, unknown and varying over the segments. In the case of unknown variance we have so far modified the a posteriori probability of the states using a non-informative prior on the parameter $\lambda$. In practice we often know something about the variance and that information should not be discarded. We will in this section present a useful choice of prior on $\lambda$ if beforehand information on $\lambda$ is available. We will assume a inverse Wishart density of $\lambda$. We will first discuss the case of constant variance over the segments, and then briefly examine the expression for the case of varying variance over the segments.

First, let us examine the inverse Wishart distribution.

The inverse Wishart distribution has two parameters and will in this thesis be denoted by $W^{-1}(m, \sigma)$. The probability density function is

$$P(\lambda) = \frac{\sigma^{m/2} e^{-\frac{\sigma}{2\lambda}}}{2^{m/2}\Gamma(m/2)\lambda^{(m+2)/2}},$$

and the mean and the variance of this distribution is given by

$$E(\lambda) = \frac{\sigma}{m-2},$$

$$Var(\lambda) = \frac{2\sigma^2}{(m-2)^2(m-4)}.$$

Fig 5.9 shows the inverse Wishart density function with mean 1 and variance 1.

![Inverse Wishart density function](image)

Figure 5.9: The probability density function of the inverse Wishart distribution.

The point with assuming an inverse Wishart distribution on the prior is its usefulness if the measurement noise variance is not exactly known, but rather we have a vague notion of its value. In that case an inverse Wishart distributed prior is assumed with the mean value as the expected noise variance and the variance of the prior chosen according to the certainty about the value of the noise variance.
If we now go through a similar development as in Section 5.2.2 in the case when the prior on $\lambda$ is inverse Wishart distributed the resulting a posteriori distribution is

$$P(z^N|y^N) = \frac{\Gamma\left(\frac{N+m}{2}\right)(2\sigma)^{m/2}}{\pi^{N/2}(\frac{m}{2})^{m/2}(\prod_{t=1}^T\det S_t)^{1/2}} \frac{P(z^N)}{(V_N + \sigma)^{\frac{N+m-N}{2}} P(y^N)}.$$  \hspace{1cm} (5.10)

A derivation of Equation (5.10) is found in Appendix A.2.

Different Inverse Wishart Distributed Priors for Each Class. Here we assume that different values of mean and variance of the prior on the measurement noise variance are used in the different segments. The likelihood of the complete data sequence, given the state sequence, expressed in the likelihoods of the individual segment is

$$P(y^N|z^N) = \prod_{i=1}^M P(y_{i\in T_i}), \hspace{1cm} (5.11)$$

where the factors on the right side of (5.11) are given by the following calculation

$$\int_0^\infty P(y_{i\in T_i}|\lambda(i)) \cdot P(\lambda(i))d\lambda =$$

$$\int_0^\infty (2\pi)^{-N(i)/2} \lambda(i)^{-N(i)/2} (\prod_{t\in i} \det S_t)^{-1/2} \cdot e^{-\frac{1}{2} \sum_{t\in i} \epsilon^T_i S_i^{-1} \epsilon_t} \cdot W^{-1}(m(i), \sigma(i)) \cdot 1.$$  \hspace{1cm} (5.12)

The likelihood (5.11) is possible to express as a product because it is assumed that data in the different segments are independent. This is a natural assumption since different segments in an image often represent different objects or background in the image. Note that this independence assumption would be valid in the case of more complex models, e.g. dynamic models, describing the different segments. The final expression for the a posteriori likelihood is given by combination of (5.11), (5.12) and Bayes' rule

$$P(z^N|y^N) = \prod_{i=1}^M \left( \frac{\Gamma\left(\frac{N(T_i)+m(T_i)}{2}\right)(2\sigma(T_i))^{m(T_i)/2}}{\pi^{N(T_i)/2}(\frac{m(T_i)}{2})^{m(T_i)/2}(\prod_{t\in T_i} \det S_t)^{1/2}} \cdot \frac{P(z^N)}{(V_N(T_i) + \sigma(T_i))^{\frac{N(T_i)+m(T_i)-N(T_i)}{2}} P(y^N)} \right).$$

5.2.3 Resulting Segmentation

To demonstrate the performance of our method we will apply it to two sets of data; one containing simulated data, and the other a data set collected by a laser range radar system.

The first signal used is white Gaussian noise with different variances. The generated signal is $y_t = \epsilon_t$, where $\epsilon_t$ is white Gaussian noise with variance 1, with the exception of samples numbered 35 through 70, where the generated signal is $y_t = 0.5\epsilon_t$. There is, thus, a drop in measurement noise variance in the middle of the signal. The test signal is shown in Figure 5.10.

Two states are used in the hidden Markov model and the transition probability matrix is

$$P = \begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix}.$$
Chapter 5. Row by Row Segmentation of Laser Range Images

Figure 5.10: Signal used for testing segmentation with respect to variance. There is a drop in noise variance in samples 35 through 75.

We chose to use AFMM with $M = 7$ (7 Kalman filters) in the search for the optimal path, and set the life length parameter to 5, i.e., no branch younger than five samples is allowed to be cut off.

The result of the simulation is shown in Figure 5.11. The algorithm finds the transitions very accurately even though it is difficult to precisely determine the jumps by simple inspection of the signal.

Figure 5.11: Resulting segmentation of the test signal in Figure 5.10. The dashed line shows the partition of the signal in segments, segment one are the parts of the signal where the measurement noise has the variance 1 and segment two the part with variance 0.25. The dotted line shows the true segmentation.
Except testing our algorithm on simulated data we have also performed tests on data obtained by the laser range radar system. We test the algorithm on test image #1, #2 and #3, all the versions with the drop-outs removed.

We begin with test image #1. Again, two states are used in the hidden Markov model to segment the image. In the 2-D case we use the one dimensional method to segment the image row by row. When the segmentation of one row is finished, the algorithm delivers the most probable surviving state sequence. Recall that it does not have to be the most probable sequence since the majority is disregarded according to the pruning algorithm. Along with the state sequence estimate the $M$ parameter vector estimates are delivered. In our case that corresponds to the $M$ different ranges, one for each state. We use the estimated values of range in the different $M$ segments as the initial values of the algorithm for the segmentation of the next row. That way the segments from row to row are not completely independent, e.g., if there are two states and two distances in row $i$, where state 1 corresponds to the distance 100 m and state 2 to the distance 400 m, the algorithm will be initialized with those values when performing the segmentation of row $i+1$. The consequence of this will be that values of distance near 100 m will still have the label 1 (associated with state 1) and distances around 400 m the label 2. The resulting segmentation is shown in Figure 5.12. The segment denoted by 2 is associated with the sign in the middle of the image and with the bushes in the upper right corner. In the lower part of the image, i.e., the first 15-20 rows, there is an irregular distribution of the two segments in the segmented image in Figure 5.12. This is because the algorithm always uses all available states. In this case a one-state “Markov chain” would model the image in the first 15-20 row sufficiently well. The algorithm, however, tries to use all the available states, here two, when segmenting, and the resulting effect is that the signal is divided between the two segments in an irregular way. A solution of this problem might be to estimate, from data, the optimal number of states beforehand.
Bayesian algorithm - resulting segmentation

Figure 5.13: Resulting segmentation of test image #2 using the one-dimensional Bayesian algorithm with a two-state Markov chain.

will increase the computational complexity, but the effects of spurious jumping will be avoided. This is a topic that is treated in Chapter 7.

We will now continue with applying the one-dimensional Bayesian algorithm to test image #2. The resulting segmentation with two states and the transition matrix

$$Q = \begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix},$$

is shown below. In Figure 5.14 we see that, except for the long thin segments caused by the fact that we are using more states than needed in some parts of the images, the pine tree can easily be found in the middle of the image. The shield is somewhat unclear and the reason for that is that grass is covering the bottom of the shield, and the left side of it is covered by the branches of the pine tree. Further the result of test image #3 is presented. Test image #3 contains a shield turned relative to the laser. It is the only object in the image. In Figure 5.16 the result after post-processing is shown. We see that the shield is correctly located.

5.3 Summary

Two different approaches to segmentation of laser range radar images have been covered in this chapter. The methods emerging from these two approaches are essentially one-dimensional methods used, with some minor modifications, on images. The segmentation is performed row-by-row and some information is passed from one row to the next. The two algorithms presented are

- the Baum-Welch algorithm
5.3 Summary

Segmentation after post processing

Figure 5.14: Result after post-processing the segmented image in Figure 5.13.

Bayesian algorithm - resulting segmentation

Figure 5.15: Resulting segmentation of test image #3 using the one-dimensional Bayesian algorithm with a two-state Markov chain.
• a Bayesian algorithm.

The Baum-Welch algorithm is identical to the forward-backward algorithm presented in Subsection 3.3.1. Examples of running the algorithm on real laser range data have been included in the presentation. The algorithm performs well and the shield in the middle of the image in Figure 2.2 is found. Figure 5.2 can be confusing, since the bushes in the upper right part of the image, and the shield in the middle, are both labeled by state one. If the parameters of the models corresponding to the two areas were compared, the result would be a significant difference between the two. This is used in a later classification stage. The objects in test image #2 (pine tree and shield) and #4 (two overlapping shields) are found by the algorithm. Details, however, are not distinguished. Both the pine tree and the shield in test image #2 are labeled with the same state label. In test image #4 the two shields are labeled identically, although the distance between them is 2 m. The distance difference is small if compared with the variations in the terrain. That implies that it will be very difficult to find the two shields, if not a very large number of states is used. If, for example, an additional label is introduced, the algorithm will rather assign the new label to a large terrain variation than use the new label for modeling the small variation that the two shields involve.

The main advantage of the Baum-Welch algorithm is its fastness and low computational complexity.

The second approach is to use a Bayesian method. The algorithm developed in this chapter is a new method for segmentation of laser range radar images. The segmentation method detects segments with different variances and ranges. The Bayesian method, as well as the Baum-Welch algorithm, is based on multiple models and a row-by-row segmentation of the image. The information taken along from row to row is the values of the previous row’s ranges in the individual segments. These ranges are used as initial values for the segmentation of the next row. Multiple models are used to model different
5.3 Summary

parts of a row, parts with different variance or range, and the switching between the models is governed by a hidden Markov chain. In the search for the optimal state sequence we use a sub-optimal search algorithm.

Our experiments show that we can detect changes of a factor four in variance of the measured range, which is useful since man-made objects often differ from its natural environment just by their smoothness. The method is also exemplified on a measured laser range radar image and shows good results. What is needed is an estimate of the number of states in the underlying Markov chain. If this number is overestimated we get the undesirable effect of spurious jumping between states since the method uses all available degrees of freedom. This effect is seen in the first rows of Figure 5.12. How to avoid this will be treated in a Chapter 7. From the results of the experiments we can conclude that the performance of the Bayesian method, in comparable experiments, is similar to the performance of the Baum-Welch algorithm.
Extension of 1-D Techniques to 2-D

In this chapter we will introduce methods that are direct extensions of the one-dimensional methods previously described in Chapter 5. The basic objective of the presented work is to retain the computational properties of the one-dimensional algorithms, while adjusting the data model towards a more realistic one, better capable of dealing with the two-dimensional correlation in data.

The majority of papers on "model based" image segmentation deal with Bayesian methods. We will begin, in Section 6.1, with a presentation of a two-dimensional version of the Baum-Welch algorithm. In Section 6.2 a new algorithm, which is a combination of AFMM, see Section 3.3.3, and a non-symmetrical half plane Markov chain, see Section 4.3, will be presented.

6.1 Extension of the Baum-Welch algorithm to Two Dimensions

The idea behind the algorithm in this section is very simple. If one takes a closer look at the Baum-Welch algorithm presented in Chapter 3, it is obvious that the equations that constitute the algorithm remain valid if all quantities involved, namely $z_t$, $y_t$ and $e_t$, are random vectors. This implies that $z_t$ is a vector Markov chain and $e_t$ is a sequence of independent, identically distributed random vectors. A vector Markov chain differs from an ordinary Markov chain in the way that in a vector Markov chain the states are vectors, with entries taking a finite number of values. Every combination of entry values is a new state of the vector Markov chain. If every entry in the state vector can assume $M$ values and there are $D$ entries the total number of states is $M^D$.

In the literature a similar idea already has been investigated. In [23] an image segmentation algorithm, based on recursive Bayes smoothing of images modeled by Markov random fields and corrupted by independent additive noise, is presented. The algorithm is an extension of an one-dimensional Bayes smoothing algorithm to two dimensions, and gives the $a$ posteriori distribution of the image value at each pixel, given the total image, recursively. It is important to note the distinction between the $a$ posteriori distribution of the image value at each pixel based on the full image, which is sought in [23], and the hidden field configuration that will maximize the joint $a$ posteriori distribution of the full image.
field given the full image. The image is processed in narrow strips, i.e., groups of rows, and the \textit{a posteriori} distributions of individual pixels in the middle row of the strips are readily obtained by integrating the \textit{a posteriori} distribution for the whole vector over all other entries. The work in [23] is inspired by the work in [6] and [74].

6.1.1 The Algorithm

Our extension of the Baum-Welch algorithm is motivated by the simplicity of the algorithm. Almost no changes in the code implementing the one-dimensional algorithm are required. The equations (3.9)-(3.12) in Subsection 3.3.1 do not have to be changed if the observations are vectors instead of scalars. The only difference is the one concerning the Markov chain which is replaced with a vector Markov chain. See Figure 6.1 for an example of a vector Markov chain with two entries in the state vector.

The number of states of the vector Markov chain is now different from the number of possible states that every entry in the vector can assume ($M$). Assume that the number of entries in the vector Markov chain is $D$, and assume further that $M$ is the number of possible states for every entry, then the total number of possible states that the vector Markov chain can assume is $M^D$.

We have chosen to process the image in strips containing three rows. We are, thus, using a vector Markov chain with three entries, i.e., $D = 3$.

The computational load is higher than in the one-dimensional case. In the one-dimensional case the memory required is the one it takes to store a $(M \cdot M) \times N$ matrix of real numbers, or to be more precise, the variable denoted by $\xi(i,j)$ in Equation (3.12),

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig6.1.png}
\caption{Vector Markov chain with two entries in the state vector. Every entry can assume two values.}
\end{figure}
is the "largest" variable in the sense that it takes the largest part of the memory required for the whole algorithm. The variable $\xi$ takes $M \cdot M$ values at each time instant, i.e., the number of combinations of the pair of indices $i$ and $j$. If the row length is $N$ then the required memory capacity, as we previously concluded, corresponds to the capacity required for storing a matrix with $M^2 \times N$ entries. For example, $\alpha$ takes only the memory required for storing $M \times N$ real numbers.

The memory required in the two-dimensional algorithm is approximately $M^4$ times the memory used by the one-dimensional algorithm. The factor $M^4$ comes from comparing the sizes of the largest matrices in the one-dimensional and two-dimensional algorithms. In the two-dimensional case $\xi(i, j)$ takes $M^3 \cdot M^3$ (recall $D = 3$) values at every time instant. The total memory required is then readily obtained by multiplying the row length by the number of possible combinations of $i$ and $j$, $M^6 \times N$. The factor $M^4$ is the quotient of the memory required by the two-dimensional and one-dimensional algorithm.

At first sight the factor $M^4$ may seem ominous. There are, however, two factors contributing to maintaining a low memory load. One is that very seldom more than 5 states are required, and the second is that the one-dimensional algorithm itself does not require considerable amounts of memory, e.g., for processing an image with dimension $512 \times 512$ pixels and using 5 states, the one-dimensional algorithm approximately requires to store a matrix of $(5 \cdot 5) \times 512$ real numbers.

When running the Baum-Welch algorithm strip-by-strip, we keep only the state estimate of the middle row of a strip. This is motivated by the fact that the dependence between the pixels usually decreases as the distance increases. We therefore assume that the influence of pixels outside the strip on the pixels in the center row is negligible. The algorithm takes one-row steps over the image and at each step an estimate of the state sequence in the center row is obtained. The algorithm is depicted in Figure 6.2.

**Figure 6.2**: Two-dimensional extension of the Baum-Welch algorithm is shown schematically. The different gray-levels correspond to the different states. When a strip is processed only the middle row is saved as the state sequence estimate.

We note that the transition matrix is somewhat different when extended to two-dimensions. The extension is straightforward. The transition matrix has now $M^D \times M^D$.
Chapter 6. Extension of 1-D Techniques to 2-D

entries, where the number of possible states that every entry in the vector Markov chain can take is denoted by \( M \) and the strip width is denoted by \( D \). A vector Markov chain is depicted in Figure 6.1.

The algorithm is applied to real laser range data and the results are presented in the next section.

6.1.2 Results

In this section results from experiments using the two-dimensional Baum-Welch algorithm on real laser range radar data are presented. We will begin with applying the algorithm to test image #1.

The transition matrix is now of dimension \( 8 \times 8 \), since we use two states per entry in the vector Markov chain. The result after segmentation is presented in Figure 6.3.

![Figure 6.3: Segmentation of test image #1 using the 2D Baum-Welch algorithm.](image)

We post process the image with the simple "median filter" to reduce the number of spurious spikes in the image. The result is presented in Figure 6.4.

If the result in Figure 6.4 is compared with the result obtained with the one-dimensional Baum-Welch algorithm in Figure 5.2, it can be noted that the segmentation is comparable in this case.

We have also applied the algorithm to test image #4. The resulting segmentation is shown in Figure 6.5. This result should be compared with the result in Figure 5.8. We can notice a small decrease in the amount of spurious single pixel segments when using the 2D algorithm.
6.1 Extension of the Baum-Welch algorithm to Two Dimensions

Figure 6.4: Result after post processing the image in Figure 6.3 with a "median filter".

Figure 6.5: Segmentation of test image #4 using the 2D Baum-Welch algorithm.
Chapter 6. Extension of 1-D Techniques to 2-D

After post processing the result in Figure 6.6 is obtained.

![Result after post-processing](image)

Figure 6.6: Result after post processing the image in Figure 6.5 with a “median filter”.

6.2 Bayesian Methods in Two Dimensions

In this section the results in Chapter 5 will be extended to two dimensions. The extension has two components. The first is the assumption on the underlying hidden field. The hidden Markov chain is extended to a non-symmetrical half-plane (NSHP) Markov chain. Using the NSHP Markov field, recursive calculations are still possible. AFMM is used for pruning the state sequences in the two-dimensional case too. The second component is changing the scanning pattern so the algorithm takes the properties of the AFMM pruning algorithm and the two-dimensional correlation in the image data in consideration.

6.2.1 NSHP Markov Field

The NSHP Markov field was defined in Section 4.3 and we will only repeat the basic facts here to refresh the reader's memory. The support of the NSHP Markov chain is depicted in Figure 6.7.

Another important issue to keep in mind is calculation of the joint probability of the hidden field. The calculation for the NSHP Markov field is similar to the one for the single-sided Markov chain. The joint probability is simply

$$P(z_{N_z}, y_N) = \prod_{i,j} P(z_{i,j} | y_{i,j}),$$

where the product is taken in the order suggested by the scanning pattern and Figure 6.7. The conditional probabilities and the support have to be adjusted at the borders of the image. The joint probability of the field, of course, does not depend on the order in which the multiplication is performed, as long as conditional probabilities are defined.
6.2 Bayesian Methods in Two Dimensions

6.2.2 The Influence of the Scanning Pattern

To understand the technique of changing the scanning pattern to improve image segmentation we must examine the pruning routine more closely. As mentioned before, the pruning routine used is AFMM and it has been presented previously in Section 3.3.3. We begin with an example.

Example 6.2.1 Assume we have the one-dimensional signal in Figure 6.8 given.

![Figure 6.8: The test signal is a simple white noise sequence with variance 1 and a jump in mean from 0 to 3.](image)
If the one-dimensional routine for segmentation, described in Section 5.2, is applied, the result depicted in Figure 6.9, is obtained.

![Resulting segmentation](image)

Figure 6.9: Resulting segmentation of the test signal.

In the segmentation routine 8 parallel filters were used ($S = 8$). This means that at most 8 different hidden state trajectories are compared and finally the most likely of those is chosen as the state sequence estimate. All state trajectories surviving at the end are depicted in Figure 6.10.

![Hidden state trajectories](image)

Figure 6.10: The 8 state trajectories that "survived" at the end of the signal. The state trajectories are identical up to the end of the signal where several alternatives are possible, i.e., several different alternatives are competing. The final segmentation of the beginning of the signal will be the same independent of which state trajectory has the highest probability.

The behavior of the segmentation routine shown in Example 6.2.1 is typical. Due to the restricted number of allowed surviving trajectories (to keep the computational complexity down) there will always be a variety of competing young alternatives. The
segmentation further back, relative the scanning order, is more or less determined. A further notable consideration is the scanning order used. Thus far, the scanning order depicted in Figure 6.11 is commonly used.

![Figure 6.11: The commonly used scanning pattern - “row-by-row”](image)

When extending the Markov chain to a NSHP Markov chain and taking in consideration the similarity of the “old” parts of the surviving state trajectories, it is obvious that all the degrees of freedom of the NSHP Markov chain are not used. When scanning, for example row $j$ and being at pixel $i$ in that row, the states denoted with $z_{i-1,j-1}, z_{i,j-1}$ and $z_{i+1,j-1}$ are fixed. They are fixed in the sense that it does not matter which sequence finally is chosen, the resulting segmentation of $z_{i-1,j-1}, z_{i,j-1}$ and $z_{i+1,j-1}$ will be the same. Hence, there are no alternative states competing. It is still, of course, an improvement when compared to the simple one-dimensional Markov chain, since now the segmentation of the previous row is taken in consideration.

We want to stress that the problem, described above, will not occur if unlimited computational capacity is available. Example 6.2.1 indicates that there are a variety of branches, or state trajectories, approximately through the $S/(M-1)$ latest pixels. For a moderate image size, say 80 pixels in width, and assuming 3 states are required for segmentation, we would require at least $160$ Kalman filters running in parallel, i.e., $S > 160$, if we do not want $z_{i-1,j-1}, z_{i,j-1}$ and $z_{i+1,j-1}$ to be fixed. That is, of course, unpracticle.

In the next section a new scanning pattern will be proposed that matches the used pruning routine and the NSHP Markov chain.

### 6.2.3 The Algorithm

The algorithm proposed in this section is a new one to our knowledge. So far no attempts to harmonize the scanning pattern with the model used, to avoid the the problem mentioned Subsection 6.2.2, have been reported.

Since the model used models the image by local neighborhoods, it is naturally desirable that many configurations of such neighborhoods compete when examining the
probability at each pixel. The scanning pattern depicted in Figure 6.12 is better suited to the model used for modeling the image.

![Figure 6.12: The proposed, modified scanning pattern better suited for the image model used.](image)

With the new scanning pattern the minimum number of Kalman filters has been drastically reduced. It is now sufficient to have a variety of non-fixed states through the last 6-7 pixels, which is a reasonable demand. We note that the algorithm processes the image in 3-row strips. When a strip is processed the algorithm takes a two-row step upwards, and in the next scan the rows marked with 1 and 2 in Figure 6.12 are assumed to be fixed. The reason for choosing to process the image by strips as opposed to not restarting the algorithm and simply continuing with row 4 in Figure 6.12 is twofold. First if we recall the introductory discussion about classification versus segmentation in Chapter 1, there is no need for numerous segments in the segmentation process. The classification of the segmented image will identify the different objects in the image. If the image is not processed in strips the number of segments required will increase. The consequence of this will be an increased computational burden. The second reason for strip processing is that if we try to process the image in wider strips, and as a special case - treat the whole image as one strip, we will again disregard the incompatibility between the local character of the image model and the pruning routine.

Similarly to the one-dimensional case, the information from one strip is transferred to the next by setting the initial values of the algorithm to the values estimated in the previous strip. In addition, when the algorithm takes a two-row step and restarts, the final segmentation of the middle row in the previous strip is used.

To conclude, the algorithm is composed of the one-dimensional algorithm presented in Section 5.2, but with the Markov chain extended to a NSHP Markov chain. The scanning pattern is changed to better accommodate the image model used and still keep the computational burden minimal.

### 6.2.4 Results

In this section the algorithm described in Section 6.2.3 is applied to real laser range data. The parameters that are the input to the algorithm are the transition matrix, initial $\theta_0$.
and the initial covariance $P_0$, see Subsection 3.2.1. The choice of $\theta_0$ and $P_0$ is not of large importance regarding the final result. However, the choice of transition matrix is important, and should reflect our prior knowledge regarding the probabilities different field configurations. When applying the algorithm in this subsection, one transition matrix is used in all the tests. The transition matrix chosen is

\[
Q = \begin{bmatrix}
0.9 & 0.1 \\
0.9 & 0.1 \\
0.9 & 0.1 \\
0.5 & 0.5 \\
0.9 & 0.1 \\
0.5 & 0.5 \\
0.8 & 0.2 \\
0.5 & 0.5 \\
0.5 & 0.5 \\
0.2 & 0.8 \\
0.1 & 0.9 \\
0.1 & 0.9 \\
0.5 & 0.5 \\
0.1 & 0.9 \\
0.1 & 0.9 \\
0.1 & 0.9 \\
\end{bmatrix}
\]

Recall that the transition matrix contains information about the quantity

\[ P(z_{i,j}|z_{i-1,j}, z_{i-1,j-1}, z_{i,j-1}, z_{i+1,j-1}), \]

i.e., the probability of the state at site $(i,j)$ given the neighbors, see Figure 6.7. Every row in the transition matrix corresponds to a combination of states of the field at the sites on which the probability is conditioned. Since we have assumed $M = 2$, i.e., the field can take one of two values at every site, the matrix $Q$ has $M^4 = 16$ rows. We have chosen the following mapping from row number to field combination. The field combination is simply the binary representation of the row number. Let us look at an example. Assume we want to interpret the number 0.2 in entry (10,1) in the matrix $Q$. Row 10 is mapped to the field configuration 2112. Hence,

\[ Q(10,1) = P(z_{i,j} = 1|z_{i-1,j} = 2, z_{i-1,j-1} = 1, z_{i,j-1} = 1, z_{i+1,j-1} = 2) = 0.2, \]

where column 1 of $Q$ corresponds to $(i,j)$ being in state 1, and similarly for column 2.

We now proceed with the presentation of the result after applying the segmentation algorithm to test image #1, #3 and #4. The segmentation of test image #1 is shown in Figure 6.13 and Figure 6.14. We see that the shield is easily spotted in the middle of the image. Contour plots of test image #1 and the resulting segmentation are included, see Figure 6.15. The contour plots show that the shield is found very accurately.

The next image is test image #3. The result can be found in Figure 6.16 and Figure 6.17. Also in this case we can see a clean segmentation, without spurious jumps.

The last test image to be presented is test image #4. The algorithm cannot distinguish between the two shields. Similar result is obtained if three states are used. The reason is that the distance difference between the two
Figure 6.13: Resulting segmentation of test image #1 using the 2D Bayesian algorithm.

Figure 6.14: Result after post-processing using a median filter to reduce the number of small, spurious segments.
6.2 Bayesian Methods in Two Dimensions

Figure 6.15: The position of the shield in test image #1 and the segmentation result are compared in a contour plot.

Figure 6.16: Test image #3 segmented with the 2D Bayesian algorithm.
Chapter 6. Extension of 1-D Techniques to 2-D

Figure 6.17: Result after post-processing using a median filter.

Figure 6.18: Segmentation of test image #4.
6.3 Summary

Two extensions of the basically one-dimensional segmentation algorithms to two dimensions have been discussed in this chapter. First a two-dimensional version of the Baum-Welch algorithm presented in Section 3.3.1 is introduced and secondly the new Bayesian method based on AFMM and the NSHP Markov chain is presented.

It is noted that the Baum-Welch algorithm is valid for vector quantities too. That fact is used to introduce two-dimensional correlation between the field elements. The image is processed in strips containing three rows. Assume one column in the strip is picked out. The column contains three entries, namely the field values in three consecutive rows. It is assumed that the three values are entries in a vector and that the vector is changing values, when going from column to column, according to a vector Markov chain. This new assumption does not demand any changes of the Baum-Welch algorithm, row three rows at the time are segmented. The correlation between vertical pixels is determined by the transition matrix. The algorithm takes one-row steps and in every step the middle row field estimate is saved. This is motivated by the assumed low correlation between pixels far away. In this case “far away” means more than one pixel-row away.

The results obtained by using the 2D Baum-Welch algorithm are slightly better than the corresponding 1D algorithm. The main advantage is the low computational

![Result after post-processing](image)

Figure 6.19: Result after post-processing the segmented image using a median filter.

shields (2 m) is negligible if compared with the distance differences existing naturally in the terrain. If another state is introduced, the two shields are still grouped into one segment, and the extra segment is used to model some other larger variation.

One important fact that is worth to stress, is that no tuning of the algorithms has been performed on a specific image basis. For example, the same transition matrix has been chosen for all test images. It is of course possible to tune the transition matrix so better results are obtained. That would, however, be an unfair comparison of the algorithms.
complexity of the Baum-Welch algorithm.

The extension of the Bayesian method presented in 3.3.2 is not so straightforward as the extension of the Baum-Welch algorithm. The extension consists two elements. The first element is to extend the support of a Markov chain. We have chosen to use the NSHP Markov chain which has a more natural support if the scanning of the image is taken into account. Using the NSHP Markov chain not much has changed from the view of the pruning routine. Still AFMM is used, but the probabilities of the different branches are calculated in a different way, now taking the new model into account. Due to the characteristic of the pruning algorithm the new model is not fully exploited. The competing branches' support located at the previous row are all identical. To overcome this problem a new scanning pattern is suggested. The image is divided into groups of rows and these rows are scanned diagonally. In that way a number of pixel configurations of pixels neighboring the visited pixel are competing.

Experiments on real laser range radar images shows an clear improvement compared to the results of the one-dimensional algorithms. There is, of course, a trade off. The algorithms in this chapter are more demanding regarding the computational complexity. It is worth to note that the results presented are not the best that can be obtained. If the algorithm is tuned for a specific image, an improvement is possible.

We have not succeeded to distinguish the two shields in test image #4, nor the pine tree and shield in test image #2. In test image #4 there is a distance change (2 m), and in test image #2 there is a variance change. The reason why the two objects are not distinguished by the algorithm is that much larger natural terrain variations existing in the image. The change in the distance respectively variance is to small to "deserve" a new segment.
We will now turn to the last question posed in Chapter 3, namely the problem exemplified by how to estimate the number of urns there are in Example 3.0.1.

As we have seen in the examples so far, a phenomenon that often occurs in the segmentation process is the spurious jumping in the state estimate of the HMM when more states than needed are used. The reason for that is that the algorithms use all available degrees of freedom, i.e., the algorithms actually segment the signal/image into $M$ segments if the underlying Markov chain has $M$ states. There is obviously a need for estimation of the number of states before applying the segmentation routine.

The problem of state order estimation for HMMs is not necessarily related to segmentation or HMMs, but is a generally interesting problem.

Example 7.0.1 Assume that a white noise sequence, depicted in Figure 7.1, is given.

![White noise sequence](image)

**Figure 7.1:** White noise sequence with variance 1.

The natural choice of number of states to model the white noise sequence is 1, since there are no jumps in the signal. If we, however, choose a two-state Markov chain and apply the
Baum-Welch algorithm to segment the signal into two segments the result is the one found in Figure 7.2.

Segmented white noise sequence

Figure 7.2. Resulting segmentation of the white noise signal in Figure 7.1, using two states.

The problem of selecting the model order of a stochastic process has been widely studied in the literature and we will apply a number of results on the estimation of the state order of the HMM. There are, however, surprisingly few papers published addressing the problem of state order estimation of MRFs. We refer the interested reader to [63] where Minimum Description Length is adopted as a complexity measure for MRF texture models of mitochondria and background regions of the Electron-Microscope Autoradiography.

The state order of a HMM is defined as the number of states in the underlying Markov chain. Assume a Markov chain with \( M \) states is given

\[
P(z_t = i | z_{t-1} = j) = q_{ji} \quad i,j = 1, \ldots, M.
\]

We further assume that \( M \) data models are given

\[
y_t(i) = \varphi^T_t(z_t) \theta(z_t) + e_t(z_t), \quad z_t \in \{1, \ldots, M\}
\]

where \( e_t \) is white Gaussian noise. More general models in state space form can be treated but we choose the simple regression model for its notational simplicity. The problem addressed here is the estimation of the model state order \( M \) from the observed process \( y_t \). This is not to be confused with the problem of order estimation of a Markov chain or Markov source. The order of a Markov chain or Markov source is \( k \) if the following relation holds for the stochastic process \( z_t \)

\[
P(z_t | z_{t-1}^{-1}) = P(z_t | z_{t-k}^{t-1}).
\]

This problem is not addressed here and we refer the reader to [52] and the references therein.
This chapter contains a brief investigation of three state order estimation techniques: information-theoretic techniques, predictive least squares (PLS) and the maximum a posteriori (MAP) estimator of the model state order.

### 7.1 Information Theoretic Techniques

The methods which are possible to associate with the term information theoretic techniques address the problem of estimating the number of states of a finite-alphabet, finite-state source. We say a source has a finite-alphabet if the observable process $y_t$, given $z_t$, can take only a finite number of values according to a discrete probability distribution.

Assume a sequence $y^N$ is given and we know it has been generated by a finite-state source, but we do not know the number of states $M$. In the sequel $M$ will denote the "true" value of the model state order, $m$ will denote the model state order and $\hat{m}$ the estimate of the model state order. Usually a criterion is calculated for different values of $m$ and then a $\hat{m}$ is chosen as an estimate. The desired result is, of course, that $\hat{m} = M$.

Consider a source whose output $Y_t$ has an unknown distribution $P$. Assume that $m_1, m_2, \ldots$ is an increasing sequence of model classes, each consisting of measures on the source alphabet sequence space. The sequence of model classes is increasing in the sense that the smaller model class is included in the larger. For example, a two-state HMM is a subclass of a 3-state HMM. We assume that $P$ is a member of one of the classes $\{m_i\}$, but we do not know which one. In the state order estimation problem one wishes to estimate the number of states $\hat{m}$ of a source relative to the different possible models $\{m_i\}$. Here we assume that $m^i$ is the class of all $i$-state HMMs.

Assume further that for every $m$ we have a code $\Phi_m$. A code can be described as a mapping from the source symbols, or groups of symbols, to a series of bits. The mapping takes into account the distribution of the source symbols. The shortest coding is obtained if the assumed distribution is the true one. All the information theoretic techniques boil down to finding an appropriate code $\Phi_m$ for coding the sequence $y^t$, calculating the code length for the different codes, and then picking the $m$ for which the code $\Phi_m$ gives the shortest code length when coding $y^t$. We refer the reader to [42] and [76] for a more thorough treatment of code-based estimators.

In [42] an state order estimator with the following form is developed

$$\hat{m} = \min \left\{ j : -\frac{1}{N} \log \max P(x) - \frac{1}{N} U_{LZ}(x) < \lambda \right\},$$

where $U_{LZ}(x)$ is the length (in bits) of the Lempel-Ziv codeword [75] for $x$ and $P(x)$. The intuitive interpretation of the expression above is that we seek the smallest model state order $\hat{m}$ for encoding $x$, such that the codeword length $-\log P(x)$ will be sufficiently close to the codeword length associated with the Lempel-Ziv algorithm, which in turn, serves as an estimate of the entropy.

Another often-used coding principle is the Minimum Description Length (MDL) principle introduced and popularized by Rissanen in [54, 55, 57, 59, 60]. The idea behind the MDL differs from usual statistical reasoning. The purpose of model building is not to approximate any underlying "true" distribution. Instead the objective is to search for a model (or model class) which best captures the properties of the data. When we say models we mean a parametric class of probabilistic models. We will later use the obtained model for encoding the data in a shorter and less redundant way. This means
that we try to find the parameter values in a selected model class capable of reproducing
the observed sequence \( y^N \) so as to minimize the number of bits it takes to describe the
observed sequence. Note that not only the data are encoded, using the model, but also
the model itself, i.e., the real-valued parameters in the model. When we say encoded, we
mean exact description or representation of observed data. For this to make sense all the
observed data points as well as the real-valued parameters must be suitably truncated
to keep the description length finite. So, the process is to find a model, more or less
complex, and then decide which coding algorithm to use.

How does this apply to the HMM case? Let us first introduce some notation. If we
code a data sequence \( y^N \) the resulting number of bits per symbol is denoted by \( L(y^N) \).
The expression for the bit length per symbol of a HMM model consists of three terms.
The first term in Equation (7.1) below, is the number of bits used for encoding the output
data given the state sequence and parameters in the output process distribution. The
second term is the number of bits used for encoding the state sequence (assuming the
transition matrix etc. given), and the last term is the number of bits for the parameters
of the output process. The overall number of bits will be the sum of the number of bits
for describing the data and the model

\[
L(y^N, z^N, \theta) = L(y^N | z^N, \theta) + L(z^N) + L(\theta),
\]

(7.1)

where the first term on the right hand side is

\[
L(y^N | z^N, \theta) = -\frac{1}{N} \log_2 P(y^N | z^N, \theta),
\]

the second is

\[
L(z^N) = -\frac{1}{N} \sum_{t=1}^{N} \log_2 (P(z_t | z_{t-1} - 1))
\]

and finally, the third is the length of the code per parameter in the parameter vector \( \theta \)

\[
L(\theta) = \log_2 (d + 1)(M(M - 1) + 1) + \sum_{i=1}^{d} \log \frac{[\theta_i]}{\delta_i} + \sum_{j=1}^{M(M-1)} \log \frac{\theta_j}{\varepsilon_j},
\]

where \( d = \dim(\theta) \) is the number of parameters in the parameter vector, \( M \) is the number
of states and hence the parameters to store are the transition probabilities (except one in
each row that can be deduced from the rest) and the initial probability vector. The last
two terms are the number of bits needed to store the truncated real values of the different
parameters. The precision used for storing the parameters and transition probabilities
is \( \delta_i \) and \( \varepsilon_j \) respectively. The expression above is very similar to the expression reported
in, for example [54] and the final approximate expression to minimize is

\[
V \approx \log_2 \left( \frac{1}{N} \sum_{i=1}^{N} e_i^2 \right) + (d + M(M - 1) + 1) \frac{\log_2 N}{N}.
\]

The full expression contains a term which can be interpreted as the sensitivity of \( L(y^N | z^N, \theta) \)
to the precision of the stored parameters. If the code length for the first term is particularly
sensitive to some parameters, these should be stored with greater precision (at the
cost of higher code length for storing these parameters). This effect however diminishes
when \( N \to \infty \).
7.2 Predictive Least Squares

In [56] a new principle of least squares estimation is described. It is based on the Predictive Minimum Description Length principle (PMDL) introduced in [58] and it is called the Predictive Least Squares principle (PLS). Whereas the usual least squares minimizes the mean prediction error, the PLS minimizes the prediction errors of the observations. The minimized criterion can be interpreted as representing the least total accumulated "honest" prediction errors, where "honest" denotes that only past data are used to identify the parameters in the predictor. In other words, when the data are predicted each prediction is based on the so-far processed data, only.

We will first demonstrate the PLS for a basic regression problem. Assume we have two sets of observations $y^N$ and $x^N(i)$, where $i = 1, \ldots, M$. The single variable $y$ is called the dependent variable or the response. The $M$ variables $x(i)$ are called the independent variables. The usual procedure when applying ordinary least squares is to introduce a model class and to pick a predictor for $\hat{y}_t$

$$\hat{y}_t = g_\theta(x_t(1), \ldots, x_t(M)), \quad (7.2)$$

where $g_\theta$ is an appropriate function with parameters $\theta$, approximating the true relationship. The function's parameters are adjusted to minimize a chosen criterion. The ideal predictor should minimize

$$E_\theta(y_t - \hat{y}_t)^2. \quad (7.3)$$

Here it is assumed that the data are generated from a "true" distribution

$$y_t = f(x_t(1), \ldots, x_t(M)) + e_t,$$

where $e_t$ is an uncorrelated sequence with zero mean. The function $g_\theta$ belongs to the same class as $f$. Hence, if we replace the expectation in (7.3) with the sample mean we obtain the following estimate

$$\hat{\theta} = \arg \min_\theta \frac{1}{N} \sum_{t=1}^{N} (y_t - \hat{y}_t). \quad (7.4)$$

Another approach is to change the predictor (7.2) to

$$\hat{y}_t = g_{\hat{\theta}_{t-1}}(x_t(1), \ldots, x_t(M)). \quad (7.5)$$

At every time instant the parameter vector $\theta$ minimizing the criterion (7.4) is calculated using past data only. The parameter vector will vary in time, since the number of data for which it is calculated grows. This makes the predictions "honest" in the way that they will be calculated from available, past, data. This is intuitive when data has a specific ordering, e.g., time, since we use all available data at one time instant to calculate the prediction for the next time instant. If then all the prediction errors are accumulated we have the following criterion

$$V_{PLS}(y|x, M) = \sum_{t=1}^{N} (y_t - \hat{y}_t)^2.$$
Assume that $M$ is not known and an auxiliary variable $m$ is introduced. The criterion $\sqrt{\text{PLS}(y|x;m)}$ may be minimized over $m$ and the $m$ at which the criterion assume the lowest value can serve as an estimate for $M$.

In system identification the objective is to build models for signals using a data set. If the model's performance is measured using the same data set as it was built from, a good performance is not surprising. The real challenge is to test the model on a new data set from the same system. This is called cross-validation. The PLS procedure, described above, can be viewed as a cross-validation technique where the cross validation takes place at every time instant. The model is built from old data, the prediction is compared with the new, in the parameter estimation not used, data at the following time instant and the error is stored.

In [56] linear regression models with Gaussian noise are considered. Based on first and second moments of some random variables of interest and relying on Chebyshev's inequality it is shown that $\hat{m}_N \to M$ in probability, where $M$ is the dimension of the regression vector, with $m \in 1, 2, \ldots, M^*$, $M^* < \infty$, and $\hat{m}_N$ is the PLS estimate at time $N$. Another study was presented in [68] where the same result is obtained for autoregressive models without requiring the Gaussian assumption. Besides giving the proof, a computationally efficient way of evaluating the PLS estimates using predictive lattice filters is presented. In [33], relying on the results for multiple regression models in [69], it is shown that the PLS estimates are also strongly consistent, i.e., $\hat{m}_N \to M$ almost surely for AR models. Further in [34] the condition of $M$ being in a finite set is dropped. It is shown that strong consistency is maintained even if $M$ is allowed to increase with rate $O((\log N)^{\gamma})$, for any $\gamma < 1/2$. Recently, in [70], the connection between PLS and traditional information-based criteria, such as Akaike's A-Information Criterion (AIC) and Akaike's B-information Criterion (BIC), used in model selection, is investigated. The traditional criteria have the form

$$\log V_N + \frac{c_N}{N},$$

where the first part is the residual variance after fitting the model and $c_N$ is a nonnegative variable measuring the complexity of the model. In AIC the penalty term is $c_N = 2m/N$, where $m$ is the dimension of the regression vector, and in BIC the penalty term is $c_N = (m \log N)/N$. In [70] it is shown that PLS can be decomposed into a sum of a term measuring the goodness of the fit and a penalty term.

PLS can be intuitively explained in the regression case. The mechanism preventing the criterion to decrease when the state order is increased is the following. At the beginning of the data set, i.e., when $t$ is small, the parameter vector is estimated from few data. If the dimension of the vector is high, worse estimates are obtained and hence the predictions are worse. The same reasoning indicates that the criterion should work when applied to HMMs.

### 7.2.1 Using PLS for Order Estimation of HMMs

In this section it is shown how PLS is applied to state order estimation of HMM. First we will calculate the one step predictor for HMM and go through the PLS procedure for the case of HMM. We will also point out difficulties in proving consistency of the criteria, although in simulations it shows good results. The procedure is to use the EM algorithm, described in Chapter 3, to estimate the state sequence and the probabilities.