2D Object Detection and Recognition:
Models, Algorithms and Networks

Yali Amit
University of Chicago

January 6, 2005
## Contents

1 Introduction ............................................. 14  
1.1 Low level image analysis and bottom-up segmentation .......... 14  
1.2 Object detection with deformable template models ............ 15  
1.3 Detection of rigid 3d objects ................................ 16  
1.4 Object recognition ...................................... 18  
1.5 Scene analysis: merging detection and recognition ............ 20  
1.6 Neural network architectures ................................ 21  

2 Detection and recognition: overview of models ................. 22  
2.1 A Bayesian approach to detection ................................ 22  
2.2 Overview of object detection models .............................. 26  
2.3 Object recognition ........................................ 30  
2.4 Scene analysis: combining detection and recognition ........... 32  
2.5 Network implementations .................................... 33  

3 1-d Models: deformable contours ................................ 34  
3.1 Inside-Outside model ...................................... 34  
3.2 An edge based data model .................................... 40  
3.3 Computation ................................................ 41  
3.4 Joint estimation of the curve and the parameters ............... 44  
3.5 Bibliographical notes and discussion ............................ 49  

4 1-d Models: deformable curves ................................ 53  
4.1 Statistical model .......................................... 53  
4.2 Local features ............................................. 53  
4.3 The likelihood ............................................. 55  
4.4 The prior and the posterior .................................. 56
4.2 Computation: dynamic programming .................................................. 57
4.3 Global optimization on a tree structured prior ........................................ 59
   The data model .................................................................................. 61
   Entropy ............................................................................................. 62
   Sequential updating of the partial posterior ........................................... 63
4.4 Bibliographical notes and discussion ...................................................... 67

5 2-d Models: deformable images ................................................................. 72
5.1 Statistical model .................................................................................. 72
   Gaussian data model ........................................................................... 73
   The prior and posterior ....................................................................... 74
5.2 Connection to the deformable contour model ........................................ 77
5.3 Computation ....................................................................................... 77
5.4 Bernoulli data model .......................................................................... 81
   Edges ................................................................................................. 81
   The data model .................................................................................. 82
   Training .............................................................................................. 84
5.5 Linearization ...................................................................................... 84
5.6 Applications to brain matching ............................................................. 87
5.7 Bibliographical notes and discussion ...................................................... 87

6 Sparse models: formulation, training and statistical properties .................. 92
6.1 From deformable models to sparse models ............................................ 93
6.2 Statistical model .................................................................................. 94
   The prior - admissible instantiations ...................................................... 95
   Likelihood and posterior .................................................................... 95
   Multiple objects ................................................................................. 96
   Computation ...................................................................................... 97
6.3 Local features: comparison arrays ....................................................... 97
   Training .............................................................................................. 98
6.4 Local features: edge arrangements ....................................................... 99
   Training .............................................................................................. 101
6.5 Local feature statistics .................................................................... 105
   Background densities ....................................................................... 106
   Statistics on object ........................................................................... 107
   False positive rates .......................................................................... 109

7 Detection of sparse models: dynamic programming ................................... 113
7.1 The prior model .................................................................................. 113
   Decomposability ................................................................................ 114
7.2 Computation: dynamic programming .................................................. 115
7.3 Detecting pose .................................................................................... 119
7.4 Bibliographical notes and discussion .................................................... 119
8 Detection of sparse models: Counting
  8.1 Detecting candidate centers ........................................ 123
  8.2 Computing pose and instantiation parameters .................. 125
  8.3 Density of candidate centers and false positives ............. 127
  8.4 Further analysis of a detection ................................... 128
  8.5 Examples ............................................................. 130
     Faces ............................................................... 130
     Randomized \LaTeX symbols ........................................ 131
     3d object - range of viewing angles .............................. 136
     MRI images .......................................................... 137
  8.6 Bibliographical notes and discussion ............................ 139

9 Object recognition .................................................. 147
  9.1 Classification trees ............................................... 149
     Training ............................................................ 151
     Testing and error .................................................. 152
     Unsupervised trees ................................................ 152
     Multiple trees ..................................................... 153
  9.2 Object recognition with trees .................................. 155
     Local features ..................................................... 156
     Absolute arrangements ............................................. 157
  9.3 Relational arrangements ........................................ 159
     Relations between local features ................................ 159
     Growing trees with relational arrangements .................... 160
  9.4 Experiments ....................................................... 162
     NIST ................................................................. 162
     \LaTeX ............................................................... 165
  9.5 Why multiple trees work? ....................................... 167
  9.6 Bibliographical notes and discussion ............................ 170

10 Scene analysis: merging detection and recognition ............ 172
  10.1 Classification of chess pieces in gray level images ........ 173
  10.2 Detecting and classifying characters .......................... 175
  10.3 Object clustering ................................................ 181
     Sequential object clustering ..................................... 182
     Tree based class clustering ..................................... 184
  10.4 Bibliographical notes and discussion ............................ 185

11 Neural network implementations .................................. 186
  11.1 Basic network architecture ..................................... 187
     Neurons and synapses .............................................. 187
     Layer connections ................................................ 187
     Visual input: edges and edge arrangements ...................... 188
  11.2 Hebbian learning ................................................. 190
  11.3 Learning an object model ....................................... 190
  11.4 Learning classifiers .............................................. 192
     Hebbian learning .................................................. 192
CONTENTS

Field dependent Hebbian learning ........................................ 194
Classification with multiple randomized perceptrons .................... 195
11.5 Detection ................................................................ 197
11.6 Gating and off center recognition .................................... 198
11.7 Biological analogies ...................................................... 200
11.8 Bibliographical notes and discussion ................................. 203

12 Software ........................................................................ 206
12.1 Setting things up ............................................................ 206
12.2 Important data structures ................................................ 208
12.3 Local features ............................................................... 210
12.4 Deformable models ....................................................... 212
12.5 Sparse models ............................................................. 218
12.6 Sparse model - counting detector -Training ......................... 219
12.7 Example - \LaTeX .......................................................... 221
12.8 Other objects with synthesized training sets ......................... 222
12.9 Shape recognition ........................................................ 223
12.10 Combining detection and recognition ............................... 226
# List of Tables

6.1 95\% confidence intervals for the coefficients of the two predictors - log-density of edges (LDE) and complexity ($n_r$), and the $R^2$ coefficient for the regressions. .................................................. 108

6.2 First column: Complexity of edge arrangements in model. Second column: Average on-object probability of the arrangements. Third column: Threshold at 5\% false negative. Fourth column: False positive density (number of false positives per pixel.) Fifth, sixth columns: 95\% confidence intervals for the regression coefficient of log-density of false positives against log-density of local features in an image. Seventh column: $R^2$ coefficient of the regression. ............................... 111

8.1 $\tau$ and $\tau_a$, candidate location and false positive rates for false negative rates under 5\%. .......... 128

9.1 NIST: Different protocols. 100,000 training data and 50,000 test. $Q$ - number of random queries sampled at a node. $s$ - size of 'spread' parameter for trees with absolute arrangements. $K_w$ - number of wedges in relative arrangements (width of wedge is $2\pi/K_w$.) $m = 3$ - number of data points in second largest class at a node to stop splitting. First row: aggregate classification rate with 100 trees. Second row: average classification rate of the individual trees. ................................. 162

9.2 NIST: Classification rates of the aggregate classifier and the individual trees as a function of the training sample size - absolute arrangements. $m = 3, s = 15, Q = 20$. ................................. 163

9.3 NIST: Absolute arrangements. Classification rates of the aggregate classifier and the individual trees with 5000 training data, as a function of the stopping criterion. $m = 3, s = 15, Q = 20$. ................................. 164

9.4 NIST: Absolute arrangements. Classification rates of the aggregate classifier with and without boosting as a function of the stopping criterion. 5000 training data are used. Boosting is not applicable for pure trees since the training error rate is 0. $Q = 20, s = 15$. ................................. 164

9.5 NIST: Absolute arrangements. Classification rates of the aggregate classifier with boosting for various protocols with boosting and 100000 training data. ................................. 164

9.6 LaTeX: Comparison of results for different values of the spread parameter $s$. Absolute arrangements are used with the 264 generic features and $Q = 20$. ................................. 165

9.7 LaTeX: Comparison of results for different values of $Q$. Top two rows randomized trees. All data points at a node were used to select the optimal query. Absolute arrangements, $m = 3, s = 15$. Bottom two rows with boosting. ................................. 166

9.8 LaTeX: Different protocols with relational arrangements. ................................. 166

9.9 LaTeX: Performance results in the presence of clutter for relational arrangements and absolute arrangements. The values in parantheses denote the percentage of test points which had the correct class among the top 5. ................................. 168
Preface

This book is about detecting and recognizing 2d-objects in gray level images. How are models constructed? How are they trained? What are the computational approaches to efficient implementation on a computer? And finally, how can some of these computations be implemented in the framework of parallel and biologically plausible neural network architectures?

Detection refers to anything from identifying a location to identifying and registering components of a particular object class at various levels of detail. For example find the faces in an image, find the eyes and mouth of the faces. One could require a precise outline of the object in the image, or the detection of a certain number of well defined landmarks on the object, or a deformation from a prototype of the object into the image. The deformation could be a simple 2d affine map or a more detailed non-linear map. The object itself may have different degrees of variability. It may be a rigid 2d object, such as a fixed computer font, or a 2d view of a 3d object, or it may be a highly deformable object such as the left ventricle of the heart. All these are considered object detection problems, where detection implies identifying some aspects of the particular way the object is present in the image, namely some partial description of the object instantiation.

Recognition refers to the classification among objects or subclasses of a general class of objects, present in a particular region of the image which has been isolated. For example after detecting a face, who is the person, or classify images of hand written digits, or recognize a symbol from a collection of hundreds of symbols. Both domains have a significant training and statistical estimation component.

Finding a predetermined object in a scene, or recognizing the object present in a particular region are only sub-problems of the more general and ambitious goal of computer vision. In broad terms one would want to develop an artificial system which can receive an image and identify all the objects or a large part of the objects present in a complex scene, from a library of thousands of classes. This implies not only detection and recognition algorithms, but methods for sequentially learning new objects, and incorporating them in the current recognition and detection schemes. But perhaps hardest of all is the question of how to start processing a complex scene with no prior information on its contents. What to look for first, and in which particular regions should a recognition algorithm be implemented. This general problem is unsolved, although our visual system seems to solve it effortlessly and very efficiently.

Deformable template models offer some reasonable solutions to formulating a representation for a restricted family of objects, estimating the relevant parameters, and subsequently detecting these objects in the image, at various levels of detail of the instantiation. Each model is defined in terms of a subset of points on a reference grid, the template, a set of admissible instantiations of these points, also referred to as deformations of the template, and a statistical model for the data given a particular instantiation of the object is present in the image. A Bayesian framework is used in that probabilities are assigned to the different instantiations. Bayes’ rule then yields a posterior distribution on instantiations. Detections are computed by finding maxima or high values of the posterior. In Chapter 2 some general and unifying elements of the Bayesian models used in all the detection algorithms are introduced, together with an overview of the models applied to a simple synthetic example. The details of the detection algorithms are provided in Chapters 3 through 8.
Chapter 9 is devoted to recognition of isolated objects or shapes, assuming some mechanism exists for
isolating the individual objects from the more complex image. The classification schemes can be viewed as
a recursive partitioning of the hierarchy of templates using classification trees. Chapter 10 is an exploration
into a possible approach to complex scene analysis by merging detection and recognition, both in terms of
training and in terms of implementation. Detectors are no longer geared to one particular class, but to
object clusters containing elements from several classes. Detection can be viewed as a way to quickly choose
a number of candidate regions for subsequent processing with a recognition algorithm. An overview of the
models of Chapters 9 and 11 are also given in Chapter 2.

Chapter 11 describes schematic neural network architectures that train and implement detection and
recognition algorithms based on the sparse models developed in Chapters 6 through 9. The goal is to
show that models based on binary local features, with built in invariances, simple training procedures, and
simple computational implementations, can indeed provide computational models for the visual system. The
Appendix provides a description of the software and data sets all of which are accessible through the web at

The Introduction is used to briefly describe the major trends in computer vision and how they stand in
relation to the work in this book. Furthermore, in the last section of each chapter references to related work
and alternative algorithms are provided. These are not comprehensive reviews but a choice of key papers or
books that can point the reader further on.

The emphasis is on simplicity, transparency and computational efficiency. Cost functions, statistical
models, and computational schemes are kept as simple as possible - Occam’s razor is too often forgotten
in the computer vision community. Statistical modeling and estimation play an important role, including
methods for training the object representations and classifiers. The models and algorithms are described at
a level of detail which should enable readers to code them on their own, however the readers also have the
option of delving into the finest details of the implementations using the accompanying software. Indeed, it
is sometimes the case that the key to the success of an algorithm is due to some choices made by the author,
which are not necessarily viewed as crucial, or central to the original motivating ideas. These will ultimately
be identified by experimenting with the software. It is also useful for the readers to be able to experiment
with these methods and discover for themselves the strengths and weaknesses, leading to the development
of new and promising solutions.

The images from the experiments shown in the book, and many more, are provided together with the
software. For each figure in the book a parameter file has been prepared allowing the reader to run the
program on the corresponding image. This should help jump start the experimentation stage. Even trying
to change parameter settings in these files can be informative, or running them on additional images. Chapter
12 should provide the necessary documentation for understanding the parameters and their possible values.

Hopefully the examples presented in the book will convince the reader that problems emerging in dif-
ferent computer vision sub-communities from the document analysis community to the medical imaging
community can be approached with similar tools. This comes at the expense of intensively pursuing any one
particular application. Still the book can be used as a reference for particular types of algorithms for specific
applications. These include detecting contours and curves, image warping, anatomy detection in medical
images, object detection, and character recognition.

Most chapters can be read separately, but Chapter 6 needs to be read before Chapters 7 or 8), and
Chapter 11 relies on the algorithm of Chapter 8. There are common themes which span several or all
Chapters, as well as discussions of connections between models and algorithms. These are in large part
found in Chapter 2, and the introductory comments and the final discussion section of each Chapter. It is
still possible to study individual models independently of the others.

The mathematical tools used in this book are somewhat diverse but not very sophisticated. Elementary
concepts in probability and statistics are essential, including the basic ideas of Bayesian inference, and
maximum likelihood estimation. These can be found in Rice (1995). Some background in pattern recognition
is useful but not essential and can be found in Duda & Hart (1973). A good understanding of multivariate calculus is needed for Chapters 3 and 5, as well as some basic knowledge of numerical methods for optimization and matrix computation, which can be found in Press et al. (1995). The wavelet transform is used in Chapters 3 and 5 where a brief overview is provided, as well as a description of the discrete wavelet transform. For a comprehensive treatment of the theory and applications of wavelets see Wickerhauser (1994). Some elementary concepts in information theory such as entropy and conditional entropy are used in Chapters 4 and 9, and are briefly covered in a section of Chapter 4. For a comprehensive treatment of information theory see Cover & Thomas (1991).

Computer vision is a fascinating subject. On one hand there is the satisfaction of developing an algorithm which takes in an image from the web or the local webcam and in under a second finds all the faces. On the other hand are the amazing capabilities of the human visual system which we experience at every moment of our life. The computer algorithms are nowhere near to achieving these capabilities. Thus every once in a while the face detector will miss a face, and quite often will select some part of a bookshelf or a tree as being a face. The visual system makes no such mistakes, the ground truth is unequivocal, and brutally confronts us at every step of the way. Thus we need to stay humble on one hand and constantly challenged on the other. Hopefully the reader will become engaged by this challenge and contribute to this exciting field.

Acknowledgements

A large part of the work presented in this book is a result of a long interaction with Donald Geman, to whom I owe the greatest debt. I am referring not only to particular algorithms we developed jointly, but also to endless conversations and exchanges about the subject of computer vision, that have been crucial in the formation of the views presented here. I am deeply thankful to Ulf Grenander for first introducing me to image analysis and deformable template models. The book as a whole is influenced by his philosophy and also by my interactions with the Pattern Analysis group in the Division of Applied Mathematics at Brown University: Basilis Gidas, Don McClure, David Mumford, and in particular Stuart Geman who through scattered conversations over the years has provided invaluable input.

The work on neural network architectures would not have been possible without the recent interaction with Massimo Mascaro. I am indebted to my father Daniel Amit for prodding me to explore the connections between computer vision algorithms and the biological visual system, and for many helpful discussions along the way.

Kenneth Wilder contributed immensely to the accompanying software which would have otherwise been unintelligible. Many thanks to Mauro Piccioni, Kevin Manbeck, Michael Miller, Augustine Kong, Bruno Jedynak, Alejandro Murua, and Gilles Blanchard who have been supportive and stimulating collaborators. I am grateful to the Department of Statistics at the University of Chicago for being so supportive over the past ten years, and to the Army Research Office for their financial support of this research.
Chapter 1

Introduction

The goal of computer vision is to develop algorithms that take an image as input and produce a symbolic interpretation describing which objects are present, at what pose, and some information on the three dimensional spatial relations between the objects. This involves issues such as learning object models, classifiers to distinguish between objects, and developing efficient methods to analyze the scene given these learned models. Our visual system is able to carry out such tasks effortlessly and very quickly. We can detect and recognize objects from a library of thousands if not tens of thousands in very complex scenes. However the goal of developing computer algorithms for these tasks is still far from our grasp. Furthermore there is still no dominant and accepted paradigm within which most researchers are working. There are a number of major trends, briefly described below, and relative to which the work in this book is placed.

1.1 Low level image analysis and bottom-up segmentation

Image segmentation is a dominant field of research in the computer vision and image analysis communities. The goal is to extract boundaries of objects or identify regions defined by objects, with no prior knowledge of what these objects are.

The guiding philosophy is that only through such low-level processing is there any chance of identifying more restricted regions in the scene for further high-level processing such as recognition. Since these algorithms operate with no higher level information about the objects they are referred to as low-level image analysis. Another commonly used term is bottom-up image processing.

Many of the early ideas which guided much of the subsequent research can be found in Duda & Hart (1973) and Marr (1982). Motivated by the connections established by Marr & Hilderith (1980) between edge detection algorithms and computations carried out in the primary visual cortex, a significant body of work in computer vision has been devoted to the specific use of edge detection for segmentation. An edge detector is used to identify all edges in the image, after which some type of local rule tells how to group the edges into continuous contours which provide continuous outlines of the objects. Other approaches to segmentation are region based. Regions with similar characteristics are identified, typically through local region growing techniques. A detailed description of a variety of such approaches can be found in Haralick & Shapiro (1992).

A statistical formulation of the segmentation problem from a Bayesian point of view was introduced in Geman & Geman (1984) combining region and edge information. An extensive review of such statistical approaches can be found in Geman (1990). The statistical model introduces global information in that the full segmentation is associated a cost or posterior probability, in terms of the ‘smoothness’ of the different regions and their contours. The various algorithms proposed to optimize this global cost are quite computationally
CHAPTER 1. INTRODUCTION

intensive. Other approaches to bottom-up image segmentation currently being proposed can be found Elder & Zucker (1996), Shi & Malik (2000), Parida et al. (1998), Ishikawa & Geiger (1998).

However there are some persistent problems with the notion of determining a segmentation of an image without any models of the objects which are expected to be present. First of all there is no agreement as to what a good segmentation really is. Furthermore continuous contours are very difficult to determine in terms of local edges detected in an image. Using local edge information alone it is very difficult to actually trace the contour of an object. For example various noise effects and occlusion can eliminate some of the edges along the contour. A local procedure for aggregating or grouping edges would encounter spurious bifurcations or terminations. Homogeneous regions are difficult to define precisely, and at times lighting conditions create artificial regions which may either cause an object to be split or merged with parts of the background.

As a result people have tried to incorporate apriori information regarding specific objects in order to assist in identifying their instantiations. This involves more specific modeling and more restricted goals in terms of the algorithms. Instead of an initial segmentation that provides the outlines of all the objects of interest, which then need to be classified, one tries to directly detect specific objects with specific models. Since shape information is incorporated into the model one hopes to avoid the pitfalls of the bottom-up approach and really identify the instantiation of these objects. This approach is called high-level image analysis information is incorporated in the models. This is the main theme of Chapters 3 through 8.

It should be emphasized that all high-level models use some form of low-level processing of the data, and often an initial edge detection procedure is performed. However such processing is always geared towards some predefined goal of detecting a specific object or class of objects, and hence are presented only within the context of the entire algorithm. In that sense there is no meaning to the notion of ‘good’ edge detection, or a ‘good’ image segmentation divorced from the outcome of the high-level algorithm.

1.2 Object detection with deformable template models

The need to introduce higher level object models was addressed in a somewhat disjoint manner in the statistics community on one hand and in the computer vision community on the other. In this section we briefly discuss the former which is the point of origin for the work in this manuscript.

High level object models, under the name deformable template models, were introduced in the statistics community in Grenander (1970) and Grenander (1978). A statistical model is constructed which describes the variability in object instantiation in terms of a prior distribution on deformations of a template. The template is defined in terms of generators and bonds between subsets of generators. The generators and the bonds are labeled with variables that define the deformation of the template. In addition, a statistical model of the image data given a particular deformation of the template is provided. The data model and the prior are combined to define a posterior distribution on deformations given the image data. The model proposed by Fischler & Elschlager (1973) is closely related although not formulated in statistical terms, and is quite ahead of its time in terms of the proposed computational tools. Much of the theory relating to these models is presented in Grenander (1978) and revisited in Grenander (1993). Some applications are presented in the latter part of Grenander (1993). The subject matter has been mostly non-rigid objects in particular objects which occur in biological and medical images.

The actual applications described in Grenander (1993) assume that the basic pose parameters such as location and scale are roughly known, namely the detection process is initialized by the user. The models involve large numbers of generators with ‘elastic’ types of constraints on their relative locations. Since deformation space - the space of bond values - is high dimensional, there is still much left to be done after location and scale are identified. The algorithms are primarily based on relaxation techniques for maximizing the posterior distributions. These types of ‘elastic’ models are described in Chapters 3 and 5. Chapter 3 draws primarily on the work presented in Grenander et al. (1991), Zhu & Yuille (1996), and Chesnaud et

Some of these ideas were developed in parallel using non-statistical formulations. In Kass et al. (1987), Terzopoulos et al. (1987) the idea of 1d deformable contours was introduced, as well as ideas of elastic constraints on deformations, and Bajcsy & Kovacic (1988) introduced the idea of image deformation as an extension of older work on image sequence analysis of Horn & Schunck (1981) and Nagel (1983). In these models a regularizing term takes the place of the prior and the statistical model for the data takes the form of a cost function on the fit of the deformed model to the data.

In much of the above mentioned work the gray level distributions are modeled directly. This can be problematic in achieving photometric invariance, invariance to variations in lighting, gray scale maps etc. At the single pixel level the distributions can be rather complex due to variable lighting conditions. Furthermore the gray level values have complex interactions requiring complex distributions in high dimensional spaces. The options are then to use very simple models which are computationally tractable, but lacking photometric invariance or to introduce complex models which entail enormous computational cost.

An alternative is to transform the image data to variables that are photometric invariant perhaps at the cost of reducing the information content of the data. However it is then easier to formulate credible models for the transformed data. The deformable curve model in Chapter 4, and the Bernoulli deformable image model in section 5.4 employ transforms of the image data into vectors of simple binary variables. One then models the distribution of the binary variables given a particular deformation rather than the gray level values. The material in Chapter 4 draws primarily from the work in Petrocelli et al. (1992) and from Geman & Jedynak (1996).

All the algorithms mentioned above suffer from a similar drawback. Some form of initialization provided by the user is necessary. However the introduction of binary features of varying degrees of complexity allows us to formulate simpler and sparser models with more transparent constraints on the instantiations. Using these models the initialization problem can be solved with no user intervention and in a very efficient way. Such models are discussed in Chapters 6, 7, and 8, based on work in Amit et al. (1998), Amit & Geman (1999), and Amit (2000).

These ideas do fit within the theoretical pattern analysis paradigm proposed in Grenander (1978). However the emphasis on image data reduction does depart from Grenander’s philosophy which emphasizes image synthesis, and which aims at constructing prior distributions and data models which if synthesized would produce realistic looking images. This image synthesis philosophy has also been adopted by people studying compositional models as in Geman et al. (to appear 2002), Bienenstock et al. (1997) and by people studying generative models such as Revow et al. (1996), as well as in Mumford (1994), Zhu & Mumford (1997). Providing a comprehensive statistical model for the image ensemble is not only a very hard task, it is not at all clear that it is needed. There is a large degree of redundancy in the gray level intensity maps recorded in an image which may not be all that important for interpreting the symbolic contents of the image.

1.3 Detection of rigid 3d objects

In the computer vision community the limitations of straight forward bottom up segmentation also led to the introduction of object models which enter into the detection and recognition process. Most of the work has concentrated around rigid 3d objects, see Grimson (1990), Haralick & Shapiro (1992), Ullman (1996). These objects lend themselves to precise 3d modeling, and the main type of deformations considered are linear or projective.

Lists of features at locations on the object at reference pose are deduced analytically from the 3d description. The spatial arrangements of these features in the image are also predicted through analytic
computations, using projective 3d geometry and local properties of edge detectors. Typical features that are used in modeling, are oriented edges, straight contour segments - lines of various lengths, high curvature points, corners, and curved contours. These paradigms describe methods for object detection using the 3d object models. Two complementary techniques for detection are searches of correspondence space and searches through pose space.

**Searching correspondence space**

One systematically searches for arrangements of local features in the image consistent with the arrangements of features in the model. The matches must satisfy certain constraints. Unary constraints involve the relationship between the model feature and the image feature. Binary constraints involve the relationship between a pair of model features and a pair of image features. Higher order constraints can also be introduced. Various heuristic tree based techniques are devised for searching all possible matchings to find the optimal one, as detailed in Grimson (1990). Invariance of the detection algorithm to pose is incorporated directly in the binary constraints. In Haralick & Shapiro (1992) this problem is called the inexact consistent labeling problem and various graph theory heuristics are employed.

Similar to the search of correspondence space, or the inexact consistent labeling problem, is the dynamic programming algorithm presented in Chapter 7, which is based on work in Amit & Kong (1996) and Amit (1997). The constraints in the models are invariant to scale and and some degree of rotation, as well as non-linear deformations. Detection is achieved under significant deformations of the model beyond simple linear or projective transformations. The full graph of constraints is pruned to make it decomposable, and hence amenable to optimization using dynamic programming, in a manner very similar to the proposal in Fischler & Elschlager (1973). The local features employed are highly invariant to photometric transformations but have a much lower density than typical edge features.

**Searching pose space**

Searching pose space can be done through brute force by applying each possible pose to the model and evaluating the fit to the data. This may be computationally expensive. But we will see in Chapter 8 that brute force is useful and efficient as long as it is applied to very simple structures, and with the appropriate data models involving binary features with relatively low density in the image.

In some cases searching parts of pose space can be achieved through optimization techniques such as gradient descent methods or dynamic programming. This is precisely the nature of the deformable models presented in Chapters 3 through 5. Note however that here objects are not assumed rigid and hence require many more pose parameters. These methods all face the issue of initialization.

A computational tool which repeatedly comes up as a way to quickly identify the most important parameters of pose such as location and scale is the Hough transform, originally proposed by Hough (1962) and subsequently generalized by Ballard (1981). The Hough transform is effectively also a ‘brute force’ search over all pose space. Since the structures are very simple the search can be efficiently implemented. The outcome of this computation provides an initialization to the correspondence space search or a more refined pose space search, see Grimson (1990) and Ullman (1996), or in our case the more complex deformable template models. In Grimson (1990) a careful analysis of the combinatorics of the Hough transform is carried out in terms of the statistics of the local features. A very appealing and efficient alternative to the Hough transform has recently been proposed in Fleuret & Geman (2001) where a coarse to fine cascade of detectors is constructed for a tree-like decomposition of pose space in to finer and finer bins. The Hough transform as a method of jump-starting more intensive algorithms, is intuitively very appealing, but did not take off as a dominant paradigm in computer vision partly because of the combinatoric problems analyzed in Grimson (1990). Testifying to this is the fact that a significant body of work in the same community did not use this
approach for face detection, see for example Rowley et al. (1998), Sung & Poggio (1998). One reason may be the use of predesigned local features.

In Chapter 6 we introduce a hierarchy of local edge arrangements of increasing complexity. Despite being more complex than simple edges, these local features are still very stable on object and quite rare in the background. The features in the model are obtained through training and do not necessarily have a clear semantic interpretation. Sparse object models are then defined as flexible arrangements of a small number of these local features. The construction and training of sparse object models in terms of these local features, and the statistical properties of these features on object and on background are also described in Chapter 6.

In Chapter 8 an efficient algorithm for detecting such models is presented, where the first step of identifying candidate locations is obtained using the Hough transform. This material is based on work in Amit et al. (1998), Amit & Geman (1999), and Amit (2000). The work in Burl et al. (1995) and Burl et al. (1998), is very similar in spirit, however the features and the statistical models are more complex, and the computation of the detection more intensive.

The dominant view in the computer vision community is still that some form of bottom-up processing involving segmentation is still necessary to jump start the detection and recognition tasks. In Ullman (1996) a case for this is made in terms of biological processes in the visual system. The point of view put forward here is that one can go a long way with a combination of model driven detections followed by more refined processing involving classification and more detailed instantiation. This is one of the main conclusions of Chapter 6 where we study the statistics of the particular local features employed in the sparse models; of Chapter 8 where we implement a version of the Hough transform for initial detection of candidate locations; and of Chapter 10 where some ideas on combining object detection and recognition for complex scene analysis are explored.

Rigid versus non-rigid objects

Much of the work on object detection is centered around rigid objects. This has led for example to detailed analysis of the specific pose space associated with 2d and 3d rigid transformations and their projections see for example Mundy & Zisserman (1992) and Arbter et al. (1990). There is also an emphasis on complete planar rotation invariance. The rigid nature of the objects has lead to reliance on ‘predefined features’ with labels such as lines, corners, junctions etc. In recent years a view based approach has become widely accepted in which 3d object detection and recognition are treated as 2d problems depending on the particular views of the objects, see Ullman (1996), Riesenhuber & Poggio (2000).

However even for 2d views of rigid objects, lines and contours or even corners can be ambiguous in the image domain. Moreover the visual system can detect and recognize rigid objects even if many of the straight lines present on the real object are deformed in the image. The message of Chapters 8 and 10 is that all objects should be studied within one framework, based on 2d views, using non-rigid 2d models. Views of the object that are substantially different are considered as different 2d objects, however the flexibility, i.e geometric invariance, introduced in the non-rigid models, implies that a wide range of views can still be accommodated by one model. This alleviates to some extent the combinatoric problem of the resulting proliferation of 2d objects that need to be modeled, detected and recognized. Some additional ideas related to this problem are presented in Chapter 10.

1.4 Object recognition

Recognition of isolated objects has been studied extensively in two main contexts: rigid 2d and 3d objects and character recognition. The latter context offers an important test bed for many ideas. Recent extensive reviews can be found in Plamondon & Sripahari (2000) and Nagy (2000). The data sets are abundant, different
forms of variability are present - rigid for printed characters and non-rigid for hand written, one can work with a limited number of classes, say only the digits, or with large numbers such as all \LaTeX\ symbols, or Chinese characters.

Deformable template models for object recognition

The problem of recognizing an image of an isolated object from among several possible classes can be addressed in a Bayesian framework using the deformable template models. These have a natural extension to a statistical model for images of the different object classes, once a prior on object classes is determined. The goal is then to compute the Bayes classifier, namely the class that maximizes the posterior on class given the data. The deformation parameters are no longer of direct interest but need to be averaged over in order to obtain the posterior on class. This type of computation is very expensive so that in real applications one deformation is estimated from each of the class templates to best fit the data and classification is then based on various metrics defined on these deformations. Such a procedure is spelled out in detail in Hastie & Simard (1998). Despite the fact that the distance of the data to each template is computed modulo the deformation, this approach still requires quite careful preprocessing and registration of the images to a standard size. The underlying assumption is that the deformations are small. It also requires explicit modeling of the prototype images, extensive computation at the classification stage and appears impractical with large numbers of shape classes. A deformable template based approach to face recognition is presented in Wiskott et al. (1997), although not based on a statistical model. There the data model is not based directly on the pixel intensities but on local features based on Gabor filters extracted at multiple scales.

In Chapter 9 we present an alternative based on elements of the sparse detection models. The main tool will be binary classification trees, see Breiman et al. (1984), where the splits are defined in terms of flexible arrangements of local features of the same nature as those defining the sparse models. Trees provide a natural mechanism for exploring arrangements of increasing complexity.

Instead of modeling the posterior distribution on deformations and on class, and then computing the posterior on-line, the trees yield partial posteriors conditional on a smaller number of variables, which are obtained off-line during training. Computation during classification is then very fast. An essential element of our approach is to produce multiple randomized classification trees. Individually the error rates of these trees can be quite large, but when aggregated a very powerful classifier emerges. The work in this chapter is based on Amit & Geman (1997) and Amit et al. (1997).

Normalization and registration

In the literature on statistical pattern recognition it is common to address geometric and photometric variations by preprocessing and normalization. A ‘standardized’ image is produced prior to classification, involving a sequence of operations which bring all images to the same size and then correct for translation, slant and rotation. This is not done using some template or model because the class of the image is unknown. Classification is then performed by one of the standard pattern recognition procedures based on the gray level intensities of the standardized image. For example penalized discriminant analysis in Hastie et al. (1995), or multilayer neural networks in Bottou et al. (1994), or classification trees in Ho et al. (1994). Difficulties in generalization are often encountered because the normalization is not robust and does not accommodate nonlinear deformations. This deficiency can only be ameliorated with very large training sets.

An alternative is to define a collection of binary features extracted from the data such as edges, contours, endings junctions etc. The feature/location pairs are collected to make a fixed sized feature vector which is fed into a standard type of classifier. These features may be designed to be more invariant to geometric deformations than the raw gray level values, using explicit disjunction (ORing). Otherwise put, a feature detected at a particular location is ‘spread’ to an entire neighborhood. The features are designed to be
photometric invariant, so that no gray level normalization is required. These matters are investigated in Chapter 9.

**Geometric invariants**

Another approach, which has been explored in the computer vision literature, is to search for functions invariant to a family of transformations, such as the affine transformations. Discrimination is possible if the functions have different values for different classes. See for example Mundy & Zisserman (1992), Forsyth et al. (1991), Binford & Levitt (1993), Lambda et al. (1988), Reiss (1993). The explicit introduction of geometric invariance is very appealing, and introduces an element which is missing in the standard pattern recognition approaches. The problem however is that the invariant functions are defined in terms of precisely located distinguished points on the object. This is not very practical on real gray level images, or for objects that are deformed non-linearly. This brings us back to the discussion above regarding rigid versus non-rigid objects. Just as in the case of detection, it is useful to consider shape classification for both categories as one problem. Focusing on invariants associated strictly with rigidity can lead to unstable algorithms. Hence the introduction of looser types of functions - flexible arrangements of local features which is also explored in Chapter 9. This however comes at the price of full rotation invariance.

### 1.5 Scene analysis: merging detection and recognition

The grand goal of computer vision is to enable the computer to detect and recognize multiple objects in a visual scene. We are still very far from achieving this goal. This is not only a function of computational limitations, it is also a result of the lack of one dominant paradigm agreed upon by most of the research community.

As indicated earlier one paradigm assumes bottom-up image segmentation as a precursor to any high level processing. At the other extreme are compositional and generative models, see (Geman et al. (to appear 2002), Bienenstock et al. (1997), Mumford (1994), Hinton et al. (1995), where people attempt to provide a comprehensive statistical model of entire scenes, both from the point of view of the components generating the scene, namely priors on complex scenes, and complex data models of the images given a particular configuration of objects. The view is that local ambiguities can only be resolved in the framework of a comprehensive explanation of the data. These models appear diametrically opposed to the segmentation models. The only way to unambiguously determine the boundary of an object is by identifying the object, its pose, all the objects in its neighborhood and their respective positions. Conceptually these models are appealing in their attempt to pose the scene analysis problem in a comprehensive Bayesian framework. Regrettably they are extremely challenging on all levels; Formulating the prior models, the data models, estimating the relevant parameters and ultimately computing the optimal interpretation given the image data.

Chapter 10 is an initial exploration into a possible middle path between these two extremes, based on a combination of the sparse detection models described in section 8 and the recognition algorithms of Chapter 9. Local features in the image are not grouped in a bottom-up manner as in standard segmentation, rather the grouping is an outcome of the detection of a particular model, and comes with an estimated pose and some additional instantiation parameters. These model driven groupings of local features can be viewed as elements of a compositional model. Recent implementations of compositional models have used very gradual compositions, from edgelets to lines or curves to small combinations of these etc. The compositions proposed here are very coarse and there is a direct jump from the local to the global model.

If the detection models are created to be less specific, either by directly training on a collection of classes, or by training on one class and then using lower thresholds, they define object clusters, as opposed to being dedicated to one particular class. This means that classification must follow detection. Detecting instances
of several coarse models and subsequently classifying them is a very efficient way to obtain a relabeling of the image into detections (involving some pose parameters) and class labels. This labeling in no way provides a final scene interpretation. There could be multiple labels at the same location, overlapping detections and so on. From the point of view of the compositional and generative models, this can be taken as a crude first pass, which provides the higher level models multiple possible scene interpretations for evaluation. In Chapter 10 we discuss possible strategies for generating this basic map of labeled detections. How to then analyze this information and produce coherent scene interpretations is beyond the scope of this book.

1.6 Neural network architectures

There have been a number of attempts to formulate parallel network architectures for higher level vision tasks such as detection. Some examples are the work in Fukushima (1986) and Fukushima & Wake (1991), Olshausen et al. (1993), and recent models such as Riesenhuber & Poggio (1999). Each of these models touches upon certain important aspects of the problem. In Fukushima & Wake (1991) hard wiring of invariance is achieved through OR-ing, ‘spreading’, which is an important component of the algorithms described in this book. However the proposed network depends too heavily on a long sequence of processing layers and on learning more and more complex features. In Olshausen et al. (1993) mechanisms for shifting data from the periphery to the center for further processing are studied but the training of classifiers or implementing object detection as a component of visual selection are not discussed. In Riesenhuber & Poggio (1999) invariant recognition is achieved through a combination of OR-ing as in Fukushima & Wake (1991), generalized to continuous variables through a MAX operation, and predefined pairwise conjunctions of features. The MAX operation is taken to an extreme where all information on the relative locations of features is lost at the highest stage. This is problematic when dealing with even simple data sets such as the NIST hand written character dataset. Moreover this approach cannot produce accurate location information in a detection problem. In Chapter 11 we explore how object representations and classifiers, trained using the principles of Hebbian learning, in a central memory module, are able to drive visual selection, over the entire scene and at a wide range of scales, as well as classify isolated objects or those present at a selected location. The material ties together the work in Amit (2000) and Amit & Mascaro (2001) into one comprehensive system.
In this chapter we present an overview of the object detection and object recognition models and introduce some notation which will help unify the ideas developed in subsequent chapters.

We will be dealing with gray level images. An image is defined on a grid $L$ of points $x \in L$ also referred to as pixels, to each of which is assigned a gray level value $I(x)$. We also use the terms pixel values or intensity values for $I(x)$. The term scene refers to a complex image containing more than one object. At times the term image will refer to the underlying grid as opposed to the gray level values, for example when talking about the ‘size of an image’ or ‘points in the image’. By the size of an image we mean the size of the grid $L$ on which it is defined. This can vary greatly even for images of an individual object. An image of a face can be as small as $16 \times 16$ pixels and as large as $500 \times 500$ or more. It is useful then to introduce the notion of a reference grid $G$ of fixed size on which prototypical images of objects are presented. We also use the term image surface which refers to the surface defined by the graph of the function $I(x), x \in L$.

2.1 A Bayesian approach to detection

Consider the example illustrated in figure 2.1, where a prototype $\mathcal{E}$ is shown together with a number of deformations, all of which are considered instances of an $\mathcal{E}$. Below is a synthetic scene with an instance of a deformed $\mathcal{E}$ present among other objects. Our goal is to detect instances of the symbol $\mathcal{E}$ in such a scene. How do we go about formulating models and developing the associated algorithms?

Start with an ordered sequence of model points $Z = (z_1, \ldots, z_n)$, also called the template, defined on the reference grid $G$. Detection is defined as finding a map from these model points on the reference grid into a set of points in the image. Since the model points are ordered, the map is uniquely determined by an ordered set of points $\theta = (\theta_1, \ldots, \theta_n)$ in the image, which is also called an instantiation of the model. This is illustrated in figure 2.2 for three models of an $\mathcal{E}$ of increasing complexity. The templates are shown overlayed on the prototype image and are mapped into another image of an $\mathcal{E}$.

In some cases the model points are chosen as points of interest or landmarks on a prototype image. For example the three ‘endings’ in the second model of figure 2.2. Intuitively a point of interest is a salient location on the prototype where the image viewed as a surface has some interesting local topography and is not simply planar. In other cases model points are chosen according to certain statistical properties of the image data in their neighborhood, evaluated on a population of training images of the object, that are presented in the reference scale on the reference grid. Models will vary in complexity in terms of the number
of points, the most complex model involving all points on the object. An instantiation of a simple model does not provide the information required for determining the instantiation of a more complex model. Knowing the approximate location of the landmark represented by the point in the left hand model of figure 2.2 does not tell us where to find the other points in the other two models. However information on the instantiation of the simple model can restrict the range of possible instantiations for the more complex models.

The prior

The model is incomplete without two additional components. The first component involves the definition of the set $\Theta$ of admissible instantiations. Through the definition of the set $\Theta$ we determine the degree of invariance expected of the associated detector. For example the range of scales and rotations the detector is expected to cover. It will be convenient to define elements of $\Theta$ in terms of a translation coupled with some instantiation around the origin. Specifically $\Theta^{(0)}$ will denote a collection of admissible instantiations more or less centered at the origin. Each instantiation $\theta \in \Theta$ is of the form $\theta_i = (x + \theta'_i), i = 1, \ldots, n$ where $x$ is a location in the image and $\theta' \in \Theta^{(0)}$. Note that $x$ and $\theta'$ are not uniquely determined by $\theta$. Since we have no prior constraints on locations $x$ all the constraints are defined on $\Theta^{(0)}$.

A useful example of such a set is given by

$$\Theta^{(0)} = \{(\theta_1, \ldots, \theta_n) : \theta_i = A z_i, A \in \mathcal{A}\},$$

where $\mathcal{A}$ is some subset of linear transformations. Here we accept only configurations which can be obtained by a linear map from $\mathcal{A}$ applied to the model configuration. A more general set of constraints is defined as

$$\Theta^{(0)} = \{(\theta_1, \ldots, \theta_n) : \theta_i = vA z_i, v \in \Upsilon, A \in \mathcal{A}\}, \quad (2.1)$$

where $\Upsilon$ is some prescribed set of non-linear deformations in the neighborhood of the identity map. A linear map is applied to the model points and the result is perturbed by a non-linear deformation $v \in \Upsilon$. A simpler extension of the first definition has the form

$$\Theta^{(0)} = \{(\theta_1, \ldots, \theta_n) : \theta_i = A z_i + C, A \in \mathcal{A}\}, \quad (2.2)$$
where $C$ is some neighborhood of the origin. For $C$ sufficiently large this last set of admissible instantiations contains the second; there are no constraints on the relative locations of the points as long as they are in the proper regions.

A prior distribution $P(\theta)$ on $\Theta$ determines which instantiations are more likely and which are less. The role of the prior is to penalize certain deviations from the model instantiation defined by the template. It is usually hard to precisely describe the distribution on instantiations, or to reliably estimate it from training data. It is therefore important to define ‘loose’ priors which do not risk precluding plausible instantiations.

The data model and the posterior

The second component of the model is a description of the image data $I(x), x \in L$ given the object is present in the scene and the specified landmarks on the object are at a particular instantiation $\theta$. Because of the variable nature of image formation, this description comes in the form of a conditional distribution often called the likelihood or data term. Given a particular instantiation there is a range of possible associated images. Lighting can change to produce different gray levels, various noise effects can occur, parts of the object may be occluded. Moreover even though the instantiation of the model points is fixed, the local and global shape of the object can still vary. For example many instances of the $E$ can have the three ‘endings’ at the same location.

The model assigns a probability distribution to the set of possible images given a particular instantiation. It is often easier to describe this probability on local transforms of the image data which are invariant to some of the gray level variations just mentioned, i.e transforms that are photometrically invariant. The data transform $\bar{I}(x)$ at pixel $x$ will be a vector of local features, namely functions applied to the gray level intensities in the neighborhood of $x$.

$$\bar{I}(x) = (X_1(x), \ldots, X_J(x))$$

$$X_j(x) = X_j(I_{N_t(x)}), \quad j = 1, \ldots, J$$

(2.3)

where $I_{N_t(x)}$ is the image data in the $t \times t$ neighborhood of $x$, and $X_j$ is a function of that data. In most cases described here $X_j$ will be binary, and we say that $X_j$ is on at $x$ if $X_j(x) = 1$. See for example the data transform applied to a sample $E$ in figure 2.4. Four operators are applied at each point. The response of a feature is 1 if the image data in a neighborhood of a point corresponds to a line at a certain range of orientations.

Having chosen a particular data transform write the likelihood or conditional probability of $\bar{I}(x), x \in L$, given an object is present at instantiation $\theta$ as $P(\bar{I}(x), x \in L|\theta)$. In most cases we will assume that conditional on the presence of an object at instantiation $\theta$ the transformed data at the different pixels is independent, so that the data term has a simple product form.

$$P(\bar{I}(x), x \in L|\theta) = \prod_{x \in L} P(\bar{I}(x)|\theta)$$

(2.4)

where we emphasize that the distribution $P(\bar{I}(x)|\theta)$ could be different for different locations $x$. The product distributions are usually not very accurate models of the data, however they allow for efficient computations and properly used lead to very useful results.

Once a prior distribution and a likelihood are defined, Bayes’ rule allows us to form a posterior distribution on the set of instantiations given the observed data.

$$P(\theta|\bar{I}(x), x \in L) = P(\bar{I}(x), x \in L|\theta)P(\theta) \cdot C,$$

(2.5)

where $C$ is a constant that does not depend on $\theta$. We typically work with the negative log-posterior which up to a constant term is given by

$$J(\theta) = -\log P(\theta) - \log P(\bar{I}(x), x \in L|\theta).$$

(2.6)
The computational task is to find one or more minima of this cost function, (hence the use of negative log-posterior,) namely instantiations that are highly likely given the observed data. Intimately related to the formulation of the model are the computational tools employed to perform this minimization. Chapters 3 through 8 describe a collection of such models and the associated computational algorithms.

In the simple example shown in figures 2.1 and 2.2, the instances of the object are produced through smooth deformations of a single prototype image. However for real objects in real images this is rarely the case. Consider faces for example. One can hardly imagine producing all faces using smooth deformations of one or even a small number of prototypes. More detailed instantiations may require specific models for subclasses. Finally we expect to detect instantiations even if part of the object is hidden or occluded, this needs to be somehow incorporated in the data models.

In some of the algorithms described below, the underlying assumption is that exactly one object is present in the image, and finding a minimum of the cost function using some optimization procedure such as gradient descent or dynamic programming will lead to the instantiation. When more than one object can be present, with a limit of say $K$ a more complex model is needed involving a prior on

$$\bigcup_{k=1}^{K} \Theta^{k},$$

where $\Theta^{k}$ is the set of $k$-tuples of instantiations from $\Theta$. This becomes practical only with the sparse models described below, see Chapter 6, where a very loose prior is used with no constraints on the relative locations of the objects. In some cases more information is available and more structure can be introduced into the prior assumptions on the configurations of multiple objects in the scene. Such matters are beyond the scope of this book.

### Statistical models versus. cost functions

Those less familiar with Bayesian modeling can interpret the first term of equation 2.6 as a penalty on deviations from the model instantiation and the second term as a measure of how well the instantiation fits the data. Often in the literature models are formulated in the form of such a cost function without bothering about the statistical setup. This can work, yet it is hard to ignore the inherent stochastic nature and variability of both the object presentation and the image formation. The same exact physical scene, captured at two consecutive moments by the same camera, can have quite variable gray level maps simply due to a slight shift in lighting caused by the movement of the tree outside the window.

The advantage of statistical modeling emerges in the formulation of the data term. We are forced to model the distribution of the entire image data or transformed image data given an instantiation. This creates a framework in which a proper weighting of different instantiations is possible. The introduction of probabilities into the modeling process forces us to systematically consider the relative weights of different events. In some models that are directly formulated in terms of a cost function, different parts of the image data are used to evaluate different instantiations. This is problematic when it comes to comparing their cost. Finally the statistical formulation provides a natural framework for estimating the unknown parameters of the model.

On the other hand it is important to stay faithful to the principle of Occam’s razor, and insist on simplicity and transparency of the models. Typically training data are not really random samples from the populations and complex statistical models will yield highly biased parameter estimates that do not generalize. Especially with the rise of Monte Carlo based simulation and Expectation Maximization methods there is a sense that ‘anything can be estimated’. This can be quite misleading. For this reason most of the statistical models presented below are simple, and the parameters are for the most part simple proportions of individual binary variables.
2.2 Overview of object detection models

The different detection algorithms described in this book involve variations in the definition of the template, i.e. the sequence of model points \( Z \), the set of admissible instantiations \( \Theta \) and the prior distribution defined on that set; the image transforms \( I \), together with data model, i.e. the likelihood of the data given an element \( \theta \in \Theta \); methods for estimating relevant parameters, and finally the computational algorithm for maximizing the posterior. These components are very tightly interlinked. The type of algorithm chosen may constrain the types of data models as well as the definitions of the set \( Z \) and \( \Theta \). Typically the set \( \Theta \) will cover a limited range of scales, say +/- 25\%, around the reference scale determined by the reference grid. This is the smallest scale at which the object is detected. For significantly larger scales the image is downsampled and the same procedure is implemented.

**Deformable models**

These models involve a rather large and dense set \( Z \), and a set \( \Theta^{(0)} \) defined using some variation on equation 2.1. The intuition is that a linear transformation of the model is smoothly deformed to produce the instantiation of the object. The set \( \Psi \) is defined through some finite dimensional parameterization of non-linear deformations of the set \( Z \) and a prior is defined which penalizes large deviations from the identity map.

The initial location and linear map from \( A \in A \) are provided by the user. This defines an initial instantiation \( \theta_{0,i} = x_0 + Az_i, i = 1, \ldots, n \). The aim is to find the instantiation \( \theta \in \Theta \), which maximizes the posterior using relaxation methods or other optimization methods in a neighborhood of \( \theta_0 \).

**Deformable contours**

In chapter 3 the set of points \( Z \) forms a closed circle in \( G \), or some other closed contour with a specific shape. The set \( \Theta^{(0)} \) is a family of smooth perturbations of the model contour. Optimization is done through gradient descent methods. Under the data model the pixel values are conditionally independent given the instantiation of the contour, one distribution for the interior of the contour, and another for the exterior. An illustration is given in figure 2.3. The left panel shows the points in the set \( Z \) forming a closed curve overlayed on the prototype image. The middle panel shows the initial contour placed in the data image and the right panel shows the final instantiation identified by the algorithm.

![Figure 2.3: Left: A contour template for the \( \mathcal{E} \) (the points of \( Z \)) overlayed on prototype. Middle: Model curve placed in image at initial location. Right: Final instantiation.](image)

**Deformable curves**

In chapter 4, the points in \( Z \) define an open curve and \( \Theta^{(0)} \) represents deformations of the model curve. A prior \( P(\theta) \) on \( \Theta^{(0)} \) penalizes irregular deviations from the model. The data term is given in terms of a collection of binary local features detecting ‘ridges’ of the image surface at a range of orientations. These are defined in terms of simple comparisons of pixel intensity differences, and are highly invariant to photometric...
transformations, see figure 2.4. Under the likelihood model the binary features are conditionally independent at all locations in the image given the instantiation of the curve. There is a probability associated to finding certain features at certain locations on the curve, and a lower probability of finding these features anywhere else in the image. These probabilities can be estimated from data.

Figure 2.4: Top: An instance of a deformed $\mathcal{E}$, and the data transform consisting of four oriented ridge detectors. Bottom: Left - A curve template for the $\mathcal{E}$ (the points of $Z$) overlayed on the prototype. Middle - The initial curve in the image. Right - Final instantiation.

Global optimization over a well defined neighborhood of the initial instantiation is achieved either by dynamic programming or with a tree based algorithm in certain cases. Figure 2.4 provides an illustration. In the top left panel is the data image of a deformed $\mathcal{E}$. The next four panels on the top show the locations where the four local ‘ridge’ features are found in the image. In other words these represent the transformed data $\hat{I}_1, \ldots, \hat{I}_4$. The bottom left panel shows the template points $Z$ overlayed on the prototype image. The middle panel shows the initial curve placed in the data image and the right panel shows the final instantiation identified by the algorithm.

Deformable images.

In chapter 5, $Z$ is the entire reference grid and $\Theta^{(0)}$ is defined through a finite dimensional parameterization of deformations of the reference grid. The prior is used to penalize irregular non-smooth deformations. Two data models are discussed. A Gaussian data model simply uses a prototype image of the object and assumes that every image in the class is obtained by first warping the prototype image (as for example in figure 2.1) and then adding independent Gaussian noise. No model for data on the object is provided. This is the classical image deformation model found in the literature.

The second model, the Bernoulli data model, uses transformed image data defined in terms of binary oriented edge type features, which like the ridge features are also defined in terms of comparisons of pixel intensity differences. In training we identify the probabilities of each edge type at each point in the reference grid assuming an instance of the object is present at the reference scale and location. There is a lower bound on these probabilities determined by the general density of edges in generic images. There are eight oriented edge types and hence eight probability maps defined on the reference grid. Conditional on a deformation $\theta$, we assume that the edges in the image occur independently according to the deformed probability maps. Optimization in both models cases is done with gradient descent methods.

In figure 2.5 the prototype image is shown in the upper left panel. No points are marked because every
point in the reference grid is in the set $Z$. The upper right panel is the data image. The bottom panel left shows the instantiation obtained by the algorithm in the form of a vector field. Every point in the reference grid is mapped according to the arrow attached to it. The bottom left panel shows the prototype image deformed according to the identified instantiation and should be compared to the data image above it. The Gaussian data model was used in this experiment.

![Figure 2.5: Top Left: The $E$ prototype. $Z$ is the entire $32 \times 32$ reference grid. Top Right: The data image. Bottom Left: Vector field describing instantiation. Each point on reference grid is mapped into some point in the image. Bottom Right: Warping of prototype image according to detected instantiation.](image)

**Global detection with sparse models**

Sparse models are defined in terms of a smaller set $Z$ and a data transform $\hat{I}$ involving binary local features that are more complex than simple oriented edges. We mainly make use of *local edge arrangements*. There is typically a different local feature $X_i$ associated to each model point $z_i \in Z$. This is in contrast to models discussed earlier involving a small number of generic binary features. Again the features are assumed independent given the object instantiation. On the background the local features are of much lower density than oriented edge features, since they involve local conjunctions defined in terms of these edge features. However the probability of occurrence on particular parts of the object is still relatively high. The object model now has the form of a *flexible arrangement of binary local features*. The degree of flexibility is determined by the set $\Theta^{(0)}$.

The conditional independence assumption in this case becomes somewhat more realistic and greatly facilitates the computation of global detections with *no initialization required*. The types of data transforms used to define the local features, methods for training the models, and an analysis of the statistics of the local features on and off the object are presented in Chapter 6. Two alternatives for computing the most likely instantiations are provided in Chapters 7 and 8.
Sparse model detection: dynamic programming

In Chapter 7, \( \Theta \) and the prior \( P(\theta) \) are defined in terms of constraints and penalties on the spatial arrangements of the points \( \theta_1, \ldots, \theta_n \). The constraints are invariant to scale and translation. This is done in terms of relative locations of triples of points in \( \theta \). The constraints have a certain decomposable or ‘peelable’ structure which permits maximization of the posterior through dynamic programming. Constraints on larger subsets of points are possible but greatly increase the computational load. The assumption in this case is that only one instance of the object is present in the image and that given the instantiation is \( \theta \), feature \( X_i \) is found at \( \theta_i \) with probability 1. All locations where local feature \( X_i \) is found are recorded in a list \( S_i, i = 1, \ldots, n \). This is the input to a dynamic programming algorithm that finds the arrangement \( \theta \in \Theta \) such that \( \theta_i \in S_i, i = 1, \ldots, n \), with highest posterior value.

Sparse model detection: counting

The previous model may be unstable. Even if the object is present at instantiation \( \theta \) not all features will be found at their respective points either due to various noise effects or occlusion. In a more realistic model the probabilities of the individual features on the object are significantly lower than 1. Under simplified assumptions on the probabilities of the features on and off the object, finding instantiations with a high posterior reduces to finding admissible sub-instantiations \( \theta_{i_1}, \ldots, \theta_{i_m} \) where \( m > \tau \) for some \( \tau > 0 \), and \( X_{i_j}(\theta_{i_j}) = 1, j = 1, \ldots, m \). By admissible sub-instantiation we mean a subsequence for which there exists some element \( \hat{\theta} \in \Theta \), such that \( \theta_{i_j} = \hat{\theta}_{i_j}, j = 1, \ldots, m \). This approach also allows us to find multiple instances of the object in the image and is studied extensively in Chapter 8.

The computation is done in two stages using a coarse to fine approach. In the first step candidate locations are detected using a much looser set of constraints \( \Theta_p^{(0)} \) which contains \( \Theta^{(0)} \) and which has the form of a product set. Each local feature is constrained relative to a center \( x \), irrespective of the locations of the other features. The structure of this set of constraints allows for very efficient detection of candidate locations using a simple counting operation, also known as the Hough transform, Hough (1962). In the second step, at each candidate location, again using a simple counting operation on detected local features, we decide whether to keep the location and simultaneously estimate pose parameters, i.e. scale, translation and other linear transformations, and identify a full instantiation of the model. The details of this approach are provided in Chapter 8.

A sparse model is shown in figure 2.6 consisting of 20 points overlaid on the prototype - top left panel. In the bottom left panel we show a graphical representation of seven of the twenty local features, which are defined through local edge arrangements. On the right we show a scene and one identified instantiation of the sparse model. Note that only a subset of the appropriate features was found, the ones that were not detected are marked as \( x \) and their location is obtained using the estimated pose parameters.

We reiterate that the sparse model represents the object at the smallest scale at which it will be detected. For much larger scales the image is subsampled and the same algorithm is applied again.

From deformable models to sparse models

The sparse models can be viewed as coarse approximations to the more detailed deformable models. In each region, instead of trying to describe the deformation of the template that best fits the data within the family of allowed local deformations, we define a binary local feature which is invariant with respect to these allowable deformations. Loosely speaking, if the local feature is present at some location on the object, than it will also be present at the corresponding location for any admissible deformation of the object. The local feature becomes a rough characterization of that part of the object within the range of admissible deformations. Only if a sufficient number of such features is found in an admissible configuration
somewhere in the image is it possible to find a more detailed deformation of the template at that location. This relationship is further discussed in Chapter 6.

The reader may have noted that the ordering of the models is not consistent with their logical ‘algorithmic’ ordering. Indeed the most sensible thing to do is to implement the sparse model first and thus obtain an initial point for the more intensive deformable models described earlier. We show examples of this in Chapter 8. The particular ordering here is chosen for historical reasons - deformable contours and deformable images have been around for quite a while and are actively used in many applications. Also these algorithms present a natural entrance to the field of high level vision and confront the user with certain problems which help motivate the sparse algorithms.

2.3 Object recognition

Detection can be viewed as a two-class classification problem. For each possible instantiation decide if the object is present or not. Namely at each instantiation classify ‘object’ or ‘not-object’. On the other hand we are also interested in classification among several classes. For example recognizing images of isolated characters. We assume the image contains only one object but do not know the class. One possibility is to train models for each of the objects, run each one on the image and find which fits best according to some criterion. This is a difficult path. Some of the models are quite crude and although quite successful in distinguishing between an object and generic background may get confused when distinguishing between similar objects. Other models may be more refined but are computationally intensive, and running the associated detection algorithm for each object class would be inefficient.

The alternative is to directly train a classifier based on examples from all the classes. The main difference in training here, compared to training object models for detection, is that now samples from all classes
are used simultaneously, and training explicitly identifies properties which discriminate between classes, as opposed to simply creating representations of the individual classes. Classification trees will serve as the classifier of choice. The basic predictors used in growing the classification trees are arrangements of local features, similar in nature to the sparse models. As before, arrangements are not rigid, rather the locations of the features are constrained in certain regions, either defined absolutely on the reference grid, and called absolute arrangements, or in terms of relations between features, and called relational arrangements. This flexibility can be controlled and determines the degree of invariance of the classifier to geometric deformations. In contrast to the detection models, information regarding the absence of certain features from certain regions is used as well.

The queries corresponding to deeper nodes involve more complex arrangements, starting at the top node of the tree with simple queries involving a single or a pair of features. Thus the trees are using models of increasing complexity to recursively partition the population among the different shape classes. All data points at a particular node share some arrangement of some complexity. As an illustration we show in figure 2.7 a number of $E$’s all of which reached the same depth 10 node in a classification tree, together with some other symbols that reached the same node. On each image the instantiation of the relational arrangement associated to that node is shown. The lines connect features which were constrained relative to each other. This tree was made using eight simple oriented edge features as the elements of the arrangements. The observed arrangements are similar to instantiations of a sparse detection model although the constraints on the relative locations are defined somewhat differently. The type of edge features at each point in the arrangement is not specified.

Training involves recursively choosing a query at each node of the tree which optimally splits the training data present in that node. A query will simply ask whether a particular feature is present in a certain region, either defined absolutely on the reference grid, or relative to other local features which have already been found in all images present at that particular node. With large numbers of feature types and possible regions there are many thousands of possible queries at each node. Only a small random sample of these is entertained and the best query is chosen in terms of some purity criterion on the empirical distribution on class. This randomization allows us to use the same training set to produce multiple trees which are different. Aggregating the information from the multiple trees leads to classification rates by far superior to those of an individual tree produced from the same training data. These issues are discussed in Chapter 9.
Figure 2.8: Classified objects detected by a sparse $E$ model.

2.4 Scene analysis: combining detection and recognition

The detection algorithms we have outlined assume we know what object to find in the image. However often we are presented with a scene with only very general notions of what we expect to find there. In principle we could develop a detector for each possible object class and run each such detector on the entire scene. This would of course be highly inefficient. On the other hand if we are presented with an image with a more or less isolated object we can train classifiers to recognize the object even from among hundreds of classes. The question is how to obtain isolated objects from a complex scene?

The traditional answer in computer vision assumes some form of bottom-up image segmentation which tries to identify regions of interest, i.e. regions corresponding to objects, using no high level model information, solely based on local information and very generic constraints on the properties of the regions. In Chapter 10 we explore an alternative approach, where detection always precedes recognition. Detectors are produced for object clusters from several classes. For example we may lower somewhat the detection threshold $\tau$ in the sparse model for the $E$’s and find that a large fraction of $B$’s, $C$’s, $G$’s, etc., are detected as well. Another possibility is to use nodes of a classification tree. So for example a sparse model would be trained for all images in the node represented in figure 2.7.

Using the detected pose the data in the neighborhood of the detection is registered to a reference grid. This is first done on training images to produce classifiers among detected classes. Then in a general scene the data around each detected pose is registered and subsequently classified. The idea is illustrated in figure 2.8 for the cluster of deformed script-style $\LaTeX$ symbols hit by the $E$ detector. Each hit of this detector in the scene is subsequently classified. The main point is that the first step in analyzing a scene is always an efficient counting detector which provides candidate poses for objects and hence input data for the classifier. The detector may be quite coarse in that it is not designed to pick out a very specific class and may result with a larger number of hits. However it is still guiding the low level processing of the image in terms of a particular set of more complex local features which need to be extracted, in a particular configuration.
2.5 Network implementations

The specific characteristics of the counting detector for the sparse model - the use of binary local features and the simple counting operations involved - make it a natural candidate for trying to model detection in a parallel network of neurons. In Chapter 11 we show that both learning and detection can be implemented in such a framework using minimal assumptions on the complexity of the units - binary neurons - and the connections between them - bounded positive synapses. The input into the system are the oriented edges detected in the image, which are then processed in local feature layers which detect simple two-edge arrangements. An object detection task implies evoking a specific object model in a central module. This causes certain intermediate layers to be primed. These also receive input from the local feature layers and can be viewed as the junction between the bottom-up information coming from the local feature layers and the top-down information coming from the central module. Summing the activities in these intermediate layers produces the candidate locations for the object. Training is achieved using simple Hebbian learning on the connections between an ‘abstract’ layer in which randomly selected populations of units code for each class, and the central module.

This architecture does not involve realistic neurons with realistic time dynamics. However it does offer a global model for how the visual system can learn object representations in a central module, and then somehow distribute the information to the entire system in order to compute detections in large scenes. In this framework we are also able to integrate a classification system which is also trained using Hebbian learning on connections feeding from a copy of the local feature layer and into the ‘abstract’ layer. Moreover we show how detection can produce the gating of attention to the selected location, enabling more detailed classification of the data at that location.
Chapter 3

1-d Models: deformable contours

Many problems in image analysis require the detection of the closed boundary of an object. In medical imaging one might need to delineate the boundary of a tumor in an X-ray or MRI image, the boundary of a ventricle of the heart in an angiogram, or any other anatomy of interest. In biological research boundaries of various microscopic organisms imaged using a variety of techniques may be of interest.

In the introduction we discussed possible drawbacks of a purely bottom-up image segmentation approach to determining the regions in the image occupied by objects. A partial solution is to introduce global constraints, in particular the notion that the ‘ideal’ contour is closed and continuous at all points. One natural idea is then to place a closed contour on the image and allow it to smoothly deform according to some data driven criterion so as to adjust to the boundary of the desired object which is present in the image. If the deformations preserve the continuity of the contour and do not tear it apart these properties will automatically be inherited by the final state, even if parts of the target contour are occluded or missing in the image. Moreover if the initial contour has a characteristic shape representing some prior knowledge regarding the shape of the target contour, then it is possible to preserve this shape with appropriate deformations. This approach is appealing in its simplicity, and is applicable in many contexts. The computations involve gradient descent techniques which in this case are quite fast because of the one dimensional nature of the problem. However there are serious limitations in the sensitivity to initialization and in parameter settings as will be illustrated below.

3.1 Inside-Outside model

In terms of the notation defined in Chapter 2 the sequence of model points $Z = (z_1, \ldots, z_n)$ defines a closed contour on the reference grid. This sequence of points is also called the template. Since optimization is done with gradient descent methods formulating the problem in the continuum facilitates the variational analysis. Thus we consider the points in $Z$ as a discrete sampling of a closed continuous curve $z(t), t \in [0, 1]$, at points $0 = t_0 < t_1 < \ldots < t_n = 1$, i.e. $z_i = z(t_i), i = 0, \ldots, n$, with $z_0 = z_n$. Instantiations $\theta \in \Theta$ will similarly be viewed as a discrete sampling of a curve $\theta(t)$ at the same time points, i.e. $\theta_i = \theta(t_i), i = 1, \ldots, n$. We abuse notation and use $\theta$ to denote both the continuous curve and the discrete instantiation in the image. The deformations of the initial contour $z(t)$ can be described as a parameterized family of continuous contours denoted $\theta(t, u) = (\theta_1(t, u), \theta_2(t, u)), t \in [0, 1], u \in \mathcal{U}$, with $\mathcal{U}$ a finite dimensional parameter space. Consequently the set of instantiations is given by $\Theta = \{\theta : \theta_i = \theta(t_i, u), i = 1, \ldots, n, u \in \mathcal{U}\}$. For each $u$ the entire contour is denoted $\theta(u)$, and the template contour $z(t) = \theta(t, u_z), t \in [0, 1]$, for some $u_z \in \mathcal{U}$. The curves are assumed oriented counter clockwise, and $\theta_{in}(u)$ denote the points inside the contour; $\theta_{out}(u)$ the points outside.
We employ the statistical approach outlined in chapter 2. A prior $P(u)$ on $U$ will assign higher probability to smooth contours that are close to the template contour. The likelihood model will assume that given the contour $\theta(u)$ the pixel intensities inside the contour are independent with distribution $f_{in}(I(x); \eta_{in})$ and the pixel intensities outside the contour are independent with distribution $f_{out}(I(x); \eta_{out})$. First we describe ways to parameterize the contours using orthonormal bases of functions, in particular the periodic Daubechies wavelet basis. The parameters of interest are the coefficients of the contour in this basis. We describe a prior on these coefficients and the likelihood model and obtain a posterior. The negative log-posterior yields a cost of the contour.

Contour parameterization

A general way to parameterize the contours, which allows us to control their smoothness, and control the degree of departure from the template contour, is to separately expand the two components $\theta_1, \theta_2$ in a basis of functions $\psi_k(t), k = 0, \ldots, d$ on the interval $[0, 1]$, truncated at some level $d$. An additional convenience of this parameterization is that it naturally defines a continuous curve defined for all $t$ using a finite number of coefficients. The parameters $u_1 = (u_{1,0}, \ldots, u_{1,d})$ and $u_2 = (u_{2,0}, \ldots, u_{2,d})$ are the basis coefficients. Thus the contours are expressed as

$$\theta_q(t, u_q) = \sum_{k=0}^{d} u_{q,k} \psi_k(t), \text{ and the template } z_q(t) = \theta_q(t, u_z) = \sum_{k=0}^{d} u_{z,q,k} \psi_k(t), \quad (3.1)$$

for $q = 1, 2$, and we assume that $U = \mathbb{R}^d$. This is called a spectral parameterization as opposed to a spatial parameterization which directly provides the locations of the points. Different bases of functions can be used depending on the specific application. Standard basis families such as the Fourier basis or wavelets are useful because the particular information conveyed by the coefficients is well understood. Furthermore there are numerically efficient algorithms for finding the coefficients of a function with respect to these bases.

Wavelets

In the experiments shown here we use a Daubechies wavelet basis (Daubechies (1988)). For convenience we adopt periodic wavelets. Such bases can be organized in a pyramid with $2^{s-1}$ functions at each level $s = 1, \ldots, S$. At the top levels the functions are smooth and supported on a large portion of the interval. The associated coefficients convey information on large scale properties of the target function. Lower down in the pyramid the basis functions have smaller support and the associated coefficients convey local information regarding the target function.

More formally, taking $d = 2^S - 1$, the periodic wavelets on the unit interval are indexed by two parameters $\psi_{s,\ell}, s = 1, \ldots, S$ and $\ell = 0, \ldots, 2^{s-1} - 1$, where $s$ denotes the level in the pyramid. The constant function is denoted $\psi_{0,0}$. At a given level $s$, the functions $\psi_{s,\ell}$ are shifts of the function $\psi_{s,0}$, covering the entire unit interval:

$$\psi_{s,\ell}(t) = \psi_{s,0}(t - 2^{-(s-1)}\ell), \quad l = 0, \ldots, 2^{s-1} - 1.$$ 

Furthermore depending on the wavelet type, there exists some $\bar{s} > 1$ such that for $s \geq \bar{s}$ we can write

$$\psi_{s,0}(t) = 2^{(s-S)/2} \psi_{\bar{s},0}(2^{(s-S)}t), \quad (3.2)$$

so that $\psi_{s,0}$ is a scaling and ‘dilution’ of the function $\psi_{\bar{s},0}$ defined at the deepest level of the pyramid. In fact this is true for $s < \bar{s}$ as well, modulo some ‘wrap around’ effects due to periodicity. Thus all basis functions
$\psi_{s, \ell}$ can be obtained by scaling, dilution and shifting of the mother wavelet $\psi_{S,0}$. From 3.2 it follows that as $s$ increases the support of the functions decreases, i.e. the resolution increases. Thus the level $s$ in the pyramid is also denoted the resolution. This pyramidal structure provides a natural coarse to fine mechanism for exploring the deformations of the contour. Figure 3.1 illustrates one function at some shift $\ell$, from each resolution of a pyramid with $S = 6$.

Very particular choices of the mother wavelet $\psi_{S,0}$ yield a set $\psi_{s, \ell}$ of orthonormal functions. The family of Daubechies wavelets Daubechies (1988) are parameterized by an integer $R$, and as $R$ increases the support of $\psi_{S,0}^R$ increases as does its smoothness. For $R = 1$ one obtains the more classical Haar basis.

The theory of the Daubechies wavelet bases can be found in Daubechies (1988), efficient algorithms for computing the discrete transforms can be found in Mallat (1989). A comprehensive description of the theory and algorithms for the much richer family of wavelet packets can be found in Wickerhauser (1994). Note that the models and algorithms described below will work with other bases such as splines or bases derived from a principle component analysis.

The prior

The prior is defined on the parameter space $\mathcal{U}$ taking the $u_{q,k}$ to be independent Gaussian random variables with variance $1/\lambda_k$ and means $u_{z,q,k}$. This completely defines a prior on curves $\theta(t, u)$ and hence a prior on discrete instantiations. However the latter is harder to write explicitly because many different curves can produce the same discrete instantiation. The role of the prior in this setting is simply to impose smoothness and prevent large deviations from the model curve $z(t)$. The log-prior has the form of a weighted quadratic.

Figure 3.1: One function from each level of a pyramid with $S = 6$. Daubechies wavelet with $R = 3$. 
penalty on the deviations of the coefficients.

\[ \log P(u) = -\frac{1}{2} \sum_{q=1}^{2} \sum_{k=0}^{d} \lambda_k (u_{q,k} - u_{z,q,k})^2 + C, \]  

where \( C \) does not depend on \( \theta \).

Basis functions of higher index are usually of higher frequency, or less smooth. The corresponding variances should be smaller implying that \( \lambda_k \) should increase with \( k \). For the Fourier basis

\[ \psi_k(t) = \frac{1}{\sqrt{2\pi}} \exp 2\pi ikt, \quad t \in [0,1] \]  

for \( k = 0, \ldots, d \) and we set \( \lambda_k = \alpha k^\rho \), for some \( \rho > 1 \). For the wavelet basis take \( k = (s,l) \) to be the two parameter index described above, with \( s = 1, \ldots, S \) and \( l = 0, \ldots, 2^s - 1 \). Then \( \lambda_k = \rho^s \), for some \( \rho > 0 \). All coefficients of functions at the same resolution \( s \) have the same variance \( 1/\rho^s \). In both cases, Fourier and wavelet bases, the larger the parameter \( \rho \) the more concentrated the prior on smooth functions.

The data model and the posterior

We assume that if the curve \( \theta(u) \) defines the contour of the object, the gray level values at each pixel inside the curve are independent and identically distributed according to a distribution \( f(\cdot; \eta_{in}) \), and outside they are distributed according to \( f(\cdot; \eta_{out}) \), where \( \eta_{in} \) and \( \eta_{out} \) are parameters. Hence the name inside-outside model. The log likelihood of the data given the curve \( u \) and the parameters \( \eta_{in}, \eta_{out} \) is given by

\[ \log P(I(x); x \in L|u, \eta_{in}, \eta_{out}) = \sum_{x \in \theta(u),_{in}} \log f(I(x); \eta_{in}) + \sum_{x \in \theta(u),_{out}} \log f(I(x); \eta_{out}). \]  

This assumption of conditional independence of the gray levels values inside and outside the true contour is clearly unrealistic. First nearby locations will typically be strongly correlated, furthermore the distributions can not be identical unless the interior and exterior are very homogeneous. This precludes any particular structures in the interior which may actually be characteristic of the object in question. Nevertheless in some contexts this model is meaningful, and its simplicity allows us to implement the types of algorithms described here.
The negative log-posterior on $\U$ can be written as

$$
- \log P(u|I(x); x \in L; \eta_{in}, \eta_{out}) = \frac{1}{2} \sum_{q,k} \lambda_k (u_{q,k} - u_{z,q,k})^2 - \sum_{x \in \theta(u)_{in}} \log f(I(x); \eta_{in})
$$

$$
- \sum_{x \in \theta(u)_{out}} \log f(I(x); \eta_{out}) + C,
$$

In the continuum formulation the second and third sums become integrals over the interior and exterior of the contour. Writing $F_{in}(x) = - \log f(I(x); \eta_{in})$ and $F_{out}(x) = - \log f(I(x); \eta_{out})$, the negative log-posterior on $\U$ has the form of a cost function

$$
J(u) = E(u) + D(\theta(u), I), \quad \text{where}
$$

$$
E(u) = \frac{1}{2} \sum_{q,k} \lambda_k (u_{q,k} - u_{z,q,k})^2
$$

$$
D(u) = \int_{\theta(u)_{in}} F_{in}(x) dx + \int_{\theta(u)_{out}} F_{out}(x) dx
$$

This continuum formulation is useful for computing the derivatives of $J$ with respect to $u$.

One important point which is at times ignored in the literature is that $D(\theta, u)$ should not depend on the time parameterization of the curve $\theta(u)$; it should be a quantity depending on the image and the physical curve determined by the parameter $u$.

The simplest example for $f_{in}$ and $f_{out}$ is the Gaussian distribution with means $\mu_{in}, \mu_{out}$ respectively and the same variance $\sigma$. The data model is then

$$
D(\theta(u), I) = \frac{1}{2 \sigma^2} \int_{\theta(u)_{in}} (I(x) - \mu_{in})^2 dx + \frac{1}{2 \sigma^2} \int_{\theta(u)_{out}} (I(x) - \mu_{out})^2 dx + C
$$

(3.8)

where again $C$ does not depend on $u$. The data model in this case tries to drive the mean gray level value inside the curve towards $\mu_{in}$ and outside the curve towards to $\mu_{out}$. The cost function is

$$
J(u) = \frac{1}{2} \sum_{q,k} \lambda_k (u_{q,k} - u_{z,q,k})^2
$$

$$
+ \frac{1}{2 \sigma^2} \int_{\theta(u)_{in}} (I(x) - \mu_{in})^2 dx + \frac{1}{2 \sigma^2} \int_{\theta(u)_{out}} (I(x) - \mu_{out})^2 dx.
$$

(3.9)

If the variances inside and outside the contour are different the data model has a more complex form involving the different normalizing constants of the respective densities. In the discrete setting, up to an additive constant, the log likelihood is given by

$$
- \sum_{x \in \theta(u)_{in}} \left[ \frac{1}{2 \sigma_{in}^2} (I(x) - \mu_{in})^2 + \log \sigma_{in} \right] - \sum_{x \in \theta(u)_{out}} \left[ \frac{1}{2 \sigma_{out}^2} (I(x) - \mu_{out})^2 + \log \sigma_{out} \right],
$$

which translates in the continuum to a data term

$$
D(u) = \frac{1}{2 \sigma_{in}^2} \int_{\theta(u)_{in}} (I(x) - \mu_{in})^2 dx + A_{in} \log \sigma_{in}
$$

$$
+ \frac{1}{2 \sigma_{out}^2} \int_{\theta(u)_{out}} (I(x) - \mu_{out})^2 dx + A_{out} \log \sigma_{out},
$$

where $A_{in}, A_{out}$ are the areas of the interior and exterior domains defined by the curve.
Variational analysis

The cost functions in equations 3.7 and 3.9 depend on the parameters $u$ in a complex way. They are not guaranteed to be quadratic or even convex. All we can hope to find is a local minimum of the cost function somewhere in the neighborhood of the initial value. This is achieved by following the downward gradient flow of the cost function. Specifically we solve the equations

$$
\frac{du_{q,k}(\tau)}{d\tau} = -\frac{\partial J(u(\tau))}{\partial u_{q,k}}, \quad q = 1, 2, \ k = 0, \ldots, d. \tag{3.10}
$$

Here $\tau$ is a time variable used for the gradient flow and has nothing to do with the parameter $t$ defining the curve. The next step is to compute the derivatives of the cost function in the variables $u_{q,k}, q = 1, 2, k = 0, \ldots, d$.

Let $F$ be a function defined on the domain and let $C_{in}(u) = \int_{\theta_{in}(u)} F(x)dx$. We assume the curve $\theta_{in}(u)$ is parameterized as in equation 3.1 and employ the following equality, the proof of which is provided in the next section.

$$
\frac{\partial C_{in}(u)}{\partial u_{1,k}} = \int_0^1 F(\theta(t,u)) \hat{\theta}_2(t, u) \psi_k(t) dt \tag{3.11}
$$

$$
\frac{\partial C_{in}(u)}{\partial u_{2,k}} = -\int_0^1 F(\theta(t,u)) \hat{\theta}_1(t, u) \psi_k(t) dt,
$$

Since the integral over the entire domain $\int_{L} F(x)$ is fixed, and does not depend on $u$, $\partial C_{out}(u)/\partial u_{q,k} = -\partial C_{in}(u)/\partial u_{q,k}$, and the derivative of the cost function $J$ is given by

$$
\frac{\partial J(u)}{\partial u_{1,k}} = \lambda_k(u_{1,k} - u_{z,1,k}) + \int_0^1 (F_{in}(\theta(t,u)) - F_{out}(\theta(t,u))) \hat{\theta}_2(t) \psi_k(t) dt \tag{3.12}
$$

$$
\frac{\partial J(u)}{\partial u_{2,k}} = \lambda_k(u_{2,k} - u_{z,2,k}) - \int_0^1 (F_{in}(\theta(t,u)) - F_{out}(\theta(t,u))) \hat{\theta}_1(t) \psi_k(t) dt. \tag{3.13}
$$

Observe that the derivatives of $D$ with respect to $u_{1,k}$ are simply the coefficients of $(F_{in} - F_{out})(\theta(t,u))\hat{\theta}_2$, in the basis $\psi_k, k = 0, \ldots, d$. Similarly the derivatives of $D$ with respect to $u_{2,k}$ are the coefficients of $-(F_{in} - F_{out})(\theta(t,u))\hat{\theta}_1$, in the same basis. Thus the gradient of $D$ is obtained from the forward transforms of two functions with respect to the chosen basis of functions.

This gradient can be interpreted as follows. If $\psi_k$ is a locally supported function at $t$ then $u_{1,k}, u_{2,k}$ describe the curve $\theta(u)$ in a neighborhood of $t$. Since the curve is oriented counter clockwise, $(-\hat{\theta}_2, \hat{\theta}_1)$ is the inward pointing normal, and if $F_{in} > F_{out}$ in the vicinity of $t$, the minimization, which proceeds along $(-\partial D/\partial u_{1,k}, -\partial D/\partial u_{2,k})$, is trying to pull the point $\theta(t)$ outward along the normal. The reason being that locally the inside of the curve fits the data better than the outside and an improved fit should involve enlarging the curve.
CHAPTER 3. 1-D MODELS

40

Proof of equality 3.11

The area integrals are rewritten as curvilinear integrals along $\theta(u)$ as follows. Define

\[ Q(x_1, x_2) = \frac{1}{2} \int_{x_1}^{x_2} F(t, x_2) dt, \]
\[ P(x_1, x_2) = -\frac{1}{2} \int_{x_1}^{x_2} F(x_1, t) dt. \]  

(3.14)

Then $\partial Q/\partial x_1 - \partial P/\partial x_2 = F(x_1, x_2)$, and by Green’s theorem we can rewrite the integral of $F$ over the interior of the curve as an integral along the curve.

\[ \frac{\partial Q}{\partial x_1} = \frac{\partial P}{\partial x_2} = F(x_1, x_2), \] and by Green’s theorem we can rewrite the integral of $F$ over the interior of the curve as an integral along the curve.

\[ C_{in}(u) = \int_{\theta_{in}(u)} F(x) dx = \int_0^1 P(\theta(t, u)) \dot{\theta}_1(t, u) + Q(\theta(t, u)) \dot{\theta}_2(t, u) dt. \]  

(3.15)

Writing $\partial_q P$ for $\partial P/\partial x_q$, for $q = 1, 2$, and differentiating in $u_{1,k}$ we get

\[ \frac{\partial C_{in}(u)}{\partial u_{1,k}} = \int_0^1 \partial_1 P(\theta(t, u)) \frac{\partial \theta_1(t, u)}{\partial u_{1,k}} \dot{\theta}_1(t, u) + P(\theta(t, u)) \frac{\partial \dot{\theta}_1(t, u)}{\partial u_{1,k}} + \partial_1 Q(\theta(t, u)) \frac{\partial \theta_1(t, u)}{\partial u_{1,k}} \dot{\theta}_2(t, u) dt \]  

(3.16)

Interchanging the order of differentiation in the second term and then integrating by parts, we have

\[ \int_0^1 P(\theta(t, u)) \frac{\partial \theta_1(t, u)}{\partial u_{1,k}} dt = \int_0^1 P(\theta(t, u)) \frac{d}{dt} \frac{\partial \theta_1(t, u)}{\partial u_{1,k}} dt \]
\[ = - \int_0^1 \frac{dP(\theta(t, u))}{dt} \frac{\partial \theta_1(t, u)}{\partial u_{1,k}} dt \]
\[ = - \int_0^1 \left[ \partial_1 P(\theta(t, u)) \dot{\theta}_1(t, u) + \partial_2 P(\theta(t, u)) \dot{\theta}_2(t, u) \right] \frac{\partial \theta_1(t, u)}{\partial u_{1,k}} dt. \]  

(3.17)

From equation 3.1 $\partial \theta_q/\partial u_{q,k} = \psi_k(t)$, and plugging equation 3.17 in 3.16 some terms cancel out and we get

\[ \frac{\partial C_{in}(u)}{\partial u_{1,k}} = \int_0^1 \left[ \partial_1 Q(\theta(t, u)) - \partial_2 P(\theta(t, u)) \right] \dot{\theta}_2(t, u) \psi_k(t) dt \]
\[ = \int_0^1 F(\theta(t, u)) \dot{\theta}_2(t, u) \psi_k(t). \]  

(3.18)

The second part of equation 3.11 is obtained in the same manner. A change in the orientation of the curves would change the signs in both equations.

3.2 An edge based data model

The original work on deformable contours in the computer vision literature, Kass et al. (1987), did not employ a statistical model. Rather a cost function was directly formulated in which the data term evaluates
how consistent the curve is with the edges in its neighborhood. The original approach simply integrated the
magnitude of the image gradient along the curve. The cost there ignores the issue of whether the direction of
the edge is consistent with the direction of the curve, moreover it is not invariant to the time parameterization
of the curve. In other word the cost function does not depend on the physical curve in the plane, rather on
the particular way the curve is described. An alternative is to parameterize the curve between 0 and 1 and
to write
\[ D(u) = -\int_0^1 \left(-\partial_2 I(\theta)\hat{\theta}_1 + \partial_1 I(\theta)\hat{\theta}_2\right) \left(1 + \partial_1 I^2 + \partial_2 I^2\right)^{-1/2}(\theta) dt, \] (3.19)
where \( \partial_1 I, \partial_2 I \) are the partial derivatives of \( I \). If the curve is oriented counter clockwise this integrand is
positive if the normalized gradient of the image has similar direction to the outward normal. Since we are
minimizing there is a negative sign before the integral. This cost is invariant to curve parameterization and
is equivalent to
\[ \int_\theta (1 + \partial_1 I^2 + \partial_2 I^2)^{-1/2} \nabla I \cdot n, \]
where \( n \) is the outward normal to the curve. The gradient of \( D \) with respect to the parameters is
\[ \frac{\partial D}{\partial u_{1,k}} = \int_0^1 K(\theta)\hat{\theta}_2 k dt, \quad \frac{\partial D}{\partial u_{2,k}} = -\int_0^1 K(\theta)\hat{\theta}_1 k dt, \] (3.20)
where \( K(x) \) is the mean curvature of the surface defined by the function \( I \) namely
\[ K(x) = \frac{\partial_2 I \cdot (1 + (\partial_1 I)^2) + \partial_1 I \cdot (1 + (\partial_2 I)^2) - 2\partial_1 I \cdot \partial_2 I \cdot \partial_{12} I}{(1 + (\partial_1 I)^2 + (\partial_2 I)^2)^{3/2}}, \]
Once again the gradient of the data part of the cost function is the forward transform in the chosen basis of
an easily calculated function. If \( I \) is not a smooth function it is possible to smooth it with some kernel and
then differentiate. Note that the derivatives need to be pre-calculated only once at every point in the image
and stored.

Edge based cost functions do not correspond to real statistical data models. The data used to evaluate
the cost function changes with the curve, and it is not clear how different curves are weighted. Any local
structure with clear boundaries will stop the evolution of contour in its neighborhood, even if the local
structure is irrelevant to the object we are seeking to fit. On the other hand the inside-outside model can in
some cases overcome such obstacles as illustrated in some of the examples.

### 3.3 Computation

The computation of the minimizer of 3.7 is obtained through some form of gradient descent algorithm. Since
the cost function is nonlinear this simply leads to a local minimum. The simplest form of gradient descent
uses a discrete approximation to the downward gradient flow. For a given function \( f(t) \) on \([0, 1]\) let
\[ \Psi(f) = \{ \int_0^1 f\psi_k dt, k = 0, \ldots, d \} \]
denote the forward transform of \( f \), namely the coefficients of \( f \) in the basis \( \psi_k, k = 0, \ldots, d \). For a given
vector of coefficients \( u = (u_0, \ldots, u_d) \) let \( \Psi^{-1}(u) \) denote the backward transform i.e.
\[ \Psi^{-1}(u) = \sum_{k=0}^d u_k \psi_k. \]
Let \( u_{q,k,m}, k = 0, \ldots, d, q = 1, 2 \) denote the \( m \)’th iterate of the coefficients of the curve.

A time step \( \Delta \) is determined (see below) and using the expression in equation 3.12 we obtain the following algorithm.

**Algorithm 3.1: Deformable contour**

1. Choose a scale \( s \) and translation \( x \) for the initial contour \( \theta^{(0)}(t) = sz(t) + x \). Initialize \( u_{q,0} = \{u_{q,k,0}, k = 0, \ldots, d\} = \Psi(\theta_q^{(0)}), q = 1, 2 \).

   Set \( m = 0 \).

2. Calculate \( \theta_1 = \Psi^{-1}(u_{1,m}), \theta_2 = \Psi^{-1}(u_{2,m}) \), and \( \hat{\theta}_1, \hat{\theta}_2 \).

3. Set \( \beta_q(t) = (F_{in} - F_{out})(\theta(t)) \cdot \hat{\theta}_q(t) \), \( i = 1, 2 \). Compute
   \[
   v_1 = \Psi(\beta_2(t)), \quad v_2 = \Psi(-\beta_1(t)).
   \]

4. Set \( u_{q,k,m+1} = u_{q,k,m} - \Delta \cdot [\lambda_k(u_{q,k,m} - u_{q,k,m}) + v_{q,k}] \) for \( q = 1, 2, k = 0, \ldots, d \).

5. If a stopping criterion is satisfied exit, otherwise \( m \to m + 1 \) go to 2.

Note that the means \( u_{z,q,k} \) of the coefficients in the prior term from equation 3.9, have been changed to the coefficients \( u_{q,k,0} \) of the initial contour \( \theta^{(0)} \), which is a scaled and translated version of the template \( z(t) \).

**Discretization**

In a real implementation the curve is discretized using the points \( \theta(t_i, u), i = 1, \ldots, n \), which also define the instantiation. The gradient of the curve calculated in Step 2 is approximated using differences \( \theta(t_{i+1}) - \theta(t_i) \), and \( \beta_q \) in step 3 are evaluated only at the points \( t_i \) yielding two \( n \) dimensional vectors. A discrete version of the original basis functions is used. For wavelets or the Fourier basis the discrete version has discrete forward and backward transforms which we again denote \( \Psi \) and \( \Psi^{-1} \). For the Fast Fourier Transform implementation of the discrete Fourier transform see Press et al. (1995). For a fast discrete wavelet transform see Mallat (1989) and Wickerhauser (1994). In the examples shown in this Chapter we have used periodic Daubechies wavelets, and for the sake of completeness we briefly describe the discrete wavelet transform, omitting any proofs.

**Discrete wavelet transform**

Associated to various choices of the mother wavelet \( \psi_{S,0} \) (see equation 3.2), there is an even integer \( R \) and two vectors of coefficients \( g = (g_0, \ldots, g_R) \) and \( h = (h_0, \ldots, R) \), with

\[
\sum_{k=0}^{R} g_k = \sqrt{2}, \quad \sum_{k=0}^{R} h_k = 0.
\]

Let \( n = 2^S \), and start with an \( M \) dimensional data vector \( x^{(S)} = x = (x_0, \ldots, x_{n-1}) \). The transform starts from the deepest level of the pyramid obtaining the coefficients \( u^{(S)}_\ell \) as follows. Set \( s = S \) and write

\[
u^{(S)}_\ell = \sum_{j=0}^{R} h_j x^{(S)}_{(2\ell+j) \mod 2^s}, \quad \text{for } \ell = 0, \ldots, 2^s - 1.
\]
CHAPTER 3. 1-D MODELS

Note that there are only $2^{s-1}$ coefficients $u^{(s)}_\ell$ due to the scaling by 2 in the summation index. Thus $u^{(s)}_\ell$ is obtained by convolving $x^{(s)}$ with the filter $h$ and subsampling. Since the coefficients of $h$ sum to zero, $h$ is like a difference operator, and this convolution has the flavor of a ‘high-pass’ filter. It has higher response at locations where the data is discontinuous or changes rapidly.

The coefficients of $g$ add to a positive value so that convolving with $g$ amounts to smoothing, namely $g$ is a ‘low-pass’ filter. It has higher response at locations where the data is smooth. The remaining levels of the transform are then obtained recursively in precisely the same way, by defining a subsampled low pass version $x^{(s-1)}$ of $x^{(s)}$ using the filter $g$ and then reapplying equation 3.21 for $s-1$. Specifically

$$x^{(s-1)}_\ell = \sum_{j=0}^{R} g_{\ell j} x^{(s)}_{(2\ell+j) \mod 2^s}, \text{ for } \ell = 0, \ldots, 2^{s-1} - 1,$$  (3.22)

The vector $x^{(s-1)}$ has dimension $2^{s-1}$ and equation 3.21 can be reapplied to obtain $u^{(s-1)}_\ell, \ell = 0, \ldots, 2^{s-2} - 1$ which are the coefficients corresponding to level $s-1$ of the pyramid. Note that due to the ‘wrapping around’ effect of the modulo operation in the index, the same element of $x^{(s)}$ can enter the sum more than once. This procedure continues until the single coefficient $u^{(1)}_0$ is obtained from $x^{(1)}$ which is a vector with only 2 elements. Furthermore $x^{(0)}$ is a weighted sum of these 2 elements, determined by applying equation 3.22 at $s = 1$ and corresponds to the coefficient $u^{(0)}_0$. The full transform consists of the vector of coefficients $u^{(s)}_\ell, s = 1, \ldots, S$, with $\ell = 0, \ldots, 2^{s-1} - 1$, and $u^{(0)}_0$.

Each $u^{(s)}_\ell$ is the coefficient corresponding to the basis function $\psi_{s,\ell}$.

The inverse transform takes coefficients $u^{(0)}_\ell$ and $u^{(s)}_\ell, s = 1, \ldots, S, \ell = 0, \ldots, 2^{s-1} - 1$ and reproduces the original data vector $x$. This is now done recursively from the top of the pyramid. Set $x^{(0)} = u^{(0)}_0$ and recursively define $x^{(s)}$ as

$$x^{(s)}_{2\ell} = \sum_{j=0}^{R/2} (h_{2\ell j} x^{(s-1)}_{(2\ell+j) \mod 2^{s-1}} + g_{2\ell j} u^{(s)}_{(2\ell+j) \mod 2^{s-1}}),$$

$$x^{(s)}_{2\ell+1} = \sum_{j=0}^{R/2} (h_{2\ell+1 j} x^{(s-1)}_{(2\ell+1+j) \mod 2^{s-1}} + g_{2\ell+1 j} u^{(s)}_{(2\ell+1+j) \mod 2^{s-1}})$$  (3.23)

for $k = 1, \ldots, 2^{s-1}$. The coefficients of $h$ and $g$ are chosen in such a way that the procedure of equation 3.23 inverts that of equations 3.21,3.22 so that at each stage the same vector $x^{(s)}$ is reconstructed from $x^{(s-1)}$ and $u^{(s)}$. In fact the step described in equations 3.21, 3.22 can be viewed as a matrix multiplication

$$y^{(s)} = Wx^{(s)},$$  (3.24)

where $W$ is a square $2^s \times 2^s$ matrix. The first $2^s-1$ elements of $y^{(s)}$ are $u^{(s)}$ and the second $2^{s-1}$ elements are $x^{(s-1)}$. The step described in equation 3.23 is then simply computing $W^t y^{(s)}$. The fact that this reproduces $x^{(s)}$ means that $W$ is an orthogonal matrix. Identifying filters $g$ and $h$ which allow for such simple inversion of the forward transform involves a very sophisticated analysis and can be found in Daubechies (1988), including the coefficients identified for a range of values of $R$. For example for $R = 2$ we have $g = (1/\sqrt{2}, 1/\sqrt{2})$ and $g = (1/\sqrt{2}, -1/\sqrt{2})$, and for $R = 4$

$$g = \frac{1}{4\sqrt{2}} (1 + \sqrt{3}, 3 + \sqrt{3}, 3 - \sqrt{3}, 1 - \sqrt{3}),$$

$$h = \frac{1}{4\sqrt{2}} (1 - \sqrt{3}, -3 + \sqrt{3}, 3 + \sqrt{3}, -1 - \sqrt{3}).$$
CHAPTER 3. 1-D MODELS

Time step

Determining the appropriate time step $\Delta$ is a non-trivial problem. On one hand it should be large enough so that the algorithm proceeds at a reasonable rate, on the other hand if it is too large the algorithm becomes unstable. Assuming the cost function is quadratic the time step could be taken as the inverse of the largest eigenvalue of the Hessian matrix of $J$. This can be bounded from above by the trace of the Hessian of $J$.

Assuming that the functions $F_{in}, F_{out}$ are bounded by 1 this in turn is bounded by $l(\theta) + \sum_k \lambda_k$, where $l(\theta) = \sum_{i=0}^{n-1} |\theta(t_i)|$, is the length of $\theta$. The time step is then taken to be $\Delta = 1/(l(\theta) + \sum_k \lambda_k)$.

It is possible to avoid the approximation of the time step by carrying out a conjugate gradient algorithm. It is then necessary to carry out searches for the minimum along a line, and hence directly evaluate the cost function which involves 2d integrals. Using equation 3.15 these can be reduced to one dimensional integrals along the contour. The functions $P$ and $Q$ defined in equation 3.14 need only be computed and stored at the beginning of the procedure.

Coarse to fine

In all examples of bases mentioned above the low index elements of the basis are smooth functions with a large support. These describe global smooth variations of the curve. The higher indices describe higher frequency or more local variations. It turns out that minimizing first on the low index coefficients until some convergence is observed, and then gradually adding in new coefficients leads to a more stable algorithm. Choose $N_1 < N_2 < \ldots < N_A \leq d$. Apply algorithm 1 but in item (4) update only $k = 0, \ldots, N_1$. When convergence is observed, update in (4) up to $N_2$ etc. This procedure will often avoid local minima in the vicinity of the initial curve. If all coefficients are updated at once the high index coefficients dominate the cost function and local fits of the curve to noise and small elements of clutter cause it to get stuck very far from the true fit. In fact when proceeding to minimize in stages it is often possible to omit the penalty term $E(u)$ altogether.

For example when a wavelet basis is used, update all coefficients corresponding to resolutions $s \leq a$. Thus for small values of $a$ only coefficients of the smoother basis elements $\psi_{s,l}$ with large support are updated. Taking $d = n - 1 = 2^S - 1$ we have $N_a = 2 \cdot 2^a$, $a = 1, \ldots, A$, for some $A \leq S$. The factor 2 comes from the two components of the contour. The results shown in this chapter were all obtained in this fashion.

In figure 3.2 the detection of the boundary of the left heart ventricle is shown and in figure 3.3 we show some detections of the contour of a chess piece. In both the template contour is a generic circular curve. In figure 3.3 two successful detections are shown alongside a failed match in which the object merges with the background. In figure 3.4 we show 3 steps in the coarse to fine process corresponding to the outcome of the algorithm for $N_1 = 2 \cdot 2^2, N_2 = 2 \cdot 4, N_3 = 2 \cdot 8$.

In figure 3.5 we show the detection of the deformable contour algorithm in a case where the initial curve has a particular shape, that of the posterior ventricles in an axial MRI brain scan. Initialized in the vicinity of the correct position the contour detects the boundaries of the ventricles. Similar experiments using an initial curve with a shape determined by a particular object class are shown in figure 2.3 in Chapter 2 and in figure 8.17 in Chapter 8. The enumerated points on the initial curve and on the final detection provide an explicit instantiation of the shape. Such a match is meaningless in the previous examples where a generic circle is deformed into some region.

3.4 Joint estimation of the curve and the parameters

In the statistical model presented in section 3.1 we assumed the parameters $\eta_{in}, \eta_{out}$ in the data model were known, namely estimated off-line, by computing means and variances in various regions identified by the user, as inside and outside, on some training images of contours. It is also possible to model the case where these
parameters are unknown and are estimated on-line. We define a prior on the parameters as well, and our task is to maximize the joint posterior on $\eta_{in}, \eta_{out}, u$. For simplicity we assume a flat prior on these parameters in some bounded domain. Then the log of the joint posterior on $u, \eta_{in}, \eta_{out}$, i.e. $P(u, \eta_{in}, \eta_{out}|I(x), x \in L)$ has the same form as in equation 3.6, with a different constant. For fixed value of $u$, namely a fixed curve, maximizing in $\eta_{in}$ and $\eta_{out}$ produces the maximum likelihood estimates of these parameters given the data inside and outside the contour respectively. For many models these estimates have analytic expressions in terms of the data. They are plugged into the log-posterior and yield a function which depends only on $u$. It then remains to maximize this function with respect to $u$. We provide the details in the case where $f(\cdot; \eta)$ are Gaussian with unknown mean and variance.

**Gaussian**

To simplify notation let $r$ denote one of the regions $\theta_{in}(u), \theta_{out}(u)$. Set $N_r$ to be the number of pixels in $r$, and write the log-density of the Gaussian as

$$\log f(I(x), \mu_r, \sigma_r) = -\frac{1}{2\sigma_r^2}(I(x) - \mu_r)^2 - \frac{1}{2} \log 2\pi - \frac{1}{2} \log \sigma_r^2.$$ 

The cost function to be minimized is given by the negative log-posterior on instantiation and distribution
parameters together,

\[
J(\mu, \sigma_{\text{in}}, \sigma_{\text{out}}) = -\log P(\mu, \sigma_{\text{in}}, \sigma_{\text{out}}|I(x), x \in L) = \mathcal{E}(u) + \frac{1}{2\sigma_{\text{in}}^2} \sum_{x \in \eta_{\text{in}}} (I(x) - \mu_{\text{in}})^2 + \frac{N_{\text{in}}}{2} \log \sigma_{\text{in}}^2
\]

\[
+ \frac{1}{2\sigma_{\text{out}}^2} \sum_{x \in \eta_{\text{out}}} (I(x) - \mu_{\text{out}})^2 + \frac{N_{\text{out}}}{2} \log \sigma_{\text{out}}^2 + C, \tag{3.25}
\]

where \(C\) does not depend on \(u, \eta_{\text{in}}, \eta_{\text{out}}\), and incorporates the uniform prior on the parameters on some bounded interval.

Fixing \(u\), for each of the regions, minimizing 3.25 in the corresponding parameters \(\mu_r, \sigma_r\) yields

\[
\hat{\mu}_r = \frac{1}{N_r} \sum_{x \in r} I(x),
\]

and

\[
\hat{\sigma}_r^2 = \frac{1}{N_r} \sum_{x \in r} I^2(x) - \hat{\mu}_r^2.
\]

These are precisely the maximum likelihood estimates of the mean and variance in the region. Using the fact that

\[
\sum_{x \in r} (I(x) - \hat{\mu}_r)^2/\hat{\sigma}_r^2 = 1,
\]

and since \(N_{\theta_{\text{in}}} + N_{\theta_{\text{out}}}\) is constant (the total number of pixels in the lattice \(L\)), we have

\[
\max_{\mu_{\text{in}}, \sigma_{\text{in}}, \mu_{\text{out}}, \sigma_{\text{out}}} J(u, \mu_{\text{in}}, \sigma_{\text{in}}, \mu_{\text{out}}, \sigma_{\text{out}}) = \mathcal{E}(u) + \frac{N_{\text{in}}}{2} \log \hat{\sigma}_{\text{in}}^2 + \frac{N_{\text{out}}}{2} \log \hat{\sigma}_{\text{out}}^2 + C'.
\]

For the continuum formulation define

\[
N_r(u) \doteq \int_r \mathbf{1} dx, \quad H_r(u) \doteq \int_r I(x) dx, \quad G_r(u) \doteq \int_r I^2(x) dx.
\]
Figure 3.5: Initial and final state of a deformable contour for detecting the posterior ventricles in two axial MRI brain scans. Top: initial state. Bottom: final state.
Then $N_r$ substitutes $N_r$ and $\hat{\mu}_r$ and $\hat{\sigma}_r^2$ respectively become

$$M_r(u) = \frac{H_r(u)}{N_r(u)} \quad S_r(u) = \frac{G_r(u)}{N_r(u)} - M_r^2(u).$$

The cost function to be minimized is then

$$J(u) = \mathcal{E}(u) + \frac{N_{in}(u)}{2} \log S_{in}(u) + \frac{N_{out}(u)}{2} \log S_{out}(u).$$

The derivative of the second and third terms with respect to $u_{q,k}$ has the form

$$\frac{1}{2} \frac{\partial N_r(u)}{\partial u_{q,k}} + \frac{1}{2} \log S_r(u) + \frac{1}{S_r(u)} \left[ \frac{\partial G_r(u)}{\partial u_{q,k}} - S_r(u) \frac{\partial N_r(u)}{\partial u_{q,k}} \right] - 2M_r \frac{\partial H_r(u)}{\partial u_{q,k}} + M_r^2(u) \frac{\partial N_r(u)}{\partial u_{q,k}}.$$

At first glance this appears a very complex derivative to compute. However each of the partial derivatives has the form given in equation 3.11 with the argument $F$ replaced by $I^2, I$ or 1, and can therefore be obtained by taking the forward transform of the appropriate one dimensional functions.

The other terms in this expression are composed of simple algebraic operations on integrals of $I^2, I$ or 1 over the domains $\theta_{in}(u)$ and $\theta_{out}(u)$. Again using Green’s theorem (equation 3.15), these can be reduced to one dimensional integrals, along the contour, of the functions

$$I_{p,1}(x_1, x_2) = \frac{1}{2} \int_0^{x_2} I^p(x_1, t)dt,$$

$$I_{p,2}(x_1, x_2) = \frac{1}{2} \int_0^{x_1} I^p(t, x_2)dt,$$

for $p = 0, 1, 2$. Specifically

$$\int_{\theta(u)_{in}} I^p(x)dx = \int_0^1 I_{p,1}(\theta(t, u))\hat{\theta}_1(t, u) + I_{p,2}(\theta(t, u))\hat{\theta}_2(t, u)dt, \quad p = 0, 1, 2$$

The functions $I_p$ do not depend on $u$ and can be pre-calculated and stored once and for all at the start of the computation. This leads to the following algorithm.

**Algorithm 3.2: Deformable contour: on-line parameter estimation**

1. Compute $I_{p,j}(x_1, x_2)$, $p = 0, 1, 2 \quad j = 1, 2$ (Simple summation), and total integrals of $I^p$ over entire domain: $T_p, p = 0, 1, 2$.
2. Initialize $u_{q,0} = \{u_{q,k,0}, k = 0, \ldots, d\}, q = 1, 2$. Set $m = 0$.
3. Calculate $\theta_1 = \Psi^{-1}(u_{1,n}), \theta_2 = \Psi^{-1}(u_{2,n})$, and $\hat{\theta}_1, \hat{\theta}_2$.
4. Calculate
   
   Integrals:
   
   $N_{in} = \sum_{i=0}^{n-1} I_{0,1}(\theta(t_i))\hat{\theta}_1(t_i) + I_{0,2}(\theta(t_i))\hat{\theta}_2(t_i)$,
   
   $H_{in} = \sum_{i=0}^{n-1} I_{1,1}(\theta(t_i))\hat{\theta}_1(t_i) + I_{1,2}(\theta(t_i))\hat{\theta}_2(t_i)$,
Figure 3.6: Top. Initial curves. Bottom: Final detection. Left and Middle: Gaussian model with equal variances and means estimated off-line, $\mu_{in} = 0, \mu_{out} = .2$ (Pixel values are scaled between 0 and 1). Right: Gaussian model with on-line parameter estimation.

Transforms: $\psi^{(1)} = \Psi(I\varphi(\theta)\dot{\theta}_1)$, $\psi^{(2)} = -\Psi(I\varphi(\theta)\dot{\theta}_2)$, $p = 0, 1, 2$.

5. Set $d_{q, r, k} = \log(\sigma^2_k)v^{0, q}_k + \frac{1}{\sigma^2_k}[v^{2, q}_k - \sigma^2_kv^{0, q}_k - 2\mu_r v^{0, q}_k + \mu^2_r v^{0, q}_k]$

$q = 1, 2, k = 0, \ldots, d$ and $r = in, out$.

6. Set $u_{q, k, m+1} = u_{q, k, m} - \Delta \cdot \lambda_k u_{q, k, m} + d_{q, in, k} + d_{q, out, k}$

$q = 1, 2, k = 0, \ldots, d$.

7. If a stopping criterion is satisfied exit otherwise, $m \leftarrow m + 1$ go to 3.

The outcomes for the two approaches, off-line and on-line parameter estimation, are compared in figures 3.6 and 3.7. We see that if the parameters are fixed off-line at unreasonable values the first approach yields erroneous results, whereas on-line parameter estimation is successful at finding the contour of the heart ventricle. On the other hand if the on-line method is initialized at a starting point with higher interior mean than that of the desired region it may converge to something entirely wrong, whereas the first method, driven by the correct parameters, obtains a good detection.

### Bibliographical notes and discussion

The idea of deformable contours goes back to Grenander (1970) and Grenander (1978). In Grenander et al. (1991) the details of the implementation are described together with theoretical aspects of the statistical
Figure 3.7: Top: Initial curves. Bottom: Final detections. Left - using fixed parameters. Gaussian model with equal variances, $\mu_{\text{in}} = 0$, $\mu_{\text{out}} = .3$ (as opposed to $\mu_{\text{out}} = .2$ in figure 3.6). The contour grows beyond the boundaries of the ventricle chamber. Right: Gaussian model with on-line parameter estimation, initialized outside the ventricle.

formulation. The curve there is parameterized directly in terms of the locations of the $n$ points, also known as the lattice parameterization. This can be viewed as one extreme example of the spectral parameterization, where the basis functions are shifts of some ‘smoothed delta’ function. In this case all elements of the basis are of equal support and in order to enforce smoothness it is necessary to introduce penalties in terms of sums of squared differences of the point locations or, in the continuum formulation, integrals of squared derivatives of some order. In this parameterization one looses a natural way to proceed from coarse to fine. Starting with a small number of points and gradually increasing their number is a possibility. However it is then necessary to determine the type of interpolation to use between the point positions in order to define the continuous curve, and how additional points are placed along the curve. Still, most formulations in the literature appear in this form. The idea of on-line estimation of the parameters appears in Zhu & Yuille (1996), and worked out in further detail in Chesnaut et al. (1999).

A deformable contour model, called ‘snakes’, based only on an edge data term was originally proposed in Kass et al. (1987). However using only an edge data model is problematic. The contours tend to perform only local adjustments unless artificially forced by additional terms in the cost function to grow or shrink, see Cohen (1991). In such models the initial contour shown for example in figure 3.5 would have trouble crossing the boundary encountered by the top part of the curve on the lower part of the ventricle. When a specific data model is expected in the interior and exterior, and the coarse to fine algorithm is implemented, the curve is able to find the correct region.

Note that when initialized with a small contour the algorithms presented here can be viewed as smoothed region growing algorithms. See for example experiment of figure 3.2. The gradient of the data term is locally looking for directions in which more pixels satisfying the ‘inside model’ can be found, and the penalty term is constraining this growth to be smooth. Region growing is a very old methodology in pattern recognition.
see for example Duda & Hart (1973) or Haralick & Shapiro (1992). In Zhu & Yuille (1996) there is further discussion on the relation between region growing and deformable contours. In this work several contours are grown simultaneously using a systematic model incorporating several closed regions, a prior on the number of regions and on the shape of the region contours. The data model is based on conditional independence of the gray level intensities in the different regions. There is also the option to merge adjacent regions. Such region growing techniques can hardly be viewed as object detection, rather as image segmentation tools. On the other hand when the deformable contour has a particular shape which is more or less preserved during the detection process, there is meaning to the match obtained between the model points on the curve and those in the image, as illustrated by the enumerated points along the model curve and matched curve in figure 3.5.

One of the problems of the algorithm presented above is that there is no inherent mechanism to prevent the self-intersection of the deforming curve, moreover there is no straightforward way to extend the work to deformable 2-d surfaces embedded in 3-d. An interesting computational approach which overcomes these limitations can be found in the level set methods Malladi et al. (1995), Caselles et al. (1997), Caselles et al. (2000). The idea is to view the deforming contour as the 0-level set of a ‘virtual’ function on the entire 2d domain. The gradient descent motion of the curve translates into an evolution of the virtual function based on a time dependent partial differential equation. At each step the estimated curve corresponds to the 0-level set of the evolved virtual function. There is never any need to define the virtual function in the entire domain. In numerical implementations it suffices to define it locally around the current estimated curve. An interesting advantage of this approach is that the topology of the curve can change in a natural way. A smooth evolution of the virtual function can produce at some point a change from one closed curve to two or more. These models have all been implemented using an edge data model and inherit the drawbacks of this type of data model. Also in the presence of noisy data their increased topological flexibility can become a liability, since it allows the algorithm to produce multiple isolated curves, instead of finding the one region of interest.

There are numerous other papers in the literature on deformable contours. The variations involve different data models, different forms of curve parameterization and different forms of smoothing penalties on the curves. See for example Chuang & Kuo (1996), Cohen (1991), Cohen et al. (1992), Cohen & Cohen (1993), Figueiredo & Leitao (1992). In Blake & Yuille (1992) several papers describe the use of deformable contours for tracking a contour in time. These ideas are further developed in Blake & Issard (1998) where a statistical model for the curve motion replaces the prior we have used in static images.

Attempts at training the parameterization of the deformations according to the particular shape and its characteristic deformations can be found in Cootes & Taylor (1992). The \( n \) points of the instantiation are marked by the user on training images registered to a fixed scale and location, to produce a \( 2n \) dimensional vector for each training image. The covariance matrix of this data is computed and the eigenvectors, i.e. principal components, of this matrix are used as the basis with which to expand the curve. Most typical variations of the curve around the mean will require only a small number of basis elements with non-zero coefficients. This is an appealing approach, although the resulting basis will not come with a fast forward and backward transform and hence may be somewhat less efficient. There is also the risk that the covariance matrix and resulting eigenvectors are too dedicated to the training data and result in a prior which does not generalize well to other instances of the object, see Wang & Staib (2000) for a related discussion.

There are certain inherent problems encountered by all implementations, the main one being the initialization of the contour, as clearly emerges from figure 3.6. Some attempts have been made to directly compute a global optimum of a cost function similar to the inside-outside model described above, see Jermyn & Ishikawa (1999), Ishikawa & Geiger (1999), using minimal cut methods on graphs. These methods are much slower than the iterative algorithms described here. Moreover they find the global optimum, over the entire image, of a cost function which is very generic and hence this optimum may not be anywhere near the desired object.
The pose initialization problems, the sensitivity of the deformable contours to noise, and the need to derive more specific data models motivate the algorithms developed in Chapters 6 through 8.
Chapter 4

1-d Models: deformable curves

The deformable contour model described in Chapter 3 is based on a non-linear cost function which is optimized using a gradient descent method. At best the algorithm will reach a satisfactory local minimum. It is easily confused by clutter. If the interior of the object, or the exterior, are not homogeneous the data models for the deformable contour are grossly incorrect and the minima are nowhere near the desired object contours.

An alternative approach involves more detailed modeling of the data along the curve. The template is again defined in terms of a sequence of points \( Z = (z_1, \ldots, z_n) \) on the reference grid. The instantiation is described directly in terms of a sequence \( (\theta_1, \ldots, \theta_n) \) of locations in the image grid \( L \), and the constraints are explicitly defined in terms of a set \( \Theta \subset L^n \) and a prior \( P(\theta) \) penalizing non-smooth deformations of the model curve. The data model is again based on a conditional independence assumption. In contrast to the previous chapter the problem is no longer considered in the continuum, rather the discrete aspect is emphasized in the computational methods. The cost function has the form of a sum of costs on pairs or triples of consecutive points, and therefore lends itself in principle to global optimization. One method of optimization is dynamic programming. The other is a tree based optimization technique from Geman & Jedynak (1996). It should be noted that in order for this approach to be computationally feasible, in terms of time and memory requirements, it is necessary to introduce some hard constraints on the locations of the \( n \) points as detailed below. In other words some user initialization is still required although the algorithms are much less sensitive to the initial configuration.

4.1 Statistical model

Here we use a transform of the gray level pixel data into a vector of binary local image features \( X_a(x), a = 1, \ldots, A \), and write \( \hat{I}(x) = (X_a(x); a = 1, \ldots, A) \). The main advantage is that such features can be chosen to be robust to monotone gray level transformations, and changes in the contrast of the data along the curve will not affect the detection. Otherwise put such features are photometrically invariant. This is in contrast to data models of the previous chapter which assumed a fixed mean intensity inside the object. Such data models would be very sensitive to global changes in the range of gray level values. Furthermore the discrete nature of the transformed data allows for simple estimation of model parameters.

Local features

We assume the curve can either traverse a pixel \( x \) at one of \( A \) different angles \( a\pi/A, a = 1, \ldots, A \), in which case we write \( \text{ang}(x) = a \); or no curve traverses the pixel in which case we write \( \text{ang}(x) = \phi \). For example take
Figure 4.1: The four local feature types detected on the axial MRI brain scan shown at the top right of figure 4.3, we use \( A = 4, \mu = 3, \nu = 1 \).

\( A = 4 \) and assume that locally each curve is either horizontal, vertical or at \( \pm 45 \) degrees. The notion of curve angle is quite loose, and covers quite a wide range. The feature \( X_a \) is expected to be ‘on’ at \( x \), i.e. \( X_a(x) = 1 \), if \( \text{ang}(x) = a \). We list here two possible definitions \( \hat{X}_a \) and \( \tilde{X}_a \), for these features, many others exist. In the experiments below we use the conjunction of these two conditions, namely \( X_a(x) = \hat{X}_a(x) \cdot \tilde{X}_a(x) \). If the curve is expected to be ‘ridge-like’, and say, brighter than its surroundings, define

\[
\hat{X}_a(x) = 1, \quad \text{if} \quad I(x) > I(x + \mu_a) \quad \text{and} \quad I(x) > I(x - \mu_a),
\]

(4.1)

where \( \mu_a \) is the vector of length \( \mu \) pixels in the direction orthogonal to \( a \), for some small \( \mu \).

If the curve can be brighter or darker than its surroundings but relatively constant in intensity we require the intensity differences within the curve to be smaller than those between the pixels on the curve and those alongside it.

\[
\tilde{X}_a(x) = 1, \quad \text{if} \quad |I(x + \nu_a) - I(x)| < \min \left( |I(x) - I(x + \mu_a)|, |I(x) - I(x - \mu_a)| \right)
\]

(4.2)

where \( \nu_a \) is a vector of \( \nu \) pixels in the direction of \( a \). In figure 4.1 we show the points obtained using the conjunction of conditions 4.1 and 4.2, on an axial MRI brain scan. The original image can be found in the top right panel of figure 4.3.
The likelihood

Clearly the probability that \( X_\alpha(x) = 1 \) will tend to be larger if a curve of angle \( \alpha \) passes through \( x \), i.e. \( \text{ang}(x) = \alpha \). For angles \( \alpha \) which are different from \( \alpha \) we would expect the probability that \( X_\alpha(x) = 1 \) to be smaller. Denote these probabilities \( p_{\alpha,a}, \alpha, a = 1, \ldots, A \). Finally if no curve passes through the neighborhood of \( x \), i.e. \( x \) is a ‘background’ pixel, the probability of \( X_\alpha(x) = 1 \) is denoted \( p_b \) and is the same for all \( \alpha \). Assume that given that the curve passes through the pixel \( x \) at angle \( \alpha \) the variables \( X_\alpha(x), a = 1, \ldots, A \) are independent, and given no curve passes through the pixel they are also independent. The conditional distribution of \( \hat{I}(x) \) at pixel \( x \) given \( \text{ang}(x) = \alpha \) is then

\[
P(\hat{I}(x)|\text{ang}(x) = \alpha) = \prod_{a=1}^{A} p_{\alpha,a}^{X_\alpha(x)} (1 - p_{\alpha,a})^{1-X_\alpha(x)}
\]  

(4.3)

and given no curve passes through the pixel

\[
P_b(\hat{I}(x)) = P(\hat{I}(x)|\text{ang}(x) = \phi) = \prod_{a=1}^{A} p_b^{X_\alpha(x)} (1 - p_b)^{1-X_\alpha(x)}.
\]

(4.4)

Given an instantiation \( \theta = (\theta_1, \ldots, \theta_n) \) of the curve, we assume the data \( \hat{I}(x), x \in L \) to be conditionally independent. Let \( L_i \) be the pixels on the segment connecting the points \( \theta_i, \theta_{i+1} \) and let \( L(\theta) \) be the union of the sets \( L_i \). Also let \( \alpha_i \) denote the angle of the \( i \)’th segment. Associated to each \( x \in L(\theta) \) there is a specific angle \( \text{ang}(\theta, x) \) - the angle of the segment to which the pixel \( x \) belongs. The conditional distribution of \( \hat{I} \), on the entire lattice, given the instantiation \( \theta \) is then

\[
P(\hat{I}|\theta) = \prod_{x \in L(\theta)} P(\hat{I}(x)|\text{ang}(\theta, x) \cdot \prod_{x \notin L(\theta)} P_b(\hat{I}(x))
\]

\[
= \prod_{i=1}^{n} \prod_{x \in L_i} P(\hat{I}(x)|\alpha_i) \prod_{x \notin L(\theta)} P_b(\hat{I}(x))
\]

(4.5)

Dividing 4.5 by \( \prod_{x \in L} P_b(\hat{I}(x)) \), which can be interpreted as the probability of \( \hat{I} \) given no curve is present, and does not depend on \( \theta \), and substituting 4.4 and 4.3, we obtain a likelihood ratio of the form

\[
\frac{P(\hat{I}|\theta)}{P_b(\hat{I})} = \prod_{x \in L(\theta)} \frac{P(\hat{I}(x)|\text{ang}(\theta, x))}{P_b(\hat{I}(x))}
\]

\[
= \prod_{i=1}^{n} \prod_{a=1}^{A} \left[ \frac{p_{\alpha,a}}{p_b} \right]^{N_{ia}} \left[ \frac{1 - p_{\alpha,a}}{1 - p_b} \right]^{(N_i - N_{ia})}
\]

(4.6)

where \( N_{ia} \) is the number of times \( X_\alpha(x) = 1 \) along the segment \( L_i \) and \( N_i \) denotes the total number of pixels along this segment. The log-likelihood is thus up to a constant given by a sum of functions of the counts along the segments of the curve.

\[
\log \frac{P(\hat{I}|\theta)}{P_b(\hat{I})} = \sum_{i=1}^{n} \sum_{a=1}^{A} \left[ N_{ia} \log \left( \frac{p_{\alpha,a}(1 - p_b)}{p_b(1 - p_{\alpha,a})} \right) + N_i \log \left( \frac{1 - p_{\alpha,a}}{1 - p_b} \right) \right].
\]

(4.7)

We are again making strong conditional independence assumptions which are clearly a gross simplification. However the data model is simple and transparent and does depend on the angle of the curve in a direct
CHAPTER 4. TRACKING

way. Furthermore, as in the previous chapters, the model is created with the computational task in mind. The log-likelihood is simply a linear function of the counts. In this equation the dependence on the data \( \hat{I} \) is through \( N_{ia} \). It is also useful to note for later computational considerations that the log-likelihood can be written in the form

\[
\sum_{i=1}^{n} \psi(\hat{I}, \theta_i, \theta_{i+1}),
\]

where the functions \( \psi \) depend only on the two consecutive points \( \theta_i, \theta_{i+1} \) and on the data \( \hat{I} \) along the segment connecting them.

Under this data model the maximum likelihood estimates of the parameters are obtained from training sample proportions. Take training subimage samples from each of the categories: \( \alpha = 1, \ldots, A \). For each \( \alpha \) obtain the proportion for which \( X_\alpha(x) = 1, a = 1, \ldots, A \) to produce an estimate \( \hat{p}_\alpha \). An estimate of \( p_b \) is obtained from subimages with no curve, estimating one pooled probability for \( P(X_\alpha(x) = 1 | \text{ang}(x) = \phi) \), for all \( \alpha \)’s.

The model can be simplified by setting \( p_{\alpha,a} = p_c \) if \( \alpha = a \) meaning that \( P(X_\alpha(x) = 1 | \text{ang}(x) = \phi) = p_c \) for any \( a = 1, \ldots, A \) and \( p_{\alpha,a} = p_b \) if \( \alpha \neq a \), meaning that the probability of \( X_\alpha(x) = 1 \) if \( \text{ang}(x) \neq \phi \) is the same as the background probability. The likelihood ratio then has the simpler form

\[
\frac{P(\hat{I} | \theta)}{P_b(\hat{I})} = \prod_{i=1}^{n} \left[ \frac{p_c}{p_b} \right]^{N_{ia}} \frac{(1-p_c)}{(1-p_b)}^{(N_{ia}-N_{ia})}.
\]

In this case parameter estimation also simplifies. Estimate one parameter \( p_c = p_{\alpha,a} \) for all \( a = 1, \ldots, A \), pooling together all subimage samples containing a curve of any angle. For each \( a \) let \( n_a \) be the number of training subimages labeled with angle \( a \), and let \( n_{a,1} \) be the number of these for which \( X_\alpha(x) = 1 \). Then estimate \( p_c \) as

\[
p_c = \frac{\sum_{a=1}^{A} n_{a,1}}{\sum_{a=1}^{A} n_a}.
\]

The parameter \( p_b \) is estimated from training subimages identified as not having any curve.

Training samples are either obtained by hand with the user pointing out pixels with curves of the \( A \) different angles and pixels with no curve. Alternatively one can start with initial parameter settings, detect the curve, and use pixels on and off the detected curve to update the parameters.

The prior and the posterior

The curve is parameterized directly through the locations of the \( n \) points \( \theta_1, \ldots, \theta_n \). It is important to include a prior penalizing irregular non-smooth curves. This can have a variety of forms. For example a penalty on high curvature can be written as

\[
P(\theta_1, \ldots, \theta_n) \propto \exp \left[ - \left( \sum_{i=1}^{n-2} \phi_i(\theta_i, \theta_{i+1}, \theta_{i+2}) \right) \right],
\]

where \( \phi_i(\theta_i, \theta_{i+1}, \theta_{i+2}) \) is the modulus of the difference between the angle of the segment \( \theta_i, \theta_{i+1} \), and that of the segment \( \theta_{i+1}, \theta_{i+2} \). Each such term depends on a triple of consecutive points and is entirely scale invariant. Such penalties are useful when there are no particular prior assumptions on the shape of the curve. If the model or template has a particular shape and we do not expect significant variations in scale or rotation, the functions \( \phi_i \) can be simplified to depend only on consecutive pairs. In the examples below we
compare angles and lengths between a pair of consecutive points in the instantiation and the corresponding pair in the model sequence:

\[
\phi_i(\theta_i, \theta_{i+1}) = A|\text{ang}(\theta_{i+1} - \theta_i, z_{i+1} - z_i)| + B|\log(|\theta_{i+1} - \theta_i|/|z_{i+1} - z_i|)|. 
\] (4.10)

and

\[
P(\theta_1, \ldots, \theta_n) \propto \exp \left[ - \left( \sum_{i=1}^{n-1} \phi_i(\theta_i, \theta_{i+1}) \right) \right].
\] (4.11)

This is a simple prior which independently penalizes deviations in angle and length of individual segments of the deformed curve from their counterparts on the template curve. The higher probability instantiations will tend to have a shape similar to the model curve \((z_1, \ldots, z_n)\).

Putting the data model from equation 4.8 together with the prior of equation 4.11 we write a negative log-posterior of the form

\[
J(\theta) = -\log P(\theta_1, \ldots, \theta_n) = \sum_{i=1}^{n-1} \Phi_i(\theta_i, \theta_{i+1}) + C 
\] (4.12)

where

\[
\Phi_i(\theta_i, \theta_{i+1}) = \phi_i(\theta_i, \theta_{i+1}) - \psi_i(\tilde{I}, \theta_i, \theta_{i+1}), \quad i = 1, \ldots, n - 1,
\]

with \(\psi_i\) defined in equation 4.8 and \(\phi_i\) defined in 4.10.

### 4.2 Computation: dynamic programming

The structure of the cost function - a sum of terms each depending on two consecutive variables - lends itself to efficient minimization using dynamic programming. Each point \(\theta_i\) along the curve is assigned a state space \(S_i \subset L\) of possible values. It is not necessary to try all possible \(n\)-tuples of points from \(S_i, i = 1, \ldots, n\). Rather the computation reduces to trying all possible pairs of points of consecutive indices, \(i, i + 1\) for \(i = 1, \ldots, n - 1\). Even with this reduction it is impractical to assume that \(S_i\) is the entire image lattice. However if some hard constraints on the location and variability of the curve are introduced the computation becomes tractable.

One approach to limiting the state space is to assume that the first and last points are given within small neighborhoods \(S_1, S_n\) in the image. Then using some heuristic determine regions \(S_2, \ldots, S_{n-1}\) such that for any reasonable configuration \(\theta_1, \ldots, \theta_n\) we would have \(\theta_i \in S_i, i = 1, \ldots, n\). Alternatively set an initial curve \(\theta^{(0)}\), determined by an affine map \(A\) applied to the model configuration, i.e. \(\theta_i^{(0)} = A z_i\), and define \(S_i\) as neighborhoods of the points \(\theta_i^{(0)}\), of some size.

Dynamic programming is based on the following simple observation. Let \(J^{(0)}(\theta) = J(\theta)\) as defined in equation 4.12. For each point \(\theta_{n-1} \in S_{n-1}\) define

\[
\theta_{n-1}^{*} = \arg \min_{S_n} \Phi_{n-1}(\theta_{n-1}, \theta_n),
\]

\[
f^*(\theta_{n-1}) = \min_{S_n} \Phi_{n-1}(\theta_{n-1}, \theta_n).
\]

Then define

\[
J^{(1)}(\theta_1, \ldots, \theta_{n-1}) = \sum_{i=1}^{n-2} \Phi_i^{(1)}(\theta_i, \theta_{i+1}),
\]
CHAPTER 4. TRACKING

with $\Phi_{i}^{(1)} = \Phi_{i}$ for $i = 1, \ldots, n - 3$ and

$$\Phi_{n-2}^{(1)}(\theta_{n-2}, \theta_{n-1}) = \Phi_{n-2}(\theta_{n-2}, \theta_{n-1}) + f^*(\theta_{n-1}).$$

A simple argument shows that $(\theta_1^*, \ldots, \theta_{n-1}^*)$ minimizes $J^{(1)}$ if and only if

$$(\theta_1^*, \ldots, \theta_{n-1}^*, \theta_n^*(\theta_{n-1}^*))$$

minimizes $J^{(0)}$. If $\theta_n^*(\theta_{n-1}^*)$ is stored for all values of $\theta_{n-1} \in S_{n-1}$ as well as the values $f^*(\theta_{n-1})$ then the $n$-dimensional problem has been reduced to an $n - 1$ dimensional problem of the same form, although one of the functions has been modified. These reductions continue until we are left to find $\theta_2^*(\theta_1)$ for every possible $\theta_1 \in S_1$. This leads to the following algorithm.

Algorithm 4.1: Curve detection with dynamic programming.

1. For $i = 1, \ldots, n$ define arrays $H_i$ indexed by elements $x \in S_i$, and with 5 columns: the first two giving the coordinates of $x$ are merely for convenience. The last three columns get updated as the algorithm proceeds.

2. Set $H_n(x, 5) = 0$ for all $x \in S_{n-1}$. Set $i = n - 1$.

3. While $i > 0$, do

   For every $x \in S_i$ find the point $z^*(x)$ in $S_{i+1}$ which minimizes $\Phi_i(x, z) + H_{i+1}(z, 5)$. Store the coordinates of $z^*(x)$ in columns 3 and 4 of the $x$ entry in $H_i$. Column 5 is used to store the value of $f^*(x) = \Phi_i(x, z^*(x)) + H_{i+1}(z^*(x), 5)$. Set $i = i - 1$.

4. Loop over the array $H_1$ and find the row with lowest value $\Phi^*$ of column 5. Let $\theta_1^*$ be the point given by the first 2 columns of that row. Set $i = 1$.

5. While $i < n - 1$, do

   Set $\theta_i^*$ to be the point given in columns 3 and 4 of $H_i$ at the row indexed by $\theta_1^*$. Set $i = i + 1$.

6. The configuration $\theta_i^*, i = 1, \ldots, n$ is the optimal configuration satisfying $\theta_i \in S_i$ and the associated cost is $\Phi^*$.

If the sets $S_i$ are large the computation and memory requirements of the algorithm get out of hand. For example for square regions of size $20 \times 20$, which is the size used below each step takes approximately 250 milliseconds on the PENTIUM III 700 Mhz. However computation here grows quadratically with the size of the state space. So for example using $10 \times 10$ regions the time reduces to 10 milliseconds per iteration. In figure 4.2 we show some of the stages of the dynamic programming for detecting an $\mathcal{E}$ in a cluttered scene. The initial curve is shown in the left panel. The optimal result of the first two stages is shown in the second panel. This would be the result if the model consisted only of the first two segments corresponding to the bottom ending of the $\mathcal{E}$. This optimal solution is clearly misplaced, and is corrected in the next stage where the optimum corresponding to the first three segments of the model is shown. A few additional stages are shown together with the final detection.

The outcome of dynamic programming for detecting the deformable curve algorithm on an axial MRI brain scan is shown in figure 4.3. The aim is to identify the exterior boundary which corresponds to the scalp. The initial curve, which has the form of a semi-circle, is shown in black, and the final curve is shown in white. The regions $S_i$ were $21 \times 21$ neighborhoods of each of the points along the initial black curve. In the lower left-hand panel the final segment of the curve is off the correct path, most likely because the
Figure 4.2: Stages of dynamic programming. The initial curve which determines the regions $S_i$ is shown in the top left panel. Then at four stages of the dynamic programming the corresponding optimal curve is shown. Final detection is in the bottom right panel.

neighborhood around the last point in the initial curve did not intersect the scalp. It is important to note that there is significant clutter in these images which is not easily visible. This can be seen in figure 4.1 where plenty of hits of the four local features are found inside as well as outside the scalp. These local features are the only input to the detection algorithm, and due to their photometric invariance, they detect ridges even in very low contrast regions.

When there is no particular prior shape to the curve and the only constraint is that of smoothness, priors involving consecutive triples are essential. Even with prior shape information such constraints may be needed. Curves detected with the simpler prior can have sharp turns since there is no constraint on the curvature. The dynamic programming procedure is similar but requires computations over triples of consecutive points. The computational and memory requirements increase and at times it is necessary to implement heuristics to prune the lists of optimal states to avoid an explosion in the computation time. In Chapter 7 we return to dynamic programming as a technique for matching a sparse model with no prior initialization. There we indeed use functions depending on triples of points in the model and provide the algorithmic details. A more efficient alternative to dynamic programming is described in the next section in the case where the curves are assumed to have low curvature.

4.3 Global optimization on a tree structured prior

When the curve changes direction very slowly we can assume that the change in angle between two consecutive segments is limited to three possible values $-\beta, 0, \beta$, where $\beta = \pi/A$ for some large $A$. The distance $N$ between consecutive points is taken to be constant. If the first two points $\theta_1, \theta_2$ are given, the path of the curve can be described as a sequence of symbols from the space $\{-1, 0, 1\}$ according to whether the next segment turns $-\beta, 0$ or $\beta$ degrees. The entire space of possible curves emanating from the initial segment can be represented as a ternary tree $T$, of depth $n$ with each node representing a possible segment. The
Figure 4.3: Examples of deformable curve detection on an MRI brain scan: In black is the original curve used to define the search regions, and in white are shown the stages of the dynamic programming.
CHAPTER 4. TRACKING

root node of the tree is the segment \( \theta_1, \theta_2 \) which is fixed. Each node is connected to three children nodes representing the three possible segments following it. A curve is represented by a path from the root node to a terminal node of the tree, and is thus completely determined by that terminal node.

Starting at the initial segment, the path on the tree up to a node \( t \) provides a sequence of turns on consecutive segments of angle \( \beta, 0 \) or \(-\beta\) all of equal length. Thus each interior node \( t \) in the tree corresponds to a physical segment \( L_t \) in the image, and has an associated angle \( \alpha_t \). If the initial segment is assigned angle 0 the angle of \( \alpha_t \) can be any multiple of \( \beta \). The node \( t \) also determines a set \( \Theta_t \subset \Theta \) of curves passing through it. We freely interchange the use of \( \theta \) for a terminal node of the tree, the path it determines in the tree, and the corresponding physical path in the image. Thus the set of possible instantiations \( \Theta \) is given by the set of paths or terminal nodes of the tree. Note that certain points in the image can be traversed by two different segments represented in the tree \( T \). This is ignored in the model developed below.

The data model

We employ the simpler model of equation 4.9, where \( P(X_a(x) = 1|\text{ang}(x) = a) = p_c \) and \( P(X_a(x) = 1|\text{ang}(x) \neq a) = p_b \), for all \( a = 1, \ldots, A = \pi/\beta \). Let \( Y_t = N_{\theta_t} \) denote the counts along the segment \( L_t \), of the corresponding feature \( X_{\alpha_t} \). Recall that \( \alpha_t \) is the angle of \( L_t \). The likelihood ratio of the transformed image data is now written as

\[
\frac{P(\hat{I}|\theta)}{P_{\theta}(I)} = \prod_{t \in \theta} [p_c/p_b]^{Y_t}(1-p_c)/(1-p_b)]^{N-Y_t},
\]

where \( N \) is the fixed number of points along each segment. Given the true curve is \( \theta \) this data model expects a higher count of the feature \( X_{\alpha_t} \) on the segments \( L_t \) for \( t \in \theta \) than on background segments, \( t \notin \theta \). Multiply the above expression by \( \prod_{t \in \mathcal{T}} p_b^{Y_t}(1-p_b)^{N-Y_t} \), which is constant, and write the likelihood of the data, only at pixels corresponding to segments of the tree, conditional on the curve as

\[
P(\hat{I}(x), x \in \cup_{t \in \mathcal{T}} L_t | \theta) = C \cdot \prod_{t \in \theta} p_c^{Y_t}(1-p_c)^{N-Y_t} \cdot \prod_{t \notin \theta} p_b^{Y_t}(1-p_b)^{N-Y_t} \quad (4.13)
\]

Through some simple algebraic manipulation we have reformulated a restricted likelihood only in terms of the data along \( \cup_{t \in \mathcal{T}} L_t \), namely the collection of image segments corresponding to the tree nodes. Recall that we are ignoring the fact that these segment may have non-trivial intersections. This likelihood depends on the data along each segment \( L_t \) only in terms of the count \( Y_t \). Moreover in our particular setting the distribution of \( Y_t \) is Binomial \( B(N, p_c) \) or \( B(N, p_b) \) depending on whether the curve passes through \( t \) or not. Since the data along the segments is still considered independent given the curve, we can write a simplified likelihood of the counts \( Y_t \) along the segments as

\[
P(Y_t, t \in \mathcal{T} | \theta) = \prod_{t \in \theta} P_1(Y_t) \prod_{t \notin \theta} P_0(Y_t), \quad (4.14)
\]

where \( P_0 \) and \( P_1 \) are the Binomial distributions \( B(N, p_b) \) and \( B(N, p_c) \) respectively. A simple computation shows that the relative weighting of the likelihood for two curves will not change using 4.13 or 4.14. Since the prior on the tree is uniform the posterior on curves given the counts along the segments of the tree is again

\[
P(\theta|Y_t, t \in \mathcal{T}) = C \cdot \prod_{t \in \theta} P_1(Y_t) \prod_{t \notin \theta} P_0(Y_t). \quad (4.15)
\]

Thus in the current setting one can start with a model which only considers the image data along the tree of paths. Assume conditional independence of the counts \( Y_t \) given \( \theta \) corresponds to the true curve, with
some distribution $P_1$ for the counts on segments in $\theta$, and another distribution $P_0$ for the counts on segments in the tree not in $\theta$. This is a somewhat weaker assumption than conditional independence of the data at each pixel, and is the approach taken in Geman & Jedynak (1996). One can then use a non-parametric form of $P_1$ and $P_0$ as opposed to the Binomial model. The distribution $P_1$ is estimated from a sample of subimages in which a curve is passing through the center. If the angle in a subimage is $\alpha$ obtain $N_\alpha$ along the appropriate segment. These counts are then pooled into a histogram to estimate the distribution $P_1$. On subimages with no curve find the counts $N_a$ for each of the angles $a = 1, \ldots, A$ and pool these into a histogram to obtain an estimate of $P_0$.

Before proceeding to describe the details of this algorithm we make a brief digression to define the notion of entropy, conditional entropy, and mutual information, which will be used in the following sections as well as in Chapter 9 in the context of classification trees. A comprehensive exposition of the theory of entropy also called information theory can be found in Cover & Thomas (1991).

**Entropy**

Given a random variable $X$ with values in a finite discrete set $S$ and with probability distribution $p(s) = P(X = s)$, we define the entropy of $X$ as

$$H(X) = - \sum_{s \in S} p(s) \log p(s). \quad (4.16)$$

Since $x \log x = 0$ if $x = 0$ this sum is well defined even if some of the probabilities are 0. This quantity is one way to express how ‘random’ $X$ is, or the degree of ‘uncertainty’ in $X$. If $X$ is uniform on $S$ then $H(X) = \log(|S|)$. This can easily be seen to be the highest attainable value for any distribution on $S$. Indeed the uniform distribution is the most ‘random’. At the other extreme if $X$ is concentrated at one point, i.e. $p(s) = 1$ for some $s \in S$, then $H(X) = 0$, which is the lowest attainable value since $-x \log x \geq 0$ for $0 \leq x \leq 1$. Note that $H(X)$ depends only on the distribution $p$ and not on the values of $X$, so we can also write $H(p)$.

Let $X_1$ and $X_2$ be two random variables with values in $S$, so that the pair $(X_1, X_2)$ has values in $S \times S$ with a joint distribution $p(s_1, s_2) = P(X_1 = s_1, X_2 = s_2)$. The joint entropy of $X_1$ and $X_2$ is defined in the same way as

$$H(X_1, X_2) = - \sum_{s_1 \in S, s_2 \in S} p(s_1, s_2) \log p(s_1, s_2). \quad (4.17)$$

Let $p_i(s_i) = P(X_i = s_i), i = 1, 2$ denote the marginal distributions of $p$. Let $p(s_1 | s_2)$ denote the conditional probability $P(X_1 = s_1 | X_2 = s_2)$. We can ask what is the randomness in $X_1$ given we know that $X_2 = s_2$, written as the entropy of the distribution $p(\cdot | s_2)$, i.e.

$$H(X_1 | X_2 = s_2) = - \sum_{s_1 \in S} p(s_1 | s_2) \log p(s_1 | s_2). \quad (4.18)$$

this could be greater or less than the original entropy of $X_1$. If we average over all possible values $s_2$ in terms of the distribution $p_2$ we get the conditional entropy of $X_1$ given $X_2$

$$H(X_1 | X_2) = - \sum_{s_2 \in S} p_2(s_2) \sum_{s_1 \in S} p(s_1 | s_2) \log p(s_1 | s_2) = - \sum_{s_1, s_2} p(s_1, s_2) \log p(s_1 | s_2). \quad (4.19)$$

Using the fact that $p(s_1 | s_2) = p(s_1, s_2)/p_2(s_2)$, and that $H(X_2) = - \sum_{s_1, s_2} p(s_1, s_2) \log p(s_2)$, we get the following relation between conditional entropy and joint entropy.

$$H(X_1 | X_2) = H(X_1, X_2) - H(X_2). \quad (4.20)$$
Furthermore using the fact that \(-\log x\) is convex we get
\[
H(X_1) - H(X_1|X_2) = - \sum_{s_1, s_2} p(s_1, s_2) \log \frac{p(s_1)p(s_2)}{p(s_1, s_2)} \\
\geq - \log \left( \sum_{s_1, s_2} p(s_1, s_2) \frac{p(s_1)p(s_2)}{p(s_1, s_2)} \right) \\
= \log 1 = 0.
\]

In other words the conditional entropy of a variable (conditioned on any other variable) is always less than the entropy. On average the randomness in \(X_1\) decreases if the value of \(X_2\) is known. Except when the variables are independent, i.e. \(p(s_1, s_2) = p_1(s_1)p_2(s_2)\), in which case it is easy to see that
\[
H(X_1|X_2) = H(X_1) = H(X_1, X_2) - H(X_2).
\]

Note that by equation 4.20
\[
H(X_1) - H(X_1|X_2) = H(X_1) + H(X_2) - H(X_1, X_2) = H(X_2) - H(X_2|X_1).
\]

The quantity
\[
\Delta(X_1, X_2) = H(X_1) - H(X_1|X_2) = H(X_2) - H(X_2|X_1)
\]

is symmetric in \(X_1, X_2\) and is called the mutual information of \(X_1\) and \(X_2\).

In our context, if we want to obtain a good guess on \(\theta\) in terms of an observation \(X\) we would pick one for which the conditional entropy of \(\theta\) given \(X\) is low, or the mutual information is high. Not much randomness is left in \(\theta\) once \(X\) is observed. This will guide us in the development of the algorithm below.

**Sequential updating of the partial posterior**

Trying to directly find the maximum of the posterior in equation 4.15 is computationally impossible for trees of depth on the order of several tens. The state space is huge and even evaluating \(Y_t\) along each segment \(L_t\) corresponding to a node \(t\) of \(T\) is non-trivial. The idea behind the algorithm described below is to evaluate a sequence of partial posteriors conditioning only on the data at a small subset of nodes of \(T\), as opposed to conditioning on all the data, and evaluating the probabilities of a limited number of coarse subsets of \(\Theta\). Gradually the size of the conditioning set increases and the subsets at which the posterior is evaluated become more refined.

At step \(m\) assume we already have chosen a subtree \(T_m\) rooted at the root node of \(T\), and \(m\) nodes \(t_1, \ldots, t_m \in T_m\) at which \(Y_t\) has been computed, with values \(y_1, \ldots, y_m\). Let \(B_m\) denote the event \(\{Y_{t_1} = y_1, \ldots, Y_{t_m} = y_m\}\). Each \(t \in T_m\) determines a subset \(\Theta_t\) of paths, and we calculate the partial posterior
\[
\pi_t^{(m)} = P(\Theta_t|Y_i = y_i, i = 1, \ldots, m),
\]

for all \(t \in T_m\). Typically there are more than \(m\) nodes in \(T_m\), but the number of segments at which we actually observe \(Y_t\) is given by \(m\) and is increased by one at each step. The subtree \(T_m\) is sequentially updated, and typically the partial posterior in equation 4.23 becomes more and more peaked at some node \(t\) of the current subtree. There is a recursive method to calculate \(\pi_t^{(m+1)}\) in terms of \(\pi_t^{(m)}\).

Moreover, as discussed in the previous section, the next segment to ‘query’, i.e. the choice of \(t_{m+1}\) at which to observe \(Y_{t_{m+1}}\), is chosen as the variable maximizing the mutual information with \(\theta\), given the data already observed, i.e. given the event \(B_m\). At some stage the partial posterior is sufficiently peaked at say \(t^* \in T_m\), the true curve can be assumed to pass through \(t^*\), i.e. \(\theta \in \Theta_{t^*}\), and the search is reinitialized at \(t^*\).
Choosing the next segment to query

Assume \( m - 1 \) segments have been queried, determining an event \( B_{m-1} \), and that the current set \( T_m \) from which to choose \( t_m \) is already determined. We are seeking \( t \in T_m \) for which the mutual information of \( \theta \) and \( Y_t \) given \( B_{m-1} \) is largest. Due to the conditional independence assumptions, the most informative node is identified solely in terms of the current partial posterior \( \pi_t^{(m-1)} \).

For \( 0 \leq \pi \leq 1 \) define the mixture distribution \( P_\pi = \pi P_1 + (1 - \pi) P_0 \) and let

\[
\phi(\pi) = H(P_\pi) - \pi H(P_1) - (1 - \pi) H(P_0),
\]

(4.24)

where \( H(P) \) is the entropy of the distribution \( P \). Since \( -x \log x \) is concave the function

\[
H(P_\pi) = \sum_{k=0}^{N} [\pi P_1(k) + (1 - \pi) P_0(k)] \log[\pi P_1(k) + (1 - \pi) P_0(k)],
\]

is concave in \( \pi \). The rest of the expression in 4.24 is linear in \( \pi \) so that \( \phi \) is a concave function of \( \pi \in [0, 1] \) and has a unique maximum at \( \pi_{\text{max}} \). It depends on the particular distributions \( P_0, P_1 \). The mutual information is maximized at the node \( t \in T_m \) for which \( \pi_t^{(m-1)} \) is closest to \( \pi_{\text{max}} \).

This is seen as follows. The distribution of \( Y_t \) given \( \theta \) depends only on whether the curve defined by \( \theta \) passes through \( t \) (\( \theta \in \Theta_t \)) and therefore the mutual information between \( Y_t \) and \( \theta \) is the same as the mutual information between \( Y_t \) and the indicator function \( 1_{\Theta_t}(\theta) \). Setting \( \pi_t^{(m-1)} = P(\Theta_t \mid B_{m-1}) = P(1_{\Theta_t}, \theta = 1 \mid B_{m-1}) \), and letting \( \Delta \) denote mutual information and \( H \) the entropy, we have

\[
\Delta(Y_t, \theta \mid B_{m-1}) = \Delta(Y_t, 1_{\Theta_t} \mid B_{m-1}) = H(Y_t \mid B_{m-1}) - H(Y_t \mid 1_{\Theta_t}, B_{m-1}).
\]

(4.25)

The conditional distribution of \( Y_t \) given \( B_{m-1} \) can be written as a mixture distribution

\[
P(Y_t = k \mid B_{m-1}) = \pi_t^{(m-1)} P(Y_t = k \mid \Theta_t, B_{m-1}) + (1 - \pi_t^{(m-1)}) P(Y_t = k \mid \Theta_t^c, B_{m-1})
= \pi_t^{(m-1)} P_1(k) + (1 - \pi_t^{(m-1)}) P_0(k),
\]

where the second equality follows from the conditional independence. The second term in 4.25 is rewritten as

\[
H(Y_t \mid 1_{\Theta_t}, B_{m-1}) = H(Y_t \mid \Theta_t, B_{m-1}) \pi_t^{(m-1)} + H(Y_t \mid \Theta_t^c, B_{m-1})(1 - \pi_t^{(m-1)})
= \pi_t^{(m-1)} H(P_1) + (1 - \pi_t^{(m-1)}) H(P_0),
\]

where again the second equality follows from the conditional independence of \( Y_t \) and \( B_{m-1} \) given \( 1_{\Theta_t} \). Therefore the mutual information reduces to

\[
\Delta(Y_t, \theta \mid B_{m-1}) = H((1 - \pi_t^{(m-1)}) P_0 + \pi_t^{(m-1)} P_1) - \pi_t^{(m-1)} H(P_1) - (1 - \pi_t^{(m-1)}) H(P_0)
= \phi(\pi_t^{(m-1)}),
\]

(4.26)

as defined in equation 4.24. We therefore need to pass through all the nodes in \( T_m \) and set \( t_m \) to be the one with \( \pi_t \) closest to \( \pi_{\text{max}} \).
CHAPTER 4. TRACKING

Updating from \( \pi_t^{(m-1)} \) to \( \pi_t^{(m)} \)

Once the next node \( t_m \in T_m \), at which to observe the data is chosen, we update the partial posterior given the larger conditioning set \( B_{m-1} \), for every node in \( T_m \). In the next section we will see how the set \( T_m \) is chosen.

Recall that the partial posterior given \( B_{m-1} \), i.e. \( \pi_t^{(m-1)} \) has already been computed and stored for every node in \( t \in T_m \). Let \( t \) be a terminal node of \( T_m \), since all the nodes \( t_1, \ldots, t_m \) are in \( T_m \), knowing that the curve passes through \( t_i \), i.e. \( \theta_\Theta_t \), we completely determines whether or not each of the nodes \( t_i, i = 1, \ldots, m \) is on the curve or not. Thus \( Y_{t_m} \) is conditionally independent of \( B_{m-1} \) given \( \Theta_t \). We can then write

\[
\pi_t^{(m)} = P(\Theta_t|B_{m-1}, Y_{t_m} = y_m)
\]

\[
= P(Y_{t_m} = y_m|\Theta_t, B_{m-1})P(\Theta_t|B_{m-1})\frac{P(B_{m-1})}{P(B_{m})}
\]

\[
= P(Y_{t_m} = y_m|\Theta_t)\pi_t^{(m-1)}\frac{P(B_{m-1})}{P(B_{m})}.
\]

The first factor can be expressed as

\[
P(Y_{t_m} = y_m|\Theta_t) = \begin{cases} \frac{P_1(y_m)}{P_0(y_m)} & \text{if } t_m \text{ is an ancestor of } t \\ P_0(y_m) & \text{otherwise } t_m \in \theta_t \end{cases},
\]

and the ratio is given by

\[
\frac{P(B_{m})}{P(B_{m-1})} = P(Y_{t_m} = y_m|B_{m-1}, \Theta_{t_m})P(\Theta_{t_m}|B_{m-1})
\]

\[
+ P(Y_{t_m} = y_m|B_{m-1}, \Theta_{t_m}^c)(1 - P(\Theta_{t_m}|B_{m-1}))
\]

\[
= P_1(y_m)\pi_t^{(m-1)} + P_0(y_m)(1 - \pi_t^{(m-1)}).
\]

(4.27)

Thus for any terminal node \( t \in T_m \) the calculation of \( \pi_t^{(m)} = P(\Theta_t|B_m) \) is given entirely in terms of of \( \pi_t^{(m-1)} \), \( \pi_t^{(m-1)} \), and \( P_i(y_m), i = 0, 1 \) all of which are known. For every internal node \( t \) of the tree \( T_m \), write

\[
\pi_t^{(m)} = \pi_t^{(m)} + \pi_t^{(m)} + \pi_t^{(m)},
\]

where \( t_1, t_2, t_3 \) are the three child nodes of \( t \), so that \( \pi_t^{(m)} \) can be recursively updated going from the terminal nodes upwards.

Defining the subtree \( T_{m+1} \)

The partial posterior \( \pi_t^{(m)} \) has been computed for all \( t \in T_m \). The set \( T_{m+1} \) is obtained from the set \( T_m \) keeping in mind the fact that subsequently we will be looking for informative nodes, i.e. nodes with \( \pi_t^{(m)} \) close to \( \pi_{max} \). First include all nodes in \( T_m \). For any terminal node \( t' \in T_m \) satisfying \( \pi_{t'} > \pi_{max} \) add the three children to \( T_{m+1} \). For these children nodes write

\[
\pi_t^{(m)} = P(\Theta_t|B_m) = \frac{1}{3}P(\Theta_{t'}|B_m) = \frac{1}{3}\pi_{t'}^{(m)},
\]

(4.29)

This is due to the uniform prior and the fact that we are conditioning on observed data at nodes in \( T_m \), so that \( t \neq t_1, \ldots, t_m \). On the other hand if \( \pi_t^{(m)} < \pi_{max} \) there is no point in adding the children since their value would be further away from \( \pi_{max} \). Even if \( \pi_t^{(m)} > \pi_{max} \) the value for the three children themselves will have to be less than \( \pi_{max} \) again because of the factor of 1/3 and because \( \pi_{max} \) is typically between .4 and .6. Therefore this extension occurs only for one level, i.e. all new nodes are children of terminal nodes of
**CHAPTER 4. TRACKING**

$T_m$. An important conclusion is that the most informative segment in the entire tree given the information in $B_m$ has to be in the set $T_{m+1}$.

The procedure now repeats. The next query $t_{m+1}$ will be that segment $t$ in $T_{m+1}$ for which $\pi^{(m)}_t$ is closest to $\pi_{max}$. The new probability $\pi^{(m+1)}_t$ is computed from $\pi^{(m)}_t$ for all elements of $T_{m+1}$ and $T_{m+1}$ is extended to $T_{m+2}$. To start we set $T_0$ as the root node and therefore necessarily $T_1$ consists of the root node and its three children, one of which is picked at random as $t_1$. At some stage the partial posterior is sufficiently peaked at some $t^* \in T_m$. We then assume the true curve passes through $t^*$, i.e. $\theta \in \Theta_{t^*}$, and the search is reinitialized at $t^*$.

The entire algorithm is summarized as follows:

**Algorithm 4.2: Curve detection: tree based algorithm** A node $t$ in the tree is an object with variables used to store the following information:

- First and second point of segment: $t.p_1, t.p_2$.
- The count: $t.Y$.
- Pointers to 3 children and parent: $t.ch_1, t.ch_2, t.ch_3, t.par$.
- The current approximate posterior: $t.\pi$.
- Depth in tree: $t.depth$.
- $\pi_{max}$ - The minimum of $\phi$ defined in equation 4.26.
- $TM$ - Pointer to node with $\pi$ value closest to $\pi_{max}$.
- $\Delta$ - Store $|TM.\pi - \pi_{max}|$.
- $TMS$ - Node at which algorithm reinitializes.
- $\pi^*$ - Lower threshold for posterior at node $t$ for reinitialize algorithm from the corresponding segment assuming $t.depth$ is greater then $d_{min}$.

1. Initialize $\pi^*, \pi_{max}, d_{min}$ Get two initial points from user: $x_1, x_2$.
2. Initialize $\Delta = 1, TM = null, TMS = null$.
   Call birth(top)
3. For step = 0 : step$_{max}$
   Evaluate data at $TM$ and store in $TM.Y$.
   Compute $R = 1/(TM.\pi \cdot P_1(TM.Y) + (1 - TM.\pi) \cdot P_0(TM.Y))$. (eq. 4.28).
   Sweep through nodes of current tree
   - If ($t$ is terminal)
     (Apply eq. 4.27)
     if ($t$ descendent of $TM$) $t.\pi \leftarrow t.\pi \cdot R \cdot P_1(TM.y)$.
     else $t.\pi \leftarrow t.\pi \cdot R \cdot P_0(TM.y)$.
     if ($t.\pi > \pi_{max}$) Call birth($t$)
   - If ($t$ is not terminal)
     $t.\pi = t.ch_1.\pi + t.ch_2.\pi + t.ch_3.\pi$
   - Call update($t$): Check if time to reinitialize, and update current choice of $TM$.

Function update($t$)
if ($t.\pi > \pi^*$) and ($t.depth > d_{min}$)
   $TMS = t$;
   $x_1 = t.p_1, x_2 = t.p_2$;
   Goto 2.
if ($|t.\pi - \pi_{max}| \leq \Delta$)
   $\pi_{\Delta} = |t.\pi - \pi_{max}|, TM = t$.

Function birth($t$)
Create $t.ch_j, j = 1, 2, 3$.
Calculate $p_{1,2}$ for each, and set $ch_j.\pi = t.\pi/3$
if $|ch_j.\pi - \pi_{max}| < \Delta$
   $\Delta = |ch_j.\pi - \pi_{max}|$
   $TM = ch_r (r\text{-random})$.

The $m + 1$th segment at which the data is observed, i.e. $Y_t$ is measured, can be characterized as one of the following possibilities. If the posterior on $t_m$ is very high it could be a randomly chosen child of $t_m$. 
This corresponds to asking whether the curve continues down the path determined by \( t_m \). If the posterior on \( t_m \) is low, the next segment could be one of the interior elements of \( T_m \) for which the updated posterior is closer to \( \pi_{max} \), recall that not all nodes of \( T_m \) have been queried. The information at that node may actually indicate where the curve does not pass. This is useful information. If there is accumulated evidence that the curve does not pass through certain parts of the tree, the posterior on other parts increases and the next query may be a randomly chosen child of some element of \( T_m \) which has high updated posterior, but which is not close to \( t_m \). The last two options are specifically what allow for backtracking and choosing a new direction of search. It is also possible to ‘jump ahead’. Assume \( t_j \) is somewhere near the root and that \( t_{j+1}, \ldots, t_m \) are not descendents of \( t_j \), and that \( T_m \) includes some subtree \( T_j \) rooted at \( t_j \). (This is possible because descendents of \( t_j \) could have been added to the sets without having been queried.) It may be that \( t_{m+1} \) is some element of \( T_j \) which is not a direct child of \( t_j \).

This algorithm performs essentially in real time. The key component of this efficiency is the very small number of segments at which the data is actually accessed and processed, and the simple recursive way in which the posterior can be updated. In figure 4.4 we illustrate the workings of the algorithm in detecting the scalp in an axial MRI scan used in figure 4.3. The parameters \( p_b, p_c \) were not estimated but set manually to .4 and .7 respectively. The angle \( \beta \) was set to 10 degrees and the length of each segment is set to 10. The algorithm tracks the entire scalp with only the two initial points provided and marked in white. In the top image this track is shown in white and in black are shown those segments in the tree for which the image data was accessed. On the right hand image we see the tracked curve with two different initial points. The computation time is a fraction of a second. In figure 4.5 a similar experiment is done to track the artery in an angiogram.

Although the tree grows exponentially with the depth, the number of nodes actually involved in the algorithm, i.e. the size of \( T_m \) never grows above several hundreds. It is also possible to prune terminal nodes for which the posterior falls below a very low threshold. Although it is hard to see, there is plenty of clutter in the background, which is able to ‘distract’ the algorithm for rather extensive periods.

4.4 Bibliographical notes and discussion

The motivation for the dynamic programming algorithm comes from Petrocelli et al. (1992), similar work can also be found in Geiger et al. (1995). The tree based algorithm is taken from Geman & Jedynak (1996) and has been implemented with great success for tracking roads in remote sensing images. It is very fast and outperforms any computationally feasible implementation of dynamic programming.

Whereas the deformation algorithms described in chapter 3 had a region growing flavor to them and hence in some coarse sense expected the initial contour to expand around a more or less homogeneous region, the curve detection algorithms described in this chapter perform some form of ‘tracking’, in that they attempt to proceed along a curve with some particular image statistics in its neighborhood.

The curve detection algorithms need some form of initialization but are much more robust to clutter and noise both due to the more systematic modeling of the data along the curve and the fact that within some restricted region the actual global optimum is found. In figure 4.6 we show an example of a randomly perturbed \( \TeX \) in a cluttered environment. Using a closed contour of the prototype \( \mathcal{E} \) we run a deformable contour algorithm initialized with the contour shown in the top left panel. The final detection is shown in the top right panel. Due to the cluttered environment the contour reaches some local minimum which does not reflect the correct instantiation. For comparison we initialize the dynamic programming algorithm for curve detection with a curve also produced from the prototype \( \mathcal{E} \). The initial curve is even further removed from the correct instantiation. And yet the final detection shown in the bottom right panel has identified a correct instantiation of the deformed \( \mathcal{E} \).

Tree based curve detection is much faster than dynamic programming and only requires an initialization
Figure 4.4: Detecting the scalp in an axial MRI scan. Left: Two initial points shown as larger squares. In black every segment that was queried, i.e. at which image data was accessed. In white the track leading to the terminal node of highest posterior at each step. Right: Final contour with two different initial points.
of two points at the beginning of the curve. It is very successful in detecting long smooth curves of low curvature. However it is hard to incorporate prior shape information in this setting, and it risks overshooting if the true curve does have large curvature at some point. This is the effect of the very strong prior. See for example the tracking of the artery in angiogram of figure 4.7. When the artery has a strong bend the detection overshoots.
CHAPTER 4. TRACKING

Figure 4.6: Top: Detection of deformable contour. Left: initial contour. Right: Detection. Bottom: Detection of deformable curve with dynamic programming. Left: Initial curve. Right: Detection.
Figure 4.7: An example of an overshoot of the tree based algorithm at a severe bend of the curve.
Chapter 5

2-d Models: deformable images

In Chapters 3 and 4 we explored the idea of fitting a one dimensional model to the boundary of an object. Prior information regarding the shape of the object is represented in terms of the initial template contour or template curve. In most cases when we analyze images of a particular object class, there is much more information available than the mere shape of one contour. Any two images of the object class viewed as surfaces would exhibit many topological and geometric similarities which are ignored when the contour alone is computed. The variations between objects often appear to be smooth, in the sense that with some pulling and stretching and squeezing one could take one image from such a collection and transform it into another. This form of stretching and squeezing, which we call image deformation, deforms not only the contour of one object to the contour of the other, it deforms the entire image surface.

Consider the image domain \( D \) as a continuum, typically the unit square, and let the prototype image be a function \( F(x), x \in D \). Let \( \phi \) denote a smooth deformation of \( D \) onto itself. What can we say about the image \( \tilde{F}(x) = F(\phi(x)) \)? If \( \phi \) is smooth extremal points are preserved, maxima are mapped into maxima, minima into minima, saddle points into saddle points, level curves into level curves. Loosely speaking the topography of the surface remains the same although the shapes of the hills and valleys may change. If \( \phi \) is sufficiently close to the identity we seem to be observing a different version of the same object. For example in figure 5.1, 25 smooth random non-linear deformations of a face still look like a face although not necessarily of the same person. The various points of interest which are characterized by some peculiar topography of the image surface, are mapped to points with the same local topography and they maintain their relative spatial arrangement. Moreover if the boundary of the object or any part of the object is marked in the prototype image \( F \), then \( \phi^{-1} \) will deform it to the corresponding boundary in the image \( \tilde{F} \).

We have reparameterized the images of the object class in terms of the deformations \( \phi \) and have established a correspondence to the prototype image. This is illustrated by picking several points on the prototype image and plotting \( \phi^{-1}(x) \) on the data image for each such point. For example note that in 5.2 the mapping \( \phi^{-1} \) maps the eyes in the prototype image to the eyes in the data image, as well as the tips of the mouth or other points on the boundary of the face.

Our goal in this section is to describe techniques for computing \( \phi \) given a prototype image \( F \) and a new data image \( I \) of the same object class.

5.1 Statistical model

In terms of the formulation in Chapter 2, we take the template points \( Z \) to cover the entire reference grid \( G \), and assume the data images are defined on a lattice \( L \), which is the same size as \( G \). The set of instantiations \( \Theta \) is then a collection of smooth one-to-one maps of \( Z = G \) onto the image lattice \( L = G \).
Gaussian data model

Let $F(z), z \in Z$ be a prototype image of the object. We distinguish here between the template which is simply the lattice structure of points $Z$ that are mapped into the image grid $L$, and the prototype image which is used to define the data model. Assume that any image of the object is generated as

$$I(\theta(z)) = F(z) + N(z), z \in Z \quad (5.1)$$

where $N(z), z \in Z$ are independent mean zero Gaussians with some fixed variance $\sigma$. In other words the pixel intensity of the prototype image at point $z$ is moved to $\theta(z)$ and some noise is added, independently at each pixel. This is not a realistic model but is very simple and leads to interesting results. Under this model the data $I(x), x \in L$ given the instantiation $\theta$ is again independent, since the additive noise is independent. It will be more convenient to write this equation as

$$I(x) = F(\phi(x)) + N(x),$$

where $\phi = \theta^{-1}$. Every pixel $x \in L$ grabs the value of the prototype image $F$ at $\phi(x)$. The gray level intensities given $\phi = \theta^{-1}$ are assumed to be independent Gaussians with variance $\sigma$, and mean given by
CHAPTER 5. IMAGE WARPING

$F(\phi(x))$. Up to a constant the log-likelihood therefore has the simple form

$$D(U) = -\frac{1}{2\sigma^2} \sum_{x \in \mathcal{L}} |F(\phi(x)) - I(x)|^2 dx. \quad (5.2)$$

Since we will be optimizing through gradient descent methods we consider the data image and the prototype image as defined on the continuum, i.e. as $I(x), F(x), x \in D$ where $D$ is the unit square. The map $\phi$ defines a smooth deformation of the domain $D$ into itself. The actual instantiation of the grid $Z$ can be recovered by taking $\phi^{-1}(z), z \in Z$.

Define the displacement field as $U(x) = \phi(x) - x$ and denote its two components as $U^{(1)}$ and $U^{(2)}$. Now rewrite $D(U)$ as

$$D(U) = -\frac{1}{2\sigma^2} \int_D |F(x + U(x)) - I(x)|^2 dx. \quad (5.3)$$

The field $U(x)$ is easier to work with since the value 0 corresponds to the identity map To simplify notation and computation we will henceforth assume that $U$ is periodic on the unit square.

Section 5.4 describes an alternative data model based on transforming the image data into binary local features similar to those used in Chapter 4. This data model will have some advantages in terms of photometric invariance and in terms of parameter estimation.

The prior and posterior

There are many solutions $U$ which will yield $D(U) = 0$. However such solutions will typically be highly irregular, discontinuous, and non-invertible. The topography of the image surface would no longer be conserved. In order to rule out the discontinuous and irregular solutions we introduce a prior on the set of displacement fields which implicitly defines a prior on the maps $\phi(x) = x + U(x)$ of the image into the reference grid. As in Chapter 3 we adopt a spectral parameterization of the displacement fields in terms of their expansion in some basis of functions. Again such a representation defines a map on the continuum in terms of a finite number of parameters, and using wavelet or Fourier functions, we obtain a natural coarse to fine parameterization of the deformations. Write

$$U^{(q)}(x) = \sum_{k=0}^{d} u^{(q)}_k \psi_k(x), \quad q = 1, 2 \quad (5.4)$$

for some finite $d$ where $\psi_k$ is some basis of functions on the unit square. This parameterization allows for a direct approach to enforcing the smoothness if we assume the coefficients $u^{(q)}_k$ are independent Gaussian, with mean zero and variance $1/\lambda_k$, with $\lambda_k$ an increasing sequence of positive values. The log-prior is then up to a constant given by

$$\mathcal{E}(U) = -\frac{1}{2} \sum_{k=0}^{d} \lambda_k [(u^{(1)}_k)^2 + (u^{(2)}_k)^2]. \quad (5.5)$$

One draw back of this parameterization is that $\phi(x) = x + U(x)$ is not guaranteed to be one-to-one and $\phi(x)$ may not necessarily be in the domain $D$. This does not cause major problems in applications as will be seen below.

Fourier basis

For the two-dimensional Fourier basis $k = (k_1, k_2)$ is a two parameter index and each basis element has the form

$$\frac{1}{2\pi} \psi_k(x) = \exp[2\pi i(k_1 x_1 + k_2 x_2)].$$
As $k_1, k_2$ increase the functions $\psi$ have derivatives of increasing magnitude. In order to ensure smoothness the variances must decrease. Thus we set $\lambda_k = (k_1^2 + k_2^2)^\rho$ for some positive $\rho$. The larger $\rho$ the smoother the resulting functions will be.

**Wavelets**

As in one dimension, the two-dimensional wavelet basis on the unit square is also arranged in a pyramid. The structure is somewhat more complex. At each level $s$, there are three functions $\psi_{\alpha,s,0,0}, \alpha = 1, 2, 3$ and all other functions of that level are given as shifts of one of these three functions:

$$
\psi_{\alpha,s,\ell_1,\ell_2}(x) = \psi_{\alpha,s,0,0}(x_1 - 2^{-(s-1)}\ell_1, x_2 2^{-(s-1)}\ell_2),
$$

for $\alpha = 1, 2, 3$ and $\ell_1, \ell_2 = 0, \ldots, 2^{s-1}$. Thus $k$ is a four parameter index $k = (\alpha, s, \ell_1, \ell_2)$, where the index $s$ defines the resolution. The constant function is denoted $\psi_{0,0,0,0}$. For $s \geq \tilde{s}$, the function $\psi_{\alpha,s,0,0}$ is a scaling and dilution of $\psi_{\alpha,S,0,0}$. Specifically

$$
\psi_{\alpha,s,0,0}(x) = 2^{(s-S)}\psi_{\alpha,S,0,0}(2^{(s-S)}x).
$$

Again this is also true for $s < \tilde{s}$ modulo some wrap around effects.

At the highest resolution $S$, the functions $\psi_{\alpha,S,0,0}$ have very small support. The support increases as the resolution $s$ decreases. For higher values of $s$ the information conveyed by the coefficients is more local. Also as $s$ increases the derivatives of the functions increase in magnitude and a smaller variance needs to be set for the coefficients to maintain the smoothness of the function. Thus we set

$$
\lambda_k = \lambda_{(\alpha,s,\ell_1,\ell_2)} = 2^{\alpha s}, \text{ for all } 0 \leq \ell_1, \ell_2 < 2^s, \alpha = 1, 2, 3. \quad (5.6)
$$

The same variance is assigned to coefficients of all functions at the same resolution. This particular form of decrease of the variance with resolution is motivated by the theory relating rates of decay of wavelet coefficients to smoothness properties of the corresponding functions, which can be found in Wickerhauser (1994) and Meyer (1990).

**Smoothing with differential operators**

Often in the literature the smoothness of the deformations is obtained by a regularizing term in the cost function. A common practice is to penalize the magnitude of the derivatives of the function. In the discretized world this corresponds to penalizing large distances between the mapping under $U$ of two nearby lattice points. Nearby pixels should be mapped to nearby locations. If on the lattice we want to minimize

$$
E_L(U) = \sum_{|x-y|=1} |U(x) - U(y)|^2, \quad (5.7)
$$

in the continuum this translates to a penalty term of the form

$$
E(U) = \int_D |\nabla U^{(1)}(x)|^2 + |\nabla U^{(2)}(x)|^2 dx. \quad (5.8)
$$

Higher order differences or derivatives lead to higher degrees of smoothness. Any penalty of the form appearing in equation 5.8 involving differential operators can be directly translated into a penalty of the form appearing in equation 5.5, with the Fourier basis, taking $d = \infty$ and using particular choices of $\lambda_k$. For example for the penalty in equation 5.8 take $\lambda_k = (k_1^2 + k_2^2)$. However the spectral parameterization offers a much richer collection of possible penalties, since other values of $\lambda_k$ can be chosen as well as other bases.
Figure 5.2: Top left: Prototype image $F$. Top right: Data image $I$. Bottom left: Deformed prototype image $F(x) = F(x + U(x))$. Bottom right: Difference $|F(x) - I(x)|$. Middle: The displacement field $U$. The seven points marked on the prototype image $F$ are mapped through $\phi^{-1}$ to points shown in the data image $I$. 
such as wavelets or wavelet packets. As in Chapter 3 this representation also has the advantage of offering a natural way to implement a coarse to fine computation, and a built in interpolation of the map \( \phi \) to the continuum.

Combining the likelihood and the prior, and ignoring the constant which does not depend on the unknowns, the negative log-posterior has the form of a cost function of the form

\[
J(u^{(1)}, u^{(2)}) = \frac{1}{2} \sum_{k=0}^{d} \lambda_k [(u_k^{(1)})^2 + (u_k^{(2)})^2] + \int \left( F[x_1 + \sum_{k=0}^{d} u_k^{(1)} \psi_k(x), x_2 + \sum_{k=0}^{d} u_k^{(2)} \psi_k(x)] - I(x) \right)^2 dx.
\]

which is now rewritten as a function of the coefficients \( u^{(q)} = (u_k^{(q)})_{k=0}^{d}, q = 1, 2 \). The variance coefficient \( 1/\sigma^2 \) from the data term has been absorbed into the coefficients \( \lambda_k \).

### 5.2 Connection to the deformable contour model

It is of interest to point out the relation of this model to the inside-outside deformable contour model presented in the Chapter 3. Given the initial contour \( \gamma^{(0)} \), define a prototype image which has constant value \( \mu_{\text{in}} \) inside the initial contour and constant value \( \mu_{\text{out}} \) outside. The data term for the two dimensional model is now identical to the data term for the one dimensional model. The difference in applying the two dimensional model is in the way the deformations of the initial contour are parameterized. Now the contour, which is the boundary between the two intensity levels is deforming together with the entire 2-d unit square. If \( \phi(x) = x + U(x) \) is the two dimensional deformation then \( \gamma(t, \phi) = \phi^{-1}(\gamma^{(0)}(t)) \). Since \( \phi \) is smooth, \( \gamma(t, \phi) \) is a smooth deformation of \( \gamma^{(0)} \). One way of illustrating the difference between these two forms of parameterization is to imagine the one dimensional model as a deforming elastic rubber band and the two dimensional model as a curve drawn on a deforming elastic sheet. It is clear for example that in the second case there is a much smaller chance of getting self intersections of the curve. In the one dimensional model, even if the perturbation is smooth it is easy to obtain self intersections.

On the other hand the advantage of the one dimensional model is that even though it uses a data-term involving the entire two dimensional domain, the actual gradient descent algorithm only uses data in the immediate vicinity of the curve, and is therefore less sensitive to whether the global data model is appropriate or not. Thus for example in figure 3.2, the outside model which assumes the outside is brighter is obviously wrong, except locally at the boundaries of the ventricle. By contrast the two dimensional algorithm always integrates over the entire domain, and tries to match the two surfaces.

### 5.3 Computation

The cost function is optimized using a gradient descent algorithm. The derivative of the cost function \( J \), of equation 5.9, in the coefficients \( u_k^{(q)} \), is obtained by interchanging integration and differentiation and applying the chain rule. This yields
for \( q = 1, 2 \) and \( k = 0, 1, \ldots, d \). As in Chapter 3 the derivative of the data term is a forward transform of a function in terms of the chosen basis. In this case the function is

\[
\frac{\partial J(u^{(1)}, u^{(2)})}{\partial u_k^{(q)}} = \lambda_k u_k^{(q)} + \int \frac{\partial F}{\partial x_i}(x + U(x)) [F(x + U(x)) - I(x)] \psi_k(x) dx,
\]

(5.10)

for \( q = 1, 2 \) and \( k = 0, 1, \ldots, d \). As in Chapter 3 the derivative of the data term is a forward transform of a function in terms of the chosen basis. In this case the function is

\[
\frac{\partial q F(x + U(x)) [F(x + U(x)) - I(x)]}{\partial x_i},
\]

where \( \partial q F, q = 1, 2 \) are the two partial derivatives of \( F \). In Chapter 3 we already introduced the idea of gradually increasing the number of updated coefficients. This idea proves to be very useful in the context of deformable images as well. Minimizing in the first few coefficients in a wavelet or Fourier basis provides large and smooth coarse deformations. Once these converge to a local minimum increase the number of coefficients and obtain more detailed matches.

Let \( \Psi \) denote the forward transform of a function and \( \Psi^{-1} \) denote the backward transform applied to a sequence of coefficients with respect to the chosen basis.

\[
\Psi(f) = \{ \int_D f \psi_k dx, k = 0, \ldots, d \}
\]

\[
\Psi^{-1}(u) = \sum_{k=0}^d u_k \psi_k.
\]

Choose \( \mathcal{N}_1 < \mathcal{N}_2 < \ldots < \mathcal{N}_A \leq d \). The coarse to fine gradient descent algorithm has the following form:

**Algorithm 5.1: Image deformation: coarse to fine**

1. Normalize \( F \) and \( I \) to have range of values in \([0, 1]\).
   
   Initialize \( u_0^{(q)} = \{u_{k,0}^{(q)}, k = 0, \ldots, d\}, q = 1, 2 \). (Typically \( u_0 = 0 \)).
   
   Set \( m = 0, a = 1 \).
2. Calculate \( U_m^{(1)} = \Psi^{-1}(u_m^{(1)}), U_m^{(2)} = \Psi^{-1}(u_m^{(2)}) \),
3. Calculate
   
   \[ W_q(x) = \partial_q F(x + U(x)) [F(x + U(x)) - I(x)], \]
   \( q = 1, 2 \).
4. Calculate \( v^{(q)} = \Psi(W_q), q = 1, 2 \).
5. Set \( u_{k,m+1}^{(q)} = u_{k,m}^{(q)} - \Delta \cdot (\lambda_k u_{k,m}^{(q)} + v_{q,k}) \) only for \( k = 0, \ldots, \mathcal{N}_a \), and \( q = 1, 2 \)
6. If stopping criterion satisfied go to 7, otherwise \( m \leftarrow m + 1 \), go to 2.
7. If \( a < A \) set \( a \leftarrow a + 1 \) go to 2, otherwise exit.

For the Fourier basis we take \( 0 < k_1, k_2 < a \), for each of the two field components, so that \( \mathcal{N}_a = 2 \cdot a^2 \), and \( a = 1, \ldots, A \). The index \( a \) corresponds to the highest frequency being updated. For the wavelet basis take \( \psi_{a,s,\ell_1,\ell_2} \) such that \( 1 \leq s \leq a \) and \( 0 \leq \ell_1, \ell_2 < 2^s-1 \). In this case \( \mathcal{N}_a = 2 \cdot 2^a, a = 1, \ldots, A \). The index \( a \) corresponds to the highest resolution currently being updated.

The choice of initial point 0 is motivated by the fact that the mapping generated by the solution is expected to be in some neighborhood of the identity map.
CHAPTER 5. IMAGE WARPING

Discretization and the 2d wavelet transform

The functions \( F \) and \( I \) are defined only on the grid \( L \). When \( x + U(x) \) does not lie on the grid, set the value of \( F(x + U(x)) \) as the linear interpolation of the four nearest lattice points around \( x + U(x) \). The derivatives of \( F \) are calculated using finite differences and stored, before the iteration procedure begins. When \( x + U(x) \) lies outside the domain \( D \), set \( F(x + U(x)) \) and \( \partial_q F(x) \) to 0. These are the ingredients needed to compute \( W_1(x), W_2(x) \) step 3, at each \( x \in L \).

Discretizing involves replacing the continuous basis with its discrete counter part and applying discrete forward and backward transforms, also denoted \( \Psi, \Psi^{-1} \). Let the image lattice be \( 2^S \times 2^S \). When using discrete wavelets the largest possible value for \( A \) is \( S \).

The discrete 2d wavelet transform is easily obtained from the 1d transforms. Given a \( 2^s \times 2^s \) data array \( F^{(s)} \) compute one step as described in equations 3.21, 3.22 for each row to obtain \( F^{(s)} \). Then compute the same for each column of \( F \) to obtain \( Y^{(s)} \). The result is a \( 2^s \times 2^s \) array

\[
Y^{(s)} = \begin{pmatrix}
  u_1^{(s)} & u_2^{(s)} \\
  u_3^{(s)} & F^{(s-1)}
\end{pmatrix},
\]

where \( u_{\alpha,\ell_1,\ell_2}^{(s)}, 0 \leq \ell_1, \ell_2 \leq 2^{s-1} - 1 \) and \( \alpha = 1, 2, 3 \) are the coefficients corresponding to resolution \( s \), i.e. the functions \( \psi_{[\alpha, \ell_1, \ell_2]} \). The same type of operation is again repeated on \( F^{(s-1)} \) and so on.

The filters \( h \) and \( g \) of equations 3.21 and 3.22 can be combined through tensor products to define the following four \( R \times R \) filters.

\[
H_{i,j}^{(1)} = h_i \cdot h_j, \quad H_{i,j}^{(2)} = h_i \cdot g_j, \quad H_{i,j}^{(3)} = g_i \cdot h_j, \quad G_{i,j} = g_i \cdot g_j.
\]

for \( i, j = 1, \ldots, R \). Then for each \( \alpha = 1, 2, 3 \) the matrices \( u_{\alpha}^{(s)} \) are simply the subsampled convolution of \( F^{(s)} \) with \( H^{(\alpha)} \). These are high pass filters of \( F^{(s)} \). The matrix \( F^{(s-1)} \) is the subsampled convolution of \( F^{(s)} \) with \( G \), which is a low pass filter.

The inverse transform is obtained in the same way. Apply equation 3.23 to each row of \( Y^{(s-1)} \) and then apply the same to each column of \( Y^{(s-1)} \). This reconstructs \( F^{(s)} \), again due to the particular properties of \( h \) and \( g \).

Time step

The time step is given as the inverse of an approximation to the trace of the Hessian matrix of \( J \). The second derivative of \( D \) with respect to the coefficient \( u^{(s)}_k \) is a sum of two integrals. One has the difference of \( F(x + U(x)) \) and \( I(x) \) in the integrand and is ignored assuming that \( F(x + U(x)) \) is relatively close to \( I(x) \). The other term is \( \int \partial_q F^2 \psi_k^2(x)dx, q = 1, 2 \). Assuming a uniform bound \( M \) on \( \psi_k, k = 0, \ldots, d \) we bound the trace of \( J \) by

\[
T = dM \int |\nabla F|^2 dx + \sum_k \lambda_k,
\]

and set the time step \( \Delta = T^{-1} \).

Smoothing

In some cases it is useful to smooth and subsample the prototype image and the data image. First of all this tends to single out global geometric structures and eliminate local ones. Second lowering the dimension of the data speeds up the algorithm. Finally if \( I \) and \( F \) have disjoint support, for example two ‘humps’ supported
on disjoint parts of the domain, there is really no reason for these two structures to attract. Smoothing can increase the support size of the two functions and create some interaction between the two.

Other optimization algorithms

The minimization algorithm is a discrete approximation to the downward gradient flow. Other more sophisticated approaches are described in the literature, (see Press et al. (1995)). The first alternative would be to implement the conjugate gradient method. This involves line searches, and turns out to yield comparable results to the gradient descent algorithm at about the same speed. Note that the computation involved in the forward or backward wavelet transform is on the order of a few evaluations of the cost itself, and thus very inexpensive. A second alternative involves Newton or quasi-Newton methods. These can be rather computationally intensive. In section 5.5 we discuss the linearization of the cost function and the associated least square solution. Implementing this procedure at any iteration of the gradient descent algorithm corresponds to a quasi-Newton type step and may speed convergence.

Computing pose parameters

The standard bases mentioned in this chapter rarely express the affine pose parameters other than translation in a direct way. Some initial updating of these parameters could be useful to improve on the initial pose parameters provided by the user. In the context of the minimization problem posed above write

\[ D(A) = \min_{A \in \mathcal{A}} \int |F((I + A)x) - I(x)|^2 dx, \]

where \( \mathcal{A} \) is the set of affine maps. If this set is directly parameterized in terms of the 4 entries of the linear matrix, and the two entries of the translation, the problem is equivalent to defining three basis functions \( \psi_1(x) = 1, \psi_2(x) = x_1, \psi_3(x) = x_2, \) and minimizing \( D \) in terms of their coefficients. The gradient has the same form as in equation 5.10 except that the integral needs to be explicitly calculated and can not be computed using a fast forward transform.

Ignoring ‘shear’, \( A \) can be parameterized in terms of scaling \( s_1, s_2 \) in the two coordinates, a rotation \( \alpha \) and a translation \((t_1, t_2)\), writing

\[ Ax = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{pmatrix} x + (t_1, t_2). \]

One can either simultaneously optimize in all 5 parameters calculating the respective partial derivatives, or iterate by first adjusting for translation, then scale and finally rotation. This cycle can be repeated several times. This is used in the Bernoulli model below.

Lattice parameterization

In order to enforce smoothness, the low dimensionality of the deformation space, and a natural coarse to fine procedure we used a spectral parameterization of the deformations in terms of some orthogonal bases. It is possible to stay with the point wise parameterization, \( U(x), x \in L \), where \( L \) is the image lattice. The penalty remains in the form of a differential operator as in equation 5.8. This procedure is very problematic in terms of obtaining the large coarse deformations and is relatively very slow. It is therefore not recommended for starting. It is however possible to use it for fine detail matching after the spectral approach has converged. This is implemented in Miller et al. (1993).

One way to implement the lattice parameterization together with a coarse to fine procedure is related to motion estimation algorithms used for image compression. The lattice is divided into large disjoint blocks
and a fixed displacement is estimated for separately for each block. This is done either through a gradient descent procedure on the two displacement parameters, or by an exhaustive search for the best match in some constrained region. A smooth interpolation of the displacements identified for each block produces the final displacement field for the current level. This serves as an initial point for the search for displacements for the next level of smaller blocks. Such methods are used in various MPEG implementations, for example see Chalidabhongse & Kuo (1997). They are very similar to the spectral approach described above if the Haar basis is used, which is a basis of piecewise constant functions, and is the simplest type of wavelet basis.

5.4 Bernoulli data model

One inherent problem in the model discussed thus far is the assumption that the gray levels have a Gaussian distribution with mean given by the deformed prototype image. It is clear that a variety of lighting conditions such as shadows or non-homogeneous light produce a more complex distribution of the gray level intensities even for a fixed object. In such cases it is desirable to find forms of the data term \( D \) which are invariant to such changes. Another problem is that it is impossible to model pixel intensities off the object as Gaussians with some fixed mean. However some form of model for pixels off the object can be helpful when the object does not occupy the entire image. In this section we will define a data model in terms of photometric invariant binary local features similar to those defined in Chapter 4 (see equations 4.1, 4.2). Those were roughly speaking ‘ridge’ detectors and were specifically designed to be highly invariant to photometric transformations. We now define discontinuity detectors commonly known in the literature as edges, again in terms of simple comparisons of pixel intensity differences.

Edges

Let \( v \) be one of the vectors \((1, 0), (1, 1), (0, 1), (-1, 1)\), and let \( w \) be the 90 degree rotation of \( v \). Let \( z \) be a pixel and \( y = z + v \), which is one of the eight nearest neighbors of \( z \). Denote \( z_1 = z + w, z_2 = z - w, z_3 = z - v \) and \( y_1 = y + w, y_2 = y - w, y_3 = y + v \). An edge is present at \( z \) if

\[
|I(z) - I(y)| > \max_{i=1,2,3} (\max (|I(z) - I(z_i)|, |I(y) - I(y_i)|)).
\]

The orientation of the edge is \( v \) if \( I(z) > I(y) \) and \(-v \) otherwise. The eight edges correspond to the four possible directions of \( v \) multiplied by the two possible signs of \( I(z) - I(y) \). The configuration of pixels being compared for a vertical edge is shown in figure 5.3. All in all six intensity difference comparisons are involved in defining the edge. These edges can be viewed as local maxima of the modulus of the gradient. It is possible to introduce a lower bound on the size of the gradient to avoid edges in extremely low contrast areas, i.e. \(|I(y) - I(z)| > \kappa\).
CHAPTER 5. IMAGE WARPING

In a given \( b \times b \) block of pixels several different edges types may be detected. This is even true for \( b = 1 \). One can decide to allow only \( n_e \) edges in each \( b \times b \) block taking those with largest gradient \(|I(z) - I(y)|\). Invariance to significant gray scale transformations is inherent in the definition of these features.

The data model

Let \( \hat{I}(x) = (X_1(x), \ldots, X_J(x)) \) (in this case \( J = 8 \)) denote the transformed image data, namely the vector of binary outputs of the edge features at point \( x \) in the image. Let \( F_1(x), \ldots, F_J(x) \) be functions of value between 0 and 1 and assume that given the object is present at deformation \( \theta \), at each location \( x \in L \),

\[
P(X_j(x) = 1|\theta) = F_j(\theta^{-1}x) = F_j(\phi(x)), \tag{5.12}
\]

independently of all other locations and features. In other words, given the object is present at deformation \( \theta \), the variables \( X_j(x), x \in L, j = 1, \ldots, J \) are all independent and have marginal distribution given by \( F_j(\phi(x)) \). The functions \( F_j \) are called probability maps, and since the features are binary we call this the Bernoulli model. Again we have a conditional independence model for the transformed image data given the instantiation of the object. The full log-likelihood, in terms of the displacement \( U(x) = \phi(x) - x \) has the form

\[
D(U) = \log P(\hat{I}(x), x \in L|U)
\]

\[
= \sum_{j=1}^{J} \sum_{x \in L} \left( X_j(x) \log[F_j(x + U(x))] + (1 - X_j(x)) \log[1 - F_j(x + U(x))] \right). \tag{5.13}
\]

Adding the prior term of equation 5.5 we have the negative log-posterior

\[
J(U) = -\log P(U|\hat{I}(x), x \in L)
\]

\[
= E(U) - \sum_{j=1}^{J} \sum_{x \in L} \left( X_j(x) \log[F_j(x + U(x))] + (1 - X_j(x)) \log[1 - F_j(x + U(x))] \right) + C. \tag{5.14}
\]

This defines a cost function which needs to be minimized and again this can be achieved using gradient descent. Parameterizing \( U \) in terms of the coefficients \( u \), the gradient has the form

\[
\frac{\partial J(U)}{\partial u^{(q)}_k} = \lambda_k u^{(q)}_k + \sum_{j=1}^{J} \left[ \sum_{x \in L} X_j(x) \frac{\partial_q F_j(x + U(x))}{F_j(x + U(x))} 
\right.
\]

\[
- (1 - X_j(x)) \left( \frac{\partial_q F_j(x + U(x))}{1 - F_j(x + U(x))} \right) \psi_k(x). \tag{5.15}
\]

for \( k = 0, \ldots, d \) and \( q = 1, 2 \). Again the gradient is in the form of the forward transform of a function with respect to the basis \( \psi_k \). Maximization can proceed just as in algorithm 5.3. The only change is in step 3. Now

\[
W_q(x) = \sum_{j=1}^{J} \left[ X_j(x) \frac{\partial_q F_j(x + U(x))}{F_j(x + U(x))} - (1 - X_j(x)) \frac{\partial_q F_j(x + U(x))}{1 - F_j(x + U(x))} \right],
\]

and is evaluated at each \( x \in L \).
Figure 5.4: Bernoulli data model. Top: Eight probability maps $F_i$ on the reference grid. Dark denotes high probability. Middle: Transformed image data, locations of the eight detected edges in the image. The probability maps and transformed data are shown smaller than the original image. They are all $64 \times 64$. Bottom: Left - Original data image. Overlayed are reference points, chosen by hand on the reference grid which should correspond to the hairline, eyes and mouth if the face is properly detected. Middle - After estimation of translation and scaling in $x$ and $y$, the reference points are shown mapped by the estimated scale and translation. Right - After finding the non-linear deformation, it is used to map the reference points into the image.
CHAPTER 5. IMAGE WARping

Training

The use of binary features is very convenient from the point of view of parameter estimation or training. At each point in the reference grid we simply estimate a proportion for each of the features $j = 1, \ldots, J$. Assume we are given a sample of training images of the object, $I^{(t)}, t = 1, \ldots, T$, each with a pose parameter $s_t$, for example scale and location. Extract the features $X_j(x), j = 1, \ldots, J$ at all locations $x \in L$ for image $t$, and register these locations to the reference grid using the pose information $s_t$ for that image. In other words if $X_j(x) = 1$ add a count to $F_j(z)$ where $z = s_t^{-1}(x)$. Some stability is gained by also adding a count to $F_j(z')$ for $|z - z'| \leq 1$. After normalizing by $T$ the maps $F_j, j = 1, \ldots, J$ are the proportions at each location on the reference grid of the $J$ different features, and serve as estimates of $P(X_j(x) = 1 | \theta)$ from equation 5.12. Set a minimal value for $F_j$ at all locations on the reference grid. This is the probability of finding feature $X_j$ in a generic background image not containing the object being modeled.

The probability maps $F_j$ determined in this way will be somewhat ‘blurred’ since the only part of the deformation variability which is ‘factored out’ in training is the pose component $s_t$. The more variable the instantiations of the object in the training set, beyond pose variability, the lower the probabilities at each location on the reference grid. In other words the instantiation variability is transferred into probabilities on the reference grid.

The use of a minimal background probability for each of the features offers a form of background model, albeit very primitive, of the data on the object. This is useful in discouraging the template from attempting to match clutter in the immediate vicinity of the object; edge activity outside the object does not incur a high penalty. For regular gray level prototypes it is not at all clear what the values of the prototype image outside the object should be.

The top panel of figure 5.4 shows the probability maps for the eight edges. Darker areas correspond to higher probabilities. The middle panels show the eight detected edge features in the data image. We express the estimated instantiation of the face using a collection of reference points that was arbitrarily chosen on the reference grid, and which are mapped by the estimated $\psi$ into the data image. The first image shows the initial instantiation which has some overlap with the background. The second shows an estimated scaling and translation, and the third shows the result of computing a non-linear deformation.

5.5 Linearization

When the deformation $U$ is known to be small in magnitude, the problem can be linearized. We provide the derivations only for the Gaussian data model. The extension to the Bernoulli data model is straightforward. Taking the first order approximation to $F(x + U(x))$ the cost function becomes

$$J(U) = E(U) + \int |F(x) + (\nabla F \cdot U)(x) - I(x)|^2 dx,$$

which is quadratic in $U$. Computing the minimum of this function is equivalent to solving a least squares problem. The solution depends on how $U$ is parameterized, however for each parameterization the solution is unique and easy to compute.

The spectral parameterization

With the spectral parameterization we solve for the first $d$ coefficients of $U$. Differentiating $J(U)$ from equation 5.16 with respect to the coefficients $u^{(1)}, u^{(2)}$ and setting the gradient to equal zero, we obtain the following system of normal equations.

$$\lambda_k U_k^{(q)} + \sum_{\ell=1}^{d} \sum_{p=1}^{2} u_\ell^{(p)} \int \partial_q F \partial_p F \psi_\ell \psi_k dx = \int (I - F) \partial_q F \psi_k dx,$$
for \( k = 0, \ldots, d, q = 1, 2 \). Assuming the symmetric \( 2d \times 2d \) matrix with coefficients \( \int \partial_p F \partial_q F \psi_k(x) \psi_k(x) \) is non-singular this can be solved using a Cholesky decomposition (see Press et al. (1995).)

The solution to the least squares problem can be obtained without calculating the integrals of equation 5.17. Assume a discrete \( N \times N \) lattice \( L \) for the spatial variable \( x \) and let \( x_l \) denote the pixel \( ([\ell/N], \ell \text{ mod } N) \), for any \( \ell = 0, \ldots, N^2 - 1 \). Define the \((N^2 + 2d) \times 2d\) matrix \( A \) as

\[
A_{\ell,k} = \begin{cases} 
\langle \partial_1 F \psi_k(x_\ell) \rangle & \text{if } k \leq d \text{ and } \ell < N^2 \\
\langle \partial_2 F \psi_k(x_\ell) \rangle & \text{if } d < k \leq 2d \text{ and } \ell < N^2 \\
\sqrt{\lambda_k} & \text{if } k - N^2 = \ell \text{ and } k \leq d \\
\sqrt{\lambda_{k-d}} & \text{if } k - N^2 = \ell \text{ and } d < k \leq 2d \\
0 & \text{Otherwise}
\end{cases}
\]

Define the \( N^2 + 2d \) vector \( b \) as

\[
b_\ell = \begin{cases} 
[I(x_l) - F(x_l)] & \text{if } \ell < N^2 \\
0 & \text{Otherwise.}
\end{cases}
\]

The cost function of equation 5.16 becomes

\[ J(u) = \|Au - b\|, \]

and the least square problem can be solved using a QR decomposition of \( A \) (see Press et al. (1995).) Note that the penalty term on the coefficients, being quadratic, has been incorporated into the least squares problem by augmenting the 'matrix of observations'.

For each different value of \( d \) we have a different linear problem and the solutions will change. Although the solution to the linearized problem is obtained directly and has a very simple form, it may be worse than the solution obtained by the coarse to fine gradient descent algorithm for the original non-linear cost function. In the top left panel of figure 5.5 we show the deformed prototype image of figure 5.2 using the linearized cost function for the first 8 coefficients, \( (a = 3) \). A small change is observed but much less than in 5.2.

It is natural to apply the coarse to fine idea in this setting by first solving the linearized problem for \( d = N_1 \), obtain the estimated displacement field \( U \), then modify the prototype image \( F \) to \( F(x + U(x)) \). Increase \( d \) to \( N_2 \), recalculate the matrix \( A \) and the vector \( b \) in terms of the new prototype image, and solve the new linearized problem etc. This greatly improves the results as shown in the bottom left panel of figure 5.5, where the same data as in 5.2 has been used. In both cases the same Daubechies wavelet basis was used. Starting with one iteration at \( N_1 = 2 \times 2^2 \) - which corresponds to expanding \( U^{(1)} \) and \( U^{(2)} \) with the four smoothest elements in the basis, one iteration at \( N_2 = 2 \times 4^2 \), and three additional iterations at \( N_3 = 2 \times 8^2 \). After each iteration the prototype image has to be updated, and the derivatives recomputed in order to update \( A \). This is not done in the coarse to fine gradient descent algorithm. The derivatives of \( F \) are computed once and for all at the beginning, and only the displacement field is changing at each iteration.

Solving each of the least squares problems is computationally more intensive than a step of the gradient descent algorithm but less iterations are needed, and the results can be similar. However with large numbers of coefficients the least squares slows down drastically. The outcome of the gradient descent algorithm shown in figure 5.2 was obtained with 400 iterations and took approximately 1.7 seconds on the PENTIUM III 700 Mhz. The 5 iterations of the regression problem for the 4 different dimensions took approximately the same time.

In low dimensions it is possible to implement one least squares step after the gradient descent iterations appear to be converging. This can help improve the result. Indeed this step can be viewed as a quasi-Newton step where the term \( \int \partial_p F \psi_k(x) \partial_q F \psi_k(x) \) is taken as the typical entry of the Hessian matrix of the...
original cost function $J$. There is an additional term $\int (F(x + U(x) - I(x))\partial_{pq} F \psi_k \psi_l dx$ in the true Hessian which is ignored, as in the time step approximation in subsection 5.3.

The lattice parameterization

When the deformation is parameterized through the displacement vector $U(x)$ at each of the lattice points, with a smoothing penalty using first order derivatives, (see equation 5.8), the Euler equations for the linearized cost function are the following:

$$\Delta U^{(1)} + (\nabla F \cdot U) \frac{\partial F}{\partial x_1} = (I(x) - F(x)) \frac{\partial F}{\partial x_1}$$
$$\Delta U^{(2)} + (\nabla F \cdot U) \frac{\partial F}{\partial x_2} = (I(x) - F(x)) \frac{\partial F}{\partial x_2},$$

where $\Delta$ denotes the Laplacian operator. These lead precisely to the same equations suggested in the optical flow and image sequence analysis literature, Horn & Schunck (1981). In that context the problem is to estimate the motion occurring between two consecutive images in a video sequence. Assuming $I$ is the image following $F$, then $I - F$ is an approximation to the time derivative, and the above equation becomes the same equation derived in Horn & Schunck (1981), using an ‘image intensity’ conservation principle. This form
CHAPTER 5. IMAGE WARPING

of lattice based linearization is problematic when dealing with larger deformations and there is no natural coarse to fine implementation.

5.6 Applications to brain matching

There has been particular interest in applying deformable image algorithms in brain research, see Friston et al. (1995), Christensen et al. (1996), Sandor & Leahy (1995).

Activity in the brain is measured in various ways, the most common being positron emission tomography (PET) and functional MRI. In order to study locations of activity in relation to various functions of the brain, it is necessary to align or standardize the data from brains of different patients. One approach has been to match the MRI data of different brains to a standard template. Even if the MRI data does not contain signatures of the activity, the estimated deformation can be applied to the PET data. Note that the PET data itself has very little anatomic information and therefore would not lead to very precise matchings.

Figure 5.6 shows the outcome of a 2d match between two axial brain slices of two different patients using a wavelet basis. We use this application to illustrate several aspects of the image deformation algorithm. The images are $128 \times 128$, but are processed at a lower resolution of $64 \times 64$ as recommend in the discussion on smoothing. Starting at $N_1 = 2 \times 2^2$ through $N_6 = 2 \times 64^2$ the coarse to fine algorithm is implemented and the result is shown in figure 5.6 on the bottom left panel. The difference between the data and the deformed template is shown in the top right panel and the deformation field is shown in the middle. Note that the deformation is able to change some of the local structure of the sulci and giri of the prototype image to match that of the target image, as well as deform and match the shape of the interior ventricles.

For comparison in figure 5.7 we show the outcome of running the algorithm with all coefficients at once i.e. $N_1 = 2 \times 64^2$ together with the resulting deformation field. The prototype image is deformed to match certain local structures but the global shapes are not adjusted at all. Finally in figure 5.8 we show two other relatively successful matches using the coarse to fine algorithm.

MRI data is often accessed in 3d form. Either as a collection of 2-d slices or as a full reconstructed 3-d image. The image deformation algorithm described above is adapted to 3-d data in a straight forward way. The deformation field now has 3 components and is expanded in a 3d wavelet or Fourier basis.

The question arises as to whether this fine detail matching has any anatomic meaning. Do these locations which are mapped to each other between two different patients really correspond to the same anatomy. Beyond the coarse shape matching is there anything in the data which really justifies a specific deformation of one sulcus or gyrus to another. This is rather debatable since the way in which, say, a short curved gyrus is deformed into a long straight one, is rather arbitrary. There is nothing in the actual functionality of the different regions in that gyrus which guides the matching.

5.7 Bibliographical notes and discussion

More details on the statistical model can be found in Amit et al. (1991). The computational approach is based on Amit (1994). Similar models in the linear setting were proposed much earlier by several authors in the context of motion estimation from sequences of images. Horn & Schunck (1981), Huang & Tsai (1981), Nagel (1983). A similar approach is described in Bajcsy & Kovacic (1988) where a spectral parameterization of the deformation is also used. More recent developments involving more complex penalty terms and requiring much more computation can be found in Hallinan et al. (1999), Christensen et al. (1996), Grenander & Miller (1998).
Figure 5.6: Top left: An axial MRI brain scan of one patient, which serves as the prototype image. Top Right: An axial MRI brain scan of another patient at the same level. Bottom Left: The deformed prototype image. Bottom Right: The residual. Middle panel: The displacement field $U$. 
CHAPTER 5. IMAGE WARPING

Figure 5.7: The outcome of the deformable image algorithm using all coefficients from the start.

**Homeomorphisms**

In the methods described here the deformations is not guaranteed to be one-to-one. A number of people have investigated the possibility of constraining the deformations to be invertible homeomorphisms of the image domain onto itself. One approach penalizes large derivatives both of the mapping \( \phi \) and of its inverse \( \phi^{-1} \), see Hallinan et al. (1999). The derivatives of the \( \phi^{-1} \) are computable without actually inverting \( \phi \) using the inverse function theorem. The penalty term involves ratios of derivatives. In this setup it only makes sense to work with the lattice parameterization, the power of the coarse to fine algorithm is lost and the computation becomes very intensive. Another approach exploits the fact that the flow of a continuous field is necessarily a homeomorphism at any time instant, due to the uniqueness properties of solutions to ordinary differential equations, see Grenander & Miller (1998) and references therein. The mathematics is quite involved but the final practical implementations is essentially the same as the image deformation algorithm described in this chapter, except that at each iteration the prototype image is changed according to the current mapping, i.e. \( F_k(x) = F_{k-1}(\phi_k(x)) \). This is related to the procedure described in section 5.5 on linearization. Note that allowing the prototype image to update at each step can lead to drastic changes in the image, without paying a high penalty. The most mathematically complete treatment of this issue is to be found in Trouvé (1998). The diffeomorphisms are treated as a Lie group and the variational formulation is done intrinsically in the Lie group in terms of the associated Lie algebra of smooth vector fields. This yields an interesting modification to the algorithms presented in Grenander & Miller (1998), where the basis functions are implicitly deformed together with the prototype image.

**Photometric invariance**

In Hallinan et al. (1999) a deformable image model is used to register prototype face images to data images. The algorithm uses intensity based data models and the prototype image is obtained by averaging images of several registered and normalized faces. Photometric invariance is addressed by taking several prototypes in the form of principle components of the population of faces imaged at a wide range of illuminations. The images used are very high resolution (512 \times 512) which are carefully normalized, and computational efficiency is not discussed. A question arises here, how much of the photometric invariance is ‘learned’ by observing examples of the object lit in a variety of lighting conditions, and how much can be hard wired
CHAPTER 5. IMAGE WARPING

Figure 5.8: Two more experiments with the coarse to fine deformable image algorithm on MRI axial scans. Left: data. Right: deformed template.

into the algorithm by using photometric invariant features as in the Bernoulli model described above. The answer to this question is not necessarily unique and depends on the type of implementation one is seeking, what type of architecture, how efficient should the algorithm be, and how fast is learning expected to occur. An advantage of the Bernoulli model is the systematic and simple form of parameter estimation, namely the training of the probability maps. This appears to be more robust than taking means of pre-normalized images. There is also the benefit of having some form of model of the data off the object.

Principal component bases

Also related to training is the choice of function basis. The standard bases will not provide an optimal parameterization of the large scale deformations typical of the object class. For example in the case of hands, the individual rotation of the fingers relative to the palm are not easily described in terms of standard bases. One approach to solving this problem has been to calculate the deformations for a training set and then find ‘eigen-deformations’, see for example Hallinan et al. (1999), in the context of faces. In other words the principal components of the deformations, computed on the training set of images, are used as an orthogonal basis of deformations. This basis, if computed from the true population of deformations, should provide the most efficient form of parameterization in the least square sense. Of course in order to estimate the principal components in the first place, it is necessary to somehow extract deformations from a large sample, using other bases or parameterizations.

Principal components will not yield a basis allowing for fast forward and backward transforms. However
if only a small number of components are to be used, substituting the fast forward and backward transforms with explicit quadrature will not significantly slow down the algorithm. It is also possible to compromise and choose an optimal wavelet packet for the given training sample. This will not be as economical a representation as the principal components basis but will provide a good approximation and will come along with a fast transform algorithm.

In any case there remains much research to be done into the problem of accurately marking out the components of a face, i.e. detecting detailed instantiation information, despite this being a trivial task for our visual system. The issue of efficient and automatic detection of coarse instantiation information of faces, including pose parameters, in a large scene is discussed in Chapters 6 and 8.

**Mutual information cost function**

In recent years a new type of data term has been proposed which deals with the situation where the pixel intensities for similar structures are not the same, see Viola & Wells (1997), Kim et al. (1997). This occurs in medical imaging when two different modalities are used to image the same structure, for example a PET image and an MRI image of the brain. In this setting the deformations are typically small however the least square distance between pixel values is entirely inadequate. The proposed solution constructs a bivariate histogram of gray levels - for each pixel, two gray level values from the two images being matched. The mutual information between the two distributions described by the histogram is computed. The algorithm tries to modify the deformation in order minimize the mutual information. In the algorithm described above we assumed that given the deformation $\phi$ the image intensities are independent with some distribution $f(\cdot|F(\phi(x)))$, where $f$ is Gaussian with mean $F(\phi(x))$ and fixed variance. Whatever distribution is chosen the log-likelihood has the form $\sum_{x \in L} \log f(I(x)|F(\phi(x)))$. If $f$ is not known ahead of time, it can be estimated from the empirical distribution of $I(x)$ at all points with the same value of $F(\phi(x))$. If this estimated is plugged in the log-likelihood the result is precisely the mutual information of the empirical bivariate distribution $(I(x), F(\phi(x))), x \in L$. In other words the mutual information cost function tries to simultaneously estimate the deformation $\phi$ and the conditional distribution of $I(x)$ given $F(\phi(x))$. If the joint distribution is estimated using a smoothing kernel the cost function is again differentiable in $\phi$, (see Hermosillo et al. (2001)).

The problems arising with regards to the two dimensional models are similar to those mentioned in the discussions regarding the one dimensional models. Initialization is crucial due to the non-linear nature of the cost function. This motivates the sparse models described in the next several Chapters.
Chapter 6

Sparse models: formulation, training and statistical properties

In previous chapters we have described a number of deformable models, all of which require some type of user initialization in terms of pose parameters, such as scale and location. The models are too complex for automatic pose estimation to be efficient. In this chapter we construct sparse models which can be detected directly, and very quickly, with no user initialization. These models involve relatively small collections of points which typically correspond to landmarks of some type on the object. When these points are identified in the image, namely the sparse model is detected, an initial state can be determined for any of the algorithms described in previous chapters. In terms of constructing these sparse models, some of the landmarks are easily defined by the user, representing a clearly defined local structure in the image, on any instance of the object. For example in figure 6.3 we show several landmarks on an axial MRI image of the brain which are chosen by the user. However other landmarks may only be found through training on a collection of examples of the object and do not necessarily have a very clear semantic definition.

In both cases each landmark has a certain characteristic local topography in terms of the image surface. Image 6.1 (left) shows the level curves around the tip of the left ventricle in an axial MRI brain scan. Imagine a binary local feature which is present at the tip of a ventricle for a large proportion of the images of the object. The feature must characterize the local topography of the image in the neighborhood of the landmark. It would be impossible to find such an operator which is on only at the tip of the left ventricle, namely with no false positives. See for example the level curves in a neighborhood of a sulcus in figure 6.1 (right). Otherwise put, parts of objects are not clearly identifiable at the local level. It is only after the global object is identified that local information can be disambiguated. It is however possible to find a local feature which is on at the tip of the ventricle with very high probability, and has a low density of false positives.

Figure 6.1: Left: Contour plot of neighborhood of frontal horn. Right: Contour plot of neighborhood of a sulcus.
positives, namely is not on in many other places in a typical MRI image.

Clearly one local feature associated to one landmark is insufficient to detect the object and its components. As mentioned there will necessarily be false positives, there is also the possibility of false negatives at the correct location. Furthermore, this one feature does not describe the instantiation of the full object but only a small part. We therefore require a collection of landmarks each with an associated local feature. A detection consists of finding instances of some or all of these landmarks in the image, which are spatially arranged in a way consistent with their expected arrangement on the object.. Each of the local features should be invariant within the range of consistent arrangements. For example if we expect to observe the anatomies in image 6.3 in a range of ±25% scaling, we would expect the local feature detecting the tip of the left ventricle to be ‘on’ at the appropriate location at all such scales. These notions are made more precise in the following section where a statistical model is again formulated in the framework of the discussion of Chapter 2.

The question is then, do local features exist which are stable on the object and of low density on the background, from which to construct such models? Two such families of local features are defined in sections 6.3 and 6.4 which are subsequently used in the detection algorithms of Chapters 7 and 8. The second family has a number of free parameters, and hence offers quite a wide range of choices in terms of the complexity and statistics of the features. A detailed study of these statistics as a function of some of these parameters is presented in section 6.5. The main conclusion which emerges from this analysis is that a wide range of possibilities exists to define features which are indeed stable on object and low density on background. Moreover it is possible to provide reasonable predictions on the density of these features, and the number of false detections of the models constructed with these features.

### 6.1 From deformable models to sparse models

Sparse models can be motivated as coarse approximations to the more complex deformable models described in earlier chapters. In the context of random \( \mathbb{L} \mathbb{E} \mathbb{X} \) images there is a complete image synthesis model; we know precisely the random mechanism producing the images. This a very rare luxury but helps illustrate some important points. Let \( \mathcal{A} \) be the set of admissible linear transformations, and \( \mathcal{Y} \) the set of admissible non-linear deformations. Random elements \( A \in \mathcal{A} \) and \( v \in \mathcal{Y} \) are drawn according to some probability and applied to the prototype image \( I_{\text{proto}} \) generating an image defined by \( vA I_{\text{proto}}(z) = I_{\text{proto}}(A^{-1}v^{-1}z) \), for each \( z \in G \). Each such image is then placed at a random location in the lattice \( L \). For simplicity let \( \mathcal{A} \), the range of admissible poses, be limited to a range of scales \([1, S] \) for say \( S = 2 \). Denote by \( \mathcal{I} \) the set of all images \( vA I_{\text{proto}}, v \in \mathcal{Y}, A \in \mathcal{A} \), all presented in the reference grid \( G \). It is in principle possible to check for each location \( x \in L \), each scale \( A \in \mathcal{A} \), and each \( v \in \mathcal{Y} \) whether a deformation under \( v \) of \( A I_{\text{proto}} \) is present at \( x \). This is equivalent to checking at each subgrid \( x + G \) of \( L \), if the data there matches any of the images in the set \( \mathcal{I} \), above some prescribed threshold, using some prescribed distance function. The distance function could be defined in terms of the sum of squares of the differences, or the match of the binary feature maps extracted from the images. This is clearly a daunting computation.

A simple way to speed up the above computation at each location is to let the data guide us in finding an optimal deformation in \( \mathcal{Y} A \), using gradient descent or some other optimization technique. However given the range of scales in \( \mathcal{A} \) it is quite unlikely that such a procedure would indeed find the global optimum. We therefore define a more restricted range of scales \( \mathcal{A}' = [1, s] \) with say \( s = 1.25 \), and let \( \mathcal{I}' \) be the smaller resulting set of images obtained by applying elements of \( \mathcal{Y} A' \) to \( I_{\text{proto}} \). Optimization in the reduced set of deformations is more reasonable, and is precisely the approach taken in Chapter 5, both for the Gaussian model which yields a least squares cost function and for the Bernoulli model.

The entire range of scales \([1, S] \) can be covered by subsequently subsampling the image at resolutions \( S/s, S/(2s), ..., S/(ks) \) where \( k = \lceil S/s \rceil \) and running the same procedure again at all locations. The existence
of local minima, namely the non-convexity of the associated cost function over the restricted range of poses is still a significant problem, and even with the advantage gained by optimizing, as opposed to brute force matching, the entire computation at all locations \( x \in L \) and \( k \) resolutions is still immense.

Assume a set of \( n \) binary local features, \( X_i, i = 1 \ldots, n \) can be defined, each of which is always on at a particular location on the object for images in \( I' \) and are quite rare on the object. These features are invariant to the smaller range of scales and the deformations in \( \Upsilon \). Specifically, a location \( z_i \) on the reference grid is associated to each feature \( X_i \), and if an image from \( I_0 \) is located at a point \( x \in L \) at a particular scale \( A \in A' \) and deformation \( v \in \Upsilon \), then \( X_i(x + vA z_i) = 1 \) for all \( 1 \leq i \leq n \). Now define a set of constraints \( \Theta \subset L^n \) on arrangements of these features which is consistent with \( \Upsilon A' \) meaning simply that any instantiation of the form \( (x + vA z_1, \ldots, x + vA z_n) \) is in \( \Theta \), for \( x \in L, v \in \Upsilon, A \in A' \). Otherwise put, the presence of an arrangement \( \theta \in \Theta \) of these features is an invariant property of the images in \( I' \). Now take an image and find all locations of the \( n \) local features. If there is an efficient way of finding consistent arrangements from \( \Theta \) and if the number of such arrangements is low, we have greatly reduced the computational load. The more intensive deformation algorithms need only be applied at these candidate instantiations.

How should the invariant local features be defined? These features need to flag local structures invariant with respect to deformations \( vA \) with \( v \in \Upsilon \) and \( A \in A' \). For example such a feature would always be on at the top ending of the \( E \) symbol used in Chapter 2. The most elementary features, in terms of which we have already defined a cost on deformations, are the coarse oriented edges. These are present in a stable way in such local structures, but are quite frequent in generic images.

A group of neighboring edges characteristic of a certain part of the object will typically maintain their relative arrangement as the object deforms smoothly. Thus features defined in terms of a flexible edge arrangement, will still be invariant to local deformations of the structures of the object. On the other hand such features are much less frequent in the background, potentially reducing the computation of detections and the number of false positives. If the edges are defined to be invariant to photometric transformations, the edge arrangements inherit this property. This is crucial for real images where the gray level maps are not as homogeneous as in the synthetic \( \LaTeX \) scenes.

How to define \( \Theta \)? Set simple constraints on the locations of the points in \( \theta \) either relative to each other or relative to a location \( x \) in the image grid, in terms of regions where we expect to find the appropriate features. The features can be anywhere in the respective region for the constraint to be satisfied. This slack in the definition compensates for the fact that we rarely have a precise definition of \( \Upsilon \) which would lead to a more specific definition of \( \Theta \). There is an explicit OR-ing being performed over all locations in the region, indeed this disjunction is the key to the invariance of the model. This is the advantage of using binary features, invariance naturally translates into an OR-ing over certain regions. This property is also exploited in the context of the classification algorithms described in Chapter 9.

Finally in this simplified model we have a straightforward mechanism to deal with occlusion and features which are not pure invariants. It is unrealistic to expect non-trivial features to always be present in the proper part of the object. Instead of looking for all local features look for a sufficiently large subset in a consistent arrangement. Thus the sparse models can be viewed as coarse approximations to the more complex ones, with the advantage of explicitly dealing with occlusion and photometric invariance, and leading to efficient detection algorithms.

### 6.2 Statistical model

Once more we formulate a model following the general recipe outlined in Chapter 2. Suppose that a collection of binary local features \( X_i, i = 1 \ldots, n \) and an associated collection of landmark locations \( Z = (z_1, \ldots, z_n) \), are identified for the object class on the reference grid. The collection of locations represents the template for
the sparse model. In section 6.3 we explain how these feature-location pairs are identified through training.

**The prior - admissible instantiations**

Without loss of generality assume

\[ z_c = \frac{1}{n} \sum_{i=1}^{n} z_i = 0. \]

We identify a set \( \Theta^{(0)} \) of admissible instantiations of the object with center at the origin, i.e.

\[ \theta_c = \frac{1}{n} \sum_{i=1}^{n} \theta_i = 0, \]

in particular \((z_1, \ldots, z_n) \in \Theta^{(0)}\). The set \( \Theta \) of admissible instantiations in the image lattice \( L \), is defined as those elements \( \theta = (\theta_1, \ldots, \theta_n) \), for which \( \theta_i = x_i + \theta_i' \), \( i = 1, \ldots, n \) for some \((\theta_1', \ldots, \theta_n') \in \Theta^{(0)}\), namely \( \theta_c = x_c \). The set \( \Theta \) consists of all translations of configurations in \( \Theta^{(0)} \), and can be identified with the set of pairs \((x_c, \theta') \in L \times \Theta^{(0)} \). This identification is not unique, two pairs can correspond to the same element in \( \Theta \).

One form of the set \( \Theta^{(0)} \) which is used in Chapter 8 has the form

\[ \Theta^{(0)} = \{ (\theta_1, \ldots, \theta_n) : \theta_i \in Az_i + C, A \in \mathcal{A} \}, \quad (6.1) \]

with \( \mathcal{A} \) some subset of the invertible 2d linear transformations and with \( C \) some neighborhood of the origin. The larger \( C \) the looser the constraints on \( \theta \), since the points can move independently anywhere within \( Az_i + C \). A uniform prior is used on this set. On the other hand, in Chapter 7 the set \( \Theta \) and the prior are defined directly in terms of hard constraints and penalties on the shape of triangles defined by triples of points in \( Z \).

**Likelihood and posterior**

The data model is written here for the transformed data \( \hat{I}(x) = (X_1(x), \ldots, X_n(x)) \) for \( x \in L \). We assume that if an object is instantiated at \( \theta \in \Theta \), the probability that feature \( X_i \) is on at \( \theta_i \), i.e. \( X_i(\theta_i) = 1 \), is at least \( p_o \), for some predetermined probability \( p_o \). As discussed in the previous section, this implies that there must be some interaction between the definition of \( \Theta \) and the definition of \( X_i \). For example it is very hard to find local features of interest which are invariant to very large ranges of scaling. Therefore one seeks to limit the range of scales covered by \( \Theta \). The full range of scales at which the object will be detected is obtained by rerunning the same algorithm at a small number of lower resolution versions of the original image, as described in section 6.1. In the sequel assume for simplicity that \( P(X_i(\theta_i) = 1|\theta) = p_o \), for all \( i = 1, \ldots, n \).

Let \( p_b \) denote the probability of detecting a feature \( X_i \) at a point \( x \) which is off the object, or even on the object but at the wrong location. Assume that \( p_b << p_o \) and that \( p_b \) is also the same for all local features \( X_i \). We again make an assumption of conditional independence. Given one object is present in the image at instantiation \( \theta \in \Theta \), all variables \( X_i(x), x \in L, i = 1, \ldots, n \) are independent. Now write the likelihood of the transformed data given an instantiation \( \theta \) of the object as

\[
P(\hat{I}(x), x \in L|\theta) = P(X_i(x), x \in L, i = 1, \ldots, n|\theta) = \prod_{i=1}^{n} \left[ p_o^{X_i(\theta_i)}(1-p_o)^{(1-X_i(\theta_i))} \right] \prod_{x \neq \theta_i} X_i(x) \left(1 - p_b \right)^{(1-X_i(x))}. \quad (6.2)
\]
The notation \(P(\cdot | \theta)\) means conditional on an object present at instantiation \(\theta\). Let
\[
P_0(X_i(x), x \in L; i = 1, \ldots, n) = \prod_{i=1}^{n} \prod_{x \in L} p_b^{X_i(x)}(1 - p_b)^{(1 - X_i(x))}, \tag{6.3}
\]
denote the probability of the data given no object is in the scene. Define
\[
\rho_1 = \log \frac{p_o}{p_b} > 0 \quad \text{and} \quad \rho_2 = \log \frac{1 - p_b}{1 - p_o} > 0.
\]
Divide 6.2 by 6.3, multiply by the prior and take logs to obtain the log-posterior given the data as
\[
\log P(\theta | \hat{I}(x), x \in L) = \log P(\theta) + n_y(\theta)\rho_1 - (n - n_y(\theta))\rho_2 + C, \tag{6.4}
\]
where \(n_y(\theta)\) is the number of features for which \(X_i(\theta_i) = 1\) and \(C\) does not depend on \(\theta\).

The posterior is thus formulated as a simple function of the number of ‘hits’, \(n_y(\theta)\) at instantiation \(\theta\), and the prior \(P(\theta)\). It is possible to write a similar form for the posterior in the more general case where \(P(X_i(\theta_i) = 1 | \theta)\) varies with \(i\). At this point however we are constructing a rather crude model, which will quickly identify a small number of candidate instantiations, the only issue of importance is then the order of magnitude of \(p_o\), and hence these probabilities are assumed to be the same.

Compared to the deformable curve model, which is very similar, here the instantiation \(\theta\) determines a fixed number of points at which the respective features are evaluated, whereas in the former model, the features were evaluated all along segments which could be of variable length. In the current model the features are more complex and of lower density. This compensates for the simpler form of the template. In some sense the current model can be viewed as ‘zero’ dimensional. Features are evaluated at a discrete set of points, not along line segments or in a 2-d domain. In the deformable curve model there was a natural way to rotate the model to any angle, since the features depended directly on the angles of the segments. The models described here, can in principle be rotated to cover larger ranges of rotations, but this is not as straightforward.

**Multiple objects**

When multiple objects may be present the model becomes more complex involving a term for each possible number of objects. The unknowns are now both the number of instantiations \(k\) and their values \(\theta^{(1)}, \ldots, \theta^{(k)}\). The set of unknowns is defined as \(\Sigma = \bigcup_{k=0}^{K} \Theta^k\), where \(\Theta^k\) is the set of \(k\)-tuples of elements of \(\Theta\), with \(K\) some upper bound on the number of possible objects in the image. There is no particular preference for any one of these states, namely the number of objects or their location in the scene. We therefore include a uniform prior on \(\Sigma\). This uniform prior implicitly favors a larger number of detections since the size of \(\Theta^k\) grows with \(k\). This is consistent with the idea that at this stage we want to avoid missing any of the objects present in the scene, perhaps at the risk of a number of false positives. Ultimately a more refined and intensive analysis should be used to distinguish real detections from the false positives. Assigning higher prior probability to \(\Theta^k\) for smaller \(k\), would increase the risk of missing real objects.

This uniform prior also imposes no constraints on the relative arrangement of the instances. In some cases more information may be available and more structure can be introduced into these prior assumptions, this is beyond the scope of this book. Given \(k\) instances at \(\theta^{(1)}, \ldots, \theta^{(k)}\) we again assume conditional independence of the features at all pixels, with probability \(p_o\) for finding \(X_i\) at \(\theta^{(l)}, l = 1, \ldots, k\) and probability \(p_b\) of finding it anywhere else. The above assumptions lead to the log-posterior on \((\theta^{(1)}, \ldots, \theta^{(k)})\) given by
\[
\log P((\theta^{(1)}, \ldots, \theta^{(k)}) | \hat{I}(x), x \in L) = \sum_{i=1}^{k} \left[ n_y(\theta^{(i)})\rho_1 - (n - n_y(\theta^{(i)}))\rho_2 \right] + C. \tag{6.5}
\]
From this expression it follows that to maximize the posterior we need to find all those $\theta \in \Theta$ for which the term in brackets is positive, namely find all $\theta \in \Theta$ for which

$$n_y(\theta) > n \frac{\rho_2}{\rho_1 + \rho_2} \triangleq \tau_m.$$ 

The number of such $\theta$’s provides the number of objects present in the image. As we will see below, since the conditional independence model is not entirely accurate, we use training data to find a threshold $\tau$ which is more conservative than $\tau_m$, ensures finding all instances of the object, and perhaps some additional false positives. More detailed processing is then necessary to determine if indeed the object is present at the detected $\theta$’s or not.

**Computation**

Two approaches for optimizing the posterior are described in Chapters 7 and 8. In the first we still assume only one object is present in the image. Local features are chosen for a collection of landmarks on the object and their parameters set so that $p_o$ is very close to 1. Unless all features are present, i.e. $n_y = n$, the posterior is 0. A list $S_i$ generated of all locations of feature $X_i$ in the image. It then remains to find the mode of $P(\theta)$ over all admissible $\theta$ such that $\theta_i \in S_i$. This can be done efficiently using dynamic programming when $P(\theta)$ is assumed to have a decomposable form, see Chapter 7. In assuming that $p_o = 1$ this method depends on the restrictive assumption that *all* features of the model are found on the object, which is often not the case.

In the second approach, described in Chapter 8 a collection of local features of moderate probability, say $p_o = .5$ is identified, at a collection of locations in the reference grid. The set $\Theta$ is defined as in equation 6.1, and since several objects can be present, following the discussion above, it is necessary to find $\theta \in \Theta$ for which the number of $\theta_i$’s with $X_i(\theta_i) = 1$ is above some threshold. This is done in two steps, the first of which involves a coarse approximation of $\Theta^{(0)}$ in terms of a product set and detection of candidate centers for the objects. In the second step, at each candidate center $\theta_c$, a scale is estimated and perhaps other pose parameters, and as many points of the model as possible are matched to the data at the estimated pose.

In the remainder of this Chapter we focus on the description of the local features used for the sparse models, methods for training the relevant parameters and some statistical properties of these features.

**6.3 Local features: comparison arrays**

There are numerous possibilities for defining local features for image data. The most common filters found in the literature employ a variety of linear filters related to differential operators. See for example Malik & Perona (1990), Wiskott et al. (1997). It is more difficult to explicitly incorporate geometric and photometric invariance with such filters. It is also more difficult to study the statistical properties of continuous valued variables as opposed to the binary valued features described below. We thus prefer to use binary features involving simple comparisons of pixel intensity differences, with which photometric and geometric invariance are easy to implement. These local features will be more complex extensions of the ridge features defined for the deformable curve model in section 4.1 and the edge features in the Bernoulli model for deformable images in section 5.4. The increased complexity is needed since the efficiency of the algorithms depends very much on the background probability $p_b$ being small. The density of the features of sections 4.1 and 5.4, in a generic image can be very high.

Define an $m \times m$ array $M$, of 1s, −1s and 0s. The *sign* of the difference between the intensity at a given pixel $x$, and the intensity at each of the pixels in its $m \times m$ neighborhood $N_m(x)$ is calculated, to yield an $m \times m$ array $A(y) = \text{sign}(I(x) - I(y))$, $y \in N_m(x)$ of 1s and −1s. A contrast threshold $\kappa$ can be
introduced and any difference of magnitude less than \( \kappa \) is set to 0. If the percentage of matched 1s and \(-1\)s between \( M \) and \( A \) are above prescribed thresholds the pixel \( x \) is considered a candidate for the corresponding landmark. The 0s region in \( M \) contains pixels where the values of \( A \) are ignored, this allows for a degree of slack which is important in obtaining the type of invariance mentioned above. In figure 6.2 are the sixteen \( 11 \times 11 \) comparison arrays used in creating the models. White means intensity greater than center value, dark means intensity less than center value, grey means ignored.

![Figure 6.2: 16 comparison arrays. White means intensity greater than center value, dark means intensity less than center value, grey means ignored.](image)

Loosely speaking these 16 comparison arrays identify ‘turns’ of level curves of the image (see figure 6.1) pointing in eight different orientations, and two gradient flow directions per orientation. These can be viewed as coarse ‘high curvature’ detectors. Recall that we are seeking features with low \( p_b \), and given the frequent occurrence of lines and smooth curves in images, ‘low curvature’ detectors or line detectors would produce a relatively large number of background instances. On the other hand high curvature points are quite convenient for representing stable local structures of interest on object.

These operators are robust to rather significant variations in pixel intensities which are bound to occur between images as well as to smooth deformations of the local topography of the landmark neighborhood in the template image. They are also robust to a certain range of scales and small rotations. Recall that this was an important requirement on the properties of the operators \( X_i \).

**Training**

The user points out a collection of landmarks \( x_{i,t}, i = 1, \ldots, n \) on each of a small number of training images \( t = 1, \ldots, T \). In figure 6.3 are two training images of axial MRI scans. Eight points have been marked on similar anatomies, the correspondences are provided by the numbering. For each landmark we seek the comparison array which, with highest threshold, has no false negatives on the training set. Higher thresholds mean fewer instances of the feature in the background. Note that here we are simultaneously choosing the local feature and estimating the relevant parameters, i.e. the thresholds. Let \( M_j, j = 1, \ldots, J \), be the collection of arrays, For training image \( t \), let \( \tau_{i,t,j,+} \) and \( \tau_{i,t,j,-} \) be the number of matches of \( A_j(x_{i,t}) \) to the positive and negative regions of \( M_j \) respectively. Let

\[
\tau_{i,j,+} = \min_t \tau_{i,t,j,+} \quad \tau_{i,j,-} = \min_t \tau_{i,t,j,-} \quad \text{and} \quad \tau_{i,j} = \tau_{i,j,+} + \tau_{i,j,-}
\]

These are the lowest matches observed with \( A_j \) at the points \( x_{i,t}, t = 1, \ldots, T \). Let \( j^* = \text{argmax}_j(\tau_{i,j}) \), and set the array corresponding to landmark \( i \) to be \( M_{j^*} \) with thresholds \( \tau_{i,j^*,+} \) and \( \tau_{i,j^*,-} \). The corresponding model locations \( z_i \), which define the template for the sparse model, are chosen on one of the training images, (for example the left panel of figure 6.3,) or through some averaging procedure. The result is a list of locations \( z_1, \ldots, z_n \), and an associated list of local features \( X_1, \ldots, X_n \), defined by comparison arrays each with two thresholds, \( M^{(i)}, \tau_{i,+}, \tau_{i,-}, i = 1, \ldots, n \).

In the example shown in figure 6.3, the first array is used for landmark ‘0’, the second for landmark ‘1’ the third for landmark ‘2’, the fourth for landmark ‘5’, the fifth for landmarks ‘4’ and ‘7’, and the sixth for landmarks ‘3’ and ‘6’. The lower panel of the figure shows a closeup on the tip of the right ventricle in four
different images. All were detected by the first array associated to landmark 0. This illustrates the flexibility of this type of local feature to various deformations of the underlying local topography of the image surface. In using comparisons of gray level intensities, and at times a low minimum contrast threshold, these features detect the same topography at very different contrasts, in other words photometric invariance is built in to the definition. In figure 6.4 we show the locations of all instances of all 6 features in an MRI image. Compared for example to the simple oriented features of Chapter 4, shown in figure 4.1, the density is much lower.

6.4 Local features: edge arrangements

We now define a larger and more flexible family of local features, with several free parameters allowing us to control the probabilities \( p_o \) and \( p_b \). A systematic analysis of these probabilities is given in section 6.5. These features are also functions of pixel intensity differences.

Start with the edges defined in section 5.4, and in terms of which the Bernoulli data model for deformations was defined. These types of edges are rather frequent in generic images. In figure 6.5 we show all instances of one vertical edge type in two images with faces which will be processed in Chapter 8, (see figure 8.5.) Only one edge \( (n_e = 1) \) is allowed in each \( 2 \times 2 \) block.

It is not efficient to base uninitialized detection algorithms directly on the locations of these edges, although it is definitely possible to do so. It turns out more efficient to produce features of lower density as functions of the initial edge map. Each feature is defined in terms of a “central edge” of some type \( e_0 \), and a number \( n_e \) of other edge types \( e_1, \ldots, e_{n_e} \) which are constrained to lie in specific subregions \( R_1, \ldots, R_{n_e} \), in the neighborhood of the location of the center edge. We refer to the number \( n_e \) of additional edges as the
complexity of the arrangement. In figure 6.6 two examples of such local edge arrangements are shown with $n_r = 2$.

The local feature is detected at a location, if the central edge is found at that location, and if an instance of each of the $n_r$ edge types is found in the corresponding region. The family of possible subregions is denoted $\mathcal{R}$. The sizes of the subregions are all approximately the same. The subregions can be wedge-shaped as indicated in Figure 6.6, or squares in the neighborhood of the center. OR-ing - allowing the $n_r$ edges to float in their respective subregions - is how geometric invariance is explicitly introduced at this level as
demonstrated in section 6.5 below. The family of $\mathcal{R}$ of regions used in the experiments reported in Chapter 8 are shown in figure 6.7, alongside a family of larger wedges which is studied for comparison in section 6.5.

Training
In contrast to the previous section, here we do not assume salient landmarks have been pointed out by the user. The locations of interest are identified as part of the training process. In this case however it is necessary to train on registered data. Three pose reference points $p_1, p_2, p_3$ are chosen on the reference grid. On each training image $t = 1, \ldots, T$ the user marks three anchor points which will be matched to the three pose reference points, determining an affine map $A_t$ from the training image $t$ to the reference grid.

Two-edge arrangements
For two-edge arrangements ($n_r = 1$), i.e. one edge in addition to the center edge, the number of possible arrangements $J$ is on the order of several hundreds. Locations of all arrangement types in each training image $t$ are found and then registered to the reference grid. Specifically a feature $j$ detected at location $x$ adds a count to $F_j(A_t x)$, for each $j = 1, \ldots, J$. Choose a probability threshold $\rho \sim .5$ and an upper bound $n$ on the number of local features. In each disjoint $3 \times 3$ box $C$ on the reference grid find the two-edge
arrangement with highest frequency in the training data, i.e.

\[ F_C = \max_{j=1, \ldots, J} \max_{C} F_j(z). \]

If the relative frequency \( F_C/T > \rho \) the corresponding two-edge arrangement is added to the model together with the location \( z_C \) corresponding to the center of the box. All such feature location pairs are identified and, if more than \( n \) are found, a random sample of \( n \) are kept. This yields a list \( X_i, z_i, i = 1, \ldots, n \) of two-edge features and associated model locations.

This is the procedure used in Chapter 11 to train object representations in the framework of a neural network implementation of these models, see Figure 11.4. A similar procedure was used to train the edge templates in section 5.4, except that all locations were kept no matter what the relative frequency.

**More complex arrangements**

When the complexity of the edge arrangements is higher, i.e. \( n_r > 1 \), the number of possible arrangements is very large and it is impractical to precompute all locations on all training images. A greedy search is implemented instead, seeking stepwise increments in the complexity of features with high frequency on the object class. Edges are first detected on each training image and their locations are registered to the reference grid using the affine map \( A_t \). Note that we do not register the images themselves but the detected locations of the edges. In each disjoint \( c \times c \) box of the reference grid the 2-edge arrangement with highest count is found. All instances of the chosen 2-edge arrangement in the box are recorded for each image. Then a loop over all possible additions of one edge/region pair selects the one with highest count in the box, until \( n_r \) edge/region pairs are found. The following algorithm provides the details.

**Algorithm 6.1: Sparse model - training edge arrangements**

Fix \( n_r \), and \( R \), the family of regions, and choose \( \rho < 1 \), (typically .5) and let \( T \) denote the training set.

1. Set feature counter \( I = 0 \). Loop over disjoint \( c \times c \) (say \( c = 3 \) or \( c = 5 \)), boxes on the reference grid. For each such box \( C \):

   (a) Detect the eight edge types in each training image and register the locations using \( A_t \) to produce registered edge maps \( E_t^{(1)}, \ldots, E_t^{(8)} \) for each \( t = 1, \ldots, T \).

   (b) For each possible triple \((e_0, e_1, R_1)\), where \( e_0, e_1 \) are any possible edge types and \( R_1 \in R \), count the number of training points in \( T \) for which an instance of the triple occurs in \( C \). This means \( e_0 \), the central edge, is located at any point \( x \in C \) and \( e_1 \) is located anywhere in \( x + R_1 \). Pick the triple with highest count and let \( T_1 \subset T \) denote the set of data points which have an instance of this triple in \( C \). For each data point \( t \in T_1 \), let \( W_{t,1} = \{x_{t,m}, m = 1, \ldots, M_{t,1}\} \) denote all locations of the first edge \( e_0 \) for which the chosen triple was found. Set \( j = 2 \).

   (c) Loop over all possible pairs \((e_j, R_j)\) and count how many data points \( t \in T_{j-1} \) have an edge of type \( e_j \) anywhere in \( x_{t,m} + R_j \) for any one of the locations \( x_{t,m} \in W_{t,j-1} \). Find the pair with highest count and let \( T_j \subset T_{j-1} \) denote the data points which have an instance of this pair. For each data point \( t \in T_j \), let \( W_{t,j} = \{x_{t,m} \in W_{t,j-1} : m = 1, \ldots, M_{t,j}\} \) denote the set of locations of the first edge \( e_0 \) for which the additional pair was found.

   (d) \( j \leftarrow j + 1 \). If \( j < n_r \) goto (c).
2. If \(|T_n|/|T| > \rho\), record the feature

\[ X_I = (e_0, e_1, R_1, \ldots, e_{n_r}, R_{n_r}) \]

at the center \(z_I\) of \(C\). For all data points in \(T_n\), there exists a location \(x \in C\) at which an instance of \(e_0\) is present and an instance of \(e_k\) is present in \(x + R_k\) for each \(k = 1, \ldots, n_r\).

Set \(I \leftarrow I + 1\).

3. Move to the next box \(c \times c\) and goto 1.

If we initially set out to pick \(n\) local features and more have been found choose a random sample of size \(n\).

In the top row of figure 6.8 we show images representing high frequency locations of the registered edges for faces. We used 300 faces of the Olivetti data base, consisting of 10 views of 30 people. For each of four edge types, the two vertical and two horizontal, a pixel is on if more than .7 of the faces had an edge of that type in the 5 \(\times\) 5 neighborhood of that pixel. In the bottom row of the same figure we show the same information for four edge arrangements from the face representation given in figure 6.9 (\(n_r = 3\)). Only locations with proportion over .5 are shown. The model location is shown as a white dot and the three reference points in black. Note how the local features are much more specific to particular parts of the face.

In figure 6.9 we show a graphical representation of 20 local features, with \(n_r = 3\) at their model locations which were used as the sparse face model for the face detection experiments shown in chapter 8. The three black dots represent the three reference points and correspond to the location of the two eyes and mouth. In other words the anchor points identified by the user on each training image were the two eyes and the center of the mouth.

Each local feature is represented by the four edges that define it. The two adjacent rectangles describe the orientation and polarity of the edge. Recall that the edges can have varying locations relative to the center. For example the upper left hand feature in the left hand panel in figure 6.9 has three horizontal edges arranged more or less horizontally and a vertical edge on their left. This appears to capture the curve of the hairline on the left part of the face. The feature directly below that, has two horizontal edges with dark part on the bottom above two horizontal edges with dark part on the top. This captures the area around
the eye, which is typically darker than its surrounding. These edge groupings not only capture contours, but other topographical structures as well. Even a smaller training set of 50 faces (from five people) will yield very similar statistics in terms of the edges and their arrangements. Other representations obtained for other types of objects can be found in Chapter 8. Figure 6.10 gives a sense of the density of these features on the same two images used in figure 6.5, and which will be processed later on in figure 8.5. Locations of the first feature in the face model above are shown.

Figure 6.10: For the two images used in figure 6.5 the locations of the first local feature in figure 6.9.
Determining the threshold $\tau$

In principle, if the local features were independent on the object, and their probabilities were given by $p_o$, a simple calculation using the binomial distribution $B(n, p_o)$ would yield the threshold $\tau$ for some chosen false negative rate $r$. But due to existing correlations between the features this analytic threshold is too high and it is better to estimate a threshold from data. Take a sample of unregistered training images for which the range of scales and rotations covers the prescribed range for the detector, determined by $A$. Choose an acceptable false negative rate $r$, say 5%.

For each training image $I^{(t)}, t = 1, \ldots, T$, let $A_t$ be the affine map taking the three reference points $p_1, p_2, p_3 \in G$ into the three anchor points $x_1^{(t)}, x_2^{(t)}, x_3^{(t)}$, in $I^{(t)}$. Strictly speaking we should find a value of $\tau$ for which 95% of the training images have a sequence $i_1, \ldots, i_\tau$ such that an instance of $X_{i_j}$ is found in the region $A_t z_{i_j} + C$ for $j = 1, \ldots, \tau$. Of course the sequence can be different from image to image. However there is quite a large degree of uncertainty regarding $A_t$ itself. The three anchor points are provided by the user on each image, at times mistakes are made, and at times the precise location of say the center of the eye can be equivocal. We therefore use a looser criterion for determining $\tau$:

Find the largest $\tau$ for which $(1 - r)T$ of the training images have an admissible instantiation $\theta \in \Theta$, with at least $\tau$ indices for which $X_{j}(\theta_j) = 1$, and the center $\theta_c$ is within $D$ pixels from the mean $x_{c,t}$ of the anchor points. The distance $D$ is chosen to be 2 to 3 pixels.

This choice ensures that we do not discount an image which does have at least $\tau$ local features in an admissible instantiation but these do not necessarily fall in the expected regions according to the affine map $A_t$.

6.5 Local feature statistics

The detection algorithms described in Chapters 7 and 8 both rely on the fact that the on-object probabilities of the local features, namely probabilities at the correct location of the object are high, and invariant with respect to a range of deformations, whereas the density of these same features in the background is low. In terms of the statistical model described in section 6.2 testing whether the object is present or not at a specific instantiation $\theta$ is reduced to a simple hypothesis test between two Binomial distributions, $B(n, p_o)$ and $B(n, p_b)$, with observations $X_{i}(\theta_i), i = 1, \ldots, n$. A threshold $\tau$ is determined, and if the number of ‘hits’ is above threshold at a certain instantiation $\theta$ it is declared object, otherwise it is declared background.

In this section we study more closely the statistical properties of the edge arrangements on background and on object. The Binomial model which derives from the conditional independence assumption is an oversimplification; some correlations exist between the features, both on object and on background. However we will show evidence of very consistent patterns in terms of the dependence of these statistics on the parameter settings. In particular false positive densities, although not directly predictable using the conditional independence assumption, can be predicted by fitting some simple linear models.

In studying the statistics of these features we take the subregion $R$ as a family of wedges of the $9 \times 9$ neighborhood of the origin. We use two sizes: Larger regions with 8 - 90 degree wedges, at increments of 45 degrees. Smaller regions with 16 - 45 degree wedges at increments of 22.5, see figure 6.7. For the large regions there are 24 pixels in each region whereas for the smaller regions there are between 10 – 14 pixels. The edges are detected in $2 \times 2$ blocks, as described in section 6.4. In order to study the dependence of the algorithm on edge density we take the maximal number of edges $n_e$, in each $2 \times 2$ block to be either one, or two. All in all there are 4 categories: $n_e = 1, 2$, and large or small wedges, determining the overall density of the local features.

We work with 40 local features identified through training on a face dataset. Very similar behavior
CHAPTER 6. SPARSE MODELS

is observed for features trained for other object types. The 40 edge arrangements were trained using the algorithm 6.4 described above. 40 locations on the reference grid were found where an edge arrangement, defined in terms of a central edge and 6 other edges in its neighborhood, was present in at least $\rho = 30\%$ of the registered edge maps of the training set. There is an ordering on the edges in each of these arrangements, determined by the order in which they were identified in the algorithm. It is possible to define partial arrangements by taking the first $n_r$ edges in the ordering for $n_r = 1, \ldots, 6$. Clearly the partial arrangements will have even higher probability, and will be of higher density on the background. Note that for the given ordering of the edges in an arrangement, as $n_r$ increases the collection of detections of the local feature is a subset of the previous stage. This allows us to study the density of these arrangements as a function of $n_r$. We start with the process corresponding to the center edge of the arrangement. At the next step we keep only those locations where the second edge in the arrangement is found in the appropriate neighborhood, so that we have filtered out the original edge locations. This filtering continues as more edges are added in.

**Background densities**

The density of the features on generic images decreases exponentially in $n_r$ - the complexity of the arrangement. The rate of decay depends on the size of the subregions in $\mathcal{R}$ and on the initial edge density which in our case depends on $n_r$ - the number of edges kept in each disjoint $2 \times 2$ block. 40 random 'background images' including outdoor images from urban and nature scenes, as well as some underwater scenes, were downloaded from the web to obtain these statistics. In addition 9 indoor office images were taken. In figure 6.11 we show the box plots for the log-densities for the four categories as a function of $n_r + 1$ for the outdoor images. The first box corresponds to the density of the edges themselves, i.e. $n_r = 0$. Each of the remaining boxes represents 1600 points, i.e. the distribution of densities over 40 images for the 40 features at the given level of complexity $n_r$.

For all four parameter categories we observe an initial large decrease of the density between $n_r = 0$ to $n_r = 1$. This corresponds to filtering out of edges resulting from random noise, leaving primarily edges which form part of some rudimentary local structure. Subsequently we observe a linear decay on the log scale of the density as a function of $n_r$, which may have to do with a gradual partitioning of the 'space' of local structures.

Of the hundreds of possible arrangements with $n_r = 1$ (two-edge arrangements), defined in terms of a central edge and one additional edge, many can actually be found in real data. Semantically some could be described as a small segment of a contour within a range of curvatures, some could be labeled as a 'slit'. For each center edge type $e_0$ there are $8|\mathcal{R}|$ possible pairs. Many such pairs describe very similar structures due to the slack in the relative locations of the edges. Still one notes that for each center edge type there is quite a large number, say $S$, of mutually exclusive structures determined by the additional edge. Assuming each of these structures is equally likely given the center edge is present, which is obviously an oversimplification, their probability is at most $1/S$ that of the center edge. This factor corresponds to the first drop in density between $n_r = 0$ and $n_r = 1$.

However once the first two edges are determined there is a much smaller number of internally consistent possibilities, representing realistic local structures for the third edge, and hence the slower drop in the density for higher values of $n_r$. Recall that the features studied here were trained on a real object, so that inconsistent arrangements do not get chosen. Indeed if the arrangements were defined by randomly choosing the edge types and the regions in $\mathcal{R}$, the density would decay more rapidly.

We fit a linear model where the response variable is the log-density of the edge arrangement (LDA) in an image. There are two predictors: the log-density of the edges (LDE) in the image, and the complexity of the arrangement, $n_r = 1, \ldots, 6$. Table 6.5 shows 95% confidence intervals for the coefficients of the two predictors as well as the $R^2$ coefficient, for each of the four categories. There are 1600 data points for each of these four regressions, consisting of the 40 local features at each of the 40 images.
We obtain the following four models for the density $DF$ of the feature as a function of the density $DE$ of the edges and complexity $n_r$.

$$DF = DE^{1.28} \cdot (45)^{n_r}$$
$$DF = DE^{1.38} \cdot (57)^{n_r}$$
$$DF = DE^{1.36} \cdot (61)^{n_r}$$
$$DF = DE^{1.23} \cdot (71)^{n_r}$$

There are sizeable overlaps among the confidence intervals for the coefficient of LDE. It appears that the crucial effect comes from the coefficient of $n_r$ namely the base of the second exponent. As the regions in the edge arrangement get larger, or as more edges are allowed per block, the rate of exponential decay decreases. The explanation offered above for these rather striking phenomena is quite heuristic. Hopefully new models on local structures will emerge that provide a more comprehensive explanation.

### Statistics on object

We now turn to the properties of the edge arrangements on object. Specifically those arrangements which were identified through training on the given object class.
### Table 6.1: 95% confidence intervals for the coefficients of the two predictors - log-density of edges (LDE) and complexity ($n_r$), and the $R^2$ coefficient for the regressions.

<table>
<thead>
<tr>
<th>Category</th>
<th>LDE</th>
<th>$n_r$</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_e = 1$, Small wedges</td>
<td>[1.21, 1.36]</td>
<td>[−.79, −.76]</td>
<td>.88</td>
</tr>
<tr>
<td>$n_e = 2$, Small wedges</td>
<td>[1.31, 1.44]</td>
<td>[−.57, −.55]</td>
<td>.87</td>
</tr>
<tr>
<td>$n_e = 1$, Large wedges</td>
<td>[1.29, 1.41]</td>
<td>[−.50, −.48]</td>
<td>.87</td>
</tr>
<tr>
<td>$n_e = 2$, Large wedges</td>
<td>[1.16, 1.30]</td>
<td>[−.35, −.33]</td>
<td>.83</td>
</tr>
</tbody>
</table>

**Probability of features as a function of complexity $n_r$**

In Figure 6.12 we show box plots of the probabilities of the 40 features on object (in this case faces) at the correct location, as a function of complexity, for the 4 categories defined above. For these plots all faces are at reference pose. The feature is considered present at the model location $z$, if its center is anywhere in a small $5 \times 5$ neighborhood $N_5(z)$ of $z$. The first important property that emerges from these plots is the existence of local features with very high probabilities. This is not only a property of the particular face database used here but of all other examples presented in later chapters. If there is a large degree of variability at the local level the probabilities will be lower. This variability could be due to non-linear deformations or to discontinuous changes. For example some people have glasses and some do not. The more variable the ensemble is after pose variation is factored out, the lower the probabilities of the best local features will be, for a fixed set of parameters, and more of these features will be needed to obtain good discrimination from background.

Another property that emerges from the data above is that probabilities of the arrangements, at the correct location in the reference grid, decay very slowly - at most *linearly* as a function of the complexity $n_r$. This is in contrast to the exponential decay of the densities on background. A heuristic explanation for this phenomenon is the following. Assume that the object ensemble is obtained from smooth local deformations of a prototype image, sampled uniformly from a set of deformations $\mathcal{U}$ of size $U$. No major linear transformations are allowed so that all deformed instances of the object are more or less at the reference pose. On the template, at a location with some ‘edge activity’, one can identify an arrangement of say 6 edges. The center edge $e_0$ in the arrangement has an absolute location $z$ in the reference grid. There is a subset $\mathcal{Y}_0$ of deformations for which there is no edge type $e_0$ in the $5 \times 5$ neighborhood of $z$. Assuming $e_0$ is found in $z_0 \in N_5(z)$, for each $k > 0$ there is then a subset $\mathcal{Y}_k$ of $\mathcal{Y}$ of deformations of the template for which the $k$'th edge $e_k$ is not obtained anywhere in $z_0 + R_k$. It is reasonable to assume that the size of these sets $\mathcal{Y}_k, k = 0, \ldots, n_r$ is more or less the same for all $k$, denoted $u$, and does not depend on the location $z_0$ of the first edge. Consequently the probability of the arrangement consisting of the first $n_r$ edges can be bounded from below by $1 - n_r \frac{u}{U}$, which is linear in $n_r$.

This argument holds only if the population is obtained through smooth deformations of a single template. This is rarely the case. However it is reasonable to assume that the population can be obtained from smooth deformations of a number of prototypes, which are similar in some parts and different in others, so that the argument presented above remains valid in spirit.

**Invariance of features with respect to pose**

When large scale deformations are introduced, such as scaling or rotation, we do not expect to find the features at the same location on the reference grid. We now compare the probability that feature $X_i$ is present at location $z_i$, when the object is at reference scale to the probability the feature is present at location $sz_i$ when the object is at scale $s$. Similarly with respect to rotation or other global linear transformations.
Clearly the probabilities change drastically with large rotations because the local features are not rotation invariant in the strict sense. The edges are constrained relative to the center edge in specific regions, and the edges themselves have coarse orientation selectivity. Yet due to the slack in the definition of the edges and their arrangements we expect a limited degree of rotation invariance. The top panel of figure 6.13 illustrates the stability of the probabilities with respect to up-scaling, and down-scaling, for the smaller wedges, at \( n_r = 3 \). The left shows \( n_e = 1 \) the right \( n_e = 2 \). The bottom of figure 6.13 illustrates the invariance with respect to rotation. The data was obtained by applying a linear transformation \( a \) to the face images, extracting the local features in those transformed images, and finding what proportion of the data had the local feature \( X_i \) on, in a small neighborhood of \( a z_i \). Observe that the probability of the features are indeed robust to a large range of poses. As would be expected, for the arrangements defined with larger wedges, the probabilities are even more robust.

**False positive rates**

The two detection schemes presented in Chapters 7 and 8 perform a search for peaks of the posterior. In the two cases a peak occurs at an instantiation \( \theta \) only when all or a certain fraction of the features is present. The question arises as to the probability of false positives, i.e. the probability of having more than \( \tau \) features present in an admissible arrangement \( \theta \) even if there is no object present at that instantiation. We study the false positive behavior assuming the set of admissible instantiations \( \Theta^{(0)} \) is defined through equation 6.1 with \( A \) covering a range of scales of \( \pm 25\% \) and rotations of \( \pm 15 \) degrees. For each value of \( n_e \) we use a set of face images to determine a threshold \( \tau \) which keeps over 95\% of the faces as detailed in section 6.4. We then search the 40 random background images for configurations of at least \( \tau \) features at any location and
at any of the prescribed linear transformations of \( A \). An efficient way to do this search is detailed in chapter 8, here we are only interested in the statistics.

As already mentioned, the statistical model described in section 6.2 reduces to a simple test between two Binomial distributions, \( B(n, p_b) \), and \( B(n, p_o) \) for every possible instantiation \( \theta \in \Theta \). If the model was accurate the probability of a false positive at any given pose could be predicted using the tails of the Binomial distribution \( B(n, p_b) \), or using the normal approximation: 
\[
1 - \Phi\left(\frac{T - \mu (1 - p_b)}{\sqrt{np_b(1-p_b)}}\right),
\]
where \( \Phi \) is the cumulative distribution function of the standard normal distribution. Note that the number of possible instantiations in the present context is hundreds of times larger than the number of locations, since at each location one could have on the order of hundreds of scales and rotation combinations. Detected instantiations typically occur in clusters arising from the same object in the image, and the instantiation with highest number of detected features is chosen to represent this cluster. These considerations make it much harder to predict the density of false positives in terms of a simple probability model. Instead we provide empirical data on the density of false positives per pixel, i.e. total number of detections divided by the area of the image.

In table 6.2, for each value of \( n_r \), we list the average probability of the 40 features on the training set; the empirical threshold \( \tau \), obtained by checking for the highest value for which 95% of the training set of faces is detected (5% false negative rate); the total false positive rate where the total number of detections in all 40 images is divided by the total area. In addition we show the 95% confidence interval for the coefficient of the log-density of the local features (per local feature type) in predicting the log-density of false positives in an image, as well as the \( R^2 \) coefficient. In figure 6.14 the scatter plots of the log-density of false positives against the log-density of the local features are shown for \( n_r = 1, 3, 5 \). The symbol "*" stands for outdoor
CHAPTER 6. SPARSE MODELS

<table>
<thead>
<tr>
<th>$n_r$</th>
<th>$p_o$</th>
<th>$\tau$</th>
<th>False Pos.</th>
<th>95% Coef. intvl.</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.77</td>
<td>25</td>
<td>$3 \cdot 10^{-4}$</td>
<td>[2.28, 3.62]</td>
<td>.82</td>
</tr>
<tr>
<td>2</td>
<td>.72</td>
<td>22</td>
<td>$5 \cdot 10^{-5}$</td>
<td>[1.34, 2.69]</td>
<td>.70</td>
</tr>
<tr>
<td>3</td>
<td>.64</td>
<td>17</td>
<td>$5 \cdot 10^{-5}$</td>
<td>[1.58, 2.64]</td>
<td>.80</td>
</tr>
<tr>
<td>4</td>
<td>.53</td>
<td>11</td>
<td>$2 \cdot 10^{-4}$</td>
<td>[1.88, 2.89]</td>
<td>.84</td>
</tr>
<tr>
<td>5</td>
<td>.39</td>
<td>6</td>
<td>$3 \cdot 10^{-4}$</td>
<td>[1.79, 2.6]</td>
<td>.87</td>
</tr>
<tr>
<td>6</td>
<td>.28</td>
<td>3</td>
<td>$8 \cdot 10^{-4}$</td>
<td>[1.50, 1.89]</td>
<td>.94</td>
</tr>
</tbody>
</table>

Table 6.2: First column: Complexity of edge arrangements in model. Second column: Average on-object probability of the arrangements. Third column: Threshold at 5% false negative. Fourth column: False positive density (number of false positives per pixel.) Fifth, sixth columns: 95% confidence intervals for the regression coefficient of log-density of false positives against log-density of local features in an image. Seventh column: $R^2$ coefficient of the regression.

images and the ‘o’ for the office images. Some points do not appear since there were no false positives. The estimates using outdoor images provide an upper bound, since there is a much higher density of edges in these. In this experiment we used the small wedges and $n_e = 1$.

If the features were indeed conditionally independent as assumed in the statistical modeling, since the probabilities on object decay linearly with $n_r$ whereas the probabilities on background decay exponentially, the larger $n_r$ a lower false positive rate could be obtained for some fixed false negative rate. However due to correlations between the features on object, which increase with their complexity, the threshold $\tau$ decreases faster than the Binomial model would predict and the number of false positives starts to increase.
Figure 6.14: Scatter plots of log-density of false positive vs. log-density of local features, \( n_r = 1, 3, 5 \).

We see that the properties of this hierarchy of local features is stable and predictable both on object and on background. Depending on the task at hand, the computational constraints and acceptable levels of the two types of error, different feature complexities can be used.

We have described two families of local features, the comparison arrays and the edge arrangements, defined in terms of simple operations involving intensity comparisons, and intensity difference comparisons. We also described two approaches for training the model. In the first the user selects the points \( x_{i,t} = 1, \ldots, n \) on each of the training images, and it remains only to identify the best local feature and estimate the associated parameters using subimages centered at \( x_{i,t}, t = 1, \ldots, T \). In the second the user only selects three anchor points on each training image, which determine an affine map for registration of local feature data to the reference grid. Training involves simultaneously finding the locations of interest \( z_1, \ldots, z_n \) and local features for those locations. The anchor points themselves need not be part of the model. In the next two chapters we describe in detail two methods for finding detections of the sparse models defined in terms of these local features.
Chapter 7

Detection of sparse models: dynamic programming

In the previous Chapter we described two families of local features which could be used to construct a model for an object. These features exhibit certain invariance properties on the object ensemble and are low density in generic images. The next step is to use knowledge on the variations of the object to constrain the possible arrangements of the instances of the features, as well as develop an efficient algorithm to detect admissible instantiations. In this chapter we explore one possible approach which is relevant when \( p_o \) is very close to 1, so that false negatives at the local level are very rare and at most one instance of the object is expected in the image. A more general approach allowing for partial occlusions and multiple objects in the scene is studied in Chapter 8.

7.1 The prior model

The prior is defined so as to favor points \( \theta \in \Theta \) with a higher degree of geometric similarity to the model configuration \( z_1, \ldots, z_n \). Define a function \( \Psi(\theta) \) which measures some form of geometric distance between \( Z \) and the instantiation \( \theta \). In anticipation of computational issues we limit ourselves to ‘triangle’ based functions \( \Psi \) of the form

\[
\Psi(\theta) = \sum_C \psi_C(\theta; i \in C),
\]

where \( C \) is a collection of triples from the set \( \{1, \ldots, n\} \). The prior is then simply

\[
P(\theta) = \exp[-\Psi(\theta)].
\]

The functions \( \psi_C \) are defined to be invariant to translation and to certain ranges of scale and rotation. They are formulated in terms of the deviation of a candidate triangle \( \theta_{i_1}, \theta_{i_2}, \theta_{i_3} \) from the model triangle \( z_{i_1}, z_{i_2}, z_{i_3} \). There are two components: hard constraints which limit the range in which the triangle can vary, and soft constraints which penalize deviations from the template triangle. The hard constraints used below are very loose, and impose a limit of \( \pm \pi/2 \) on the deviation of each angle of the candidate triangle from the model triangle, and a limitation on the scaling of any of the edges between 1/5 and 5. The set \( \Theta \) is defined as those sequences \( \theta \) which satisfy the collection of hard constraints on \( \theta_C, C \in \mathcal{C} \).
For a triple $C = \{i_1, i_2, i_3\} \in \mathcal{C}$, the soft constraints have the form

$$\sum_{j=1}^{3} \left( (\alpha_{i,j} - a_{i,j}) \mod 2\pi \right)^2,$$

where $\alpha_{i,j}$ is the angle at vertex $\theta_{i,j}$ and $a_{i,j}$ is the angle at vertex $z_{i,j}$ in the model triangle. There are many alternatives proposed in the statistical shape literature see for example Bookstein (1991) or Dryden & Mardia (1998). These penalties are translation and scale invariant. As defined above they are also rotation invariant, however the local features themselves are not. Since the orientation is usually known in the present context it is useful to add a rotational constraint on the functions. For example constraining the absolute angle of an edge to lie within a certain range of the angle of the corresponding edge in the model. Note that since these constraints are translation invariant, there is no particular need to specifically define the set $\Theta^{(0)}$ of admissible instantiations centered at the origin, although the set is implicitly defined through the hard constraint component of the cost function $\Psi$. Furthermore, since the hard constraints are defined in terms of the same triangles as the soft constraints, we can incorporate them directly into the cost function $\Psi$ by setting $\psi_C(\theta; i \in C) = \infty$ if $(\theta; i \in C)$ does not satisfy the hard constraints.

**Decomposability**

When $p_o = 1$ all $n$ local features will appear in an admissible configuration if the object is in the image. The maximum of the posterior is obtained by generating a list $S_i$ of all instances of each local feature $X_i, i = 1, \ldots, n$ in the image. Then for each $\theta$ such that $\theta_i \in S_i$, evaluate $P(\theta)$ and find the minimum. Typically this is not computationally feasible, even if $n$ is on the order of 10 points and the size of $S_i$ is on the order of 50 to 100. However when the collection of triples $\mathcal{C}$ has a **decomposable** form minimization can be efficiently done through dynamic programming.

Decomposability in the present context means that there are $n - 2$ triples defined on the set $I_n = \{1, \ldots, n\}$, and there exists an ordering of the triples $C_1, \ldots, C_{n-2}$ and a reordering $\sigma_i \in C_i$ of the elements in $I_n$, with the following properties. $\sigma_1$ is only a member of $C_1$. If we remove $\sigma_1$ from the list of points and $C_1$ from the list of triples then $\sigma_2$ belongs only to $C_2$, and the two other points in $C_1$ together belong to at least one other triple. Eliminate $\sigma_2$ and $C_2$ and so on. At stage $j - 1$ the point $\sigma_{j-1}$ belongs only to the triple $C_{j-1}$ and after eliminating $\sigma_{j-1}$ from the list of points and $C_{j-1}$ from the list of triples, $\sigma_j$ belongs only to $C_j$, and the two other points in $C_{j-1}$ together belong to some other triple. This elimination or ‘peeling’ procedure continues until all that is left are $\sigma_{n-2}, \sigma_{n-1}, \sigma_n$ which make up the triple $C_{n-2}$.

For example given $n$ indices the simplest decomposable collection would have the triples $i, i + 1, i + 2$ for $i = 1, \ldots, n - 2$, which has a simple **linear** form. There are other examples where the collection of triples is decomposable but does not have a linear form. The notion of decomposability can be defined in terms of graphs, which have edges between any two indices belonging to the same triple. Thus in figure 7.1 the triples in the collection are those for which all three connecting edges are present in the graph. Decomposability of the graph is equivalent to decomposability of the collection of triples. The numbers in the graphs correspond to the peeling order, and differ for different graphs. The graph on the left of figure 7.1 has the simple linear form. The graph on the right is also decomposable, but is not linear. Decomposability can be easily extended to collections of larger subsets with varying numbers of points (see Bertele & Brioschi (1969) and Rose et al. (1976).) However in the present context such extensions would lead to massive slow down in computation. We therefore limit ourselves to decomposable collections of triples.
7.2 Computation: dynamic programming

The description of dynamic programming in this context is very similar to that of Chapter 4, however it is worth repeating due to the somewhat more complex structure of the function. We assume that the indices and triples in the model are now relabeled according to the 'peeling' order described above. The state space $S_i$ associated to each index $i = 1, \ldots, n$ is the list of locations $i; j; k = 1, \ldots, J_i$ for which $X_i(i; j) = 1$.

Dynamic programming proceeds as follows. Since 1 is the vertex in $C_1$ which is not in any other triple (the first to be eliminated), we can write

$$\Psi(\theta_1, \ldots, \theta_n) = \sum_{i=1}^{n-2} \psi_{C_i}(\theta_k; k \in C_i)$$

$$= \psi_{C_1}(\theta_k; k \in C_1) + \sum_{i=2}^{n-2} \psi_{C_i}(\theta_k; k \in C_i)$$

$$\overset{\text{def}}{=} \psi_{C_1}(\theta_k; k \in C_1) + \Lambda(\theta_2, \ldots, \theta_n), \quad (7.2)$$

where $\theta_i \in S_i, i = 1, \ldots, n$. Assume $C_1 = \{1, a, b\}$. For any fixed pair $\theta_a \in S_a, \theta_b \in S_b$ of candidate points for landmarks $a$ and $b$, let $\theta_1^* [\theta_a, \theta_b]$ be the choice of $\theta_1 \in S_1$ which minimizes $\psi_{C_1}(\theta_1, \theta_a, \theta_b)$. Since $\Lambda(\cdot)$ does not depend on $\theta_1$, it is easy to see that the optimal instantiation $(\theta_1^*, \ldots, \theta_n^*)$ satisfies $\theta_i^* = \theta_i^* [\theta_a^*, \theta_b^*]$. Note that finding $\theta_i^* [\theta_a, \theta_b]$ for all possible pairs $\theta_a \in S_a, \theta_b \in S_b$ requires evaluating $\psi_{C_i}(\theta_1, \theta_a, \theta_b)$ for all possible combinations of $\theta_1, \theta_a$ and $\theta_b$. Hence the amount of computation is proportional to $m^3$, where $m$ is an upper bound on the number of elements in each of the sets $S_i$. For each pair $(\theta_a, \theta_b)$ the index $\theta_i^* [\theta_a, \theta_b]$ is stored as well as $\psi_{C_i}(\theta_i^* [\theta_a, \theta_b], \theta_a, \theta_b)$, so that the amount of storage required is proportional to $m^2$.

According to the decomposability requirements there exists a triple $C_{u_1}$ which contains both vertices $a$ and $b$. The index $u_1$ is not necessarily the next one in the ordering, except in the case where the collection of triples has a linear structure. Let $C_{u_1} = \{a, b, c\}$. Define

$$\hat{\psi}_{C_{u_1}}(\theta_k; k \in C_{u_1}) = \hat{\psi}_{C_{u_1}}(\theta_a, \theta_b, \theta_c) = \psi_{C_{u_1}}(\theta_a, \theta_b, \theta_c) + \psi_{C_1}(\theta_1^* [\theta_a, \theta_b], \theta_a, \theta_b),$$

$$\hat{\psi}_{C_i}(\theta_k; k \in C_i) = \psi_{C_i}(\theta_k; k \in C_i) \quad \text{for} \quad i = 2, \ldots, n - 2, i \neq u_1,$$

Figure 7.1: Two decomposable collections of triples with the eight model points. Each triangle in the above graphs corresponds to one of the triples. The left hand collection is linear the right hand is not.
and

\[ \Phi(\theta_2, \ldots, \theta_n) = \sum_{i=2}^{n-2} \hat{\psi}_{C_i}(\theta_k; k \in C_i). \]

Note that \( \psi_{C_1}(\theta_1^* \theta_a, \theta_b, \theta_a, \theta_b) \) is a function of \( \theta_a \) and \( \theta_b \) only, and hence \( \hat{\psi}_{C_{u_1}}(\cdot) \) is a function of \( (\theta_k, k \in C_{u_1}) \).

It is easily seen that \( \Phi(\theta_2^*, \ldots, \theta_n^*) = \Psi(\theta_1^*, \ldots, \theta_n^*) \) and \( (\theta_2^*, \ldots, \theta_n^*) \) is the minimizer of the new function \( \Phi(\cdot) \).

The original problem is thus reduced to a similar one with one vertex eliminated. The new function is still a sum of local functions on a decomposable collection of triples of indices.

By eliminating the vertices one at a time in increasing order of the labels, the problem can be solved when only the last 3 vertices \( n-2, n-1, n \) are left. When \( \theta_{n-2}^*, \theta_{n-1}^*, \theta_n^* \) are determined, we can go backwards and determine \( \theta_{n-3}^*, \ldots, \theta_1^* \) sequentially. At step \( j \) set \( \theta_j^* \) as \( \theta_j^*[\theta_k^*, \theta_l^*] \) (which is already stored) where \( k, l \) are the other two elements of \( C_j \). The indices \( k, l \) are necessarily greater than \( j \) so that \( \theta_k^* \) and \( \theta_l^* \) have already been determined. Because the amount of computations needed to eliminate one vertex is proportional to \( m^3 \), the total amount of computations is proportional to \( nm^3 \). We summarize this procedure in the following algorithm.

**Algorithm 7.1: Sparse model detection: dynamic programming**

- \( trip(i, j) \) - \( j \)’th vertex in \( i \)’th triple.
- Points ordered by peeling order: \( trip(i, 1) = i \).
- \( locs \) - vector of \( n \) arrays:
locations of the $n$ local features in the image.

$\text{locs}[i][k,0], \text{locs}[i][k,1]$ - coordinates of $k$th instance of $i$th feature.

$num[i]$ - number of instances of feature $i$.

- function $\text{par}(n_1, n_2, n_3)$ - return index $m$ if $m < n_3$ and $m$ belongs to a triple with $n_1$ and $n_2$, if no such $m$ exists return $-1$.

- $\text{rec}[i], i = 1, \ldots, n$ - arrays of dimensions $\text{num}[n_2] \times \text{num}[n_3] \times 2$:
  
  $n_2 = \text{trip}(i, 2), n_3 = \text{trip}(i, 3)$.
  
  $\text{rec}[i][l, k, 1]$ - optimal instance of $i$ for the $l$th instance of feature $n_2$,
  
  and $k$th instance of feature $n_3$.
  
  $\text{rec}[i][l, k, 2]$ - stores the value (initialized to 0).

- for $n_1 = 1 : n - 2$
  
  $n_2 = \text{trip}(n_1, 2), n_3 = \text{trip}(n_1, 3)$

  Find index of triples that have already been peeled based on any of the edges of the current triple,

  $m_2 = \text{par}(n_1, n_2, n_1), m_3 = \text{par}(n_1, n_3, n_1), m_4 = \text{par}(n_2, n_3, n_1)$

  $(m_2, m_3, m_4 < n_1)$

  for $l = 1 : \text{num}[n_2]$

  for $k = 1 : \text{num}[n_3]$

  $\text{currmin} = 0$.

  for $j = 1 : \text{num}[n_1]$

  $a = \text{rec}[m_2](j, k, 2), b = \text{rec}[m_3](j, l, 2), c = \text{rec}[m_4](l, k, 2)$

  $v = \psi_{n_1}(j, k, l) + a + b + c$ (\psi_{n_1} - cost function of triple $n_1$.)

  if ($v < \text{currmin}$) $\text{currj} = j, \text{currmin} = v$.

  end

  $\text{rec}[n_1][l, k, 1] = \text{currj}, \text{rec}[n_1][l, k, 2] = \text{currmin}.$

- $\text{opt}$ - vector of length $n$. Initialize to $-1$. Will contain optimal configuration.

- $V AL = \min_{k,l} \text{rec}[n - 2][k, l, 2]$, $\text{opt}(n - 1) = k_{\text{min}}, \text{opt}(n) = l_{\text{min}}$.

- for $n_1 = n - 2 : 1$

  $n_2 = \text{trip}(n_1, 2), n_3 = \text{trip}(n_1, 3)$

  $m_2 = \text{opt}(n_2), m_3 = \text{opt}(n_3)$

  $\text{opt}(n_1) = \text{rec}[n_1][m_2, m_3, 1]$.

end

In figure 7.2 we show the six steps of the dynamic programming algorithm for matching the linear model of figure 7.1. At each step we recover the optimal match corresponding to the subgraph covered up to that step. It is interesting to observe that due to the translation and scale invariance of the triangle cost functions the initial optimal matches are far removed from the actual structures we are seeking to detect. Only after a sufficient amount of information is integrated does the optimal match settle down at the correct location. The structure of interest is typically close to the center of the image, however this information is not used in the algorithm. The structure would be detected anywhere and at a large range of scales.

In figure 7.3 we show matches of the two models shown in figure 7.1 to several axial MRI images. In figure 7.4 we show some failed matches of the linear model. Most often failed matches will contain a large subset of correct matches but one or two which are wrong. This is due to one hand to the very loose hard constraints introduced in the cost functions shown here. A more judicious estimation of the variability of each of the triangles in the model could probably avoid these partial mismatches. On the other hand false negatives do occur at the local feature level, and necessarily lead to a wrong detection.
Figure 7.3: Top: Matches of the linear graph in figure 7.1 (left) to three axial MRI brain images. Bottom: Matches of the non-linear graph in figure 7.1 (right) to three other images.

Figure 7.4: Three partial mismatches of the linear graph model.
Coarse to fine computation

The efficiency of the computation of the model instantiation depends critically on the decomposability property of the collection of functions, and the low density of the features in the background. Ideally however one would like to evaluate a candidate instantiation in terms of the cost of all possible triangles in the model which correspond to the full graph on the \( n \) model points. The decomposable collection can be viewed as a coarse model which is easily matched. A more sophisticated implementation of dynamic programming can yield not only the top match but the top \( M \) matches, see for example Rabiner & Juang (1993) in the case of the linear graph. One can also run several different graphs as shown in figure 7.1 to obtain different candidate instantiations. The role of the coarse model is then to provide a small number of candidates likely to have low cost for the full function. At each of these it would be very easy to evaluate the full cost function and determine the optimal match in terms of this more complete model. Estimating an upper threshold of this cost in terms of a training sample of object images could provide a way of ruling out false matches in images which do not contain the object. We continue to pursue this notion of a coarse to fine implementation of the detection problem in Chapter 8.

7.3 Detecting pose

When the main source of variability in object instantiation is described in terms of affine transformations, i.e. pose, with perhaps a small degree of non-linear variability, it is possible to avoid dynamic programming. Given a list of features and a list of model locations on the reference grid \((X_i, z_i, i = 1, \ldots, n)\), one searches for triples of features in the image. A triple \( \theta_{i_1}, \theta_{i_2}, \theta_{i_3} \) of instances of features \( i_1, i_2, i_3 \) in the image determines a unique affine map \( a \) taking \( \theta_i \) to \( z_i \) for \( j = 1, \ldots, 3 \). This map provides a prediction for the location for all other features, \( \theta_i = a z_i \), and if there is an instance of feature \( i \) in a small neighborhood of \( \theta_i \) for each \( i = 1, \ldots, n \), then an instantiation of the object has been found. One can even allow some omissions, as long as some given fraction of the features are found in the expected location \( a z_i \). Once omissions are permitted it is necessary to carry out a systematic search over all possible triples \( i_1, i_2, i_3 \) in the model until a triple of instances \( \theta_{i_1}, \theta_{i_2}, \theta_{i_3} \) is found, providing an affine map \( a \).

In this type of search we assume that the affine map provided by any triple of detected features is reliable in predicting the locations of the others. This would not work for example with the ventricle structures in the MRI brain scans, due to the wide variability in the scale of the substructures. For example in 7.3 the scale of the two ‘arms’ defined by the top ventricle varies in a range of approximately 2:1 whereas the entire central ventricle structure is more or less the same size across images. Using the affine map determined by the two tips of the upper ventricles and the cusp in between, could result in wrong predictions for the locations of the other features. On the other hand this type of scheme can be relatively successful for objects such as faces as suggested in Amit & Geman (1997), Burl et al. (1998), and is closely related to the alignment method in Ullman (1996).

The loop through all possible triples can be computationally intensive, and even on relatively rigid objects, one particular triple of detected features can yield rather erroneous estimates of the correct pose. A more systematic and efficient computational approach to detection of instantiations with missing features is described in the next chapter.

7.4 Bibliographical notes and discussion

There is a vast literature on graph matching using a variety of other techniques. These techniques are more general than dynamic programming in that they are not constrained to special decomposable graphical structures. In Haralick & Shapiro (1992) the term used is inexact consistent-labeling and several algorithms
are described using heuristics to overcome the immense combinatorial burden. In Grimson (1990) the same problem is called searching correspondence space and again a number of algorithms are suggested mainly in the context of rigid objects. These themes are revisited in Ullman (1996). Recently methods which produce a ‘soft’ assignment of model points to data points has been studied, see (Gold & Rangarajan (1996), Rangarajan et al. (1997)). However all these methods are necessarily slow due to the inherent complexity of the general graph matching problem. They are usually applied to hand picked collections of points. They are rarely applied to images where there is a substantial amount of clutter, which significantly increases the number of detected features.

The idea of using dynamic programming to find optimal feature arrangements was proposed in Fischler & Elschlager (1973), although there is no specific recipe for defining local features. The computational procedure described in this Chapter is essentially the same. More details on the application of this approach to matching deformable structures in MRI brain scans can be found Amit (1997).

We have started exploring the use of more complex local features of relatively low density in the background. Fortunately it is possible to identify such features which are at the same time highly probable at certain locations on the object. The resulting models are sparse and given the appropriate structure can be computed very efficiently. The dynamic programming on decomposable graphs for the MRI images, takes no more than 100 milliseconds on the PENTIUM III 700Mhz, for each step. This grows linearly with the number of vertices in the graph.

The use of dynamic programming has already been introduced in Chapter 4. And in principle the dynamic programming algorithm described here is the same. However there is a major difference in terms of the definition of the state space at each step. In the model described in Chapter 4 the state space at step $k$ is the region $S_k$. These regions can be taken of moderate size, on the order of hundreds, due to the initialization provided by the user. In the models described in this chapter we seek to avoid user initialization. This means that in principle every point in the image is in the state space for each step of the dynamic programming. Clearly this is computationally impossible. The solution comes in the form of the low density local features. Indeed one could still imagine that the entire image is the state space, however we add a cost to the match of a point in the model to a point in the image, which is infinite if the latter does not pass the test associated to the local feature. The proper way to compute the optimal match is then to find the points which pass the test associated to each local feature, and those subsequently become the state space for the corresponding step in the dynamic programming.