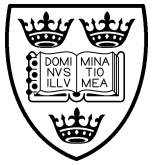
Learning Dynamical Models for Visual Tracking



Ben North

Robotics Research Group,
Department of Engineering Science,
University of Oxford.

October, 1998

This thesis is submitted to the Department of Engineering Science, University of Oxford, for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise indicated, describes my own research.

Ben North
New College

Doctor of Philosophy
Trinity Term, 1998

Learning Dynamical Models for Visual Tracking

Abstract

Using some form of dynamical model in a visual tracking system is a well-known method for increasing robustness and indeed performance in general. Often, quite simple models are used and can be effective, but prior knowledge of the likely motion of the tracking target can often be exploited by using a specially-tailored model. Specifying such a model by hand, while possible, is a time-consuming and error-prone process. Much more desirable is for an automated system to learn a model from training data. A dynamical model learnt in this manner can also be a source of useful information in its own right, and a set of dynamical models can provide discriminatory power for use in classification problems.

Methods exist to perform such learning, but are limited in that they assume the availability of 'ground truth' data. In a visual tracking system, this is rarely the case. A learning system must work from visual data alone, and this thesis develops methods for learning dynamical models while explicitly taking account of the nature of the training data — they are noisy measurements. The algorithms are developed within two tracking frameworks. The Kalman filter is a simple and fast approach, applicable where the visual clutter is limited. The recently-developed Condensation algorithm is capable of tracking in more demanding situations, and can also employ a wider range of dynamical models than the Kalman filter, for instance multi-mode models.

The success of the learning algorithms is demonstrated experimentally. When using a Kalman filter, the dynamical models learnt using the algorithms presented here produce better tracking when compared with those learnt using current methods. Learning directly from training data gathered using Condensation is an entirely new technique, and experiments show that many aspects of a multi-mode system can be successfully identified using very little prior information.

Significant computational effort is required by the implementation of the methods, and there is scope for improvement in this regard. Other possibilities for future work include investigation of the strong links this work has with learning problems in other areas. Most notable is the study of the 'graphical models' commonly used in expert systems, where the ideas presented here promise to give insight and perhaps lead to new techniques.

Acknowledgments

Many thanks are owed to my supervisor, Professor Andrew Blake, for the great deal of assistance and direction he has given me over the course of the development of this work. Thanks also go to Professor Brian Ripley, my co-supervisor, for helpful discussion on the statistical aspects of some of the work. Past and present members of the Visual Dynamics Group have provided a friendly and inspiring environment to work in, to whom go thanks: Bob Kaucic (for many games of pinball), Andy Wildenberg (for unwise amounts of rum), Michael Isard (for great party clothes, and many valuable insights), Colin Davidson (for drinking partnerships and countless plans for world domination), John MacCormick (for net practice at beard cricket), Benedicte Bascle (for enjoyable collaboration on human motion capture), Josephine Sullivan (for Irish joviality) and Jens Rittscher (for steering the lab towards Linux).

I am also grateful to my college advisors — David Gavaghan, whose help was invaluable during my change of topic, and Wilson Sutherland, who was also my undergraduate tutor and whose geniality I have appreciated throughout my time at New College; and to the EPSRC, Oxford Metrics Ltd (through Julian Morris), and New College (through Alastair White) for providing me with funding. Finally, I owe a large debt of thanks to my parents for their support and encouragement over the years.

Contents

1 Introduction and Background **1**

1.1 Introduction 1

1.2 Outline of this Thesis 2

1.3 Background 3

1.3.1 Parametric Models 3

1.3.2 Snakes 8

1.3.3 Active Contours 10

1.3.4 Optic Flow 11

1.3.5 Active Shape Models 12

1.3.6 Bayesian Filtering 13

1.3.7 Dynamical Models and System Identification 14

2 Tracking with Active Contours **16**

2.1 Details of Tracking Framework 16

2.1.1 B-Splines 16

2.1.2 Model Space Representation 17

2.1.3 Dynamics 19

2.1.4 Measurement 20

2.1.5 The Kalman Filter 21

2.1.6 Metrics in Spline Space 22

2.2 Model Space Specification 24

2.2.1 PCA using the L_2 Norm 24

2.2.2 Determination of Subspace Dimension 26

2.2.3 Experiments 28

3 Basic Learning **35**

3.1 Introduction 35

3.2 Learning a Dynamical Model from a Training Sequence 36
3.2.1 Maximum-likelihood Estimate 36
3.2.2 One-dimensional First-order Systems 37
3.2.3 Vector Second-order Systems 38
3.2.4 General-order Systems with Unknown Mean 40
3.3 Filtered Learning 41
3.4 Classification using Learnt Dynamical Models 41
3.4.1 Learning the Models 43
3.4.2 Classifying with the Models 46

4 Learning Dynamical Models using EM **52**
4.1 Introduction 52
4.2 The Expectation-Maximisation Algorithm 53
4.3 Learning Dynamical Systems from Data Sequences with Missing Points . . 58
4.3.1 First-order Scalar System 58
4.3.2 First-order Vector System 64
4.3.3 Conclusions 68
4.4 Learning Dynamical Systems from a Measurement Sequence 69
4.4.1 Maximum Likelihood Estimation of Parameters 69
4.4.2 The Augmented-state Smoothing Filter 70
4.4.3 Summary of Algorithm 73
4.4.4 Experiments with Synthetic Data 73
4.5 Second-order Systems 75
4.5.1 Augmented-state Smoothing 76
4.5.2 Learning the Process Mean 79
4.6 Missing Data as an Infinitely Noisy Measurement Process 80

5 EM-Learning for a Kalman-filter-based Tracker **83**
5.1 Applying EM Learning to Dynamic Contour Tracking 83
5.1.1 Implementation of EM Learning in the Tracker 84
5.1.2 Example Application: Hand Tracking 85
5.1.3 Example Application: Lip Tracking 98
5.2 Dynamics with Stationary Noise 108
5.2.1 Second-order Systems 108
5.2.2 Example with Real Data — Heartbeat Data 110

5.3 Discussion 112
5.3.1 Learning Measurement Noise 112

6 Dynamical Models for Condensation **117**
6.1 Introduction 117
6.1.1 The Condensation Tracking Algorithm 118
6.1.2 Higher-order AR Models 120
6.1.3 Multi-mode Dynamics 121
6.2 EM Learning for Condensation 121
6.2.1 Maximisation Step 122
6.2.2 Expectation Step 124
6.2.3 Summary of Algorithm 127
6.3 Condensation-EM in Practice 127
6.3.1 Simple Motion — a Juggled Ball 127
6.3.2 Robustness in the EM Step 137
6.3.3 Longer Example Sequences 140
6.3.4 Robustifying 'Across Iterations' 140

7 Applications of Condensation-EM Learning **144**
7.1 Multi-model Learning 144
7.1.1 Starting Conditions 144
7.1.2 Results — Asymmetric Prior 145
7.1.3 Results — Symmetric Prior 149
7.1.4 Classification of Test Sequences 154
7.1.5 Discussion 161
7.2 More Complex Single-mode Motion — Hand-tracking 161
7.2.1 Preliminaries 162
7.2.2 Convergence of Model 162
7.2.3 Simulation of Learnt Model 162
7.2.4 Tracking Results 167
7.2.5 Performance on a Test Sequence 168
7.2.6 Performance on Multiple Test Sequences 169
7.2.7 Discussion 175
7.3 Discussion 175
7.3.1 History-based Smoothing 175

7.3.2 Limited-history Smoothing 182
7.3.3 Reduced Forward-backward Smoothing 183
7.3.4 Behaviour of One-dimensional Condensation 184
7.3.5 Pedestal Condensation 188

8 Discussion **190**
8.1 Relation to other Model-learning Frameworks 190
8.2 Future Work 192
8.3 Conclusions 196

A Derivations and Proofs **197**
A.1 Ljung's Method 197
A.2 Alternative Approaches to EM Learning for a Kalman Filter 199
A.3 Multi-dimensional Yule-Walker 200
A.3.1 Zero-mean case 200
A.3.2 Unknown Mean 201
A.4 Learning from Multiple Sequences 203
A.5 Maximum-likelihood Estimate for Multi-mode Models 203

Chapter 1

Introduction and Background

1.1 Introduction

Visual tracking can be characterised as extracting information about moving objects from a sequence of images taken over time. Usually, this involves analysing a sequence of video fields, which are produced at a rate of 50 or 60 per second. It is a difficult problem, and any solution must be able to process the vast amount of data in a video stream in an efficient manner.

Good use can be made of the temporally-coherent nature of video sequences. Typically, fairly accurate predictions can be made about the position and configuration of an object in a video field, based on its behaviour in the immediate past. Such a predictive system is referred to as a dynamical model. Tracking of an object's motion when a precise dynamical model is available is significantly easier than trying to do so when no such model exists — consider the greater ease with which a human can watch a ball thrown through the air than follow a bluebottle flying round a room.

This thesis addresses the problem of how a dynamical model can be learnt from training data, the idea being that expending some computational effort on studying a class of motion will be repaid in more successful tracking of other objects performing motion of that type. The systems described in this thesis, and the learning framework in particular, operate probabilistically. When tracking, instead of calculating merely an estimate of the object's configuration at any given time, the entire probability distribution for that configuration is computed. This approach has as its basic premise the assumption that the quantity of interest, namely the object's configuration, is a random variable about which inferences can be made based on measurements made from images. When learning, the probabilistic interpretation of the aim is that it is to calculate the most likely description of the object's motion, based on the training sequence.

As well as the use of learnt dynamical models for the predictive component of a tracking system, other problems are considered in varying amounts of detail. Segmentation of a sequence into different modes of motion is covered, as well as classification of sequences based on how likely they are to have arisen from each of a set of dynamical models.

The work has raised several issues in the course of its development. Some are improvements to the techniques whose development will be described, but there are also interesting parallels in other fields where models used for inference are important.

1.2 Outline of this Thesis

The remainder of this chapter looks at some related work in the area of image analysis and visual tracking. Following on from this background, chapter 2 covers the development of the 'active contour' tracking framework to be used for most of the work in later chapters. The use of B-splines to represent the outline of an imaged object is introduced, and a tracking algorithm based on the Kalman filter presented. The chapter also develops a mechanism for determining the range of deformations of the object's outline, when a video stream is treated simply as a large (unordered) set of static images.

Chapter 3 addresses the problem of how the temporal coherence of a video sequence can be exploited. The idea of a 'dynamical model' is developed more fully, and techniques given for 'learning' such a model from training data, making certain assumptions. The usefulness of such learnt models for tracking applications is summarised, and their application to a slightly different problem domain — that of classification of sequences — is discussed.

Chapter 4 examines the assumptions made in chapter 3 for learning dynamical models, and the circumstances in which their validity is questionable. A mathematically rigorous algorithm is developed which learns a dynamical model while explicitly taking account of the fact that 'ground truth' data of the tracked object's state are not available. The statistical tool of 'Expectation-Maximisation' is used to solve this 'missing data' problem.

Chapter 5 presents some results obtained by applying the general theory of chapter 4 to real tracking problems. Successful tracking using models learnt in this manner is demonstrated, as well as improvements over using models learnt with existing methods. Hand- and lip-tracking examples are considered. Some possible extensions to the algorithm are discussed.

Chapter 6 considers a recently-developed tracking framework, the Condensation algorithm, which uses a stochastic representation of the underlying probabilistic systems. A Condensation-based tracker requires a dynamical model just as the Kalman-filter-based

systems of earlier chapters do. A greater variety of models is possible, though, and this chapter both applies the theory of previous chapters to this new framework, and extends them to cover 'multi-mode' dynamical models.

Chapter 7 demonstrates the effectiveness of the techniques of chapter 6 when applied to tracking problems where a Kalman filter would fail. This includes trackers incorporating multi-mode dynamical models, where automatic simultaneous learning and classification of a juggling sequence is presented. Some points concerning improving the efficiency and performance of the learning algorithms are raised.

Concluding discussion is given in chapter 8, including of the relation of the learning algorithms presented in this thesis to the wider problem of inference and learning on 'graphical models', a well-established framework in the statistical and artificial-intelligence literature. Areas where further research would be interesting and fruitful are also discussed.

1.3 Background

This section looks at the various approaches taken to the problem of visual tracking, and image analysis in general, in some related work. What might be dubbed 'hypothesis-driven' analysis operates by generating and refining the fit of a model to the image data. This includes the work described below on parametric and active shape models. A more 'bottom-up' approach uses much less information about the likely configurations of the object, including the earlier forms of snakes and some of the work using optic flow. Ideas from both parametric modelling and snake-based systems are combined in the 'active contour' approach; some work using this method is described below, and a fuller development presented in chapter 2.

1.3.1 Parametric Models

Work on tracking using detailed physical information about the object being tracked includes the use of the position in the image of the projection of easily identifiable features of the object, such as corners and edges, e.g., (Yuille and Hallinan, 1992; Harris, 1992; Lowe, 1992; Azarbayejani et al., 1993; Wachter and Nagel, 1997). A parametric model (consisting of, say, the object's three spatial translation coordinates, three rotation parameters and any internal parameters the object may require to completely specify its structure) is then fitted to these data, using statistical techniques. Often, e.g., (Azarbayejani et al., 1993; Bregler, 1997; Terzopoulos and Metaxas, 1992), in tracking as opposed to pattern recognition, the final estimate of the configuration in the model in one frame is used to provide

the starting point in an optimisation algorithm for the next frame, either directly, or using some predictive model of the motion of the object. This makes some use of the continuity present between one image frame and the next; section 1.3.7 returns to this topic.

Instead of modelling the object itself in three dimensions, one can use a model of the resulting image. This approach has the advantage of greater simplicity (being two-dimensional), and can work well if the object being modelled is almost planar, or is always viewed in the same orientation. Both model-based approaches have the advantage of including kinematic knowledge of the object being tracked, reducing the problem of the tracker generating impossible estimates of the configuration of the object, as can happen when no information about the structure of the target is included.

Eyes are used by Yuille and Hallinan (1992) to illustrate a type of parametric representation which they refer to as a 'deformable template'. As mentioned above, this model, in contrast to those described below, works in the image plane as opposed to being a three-dimensional representation of the tracked object. The outline and interior of the image of an eye are represented with a small collection of geometrically simple curves, and 11 parameters are chosen to control their location and shape. A function is defined based on the fit of the model to the image (for example, the predicted location of the iris of the eye in the image is expected to be dark; an edge is expected at the predicted location of the junction between the white of the eye and the eyelid), which is optimised starting from a hand-selected initial estimate. In this way, the position of the eye and the direction of the gaze can be determined. Tracking is accomplished by using the final template configuration in one frame as the starting point for the optimisation algorithm in the next. The authors also mention how the techniques can be applied to the extraction of tracks in the output from particle accelerators.

Another example of the use of these two-dimensional models is the work of Bennett and Craw (1991), which deals with finding hands and faces in both clean and noisy (signal : noise ratio of $1:1$) images. Their models of the hand and of the face (including the position of the features with respect to the outline of the face as a whole) is statistical, with a distribution estimated from around a thousand samples. They define a 'fitness function', which is designed to be a good indication of how well the proposed set of parameter values fits both the prior distribution and the image data. A global maximum for this function is sought, using a two-stage approach: first an initial estimate is obtained by sampling from the prior, which is then improved upon by making deviations from it. The work is solely on static images; no tracking is done.

Harris (1992) considers the problem of tracking the motion in three dimensions of a known object, using a description of its edges. His system tracks, in the examples given, model and real aircraft, as well as a runway for the automatic guidance of aircraft during landing. Fitness of a hypothesised model pose to the image data is determined by casting normals (in the image plane) to control edges (set up in the model), with some approximation to enhance speed, and seeking to minimise a function based on the perpendicular distances. He uses quaternions to reduce the effect of errors introduced by incrementally updating rotations, and a Kalman filter to perform smoothing over time of the estimate. He describes work on using profile edges of a limited class of surfaces in a similar algorithm. The resulting tracker is robust and reasonably fast. He employs a constant-velocity dynamical model for prediction.

A more general problem is addressed by Lowe (1992), that of tracking an object which can be represented as a polyhedron, the increased generality coming from the possibility that the objects have internal parameters: he tracks a hinged file box as an example. He uses a simple velocity prediction method to obtain the starting point for the model-fitting algorithm, and uses the predicted configuration of the object to drive the matching process of edges of the model to lines of high contrast in the image. The matches are tried in an order decided by the closeness of the image lines to the predicted projections of the model edges; in this way, an overall match for the whole model is found quickly, and is robust to false image-line/model-edge matches, as well as being able to cope well with occlusion of some of the model edges.

Azarbayejani et al. (1993) use point features (instead of lines) to track the position and orientation of the head. They describe a 'virtual holography' application, in which the displayed view of a virtual three-dimensional object on a stereoscopic monitor varies to reflect the location of the viewer's head. In this way, the user can look at the object from different angles, and see what would be seen if the object were present. Applications to teleconferencing are also mentioned. They also describe the possibility of gathering the information required to construct a physical model on-line, i.e., without any prior knowledge of the structure of the object being tracked. In the paper, though, features which come into view as the head, for example, rotates have their 3D position calculated based on a simple ellipsoidal model of the shape of the head. An interesting aspect of their work is the use of the predictive dynamical model to reduce the lag (the time between the head moving and the display changing), which can be a cause of reduced usability or even motion sickness.

In one of the more impressive demonstrations of the potential of this model-based

approach, Sullivan (1992), tracks several different types of vehicles simultaneously in an airport. He also describes the tracking of road vehicles at a road junction with the aim of automating the production of traffic-flow statistics. His system, as well as coping with the tracking of many objects, is robust to the occlusion of the tracked object, either by another tracked object or by some other object in the scene (for example, the occlusion of a car by a lamp-post). A simple motion detector is used to detect when a new vehicle enters the field of view of the camera, and identify the likely location of the vehicle. An initial estimate of its configuration is produced, which is refined until a best match is found between the model pose and the image data. Again, a dynamical model based on a Kalman filter is used to provide some predictive power. He also makes comparisons with the human visual system, in which there are extensive connections from the higher-level areas of the brain to the lower-level information-receiving areas, and summarises that both hypothesis-driven and data-driven processes are required in a vision system.

Terzopoulos and Metaxas (1991; 1992) recover the structure of the surface of an object, either from two-dimensional image data, or from three-dimensional range data. Their approach allows both recognition (and also pose computation, although this aspect is not explored in great detail) and surface reconstruction at the same time, within the same framework. In this way, their work is a hybrid between the model-based and image-based approaches. They recover the surface of an object from either two-dimensional image data or three-dimensional range data, using a model for the surface which combines a parameterisation of the object's pose (translation and orientation) with one of its global shape (the example they use is a super-ellipsoid, with parameters describing its size, aspect ratios, and 'squareness') and one of local deformations (expressed as a linear combination of a set of basis deformation functions). These parameters are all combined into a single overall parameter vector, which evolves into a minimum-energy configuration, where the energy takes into account such things as membrane elasticity and stiffness of the surface, and a potential derived from the (image or range) data. The results are good; examples given include the recovery of surface information about a pestle and a doll (in this case the image is first manually segmented into eleven parts, although they mention that this could be done automatically). They apply their system to the tracking, over a sequence of 120 frames, of a human torso and arms from sparse range data (Terzopoulos and Metaxas, 1992).

Human Body Tracking

The tracking of human motion is a problem with applications in medicine (particularly gait analysis), sports (for instance, analysis of tennis serving action), and the film industry (motion capture for animation), as well as being an interesting problem in its own right. Bregler (1997) describes a hierarchical tracking system, employing dynamical models throughout the hierarchy. At the very top level of the model are Hidden Markov Models (Rabiner and Juang, 1986) containing states for each major subdivision of a type of motion (for example, the motion of a leg during walking consists of a state describing the leg while in contact with the ground, and one for the leg's motion while swinging). Each state has an emission model corresponding to a second-order linear dynamical model. These models form the second layer. The states of these linear models are the outputs of the lowest layer, a blob-tracker, including velocity estimates, which are Kalman-filtered using a constant-velocity model. Using this system, human gaits are classified into 'running', 'walking', and 'skipping', with success rates around 90%. Learning is accomplished from specially-gathered ground truth data.

Later work by Bregler (1998) uses the elegant formulation of twists and exponential maps (Murray et al., 1994) for the 3D transformations involved in modelling human motion. Measurement is achieved by estimating a parameterised motion field, using optic flow (see below). Prediction is performed using a constant-position model for the motion from time-step to time-step. Although this does not cause problems in the results shown, one would expect it to be a limitation when tracking of humans performing rapid motion is attempted.

Wachter and Nagel (1997) combine contour- and region-based measurements to perform tracking of a human body, estimating a model with 10 degrees of freedom. Results are quite good, especially as information from only one camera is used. Motion where the subject walks across the field of view is better captured than when the subject turns towards the camera, however. An extended Kalman filter is used, with a constant-velocity prediction model. Full 3D tracking of the entire human body in all its complexity (Williams, 1989) remains a challenging problem.

Essa and Pentland (1995) focus on the face, presenting a system for recognising human facial expressions. A motion model based on the physics of the muscles of the face is used to constrain estimation of optic flow. An improvement over simply using optical flow measurements directly is claimed — the model constrains the estimated motion by means of the physical model. The resulting system performs well, producing classification with 98% accuracy for six facial expressions.

1.3.2 Snakes

Important information in scenes is often conveyed by lines or curves across which there is high contrast. Here, 'contrast' is usually in intensity, although contrast in other image properties, including texture, has been used. For example, at an edge of a polyhedral object, the light reflected from the two meeting surfaces changes owing to their different orientation and there will be a corresponding line in the image across which the intensity changes suddenly. Similarly, the projection of the outline of any sort of object will typically be a curve in the image across which there is a sharp intensity change.

There is also often useful information to be gathered from lines of locally maximum or minimum intensity in the image, for example a marking on the surface of a tracked object. Snakes, as described briefly below and on which the tracking system used in this thesis is based, provide one method whereby information about these essentially one-dimensional features can be extracted.

The term 'snake' is coined by Kass et al. (1987) to describe a parameterised family of curves in the image plane which is deformed in such a way as to locally minimise an energy function. The minimisation is then done by treating the snake as a physical system which is simulated numerically — it is given similar characteristics to a wire having elasticity and stiffness, in addition to being attracted to certain image features. These features include dark areas, light areas, areas of high contrast and the ends of lines and the corners of regions. The snake is initialised near the feature of interest, in order to indicate the approximate location of the desired local minimum, and then allowed to deform over time under the influence of the potential created by the energy function described. Variations include providing an additional energy term for kinetic energy in the case that the snake is required to follow a moving feature. An example of a snake which has found its final position in an image, in this case being attracted towards areas of high contrast, is shown in figure 1.1. (The development of the concept of a snake into the active-contour tracking framework used here will be developed in chapter 2.)

There has been considerable success in using snakes in computer vision, primarily in areas where a physical model of the object being tracked would be difficult to formulate, or where no such model is available. Their ability to deform almost arbitrarily allows them to follow the motion of non-rigid bodies much more easily than a system based on a necessarily complex physical model would be able to.

Ayache et al. (1992) describe a system for the automatic analysis of images gained by ultrasound scans of the heart. There is an additional complication in this case in that

Figure 1.1: **A snake in a locally minimum energy configuration.** *An example of a snake which has been allowed to 'relax' into a configuration with locally minimum energy on a static image. The white line shows the snake, a closed quadratic B-spline, attracted to the high-contrast areas at the edges of the radio.*

the images are gathered using a polar coordinate system, but the analysis takes place in Cartesian coordinates. They develop filtering and conversion techniques to map the image into Cartesian coordinates, and then analyse the resulting image sequence using snakes to follow the movement of the mitral valve and the left auricle. They also develop a method whereby a pointwise correspondence can be set up between the contours in successive frames which reflects intuition about curve-matching by placing greater emphasis on matching regions of high curvature, and favouring greater smoothness between these regions. More recent work in this area includes that by Jacob et al. (1998), who use the active contour framework together with carefully-chosen image filtering techniques to produce robust tracking of the left ventricle.

An example of the use of snakes where the object is to *create* a physical model of (at least some aspects of) the world in which the robot is required to work is provided by Blake et al. (1992). Here, a robot's arm is required to manoeuver around a work area which contains several obstacles, with the purpose of picking up an object on the far side of its environment. Snakes are used to track markings on, and the outlines of, the obstacles to gain information about their surface. In this case, the image motion is generated not by the

motion of the objects in the world, but by motion of the camera, attached to the robot's arm, near its actuator. To a large extent, therefore, the computer is in control of the visual information it receives. It moves the robot around in a small volume around its current position to gain information about its immediate surroundings. As the snake tracks the outlines and object markings, calculations are applied to the moving two-dimensional curves to compute three-dimensional surface normals, curvature and torsion, from which sufficient information is available to compute an approximate geodesic a small but safe distance from the surface of the object. The robot can then move into the area of free space it has just 'discovered', and repeat the exercise from its new vantage point. Eventually, it will have a direct route to the goal object, possibly after some backtracking when dead-ends are discovered.

One instance in which a quite complicated physical model *is* used in conjunction with a snake-based tracker is described by Terzopoulos and Waters (1990). Their objective is to perform analysis and re-synthesis of facial expressions for their low-data-rate transmission over low-bandwidth lines with the consequent applications in teleconferencing. They develop a physical model for the human face, based on anatomical knowledge of the bone and muscle structure, and of the material properties (elasticity, stress/strain relationship) of skin and the underlying fascia tissue. They choose a set of five muscle groups from the 268 voluntary muscles in the face which allow a reasonable representation of the range of expressions seen. Specifying the contractions of these muscles allows the re-synthesis part of the problem to be solved (within the approximation used; completely convincing synthesis of virtual faces is a difficult problem) — the resulting face model is rendered at frame rate. The analysis part is approached by tracking, using snakes, nine facial features, including the hairline, the eyebrows, and the lips. From the positions of the features, estimates are made of the states of the five muscle groups used, allowing the model to be driven; the results presented show quite realistic images of the model face. The physical model is only used in the re-synthesis phase, however; it is not used to influence the tracking of the features.

1.3.3 Active Contours

One method of incorporating some limited form of prior knowledge into a snake-based tracker is to specify a 'template' for the snake, as described by Curwen et al. (1992). The type of snake described in section 1.3.2 has, in a sense, a template, in that in the absence of a potential from the image, the snake will relax into a certain configuration — typically a straight line or a circle. A more interesting template must be incorporated into the snake's

behaviour. Another energy term is added to the snake potential favouring similarity to a pre-specified template shape, in terms of trying to minimise (along with fitness to the image) the deviation from the template. Without such a template, the snake exhibits a tendency to remain static in the image, which, when a moving object is present, means that the snake slides along its length, round the outline of the object, making recovery of the object's position difficult. When the snake is coupled to a template having the shape of the outline of the object, the snake is dragged along by the moving object, as desired.

The type of snake used in this thesis follows the variation developed by Blake et al. (1995), where hands and lips are tracked using a snake based on a contour which is represented as a B-spline. The spline is allowed to deform only in certain pre-defined ways, expressed as a set of displacements of the control points which can be used in linear combination. The size of the set thus determines the number of degrees of freedom which the contour has. The examples presented include the tracking of hands (mostly rigid motion; some articulation about the knuckles) and lips (non-rigid deformation). They promote the use of a Kalman filter in conjunction with a dynamical model; the dynamical model is learnt statistically from training video sequences, and produces an increase in tracking robustness. These ideas will be investigated more fully in the remainder of this thesis.

Reynard et al. (1996) develop a method by which the first few frames of tracking define the template which will be used subsequently. A few parameters control the shape of the template, and these are allowed to vary over the first few tenths of a second (of real-time tracking); thereafter their dynamics are such that they are effectively fixed and the tracker then proceeds with the decided-on template. In this way, the tracker can cope with variations from object to object, without bringing in too many extra degrees of freedom with the concomitant decrease in stability.

1.3.4 Optic Flow

Knowing the two-dimensional velocity of an image point corresponding to a real-world point in motion provides information about the shape of a rigid body (Koenderink and van Doorn, 1975), and can be used to segregate an image into separate objects (Black et al., 1997). The fundamental assumption made when determining this 'optic flow' is that changes in image intensity $I(x, y, t)$ are caused only by motion of objects with fixed texture. The total derivative of the intensity when moving with the optic flow is then zero — this leads to the

'motion constraint equation' given by Horn (1981):

$$u\frac{\partial I}{\partial x} + v\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0. \tag{1.1}$$

While this is often violated (for instance, by occlusions or semi-transparent objects, or deforming objects), it can be a useful approximation.

Eqn (1.1) does not by itself provide enough information to completely determine the optic flow field (u, v), so methods for finding this optic flow field from a pair of images (for instance consecutive frames of a video sequence) rely on additional constraints. Black (1993) mentions two. The 'data-conservation' constraint involves minimising an error term consisting of the left-hand side of eqn (1.1) summed over some region, with fixed (u, v) for that region. The other constraint is to add in a regularisation term to the error; typically a term corresponding to 'membrane energy'. This has the disadvantage of smoothing over object boundaries, when sometimes the location of discontinuities in the flow field is precisely the desired information. Black then introduces robust methods for finding the optic flow when it has motion discontinuities.

An alternative constraint mechanism for solving eqn (1.1) is given by Black et al (1997). Here, PCA is performed, on either synthetic or real training sets of flows. The problem can then be expressed in the resulting PCA space, with its greatly reduced dimensionality (7 or 9 components are used in the experiments, as opposed to two components of the flow field per pixel of the image). Successful motion estimation and also recognition of facial expressions is then demonstrated.

1.3.5 Active Shape Models

The notion of a Point Distribution Model (PDM) was introduced by Cootes et al. (1992) as an effective mechanism for producing a model for the variable shape of objects drawn from some class (for example, resistors on a circuit board). Principal Component Analysis is used to capture the variation of a training set, with any given shape being represented by a set of hand-chosen and -labelled points on the object's silhouette. These PDMs are then developed by Cootes and Taylor (1992) into 'Active Shape Models', where a learnt PDM is used to guide the localisation of an object in an image. Edge-detection along model normals is used to generate an update vector, whereby translation, rotation, scaling, and deformation (within the constraints given by the learnt PDM) are iteratively estimated. Results on the examples of resistors and hands are good.

More recent work using PDMs includes the incorporation of grey-level information by Lanitis et al. (1995) to analyse images of faces. Pose recovery, person identification, and

expression recognition are all performed. Heap and Hogg (1997) point out a significant limitation of the original PDM approach. The fundamental assumption is that a set of training points in some space can be adequately modelled by a Gaussian distribution, or, equivalently, as being roughly contained within a hyperboloid. For an object undergoing non-linear deformations, for example articulation, this is not the case, and the resulting PDM is either too inaccurate (does not cover all valid instances) or too non-specific (allows invalid shapes). To counter this, they use a 'hierarchical PDM' as introduced by Bregler and Omohundro (1994), clustering the training points into subregions, on each of which PCA is performed. This produces a much better representation of the variation in shape of a class of objects, or a class of deformations of a single object.

Developing the hierarchical PDM idea further, Heap and Hogg (1998) address the problem of tracking the outline of a hand, including the more difficult case of discontinuous changes in the apparent contour, for example when the fingers move from being spread apart to being closed together. For filtering through time, the Condensation algorithm discussed in chapters 6 and 7 of this thesis is used, with an extension to allow jumps between non-adjacent areas of shape-space. This is very similar to the multi-mode Condensation tracking used in this thesis.

1.3.6 Bayesian Filtering

Within a probabilistic framework, the tracking problem can be phrased in terms of filtering: The (unknown) state of the target at time t is x_t, and measurements z_t are available. Using the measurements, inferences are to be made about the state. The basic problem, then, is of finding the a posteriori distribution $p(x_t \,|\, z_1, \ldots, z_t)$. The recursive approach is to write

$$\begin{aligned} p(x_t \,|\, z_1, \ldots, z_t) &\propto p(x_t \,|\, z_1, \ldots, z_{t-1}) p(z_t \,|\, x_t); \\ p(x_t \,|\, z_1, \ldots, z_{t-1}) &= \int p(x_{t-1} \,|\, z_1, \ldots, z_{t-1}) p(x_t \,|\, x_{t-1}) \, \mathrm{d}x_{t-1} \end{aligned}$$

thereby expressing the posterior distribution at time t in terms of that at time $t-1$.

The evolution density $p(x_t \,|\, x_{t-1})$ is assumed to be of the form

$$x_t = f_t(x_{t-1}, w_t)$$

for process noise w_t, and the observation density of the form

$$z_t = h_t(x_t, v_t)$$

for measurement noise v_t. When the process function f_t and the measurement function

h_t are both linear, and the distributions of each w_t and v_t are Gaussian, as is the initial distribution, of x_0, there is a closed-form solution, namely the Kalman filter (Gelb, 1974).

For non-linear f_t or h_t, or where the noise distributions are not Gaussian, no straightforward solution is possible, and various techniques have been developed to propagate densities in these circumstances. The introduction to chapter 6 covers some of these.

In the case of visual tracking, the process can often be assumed to be linear and Gaussian, and it is the non-Gaussian nature of the observation density which invalidates the assumptions required for the Kalman filter. The 'Probabilistic Data Association Filter', as described by Bar-Shalom (1988) and used by Rao (1992) for visual tracking, combines multiple observations into a single measurement by weighting each with its probability of arising from the target. One can instead try to decide which features have arisen from clutter and which from the tracked object — in other words, to distinguish between outliers and inliers. This is the approach taken in the RANSAC algorithm of Fischler and Bolles (1981). It has been used for vision applications including the work on the computation of the 'Fundamental Matrix' (the calibration-free representation of camera motion between two images) of Torr and Murray (1993), the line-feature tracker of Clarke et al. (1996), and a system described by Fornland (1995) for the detection of obstacles for an autonomous vehicle.

1.3.7 Dynamical Models and System Identification

The use of dynamical models for prediction in visual tracking applications has been mentioned in the above descriptions of work in the area. A common technique is to apply a simple model, such as constant acceleration, e.g., (Azarbayejani et al., 1993), constant velocity, e.g., (Harris, 1992; Lowe, 1992; Bregler, 1997; Wachter and Nagel, 1997) or even constant position, e.g., (Yuille and Hallinan, 1992; Bregler and Malik, 1998), and these have been successful, but the benefits of using a more specific dynamical model are known (Sullivan, 1992; Terzopoulos and Metaxas, 1992; Bregler, 1997; Essa and Pentland, 1995; Jacob et al., 1998; Blake et al., 1995; Heap and Hogg, 1998).

The field of control theory makes use of dynamical models, and this is covered by, for instance, Jacobs (1993). The problem of how to find a dynamical model is referred to as 'system identification', and Ljung (1987) provides a good description of this area. He mentions various types of dynamical model, ranging from the informal 'mental model' employed by a human driving a car, through a model with only a numerical description (called by Ljung a 'graphical model'; a different use of the term than will be employed in chapter 8) and models with concise mathematical descriptions ('analytical models', a

subset of which will be employed for the work in the remainder of this thesis), to 'software models', which although fundamentally mathematical, are too complex to be written down neatly.

Ljung's three 'basic entities' of system identification involve data-gathering, model-set selection, and the determination of the 'best' model in the chosen set. From the point of view of this thesis, the facet of model-set selection is perhaps least explored. A fairly specific type of dynamical model will be employed throughout, namely a (possibly multi-mode) linear model with Gaussian noise.

A class of dynamical model with a slightly different emphasis to those used in this thesis is the 'Hidden Markov Model', with well-known applications in speech recognition, as covered by Rabiner (1986), Rabiner and Juang (1993), and Huang et al. (1990). The relations between discrete-valued HMMs and the continuous-valued dynamical models used in this thesis, and between the problem of learning within the two frameworks, will be investigated in chapter 8.

Chapter 2

Tracking with Active Contours

2.1 Details of Tracking Framework

This section describes the tracking framework within which most of the research in this thesis will be carried out, which will be a type of 'contour tracker' as introduced in chapter 1 and promoted by Blake and Isard (1998). This type of tracker operates entirely in the two-dimensional image, seeking to maintain an estimate through time of the outline of the object being tracked. The particular type considered here represents this outline by means of a B-spline.

2.1.1 B-Splines

Let a B-spline (Bartels et al., 1987) be described by a set of N_c control points, (Q_i^x, Q_i^y) for $1 \leq i \leq N_c$. Then the spline curve $\mathbf{r}(s)$ itself is given by

$$\mathbf{r}(s) = (\mathbf{B}(s)\mathbf{Q}^x, \mathbf{B}(s)\mathbf{Q}^y), \tag{2.1}$$

where $\mathbf{Q}^x$ is the column vector $(Q_1^x, Q_2^x, \ldots, Q_{N_c}^x)^\top$ of the x coordinates of the control points, and $\mathbf{Q}^y$ similarly is the column vector $(Q_1^y, Q_2^y, \ldots, Q_{N_c}^y)^\top$ of their y coordinates. Here $\mathbf{B}(s)$ is the row vector of 'blending functions', which depend on the order of the spline and its knot configuration.

The configuration of the contour is described entirely in terms of its control points' positions, so a configuration column vector $\mathbf{Q}$ is defined in the so-called 'control-point space' as

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}^x \\ \mathbf{Q}^y \end{pmatrix}. \tag{2.2}$$

Knowledge of the shape of the object being tracked is incorporated by specifying a 'typical' configuration. This is usually a hand-drawn spline, fitted round a single video frame

of the object. This distinguished configuration of the contour is referred to as the *template*, and its control-point configuration vector will be written as $\overline{\mathbf{Q}}$. All other configurations will be dealt with as displacements in control-point space from this template. This will be described in more detail in section 2.1.2.

At several stages in the discussion it will become necessary to have some sensible notion of the 'distance' between two splines, and also of 'projection' onto a subspace of splines. This raises the question of how one defines a metric and an inner product on the space of splines, which is dealt with in section 2.1.6, where a matrix $\mathcal{H}$ is defined such that the inner product $\langle \mathbf{Q}_1, \mathbf{Q}_2 \rangle$ of two splines, represented by their control-point vectors, is given by $\mathbf{Q}_1^\top \mathcal{H} \mathbf{Q}_2$. This inner product and metric reflect intuition concerning the properties which a 'distance' function between two curves should possess.

2.1.2 Model Space Representation

Allowing the contour to deform arbitrarily (in other words, allowing the control-point configuration vector to assume any value in the $2N_c$-dimensional control-point space) has the advantage of being able to model the largest range of movements and deformations of the object. However, in many cases there is a much smaller set of motions to be tracked — for example, translation, or motion resulting in affine deformation of the contour in the image. Restricting the tracker to just those motions provides an increase in stability, as can be verified experimentally, and seen from the fact that the more degrees of freedom the contour has, the less over-constrained the system will be, with the corresponding reduction in robustness to noisy measurements.

This restriction is implemented by first effectively shifting the origin of control-point space to $\overline{\mathbf{Q}}$, and then choosing a linear subspace of this translated control-point space, to give the *model space*. The subspace is represented by means of a set $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n\}$ whose linear span is the subspace $\mathcal{W}$ required. Define the matrix W to have $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n$ as its columns.

The contour, when its configuration lies in the subspace $\mathcal{W}$, can be represented by the components of the $\mathbf{w}_i$, defining the vector $\mathbf{X}$. The control-point configuration vector $\mathbf{Q}$ is then given by

$$\mathbf{Q} = \overline{\mathbf{Q}} + W\mathbf{X}. \tag{2.3}$$

Conversely, the best representation $\mathbf{X}$ for an arbitrary control-point vector $\mathbf{Q}$ is given by the components (with respect to the columns of W) of the orthogonal (with respect to the

inner product defined by $\mathcal{H}$) projection of $\mathbf{Q}$ onto $\mathcal{W}$:

$$\mathbf{X} = W^{\dagger}(\mathbf{Q} - \overline{\mathbf{Q}}), \tag{2.4}$$

where

$$W^{\dagger} = (W^{\top}\mathcal{H}W)^{-1}W^{\top}\mathcal{H}. \tag{2.5}$$

(To see this, note that $W\mathbf{X}$ is to be the orthogonal projection of $\mathbf{Q} - \overline{\mathbf{Q}}$ onto $\mathcal{W}$. It is clearly contained in $\mathcal{W}$, so all that remains is to check that $\langle W\mathbf{X}, \mathbf{w}_i\rangle = \langle(\mathbf{Q} - \overline{\mathbf{Q}}), \mathbf{w}_i\rangle$ for each i, or equivalently, that $(W\mathbf{X})^{\top}\mathcal{H}W = (\mathbf{Q} - \overline{\mathbf{Q}})^{\top}\mathcal{H}W$:

$$\begin{aligned}(W\mathbf{X})^{\top}\mathcal{H}W &= (W(W^{\top}\mathcal{H}W)^{-1}W^{\top}\mathcal{H}(\mathbf{Q} - \overline{\mathbf{Q}}))^{\top}\mathcal{H}W \\ &= (\mathbf{Q} - \overline{\mathbf{Q}})^{\top}\mathcal{H}W((W^{\top}\mathcal{H}W)^{-1})^{\top}W^{\top}\mathcal{H}W \\ &= (\mathbf{Q} - \overline{\mathbf{Q}})^{\top}\mathcal{H}W(W^{\top}\mathcal{H}W)^{-1}W^{\top}\mathcal{H}W \\ &= (\mathbf{Q} - \overline{\mathbf{Q}})^{\top}\mathcal{H}W\end{aligned}$$

as required.)

The reduced-dimension $\mathcal{W}$ space will be used to describe the configuration of the spline henceforth, replacing the control-point representation. The question of how $\mathcal{W}$ should be specified is addressed in section 2.2.1.

Note that the parameterisation of the curve $\mathbf{r}(s)$ may be recovered from the model space configuration vector $\mathbf{X}$ by combining equations (2.1), (2.2) and (2.3):

$$\mathbf{r}(s) = U_0(s)\overline{\mathbf{Q}} + U(s)\mathbf{X}, \tag{2.6}$$

where

$$U_0(s) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \mathbf{B}(s); \qquad U(s) = \left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \mathbf{B}(s)\right] W,$$

and $\otimes$ represents the Kronecker product of two matrices: If A is an $m \times n$ matrix and B a $p \times q$ matrix, $A \otimes B$ is $mp \times nq$, constructed by replacing a_{ij} in A with the matrix $a_{ij}B$.

State Vector for Second-order System

Section 2.1.3 below will introduce the notion of a dynamical model, used for prediction within tracking. Often, this predictive model will predict the state at time t based on states at times $t-1$ and $t-2$: a second-order model. For ease of notation in this case, a state vector $\mathbf{x}_t$ is defined:

$$\mathbf{x}_t = \begin{pmatrix} \mathbf{X}_t \\ \mathbf{Y}_t \end{pmatrix}.$$

Here, $\mathbf{X}_t$ is the model space configuration vector (at time t) defined in section 2.1.2 above, and $\mathbf{Y}_t$ will be $\mathbf{X}_{t-1}$ for a discrete-time model. Alternatively, $\mathbf{Y}(t)$ might be $\dot{\mathbf{X}}(t)$ for a continuous-time model. Discrete-time models will be the only ones considered in detail in this thesis, as tracking is performed on a sequence of discrete images, captured at discrete intervals. Video-rate sequences are used, giving a time-step of 0.02 s.

2.1.3 Dynamics

A discrete-time model of the motion of the contour is used, in a stochastic dynamical model. This is second-order in terms of the original configuration vector $\mathbf{X}$, but first-order in terms of the state vector $\mathbf{x}$. The dynamical model therefore takes the form

$$(\mathbf{x}_t - \overline{\mathbf{x}}) = A(\mathbf{x}_{t-1} - \overline{\mathbf{x}}) + B\mathbf{w}_t,$$

where $\overline{\mathbf{x}}$ is the mean of the system, A is a matrix giving the deterministic part of the dynamics, and $\mathbf{w}_t$ is a vector giving the stochastic part. It is assumed to be Gaussian: $\mathbf{w}_t \sim N(0, I)$. The covariance matrix of the term $B\mathbf{w}_t$ is therefore $C = BB^\top$.

Since $\mathbf{x}_t = (\mathbf{X}_t, \mathbf{X}_{t-1})$, it follows that

$$\overline{\mathbf{x}} = \begin{pmatrix} \overline{\mathbf{X}} \\ \overline{\mathbf{X}} \end{pmatrix}, \tag{2.7}$$

and that A and $\mathbf{w}_t$ must be as follows.

$$A = \begin{pmatrix} A_1 & A_2 \\ I & 0 \end{pmatrix}; \qquad \mathbf{w}_t = \begin{pmatrix} \mathbf{w}'_t \\ \mathbf{0} \end{pmatrix}.$$

In terms of the configuration vector $\mathbf{X}$, then,

$$\mathbf{X}_t = A_1\mathbf{X}_{t-1} + A_2\mathbf{X}_{t-2} + (I - A_1 - A_2)\overline{\mathbf{X}} + B\mathbf{w}'_t. \tag{2.8}$$

So A_1 and A_2 describe the deterministic part of the dynamics, and the 'shape' of the noise is given by B.

The deterministic part of equation (2.8) is then used as the prediction phase of the tracking algorithm, described in section 2.1.5 below. The dynamics of the system (as described by A and C) need to be specified; in the absence of other information, plausible default dynamics are chosen. For example, a constant-velocity model might be employed, or a system whose modes are damped harmonic motion with time-constants appropriate to the motion being tracked.

The remainder of this thesis will examine how the dynamical model can be *learnt* from training sequences, and the benefits of doing so. For the remainder of the development of the active contour tracking framework, though, it will be assumed that a suitable model has somehow been specified.

2.1.4 Measurement

As well as predicting the expected motion of the contour by means of the dynamical model given in section 2.1.3, measurements taken from the image are to be incorporated into the estimation of the contour's position. This is done by examining the image along normals to the contour, and searching for features, a feature being defined as a location where, typically, the image intensity varies suddenly with distance along the normal. Other types of feature include valleys or peaks of intensity; signed edges (those which go specifically from, e.g., light to dark) can be used to reduce mismatches.

Along each normal there may well be more than one such feature, and there needs to be some mechanism for picking the one to be used in subsequent calculations. Typically, the strongest feature will be selected; other methods include weighting this choice towards the position predicted by the dynamical model and choosing the innermost or outermost feature. (These last methods only strictly makes sense for closed splines, but can be adapted for use with open ones, by arbitrarily treating, say, the left-hand side of the spline as it is traversed in the direction of increasing parameter value as the inside.)

This measurement innovation ν is defined in terms of s, the parameter along the contour, as follows.

$$\nu(s) = \mathbf{n}(s).(\tilde{\mathbf{r}}(s) - \hat{\mathbf{r}}(s)) + v(s), \tag{2.9}$$

where $\hat{\mathbf{r}}(s)$ is the predicted position of the contour at parameter value s and $\tilde{\mathbf{r}}(s)$ is the measured position of the feature along the contour normal $\mathbf{n}(s)$. Additive noise $v(s)$, assumed Gaussian (and independent for different values of s) with variance σ^2, is included in equation (2.9) to account for inaccuracies in the measurement $\tilde{\mathbf{r}}(s)$. This is illustrated in figure 2.1.

To relate the values of ν to the configuration vector estimate $\hat{\mathbf{Q}}$ and hence update the estimate using the measurements, a measurement co-vector $\mathbf{H}$ is required for use in the Kalman filter below. This $\mathbf{H}$ is to be defined in such a way that

$$\nu(s) = \mathbf{H}(s)(\tilde{\mathbf{x}} - \hat{\mathbf{x}}) + v(s).$$

From eqn (2.6),

$$\begin{aligned}
\mathbf{n}(s).(\tilde{\mathbf{r}}(s) - \hat{\mathbf{r}}(s)) &= \mathbf{n}(s).(U(s)\tilde{\mathbf{X}} - U(s)\hat{\mathbf{X}}) \\
&= \mathbf{n}(s).(U(s) \begin{bmatrix} I & 0 \end{bmatrix} \tilde{\mathbf{x}} - U(s) \begin{bmatrix} I & 0 \end{bmatrix} \hat{\mathbf{x}}) \\
&= \mathbf{n}(s)^\top U(s) \begin{bmatrix} I & 0 \end{bmatrix} (\tilde{\mathbf{x}} - \hat{\mathbf{x}}).
\end{aligned}$$

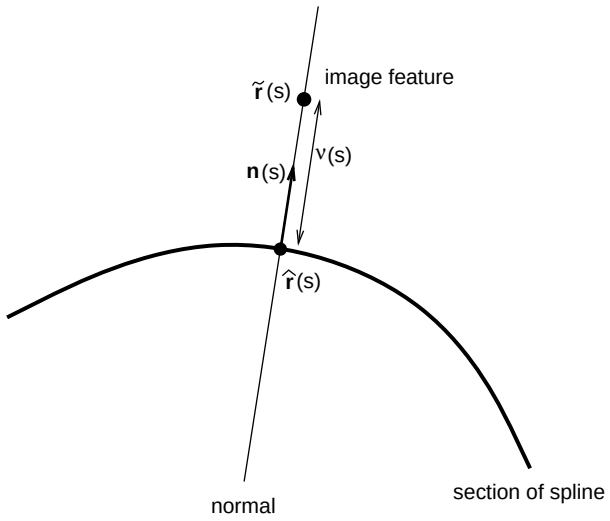


Figure 2.1: **Making measurements in the image.** *An innovation $\nu(s)$ is defined as the distance along the image normal between the predicted position of the contour and an image feature.*

Defining

$$\mathbf{H}(s) = \mathbf{n}(s)^\top U(s) \begin{bmatrix} I & 0 \end{bmatrix}$$

will therefore produce the desired results.

2.1.5 The Kalman Filter

The update of the estimate takes place in two phases: prediction and measurement. The current values of the mean and variance of the distribution are maintained. The mean is the estimated state vector $\hat{\mathbf{x}}$; denote the variance P. The variance is essential for the update of the Kalman gain. It is also useful for determining a 'validation gate', the range over which the search for features is carried out. As P increases, the length of the normals along which the feature search is carried out can be increased, reflecting the increased uncertainty.

Prediction (estimating $\mathbf{x}_t$ given $\mathbf{z}_1, \ldots, \mathbf{z}_{t_0}$ for $t > t_0$) and filtering (estimating $\mathbf{x}_t$ given $\mathbf{z}_1, \ldots, \mathbf{z}_t$) produce different distributions; introduce the notation

$$\hat{\mathbf{x}}_{t_0}^{t_1} = \mathcal{E}[\mathbf{x}_{t_0} \,|\, \mathbf{z}_1, \ldots, \mathbf{z}_{t_1}]; \qquad P_{t_0}^{t_1} = \mathrm{Var}[\mathbf{x}_{t_0} \,|\, \mathbf{z}_1, \ldots, \mathbf{z}_{t_1}].$$

where $\mathbf{z}_t$ is the measurement (or set of measurements) made at time t.

The prediction phase is based on the dynamic model described in section 2.1.3:

$$\begin{aligned} \hat{\mathbf{x}}_t^{t-1} &= A\hat{\mathbf{x}}_{t-1}^{t-1} + (I - A)\overline{\mathbf{x}}; \\ P_t^{t-1} &= AP_{t-1}^{t-1}A^\top + C. \end{aligned}$$

A number of measurements are then taken as described in section 2.1.4, at various values of the spline parameter s. These measurements must be used to influence the predicted position and variance. This is done by means of the Kalman gain, a vector $\mathbf{K}(s)$ defined as

$$\mathbf{K}(s) = P_t^{t-1}\mathbf{H}(s)^\top(\mathbf{H}(s)P_t^{t-1}\mathbf{H}(s)^\top + \sigma^2 I)^{-1}.$$

This is used to produce estimates given the new measurement as follows:

$$\hat{\mathbf{x}}_t^t = \hat{\mathbf{x}}_t^{t-1} + \mathbf{K}(s)\nu(s); \qquad P_t^t = [I - \mathbf{K}(s)\mathbf{H}(s)]P_t^{t-1}.$$

(These are applied at each value of the parameter s where a measurement has been taken. In the current implementation, three measurements are typically taken per span of the spline.) The resulting value of $\hat{\mathbf{x}}_t^t$ is then an optimal estimate of the state given the measurements (Jacobs, 1993).

2.1.6 Metrics in Spline Space

This section returns to the question of defining an inner product (and hence metric) on the space of all splines with a given knot configuration. This will be done by addressing the issue of a suitable metric, from which the inner product can be deduced.

Given two splines b_1 and b_2, with parameterisations $\mathbf{r}_1(s)$ and $\mathbf{r}_2(s)$ and control-point representations $\mathbf{Q}_1$ and $\mathbf{Q}_2$ respectively, there are two obvious choices for the definition of 'distance' between them.

The first is to consider the Euclidean distance between the two points $\mathbf{Q}_i$ in the $2N_c$-dimensional control point space:

$$d_1(b_1, b_2) = \left[\sum_{i=1}^{N_c}\Big([(Q_1^x)_i - (Q_2^x)_i]^2 + [(Q_1^y)_i - (Q_2^y)_i]^2\Big)\right]^{1/2}.$$

The second is to find the mean squared distance between the curves themselves, and use this to define the L_2 metric:

$$d_2(b_1, b_2) = \left[\frac{1}{L}\int_0^L \|\mathbf{r}_1(s) - \mathbf{r}_2(s)\|^2 \,\mathrm{d}s\right]^{1/2}.$$

Since it is the curves themselves which are of interest, a more sensible choice is to take d_2 as the metric function. This metric can be calculated in terms of the control-point vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$ as

$$d_2(b_1, b_2) = (\mathbf{Q}_1 - \mathbf{Q}_2)^\top \mathcal{H}(\mathbf{Q}_1 - \mathbf{Q}_2),$$

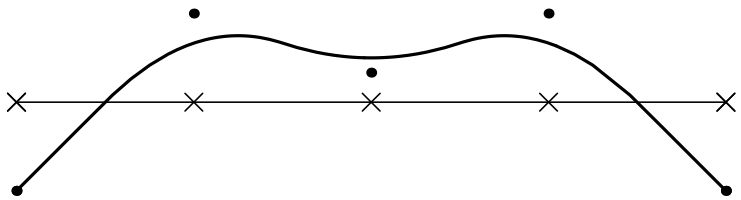
where

$$\mathcal{H} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \frac{1}{L} \int_0^L \mathbf{B}(s)\mathbf{B}(s)^\top \, \mathrm{d}s.$$

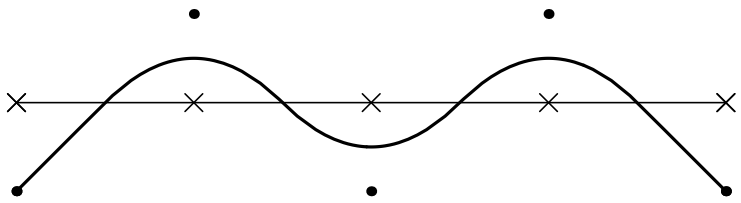
Similarly, the inner product $\langle \mathbf{Q}_1, \mathbf{Q}_2 \rangle$ is defined to be

$$\langle \mathbf{Q}_1, \mathbf{Q}_2 \rangle = \mathbf{Q}_1^\top \mathcal{H} \mathbf{Q}_2.$$

As an illustration of the difference between them, and why it is more sensible to choose the L_2 metric, consider figure 2.2, which shows the difference measured between two splines and a fixed reference spline. Table 2.1 shows the results of computing the distance between the two pairs of splines with the two different metrics.



(a) The two splines in their first configuration



(b) The two splines in their second configuration

Figure 2.2: **Care must be taken in the definition of a metric on the space of splines.** *An illustration of the difference between the Euclidean metric in control-point space and the L_2 metric for two splines b_1 (shown bold, control points shown as $\bullet$) and b_2 (shown light, control points shown as $\times$) in two different configurations. The distances between the two pairs of curves are given in table 2.1.*

As the results show, the distance between the two curves increases as measured by the Euclidean metric in control-point space, but decreases when measured with the L_2 metric. Since the two curves have intuitively moved closer together in figure 2.2(b) as compared to their positions in figure 2.2(a), the choice of the L_2 metric is seen to be consistent with intuition, as well as being more logically founded mathematically.

Metric	Distance in (a)	Distance in (b)	Multiplicative change
Euclidean	6.11	7.00	1.15
L_2	2.09	1.47	0.70

Table 2.1: **The L_2 norm produces a metric representing a more accurate concept of the distance between splines.** *The distances between the two curves shown in figure 2.2 in their two configurations. The L_2 metric matches intuition, which suggests that the two curves have moved closer together, better than the Euclidean metric in control-point space.*

2.2 Model Space Specification

2.2.1 PCA using the L_2 Norm

Section 2.1.2 introduced the idea of a model space — a linear subspace of the full control-point space which contains all spline configurations of interest. This section describes the usual method whereby such a subspace is specified. Note that in some cases, the subspace can be written down immediately, for example if translation represents the only degrees of freedom the spline is to possess. This will often not be the case, especially for deformations undergone by a non-rigid object, so some learning must be performed.

Problem Description

Suppose a set of curves $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$ forming a data-set is given, with corresponding representation $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_N$ in configuration space. The representation is linear, so $\mathbf{r}(s) = \mathbf{B}(s)\mathbf{Q}$ as usual. Assume (without loss of generality) that the mean

$$\begin{aligned}\overline{\mathbf{Q}} &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{Q}_i \\ &= \mathbf{0}.\end{aligned}$$

Then the Principal Components Analysis (PCA) problem is to 'explain' as much of the data as possible using fewer than N components. In fact, the problem can be simplified to that of choosing just *one* component, the one with the most explanatory power. Choosing the best n components can then be thought of recursively: choose the best one, remove it from each item in the data set, and repeat the process with the 'flattened' data-set. (In practice, PCA will not be performed in this recursive manner; this is just an argument to show that it is sufficient to consider just the first principal component.)

The first principal component can be defined with respect to any metric on the space of curves $\mathbf{r}(s)$. For consistency with the contour-tracking framework, it is natural to choose

the L_2 norm, which measures the difference between curves in a mean-squared sense, as explained in section 2.1.6.

The first principal component with respect to the L_2 norm, then, is the curve $\mathbf{r}(s)$, represented parametrically by $\mathbf{Q}$, that minimises the remaining mean-square unexplained portion of the data-set:

$$\min_{\mathbf{Q}} \min_{\lambda_1,\ldots,\lambda_N} \sum_n \|\mathbf{Q}_n - \lambda_n \mathbf{Q}\|^2,$$

where the λ_n are the proportions of the component $\mathbf{r}(s)$ contained in each data-curve $\mathbf{r}_n(s)$, and are chosen to minimise the unexplained data. Note that $\mathbf{Q}$ must be found subject to the constraint $\|\mathbf{Q}\| = 1$, otherwise the problem is underspecified, as any scaling of $\mathbf{Q}$ would be equally as good.

Solution

First the minimisation over the λ_n can be done independently, for each n, to give:

$$\lambda_n = \frac{\langle \mathbf{Q}, \mathbf{Q}_n \rangle}{\|\mathbf{Q}\|^2}$$

so the minimisation of $\mathbf{Q}$ is now, using the optimal λ_n,

$$\min_{\mathbf{Q}} \sum_{n=1}^{N} \|\mathbf{Q}_n - \lambda_n \mathbf{Q}\|^2.$$

Completing the square and substituting expressions for the λ_n, it can be seen that the desired $\mathbf{Q}$ is that for which $\mathcal{L}$ is at a maximum, where

$$\mathcal{L} = \sum_{n=1}^{N} \frac{\langle \mathbf{Q}_n, \mathbf{Q} \rangle^2}{\|\mathbf{Q}\|^2}$$

is the mean-square displacement of the component $\mathbf{Q}$ of the data-set. Defining

$$R_{00} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{Q}_n \mathbf{Q}_n^T$$

gives

$$\mathcal{L} = (\mathbf{Q}^T \mathcal{H} R_{00} \mathcal{H} \mathbf{Q}) / \|\mathbf{Q}\|^2$$

and the maximisation can be done using a Lagrange multiplier μ on the constraint $\|\mathbf{Q}\| = 1$ as

$$\max_{\mathbf{Q}} \left[(\mathbf{Q}^T \mathcal{H} R_{00} \mathcal{H} \mathbf{Q}) - \mu \mathbf{Q}^T \mathcal{H} \mathbf{Q} \right].$$

Finally, differentiating with respect to $\mathbf{Q}$ gives

$$\mathcal{H}R_{00}\mathcal{H}\mathbf{Q} = \mu\mathcal{H}\mathbf{Q},$$

which simplifies to

$$R_{00}\mathcal{H}\mathbf{Q} = \mu\mathbf{Q}.$$

The first principal component is therefore an eigenvector of $R_{00}\mathcal{H}$ and its eigenvalue μ gives the corresponding mean-square displacement as μ/L.

2.2.2 Determination of Subspace Dimension

In this section a framework is established within which the size of the linear subspace required for tracking various classes of motion can be predicted, particularly of articulated structures. This is an important part of the specification of the model space: when using PCA as described in section 2.2.1, a choice must be made as to the number of components to retain. Here a theoretical grounding for this decision is developed. Of particular interest will be the motion of planar objects, as this is a good approximation in several cases, including a hand moved in a plane perpendicular to the optical axis of the camera, as will be used in illustrative experiments below. The theory and experiments described in the remainder of this chapter are original contributions of the thesis.

Introduction

The range of possible transformations of the object is typically known; the question is how this dictates $\mathcal{W}$. The discussion below uses points in the body; because of the behaviour of B-splines under linear transformations, the results will remain valid when applied to sets of control points of splines, as used in the contour-tracking framework described.

Motion in the Plane

Consider firstly the motion of a planar object in its plane, assuming orthographic projection. Suppose the points of interest in a homogeneous (2-dimensional) coordinate frame fixed in the body are $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n$; then the points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$ in the image, when the body has been rotated through an angle θ and translated by the vector $\mathbf{u}$, are given by

$$\begin{aligned}\mathbf{p}_i &= \begin{pmatrix} \cos\theta & -\sin\theta & u_x \\ \sin\theta & \cos\theta & u_y \\ 0 & 0 & 1 \end{pmatrix} \mathbf{r}_i \\ &= \begin{pmatrix} R(\theta) & \mathbf{u} \\ \mathbf{0}^\top & 1 \end{pmatrix} \mathbf{r}_i \\ &= t(\theta, \mathbf{u})\mathbf{r}_i. \end{aligned} \tag{2.10}$$

A transformation of all points at once is required; define vectors consisting of the coordinates of all points, $\mathbf{r} = (x_1, y_1, 1, \ldots, x_n, y_n, 1)$ and a vector consisting of all image points, $\mathbf{p} = (p_{11}, p_{12}, 1, \ldots, p_{n1}, p_{n2}, 1)$. Then the transformation (2.10) is

$$\mathbf{p} = I_n \otimes t(\theta, \mathbf{u})\mathbf{r}. \tag{2.11}$$

The important vector now is the difference $\mathbf{p} - \overline{\mathbf{p}}$, where $\overline{\mathbf{p}}$ is the set of image points under $T(0, \mathbf{0})$, and how it can be expressed as a linear combination of vectors in some linearly independent set $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k\}$, for any given value of $(\theta, \mathbf{u})$. In this case, a suitable set can be generated by applying the transformations $E_1, E_2, \ldots, E_4$ to $\mathbf{r}$, where

$$E_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \; E_2 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \; E_3 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \; E_4 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \tag{2.12}$$

since then

$$\mathbf{p} - \overline{\mathbf{p}} = (\cos\theta)\mathbf{e}_1 + (\sin\theta)\mathbf{e}_2 + u_x\mathbf{e}_3 + u_y\mathbf{e}_4.$$

So in this case, the dimension of $\mathcal{W}$ is 4.

Articulating Planar Body

Next consider the case where the body is made up of two parts, hinged together. Suppose the first a points belong to the first body, and the remainder, b, to the second. (So $n = a+b$; b is introduced for ease of notation.) Then equation (2.11) is replaced by

$$\mathbf{p} = \begin{pmatrix} I_a \otimes T_1 & 0 \\ 0 & I_b \otimes T_2 \end{pmatrix} \mathbf{r},$$

where T_1 and T_2 are the transformations undergone by the first and second parts of the body respectively.

If $\mathbf{u}_1$ is the position of the hinge, in the frame of the first body, and θ_1 and θ_2 are the rotations of the two parts (with respect to some fixed reference frame), then (taking the origin in the second body's frame to be at the hinge)

$$T_1 = \begin{pmatrix} R(\theta_1) & \mathbf{u} \\ \mathbf{0}^\top & 1 \end{pmatrix}; \quad T_2 = \begin{pmatrix} R(\theta_2) & \mathbf{u} + R(\theta_1)\mathbf{u}_1 \\ \mathbf{0}^\top & 1 \end{pmatrix}.$$

Noting that $\mathbf{u}_1$ is fixed, a suitable $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k\}$ can be generated by using the trans-

formation pairs $E_i = \begin{pmatrix} T_{1i} & T_{2i} \end{pmatrix}$, where

$$E_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & u_{1x} \\ 0 & 1 & 0 & 0 & 0 & u_{1y} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \quad E_2 = \begin{pmatrix} 0 & -1 & 0 & 0 & 0 & -u_{1y} \\ 1 & 0 & 0 & 0 & 0 & u_{1x} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$E_3 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \qquad E_4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$E_5 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \qquad E_6 = \begin{pmatrix} 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

since then

$$\begin{aligned} \mathbf{p} - \overline{\mathbf{p}} = (\cos\theta_1)\mathbf{e}_1 + (\sin\theta_1)\mathbf{e}_2 + u_x\mathbf{e}_3 + u_y\mathbf{e}_4 \\ + (\cos\theta_2)\mathbf{e}_5 + (\sin\theta_2)\mathbf{e}_6. \end{aligned} \tag{2.13}$$

Therefore, for a singly-articulated planar object, a six-dimensional $\mathcal{W}$ is required. This argument can be extended to show that for an n-fold-articulated planar object, $\mathcal{W}$ must be $(2+2n)$-dimensional.

Other Results

Extending the case of a planar object by allowing motion towards and away from the camera, while maintaining the orientation of the plane the object lies in, creates an additional degree of freedom. The subspace required does not, however, need to be larger, since zoom is already permitted by virtue of the vector $\mathbf{e}_1$ (using the non-articulating case as an example) derived from equation (2.12). Therefore, the tracker is capable of following this type of motion with no change to $\mathcal{W}$.

2.2.3 Experiments

To test the validity of the theory developed in the previous section in a practical tracking situation, consider the motion of an outline of a left hand, as held in a 'mitten' shape, with the thumb treated as a pivoted extra body. The overall structure should therefore behave approximately as two pivoted bodies from the point of view of the theory. This example has applications in a 'mousing' application, where an unadorned hand can be used as a control in a user-interface.

Procedure

Several classes of motion are considered, as given in table 2.2. Here and subsequently, 'articulation' means of the thumb.

Class of motion in two dimensions	**Size of linear space**
Translation	2
Rotation	2
Articulation	2
Translation and rotation	4
Translation and articulation	4
Rotation and articulation	4
Translation, rotation and articulation	6

Table 2.2: **Predictions for subspace dimension.** *The dimension of the linear subspace of the full control-point space predicted to describe motions of the class indicated. These results were derived in section 2.2.2.*

To perform the verification, one base frame of a hand was captured and a spline manually drawn around the outline of the hand. Extra frames of the type of motion being examined were created in the same way; typical frames are shown in figure 2.3. Four frames more than the expected dimension of the linear subspace were drawn. PCA was then performed on the frames, and a drop in the eigenvalues after the first n values was sought, where n is the value given in table 2.2.

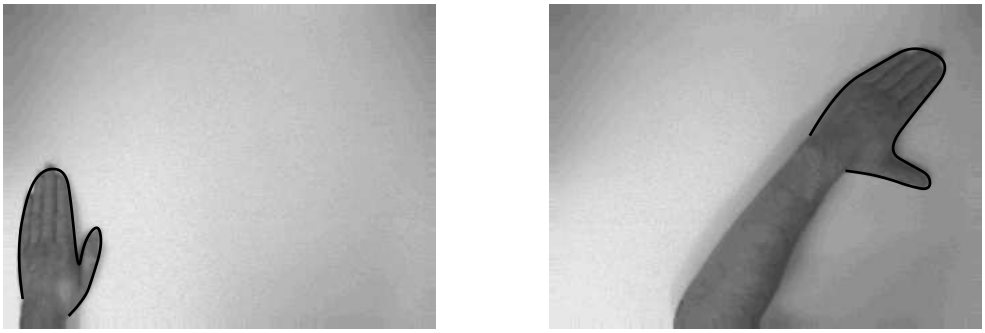


Figure 2.3: **Typical configurations used for experiments on determination of subspace dimension.** *Frames showing the hand in different configurations, with hand-drawn splines superimposed. The hand has been translated and rotated between these two frames, and the thumb has been pivoted.*

Class of motion	**Residue (%)**
Translation	5.8
Rotation	14.5
Articulation	16.6
Translation and rotation	2.0
Translation and articulation	2.3
Rotation and articulation	6.8
Translation, rotation and articulation	1.9

Table 2.3: **Residue — motion not accounted for by the predicted linear subspace.** *The 'residue' calculated as a measure of how much of the motion is not accounted for by a linear subspace of the dimension predicted. The hand was moved in the class of motion listed, and PCA performed to determine how much of the motion was left unexplained by the predicted linear subspace.*

Results

The base frame and extra frames described above were examined. PCA was performed on the sets of frames, and the resulting eigenvalues of the moments matrix plotted. In fact, the plots give the square root of the eigenvalue, since the eigenvalues are of the matrix $R_{00}\mathcal{H}$ (where R_{00} is the moments matrix), which have, therefore, the dimensions of squared distance. To get a value with the dimensions of distance, the square root is extracted.

The seven plots are given in figure 2.4, where the dotted line indicates the hoped-for separation between motion within the predicted linear subspace and any 'noise'. Table 2.3 shows the 'residue' of the eigenvalues beyond the expected number; i.e., writing λ_i for the eigenvalues, where $i = 1, \ldots, n$, and n is the size of the linear space for the class of motion in question, the value r, defined as

$$r = \frac{\sum_{i=n+1}^{n+4} \lambda_i^2}{\sum_{i=1}^{n+4} \lambda_i^2}.$$

Ideally, this value would be small, indicating that most of the motion of the hand is accounted for by the predicted number of vectors.

As can be seen, these residues are indeed low; mostly under 5% or 6%, with rotation and articulation slightly higher. Possible reasons for the larger residues in the rotation and articulation cases and for the lack of the theoretical 'knee' in the curve are given below.

Also given, in figure 2.5, are the eigenvectors resulting from the PCA of the last motion class (translation, rotation, and articulation); a sample of the span of each vector is given. The vectors are given in order of decreasing eigenvalue, scaled according to the eigenvalue.

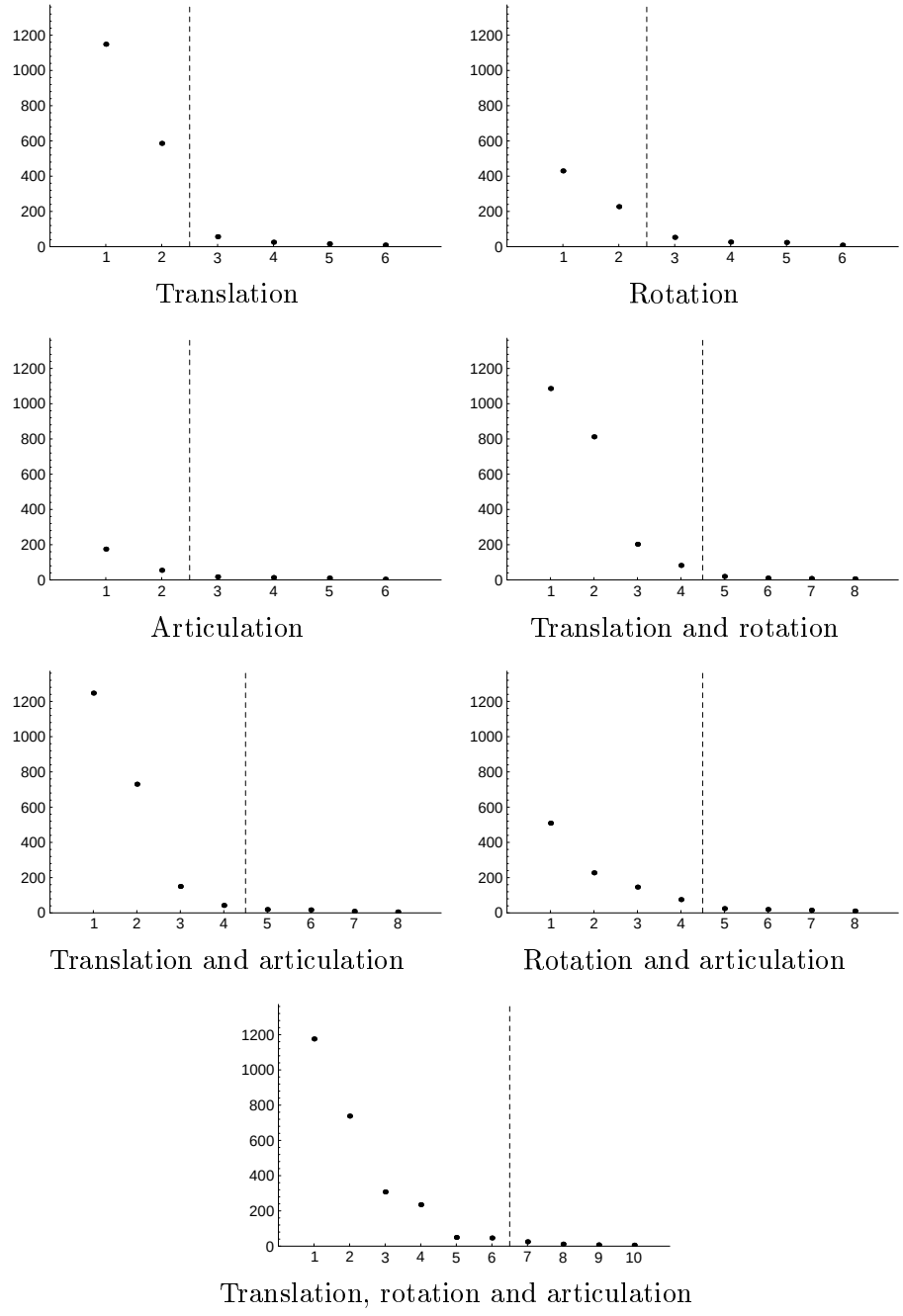


Figure 2.4: **Eigenvalues of principal components for various classes of motion of a hand.** *Plots of $\sqrt{\lambda_i}$ against i, where λ_i are the eigenvalues resulting from principal component analysis of sets of splines drawn round the hand in several positions while carrying out the class of motions indicated. The dotted line indicates the theoretical separation between motion within the linear subspace of the predicted size and any 'noise'.*

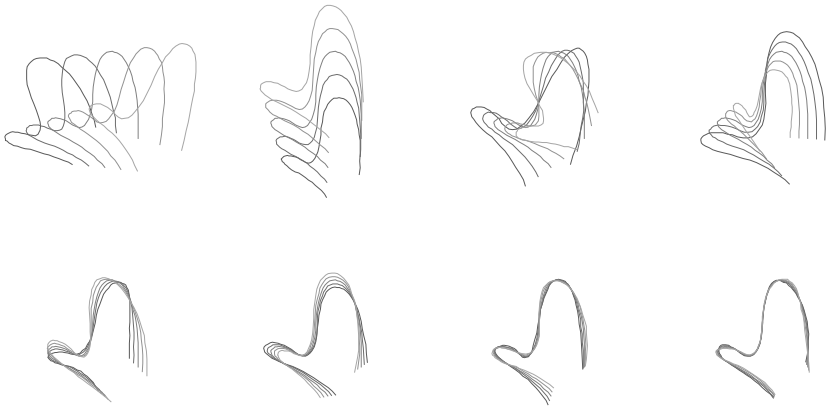


Figure 2.5: **Principal components analysis on configurations of a moving hand.** *The first eight principal components for a translating, rotating hand with articulating thumb, scaled according to the corresponding eigenvalue.*

Deformation

Also of interest are the effects of the articulation alone, without the translational and rotational parts, in the last case. Some means of 'subtracting' the translational and rotational contribution to the motion is therefore required. This is done by defining 'zoom' and 'rotated zoom' vectors for a template. The 'zoom' should correspond to enlargement about some sensible point; the centroid is the obvious choice for this, defined as follows. The centroid (x, y) of a control-point vector $\mathbf{Q} = (\mathbf{Q}^x, \mathbf{Q}^y)$ is defined by

$$x = \frac{\mathbf{1}\mathcal{H}_0\mathbf{Q}^x}{\mathbf{1}\mathcal{H}_0\mathbf{1}}; \quad y = \frac{\mathbf{1}\mathcal{H}_0\mathbf{Q}^y}{\mathbf{1}\mathcal{H}_0\mathbf{1}},$$

where $\mathbf{1}$ is a vector of length equal to the number of control points with each entry 1 ($\mathbf{0}$ is defined in a similar fashion) and $\mathcal{H}_0$ is the upper-left square submatrix of $\mathcal{H}$ of size equal to the number of control points.

Define control-point-space vectors corresponding to x- and y-translation as follows.

$$\mathbf{w}_x = (\mathbf{1}, \mathbf{0}); \quad \mathbf{w}_y = (\mathbf{0}, \mathbf{1}).$$

Now define the 'zoom' and 'rotated zoom' vectors for a template $\overline{\mathbf{Q}}$ to be

$$\mathbf{w}_z = \overline{\mathbf{Q}} - \bar{x}\mathbf{w}_x - \overline{y}\mathbf{w}_y; \quad \mathbf{w}_r = \mathbf{w}_z^{\perp}, \tag{2.14}$$

where the perpendicular operator $()^{\perp}$ is defined as

$$(\mathbf{x}, \mathbf{y})^{\perp} = (-\mathbf{y}, \mathbf{x}). \tag{2.15}$$

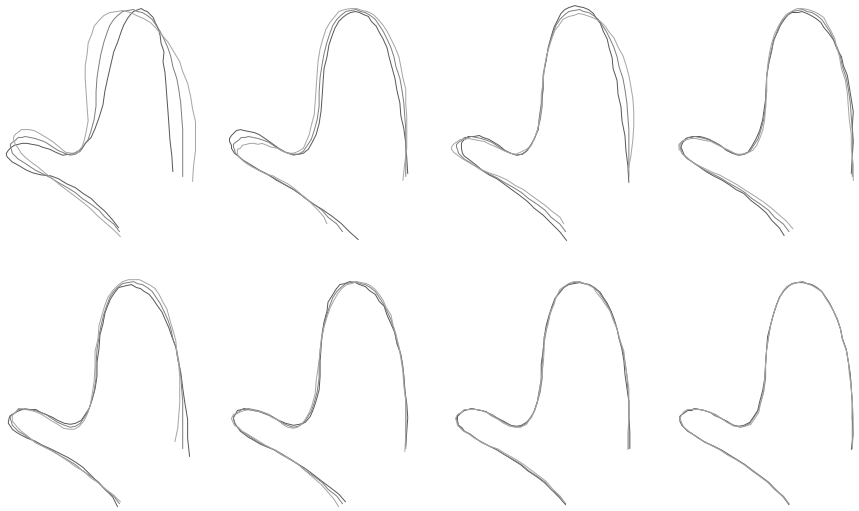


Figure 2.6: **Components of the vectors in figure 2.5 orthogonal to the space of Euclidean similarities, with PCA performed.** *The first eight principal components for a translating, rotating hand with articulating thumb, with the translational and rotational components removed (see text). The vectors are scaled according to the corresponding eigenvalue.*

Now for a control point vector $\mathbf{Q}$ define $\mathbf{Q}'$ to be the component of $\mathbf{Q}$ perpendicular to the space spanned by $\{\mathbf{w}_x, \mathbf{w}_y, \mathbf{w}_z, \mathbf{w}_r\}$. Here the inner product is formed as described in section 2.1.6.

This calculation was applied to each of the set of vectors drawn for the last case, and PCA performed on this new set of vectors. Plots of spans from the resulting eigenvectors are given in figure 2.6.

Discussion

It can be seen from figure 2.4 that the translation plot shows the clearest demarcation between the top n eigenvalues and the remainder, with the other motion classes having a less clear partition. There are several factors which contribute to this discrepancy between theory and practice.

One problem which arises in calculating the distance between two splines as defined using the metric of section 2.1.6 is that it is derived by integrating the squared distance between points corresponding to the same spline parameter. It is possible to produce two splines such that the curves are similar but the positions of the control points are quite different. The two splines will then have a significant 'distance' between them despite being intuitively similar. Each outline in the experiments was produced independently, and so

this situation could easily have arisen. An automatic spline-fitting system, together with some 'matching' similar to that used in (Ayache et al., 1992), would reduce this cause of error.

A second factor is that the distances considered occur at quite different scales — a translation from one edge of the field of view to the other produces a much larger distance than an articulation of the thumb. Thus a knee in the eigenvalue plot for 'translation with articulation' is unlikely. The lack of a knee in the curve is not an indication of the failure of the theory, however; of much greater importance is the size of the residue beyond the first n components.

The motion of the spline under articulation of the thumb does not correspond exactly to the type of motion considered in section 2.2.2 as pivoting of two rigid bodies, since the section of skin joining the thumb to the rest of the hand deforms significantly under such articulation. This is apparent in figure 2.5 — for a perfectly articulating body, only the first six eigenvectors should have any significant effect (when scaled according to the eigenvalue), and yet all of the first eight, shown, noticeably affect the contour. Similarly, only the first two vectors in figure 2.6 should show any marked deformation, and not the first five, as is the case.

Chapter 3

Basic Learning

3.1 Introduction

Section 2.1.3 introduced the idea of a dynamical model, to be used for the prediction phase of the Kalman filter used in one version of the 'active contour' tracker. The development of the tracking system then proceeded without undue consideration of how the dynamical model should be specified. When there is enough prior knowledge of the problem, it may in fact be possible to calculate an appropriate model explicitly.

Manually specifying the dynamical model has limitations, to be raised below, and this chapter will look at the concepts involved in 'learning' such a model from a training sequence, treating the problem as one of parameter estimation in the statistical sense. The idea is that a sequence of states $(x_1, x_2, \ldots, x_T)$ is treated as a (vector-valued) random variable, whose distribution depends on a set φ of parameters. The learning problem is then to estimate the parameters given a realisation $(x_1^*, x_2^*, \ldots, x_T^*)$, or in some cases a set of realisations, of this random variable, namely a training sequence. Within the constraints of an auto-regressive model, the problem has a relatively straightforward solution, whose derivation is given.

The beneficial effect of using a correctly-learnt model in a tracking system will be illustrated, although not in any great detail as these techniques have been known since the work of Blake et al. (1995), and an example of a different use — classification — of dynamical models will then be considered. In this application, the aim is to determine which model of a set of M candidates best explains a previously unseen sequence. Within the statistical framework, this problem is formalised as follows. There are M sets of parameters $\varphi_1, \varphi_2, \ldots, \varphi_M$, one for each model, learnt from training examples. The likelihood that the unknown sequence originated from each model can then be calculated, and the most likely model selected as the system's classification output for that sequence. The technique will

be illustrated with example data from electroencephalogram sequences used in classification of sleep patterns.

3.2 Learning a Dynamical Model from a Training Sequence

Section 2.1.3 introduced the dynamical model used in all tracking applications considered in this thesis. It takes the form of an auto-regressive model:

$$(\mathbf{x}_t - \overline{\mathbf{x}}) = A(\mathbf{x}_{t-1} - \overline{\mathbf{x}}) + B\mathbf{w}_t$$

for a suitably-dimensioned state-vector $\mathbf{x}_t$.

The deterministic component A and the coupling B of the noise into the system together constitute the parameters of the model. As discussed, they can be set by hand, but this can be impractical — the number of parameters to be set goes up as the square in the dimension of the state vector $\mathbf{x}$. An important problem is therefore that of automatically setting the dynamical model; in other words, *learning* an appropriate dynamical model from training data.

3.2.1 Maximum-likelihood Estimate

A sequence of states $(\mathbf{x}_1^*, \ldots, \mathbf{x}_T^*)$ can be considered as a realisation of a random process depending on a set of parameters φ. Here, $\varphi = (A, B)$:

$$(\mathbf{x}_1^*, \ldots, \mathbf{x}_T^*) \sim f(\mathbf{x}_1, \ldots, \mathbf{x}_T; \varphi). \tag{3.1}$$

The problem is to produce an estimate of φ given the realisation $(\mathbf{x}_1^*, \ldots, \mathbf{x}_T^*)$. The topic of how one can formulate an 'estimator', in other words a function $\delta()$ which yields an estimate of φ given a particular set of values $(\mathbf{x}_1^*, \ldots, \mathbf{x}_T^*)$ for the random variables in question, is covered by, for example, DeGroot (1985). The theory of 'Bayes estimators' requires one to specify both a prior distribution on the parameters φ, and a 'loss function' $L(\varphi, \alpha)$ which describes the cost of using a value α when the true value of the parameters is φ.

A relatively simple method of avoiding the necessity to specify these functions is to use the 'Maximum Likelihood' estimator, originally introduced by Fisher (1912). It is based on the idea that one should choose the φ which makes the realisation most likely, where the likelihood of certain value φ given the realisation is

$$f(\mathbf{x}_1^*, \ldots, \mathbf{x}_T^*; \varphi).$$

The maximum likelihood estimator of φ is then

$$\varphi_{ML} = \arg\max_{\varphi} f(\mathbf{x}_1^*, \ldots, \mathbf{x}_T^*; \varphi).$$

This estimator will be used here.

The Maximum A Posteriori Estimate

Within a Bayesian framework, the parameter vector φ is itself considered as a random variable. Suppose it has a prior distribution with density function g. Then the distribution in eqn (3.1) can be thought of as the posterior distribution for the state sequence given the parameters:

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_T; \varphi) = p(\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \varphi).$$

Now, by Bayes' rule, the distribution for φ given the sequence can be written as

$$\begin{aligned} p(\varphi \mid \mathbf{x}_1, \ldots, \mathbf{x}_T) &= \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \varphi) g(\varphi)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_T)} \\ &\propto f(\mathbf{x}_1, \ldots, \mathbf{x}_T; \varphi) g(\varphi). \end{aligned}$$

In the case of little prior information, the influence of $g(\varphi)$ on the posterior distribution can be neglected, reducing the maximum a posteriori estimate to the maximum likelihood one. In the case of learning a dynamical model for a tracking system, there is little that can be said a priori about the parameters, so using the maximum likelihood estimate is justified within a Bayesian setting. The results developed in this section have their ultimate roots in the work of Yule (1927) and Walker (1931), and are well understood (Gelb, 1974; Goodwin and Sin, 1984).

3.2.2 One-dimensional First-order Systems

To introduce the ideas used, the problem of finding the maximum-likelihood estimate of a first-order dynamical model for a one-dimensional system will be addressed. The dynamical model is then of the form

$$x_t = a x_{t-1} + b w_t,$$

where the w_t are independent and each $w_t \sim N(0, 1)$. Here only systems with zero mean are considered. The analysis extends easily to systems with known mean (by subtracting the mean from each datum); the extension to unknown mean is given later.

A training sequence $x_1^*, x_2^*, \ldots, x_T^*$ of states is given. The problem is to find the values of a and b which maximise the likelihood $p(x_1^*, x_2^*, \ldots, x_T^* \,|\, a, b)$, or, equivalently, the log-likelihood L. From the independence of the w_t, this can be written as

$$L(x_1^*, x_2^*, \ldots, x_T^* \,|\, a, b) = \sum_{t=2}^{T} \log[p_{bw_t}(x_t^* - a x_{t-1}^*)].$$

Now,

$$p_{bw_t}(x) = \frac{1}{\sqrt{2\pi}b} \exp\left[-\frac{1}{2b^2}x^2\right],$$

so, up to an additive constant,

$$L = -(T-1)\log b - \frac{1}{2b^2}\sum (x_t^* - a x_{t-1}^*)^2.$$

Expanding the squared terms inside the sum gives:

$$L = -(T-1)\log b - \frac{1}{2b^2}(r_{00} - 2a r_{01} + a^2 r_{11}),$$

where the moments r_{ij} are given by

$$r_{ij} = \sum_{t=2}^{T} x_{t-i}^* x_{t-j}^*.$$

Setting $\partial L/\partial a = 0$ gives

$$a = \frac{r_{01}}{r_{11}},$$

and setting $\partial L/\partial b = 0$ gives

$$b^2 = \frac{1}{T-1}(r_{00} - 2a r_{01} + a^2 r_{11}).$$

There is an inevitable ambiguity of sign in b, since $\pm b w_n$ are identically distributed.

3.2.3 Vector Second-order Systems

For the purposes of prediction with a tracking system, a first-order model is inadequate — its range of modes is limited to simple exponential decay or growth. A second-order model allows the resulting system to display oscillatory behaviour, and is often a more appropriate representation of the real motion. Also, vector systems must be considered. This section develops the theory for learning such a system.

The problem now is to find the maximum likelihood estimate of a system of the form

$$\mathbf{x}_t = A_1 \mathbf{x}_{t-1} + A_2 \mathbf{x}_{t-2} + B\mathbf{w}_t$$

from a training sequence $\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_T^*$. Here, A_1, A_2, and B are matrices, and $\mathbf{w}_t$ are independent multivariate Gaussians; each $\mathbf{w}_t \sim N(0, I)$.

As before, the log-likelihood can be written up to an additive constant as

$$L = -\frac{1}{2}\sum_{t=3}^{T}(\mathbf{x}_t^* - A_1\mathbf{x}_{t-1}^* - A_2\mathbf{x}_{t-2}^*)^\top B^{-\top}B^{-1}(\mathbf{x}_t^* - A_1\mathbf{x}_{t-1}^* - A_2\mathbf{x}_{t-2}^*) - (T-2)\log|B|.$$

Following Blake et al. (1995), this can be re-written as follows, using $x^\top Ay = \mathrm{tr}[(xy^\top)A]$ (where $\mathrm{tr}(A)$ is the trace of a matrix A).

$$L = -\frac{1}{2}\mathrm{tr}(ZC^{-1}) - \frac{1}{2}(T-2)\log|C|,$$

where $C = BB^\top$ and

$$\begin{aligned} Z = {} & R_{00} - R_{01}A_1^\top - R_{02}A_2^\top \\ & - A_1R_{10} + A_1R_{11}A_1^\top + A_1R_{12}A_2^\top \\ & - A_2R_{20} + A_2R_{21}A_1^\top + A_2R_{22}A_2^\top, \end{aligned}$$

the matrix moments R_{ij} being given by

$$R_{ij} = \sum_{t=3}^{T}\mathbf{x}_{t-i}^*(\mathbf{x}_{t-j}^*)^\top.$$

L must be differentiated with respect to both the A_k matrices and C; the following derivatives will be needed.

$$\begin{aligned} \frac{\partial}{\partial A}\left[\mathrm{tr}(ASC^{-1})\right] &= C^{-\top}S^\top; \\ \frac{\partial}{\partial A}\left[\mathrm{tr}(SA^\top C^{-1})\right] &= C^{-\top}S; \\ \frac{\partial}{\partial A}\left[\mathrm{tr}(ASA^\top C^{-1})\right] &= C^{-\top}A\left(S + S^\top\right). \end{aligned}$$

This results in the following equations for the deterministic parameters A_1 and A_2.

$$\begin{aligned} A_1R_{11} + A_2R_{21} &= R_{01}; \\ A_1R_{12} + A_2R_{22} &= R_{02}. \end{aligned}$$

Using $\partial|M|/\partial M = |M|M^{-\top}$ gives the following expression for C.

$$\begin{aligned} C = {} & \frac{1}{T-2}\Big(R_{00} - R_{01}A_1^\top - R_{02}A_2^\top \\ & \qquad - A_1R_{10} + A_1R_{11}A_1^\top + A_1R_{12}A_2^\top \\ & \qquad - A_2R_{20} + A_2R_{21}A_1^\top + A_2R_{22}A_2^\top\Big) \\ = {} & \frac{1}{T-2}\left(R_{00} - A_1R_{01} - A_2R_{02}\right), \end{aligned}$$

where the simplification comes from the equations satisfied by the A_k. Note that only C can be found, as any orthogonal transformation of B will produce the same distribution for $B\mathbf{w}_t$. This is the same phenomenon as the sign ambiguity in the scalar case.

3.2.4 General-order Systems with Unknown Mean

The general form of a system with unknown mean and model order K is

$$\mathbf{x}_t = \sum_{k=1}^{K} A_k \mathbf{x}_{t-k} + \mathbf{d} + B\mathbf{w}_t,$$

where the offset $\mathbf{d}$ is related to the system mean $\overline{\mathbf{x}}$ by

$$\mathbf{d} = (I - \sum_{k=1}^{K} A_k)\overline{\mathbf{x}}.$$

Specifying $\mathbf{d}$ rather than $\overline{\mathbf{x}}$ is more general, as it allows systems which have no mean, such as a constant-acceleration system. In this case, the maximum-likelihood estimate of the system parameters $(A_k, C, \mathbf{d})$ is derived in appendix A.3.2. The result is that

$$\begin{aligned} AR^0 &= \mathbf{R}_0^0; \\ C &= \frac{1}{T-K}\left(R_{00}^0 - A(\mathbf{R}_0^0)^\top\right); \\ \mathbf{d} &= \frac{1}{T-K}(R_0 - A\mathbf{R}), \end{aligned}$$

where

$$\mathbf{R} = \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_K \end{pmatrix}; \qquad R_i = \sum_{t=K+1}^{T} \mathbf{x}_{t-i}^*,$$

and

$$R^0 = \begin{pmatrix} R_{11}^0 & R_{12}^0 & \cdots & R_{1,K}^0 \\ R_{21}^0 & R_{22}^0 & \cdots & R_{2,K}^0 \\ \vdots & \vdots & \ddots & \vdots \\ R_{K,1}^0 & R_{K,2}^0 & \cdots & R_{K,K}^0 \end{pmatrix}; \qquad \mathbf{R}_0^0 = \begin{pmatrix} R_{01}^0 & R_{02}^0 & \cdots & R_{0,K}^0 \end{pmatrix},$$

with

$$R_{ij}^0 = R_{ij} - \frac{1}{T-K} R_i R_j^\top; \qquad R_{ij} = \sum_{t=K+1}^{T} \mathbf{x}_{t-i}^* (\mathbf{x}_{t-j}^*)^\top.$$

3.3 Filtered Learning

In many situations, and in particular the domain of visual tracking considered here, a 'ground truth' sequence of state vectors is not available. All that is available is a sequence of images. If the environment within which the training image sequence is gathered can be controlled, for example by recording it in a situation with low clutter, a sensibly-specified default dynamical model may well result in adequate tracking. This filtering produces a sequence of *estimated* state vectors, and one obvious approximation to make is that these estimates are good enough to treat as the true state vectors in the learning algorithm. The calculation of the maximum-likelihood estimate of a dynamical model from the estimated state vectors of a training sequence will be referred to as 'filtered learning'.

This is the approach taken by Blake et al. (1995), where dynamical models were learnt for a hand-tracking application, and also a lip tracker. In all cases, using the filtered learning algorithm produced a more reliable tracker. As an illustration, one of the lip-tracking results is shown in figure 3.1.

Unavoidably, however, when the tracker has been trained to follow one type of motion (for example, horizontal (in the image plane) oscillation), it performs less well when presented with motion of a different type (for example vertical oscillation). It is difficult to create a dynamical model which at the same time covers all possible motions of the object without having so much generality that the benefits of having a model are lost.

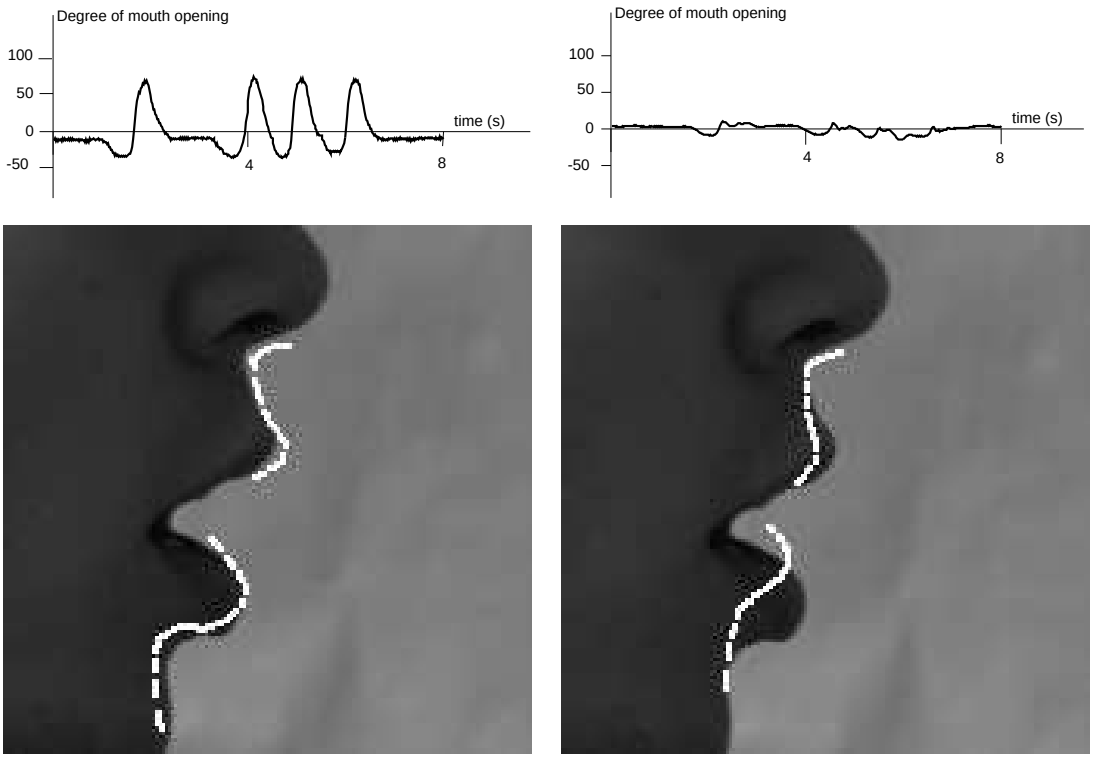
3.4 Classification using Learnt Dynamical Models

The use of a dynamical model within a Kalman filter, for example the vision-based trackers discussed, is for prediction. A model is chosen which best fits training data, within some constraints (such as fixed-order auto-regressive models with Gaussian noise). This model is then used to provide a prior for tracking motion of a similar type.

Once a dynamical mode has been learnt for a particular type of motion, it can be used to judge how likely it is that a test sequence has arisen from that model, by means of the log-likelihood:

$$L = \log[p(\text{sequence} \,|\, \text{model})].$$

With more than one model, each learnt to represent a different type of motion, this technique can be used to classify a test sequence — evaluate the log-likelihoods for each model, and choose the model with largest L.



(a) Trained on 'Pah' (b) Trained on 'Ooh'

Figure 3.1: **Filtered learning produces a more reliable tracker.** *A profile view of a speaker repeating the syllable 'Pah' is tracked, using two models produced using filtered learning. The first, shown in (a), was trained using a sequence of repeated 'Pah' utterances. The second model, shown in (b), was trained on a series of 'Ooh' syllables. The model learnt from matching training data performs much better, as can be seen both in the estimated time-series (showing estimated mouth opening as a function of time) and the snapshots. Figures courtesy of A. Blake and M. Isard.*

EEG Data

As an example (non-vision) application, the problem of classifying sleep types from EEG data is addressed. The experiments described are the original work of the author. An electroencephalogram (EEG) has been successfully used for automatic classification of sleep types, with the system involving a neural network (Pardey et al., 1995; Pardey et al., 1996). This section investigates a simpler alternative based on likelihoods. A subset of the problem addressed in (Pardey et al., 1996) will be used, that of distinguishing between three types of sleep state — wakefulness (dubbed `Wake` henceforth), REM sleep (`REM`), and deep sleep (`Sleep4`).

Data used in this section were kindly provided by Neil Townsend. The data consist of samples at 128 Hz, digitised to 8 bits, unsigned. Data from nine healthy and normal subjects were available, with sequence lengths ranging from just over 5 hours to around 10 hours. This corresponds to a few million time-steps at 128 Hz. These data sequences were divided into epochs of 30 s, and each epoch was given a 'ground truth' classification, decided upon by human experts.

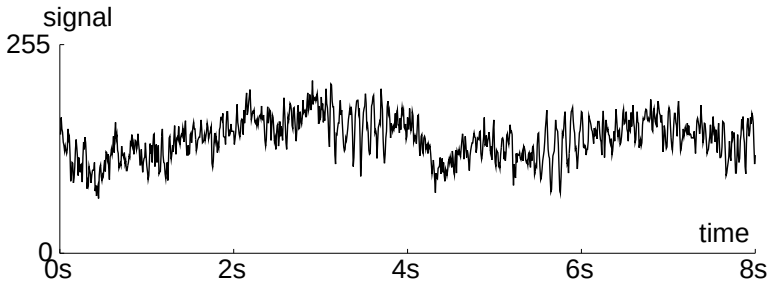
3.4.1 Learning the Models

For each of the three types of sleep under consideration, 10 epochs of that class were chosen at random from the sequences. For ease of spectral analysis, the first 16 s section of each epoch only was used, this corresponding to 2048 time-steps.

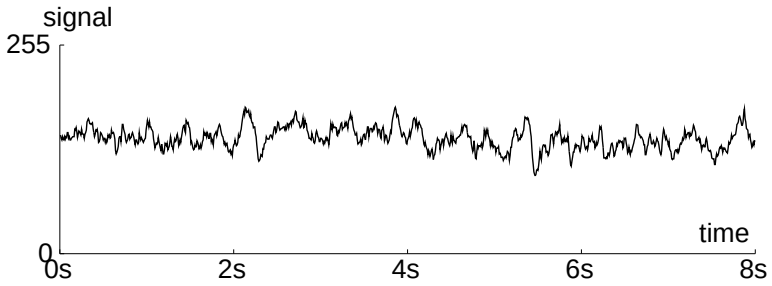
Example Training Data

Traces for examples drawn from each training set are shown in figure 3.2. As noted, the sequences were actually 2048 steps long, but only the first 1024 are shown here for clarity. Approximations to the spectra of the signals, gained by operating on the whole (2048) sequence with the Discrete Fourier Transform, are shown in figure 3.3. The graphs have been smoothed slightly. The large DC component has been subtracted off these and all subsequent spectra.

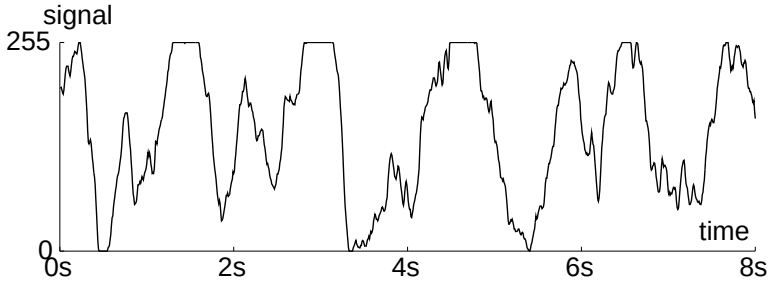
From these training sequences, models were learnt using the standard Yule-Walker equations, with moments added from the 10 examples (see appendix A.4). Even model orders from 2 to 12 were used.



(a) Example of `Wake` training data

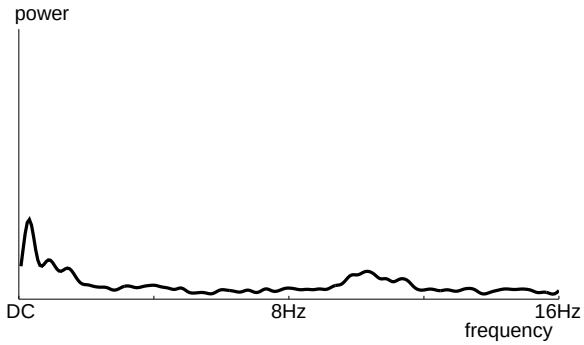


(b) Example of `REM` training data

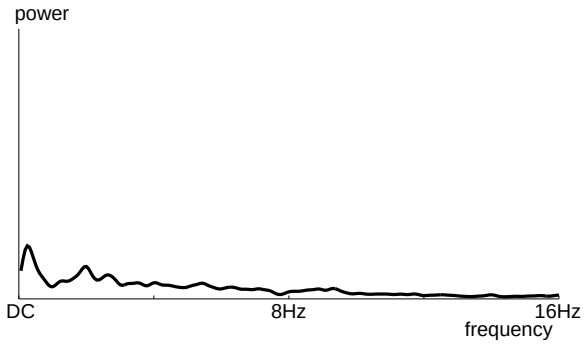


(c) Example of `Sleep4` training data

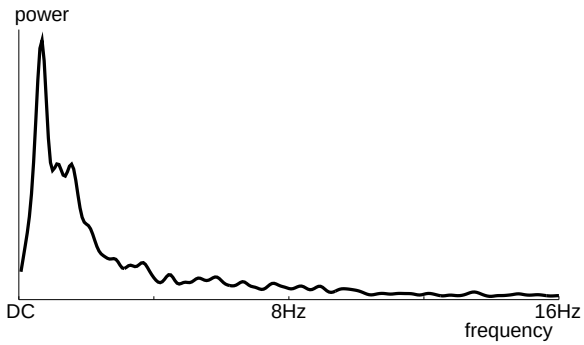
Figure 3.2: **Time-series of training data for `Wake`, `REM`, and `Sleep4` states exhibit significant differences.** *For each sleep state, one of the ten epochs used for training purposes is shown. The behaviours of the categories are quite distinct. The saturation of the sensor in (c) is of interest, and is discussed later.*



(a) Discrete Fourier Transform of example `Wake` training data



(b) Discrete Fourier Transform of example `REM` training data

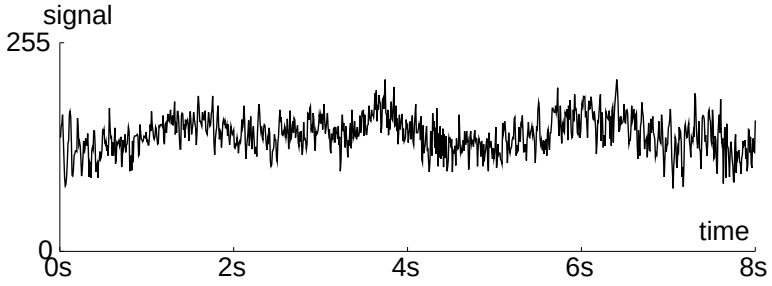


(c) Discrete Fourier Transform of example `Sleep4` training data

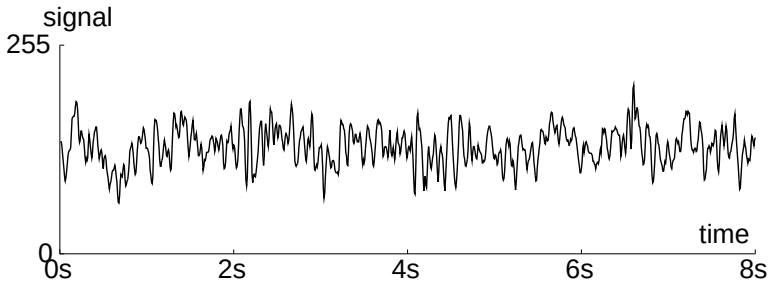
Figure 3.3: **Spectra of example training data have different characteristics.** *The sequences shown in figure 3.2 have been subjected to a discrete Fourier transform. The differing power distributions between the sleep states can be seen.*

3.4.2 Classifying with the Models

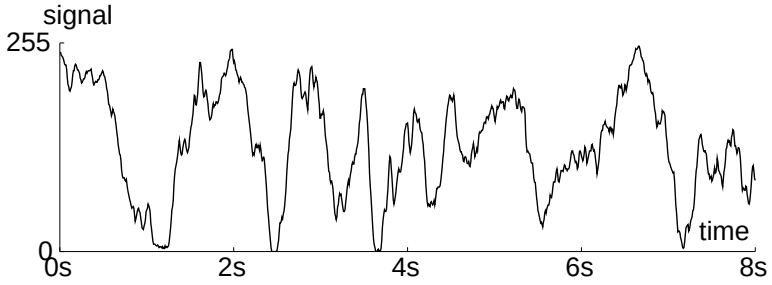
From the data available, 10 test sequences of each type were chosen at random (with the condition that the test sequences be disjoint from the training sequences). Time series of example test sequences are shown in figure 3.4.



(a) Example of `Wake` test data



(b) Example of `REM` test data

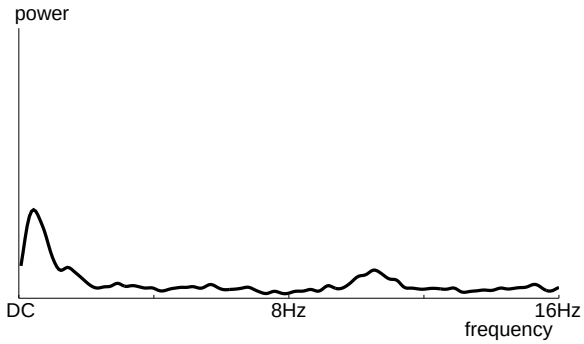


(c) Example of `Sleep4` test data

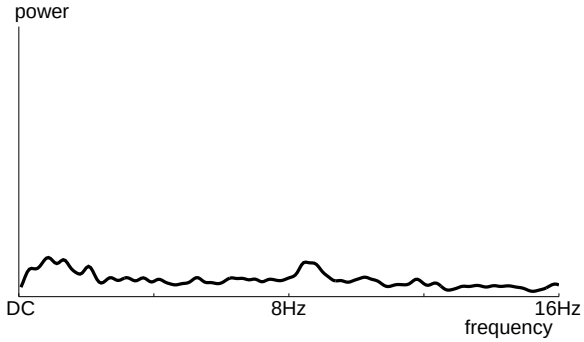
Figure 3.4: **Test data show similarities to the training sequences.** *The sequences shown here bear a strong resemblance to the corresponding time-series of figure 3.2, indicating that the use of dynamical models should be a sensible approach to the classification problem.*

Approximation to the spectrum of the signals, gained by operating on the whole (2048) sequence with the Discrete Fourier Transform, are shown in figure 3.5. Again, the graphs have been smoothed slightly.

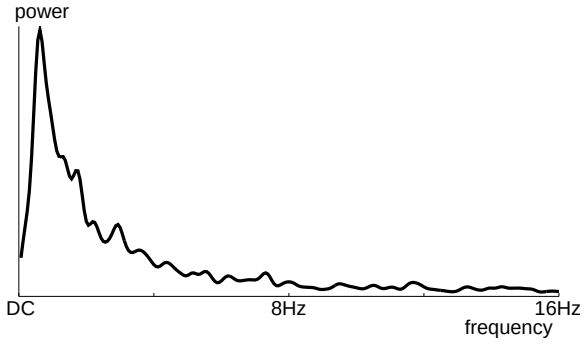
The learnt dynamical models of even orders 2–12 were used to evaluate the likelihood of



(a) Discrete Fourier Transform of example `Wake` test data



(a) Discrete Fourier Transform of example `REM` test data



(a) Discrete Fourier Transform of example `Sleep4` test data

Figure 3.5: **Spectra of example test data exhibit similar characteristics to those of training data.** *The discrete Fourier transform has been used to approximate the spectra of the time-series of figure 3.4. The similarity between these and the spectra of the training sequences shown in figure 3.3 is apparent.*

these 30 test sequences. The results of classifying the test sequences using models of each order are summarised in table 3.1.

		Classifier output		
		Wake	Sleep4	REM
True State	Wake	0.9	0.0	0.1
	Sleep4	0.0	1.0	0.0
	REM	0.1	0.0	0.9

2nd-order model

		Classifier output		
		Wake	Sleep4	REM
True State	Wake	0.9	0.0	0.1
	Sleep4	0.0	1.0	0.0
	REM	0.1	0.0	0.9

4th-order model

		Classifier output		
		Wake	Sleep4	REM
True State	Wake	0.9	0.0	0.1
	Sleep4	0.0	1.0	0.0
	REM	0.1	0.0	0.9

6th-order model

		Classifier output		
		Wake	Sleep4	REM
True State	Wake	0.9	0.0	0.1
	Sleep4	0.0	1.0	0.0
	REM	0.1	0.0	0.9

8th-order model

		Classifier output		
		Wake	Sleep4	REM
True State	Wake	0.9	0.0	0.1
	Sleep4	0.0	1.0	0.0
	REM	0.1	0.0	0.9

10th-order model

		Classifier output		
		Wake	Sleep4	REM
True State	Wake	0.9	0.0	0.1
	Sleep4	0.0	1.0	0.0
	REM	0.0	0.0	1.0

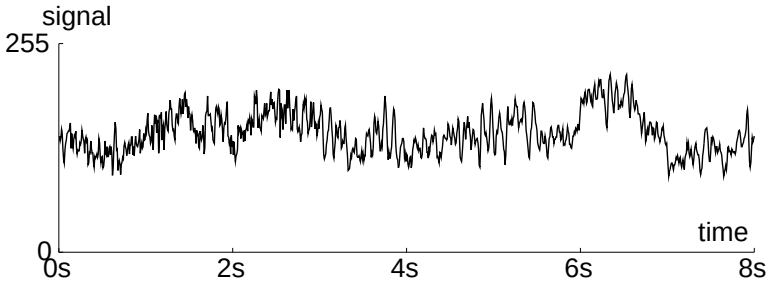
12th-order model

Table 3.1: **Classification of epochs using dynamical models is successful.** *Ten examples of each type of sleep were presented to each learnt model, and the proportions of each type which gave each output classification are shown here. (For example, a second-order model, when presented with 10 known* `Wake` *sequences, gave the classification* `Wake` *for 9 of them and* `REM` *for the remaining 1.) The results are good using models with as low an order as* 2*. Increasing the model order does not bring any improvement in the classification results until it reaches 12, at which stage the mis-classification of one of the* `REM` *sequences as* `Wake` *disappears.*

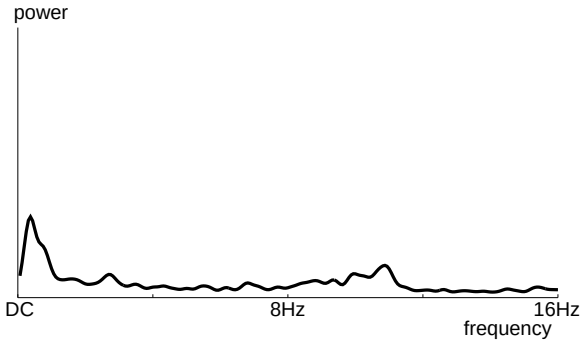
Even a second-order model provides good discriminatory power between the three states. Increasing the model order as far as 12 does cause one of the misclassifications to become correct, but the other remains. In general, though, classification by straightforward log-likelihood ranking is very successful. More experiments would be required to fairly compare with the neural-network-based system of (Pardey et al., 1996) — considering all six states instead of just three, and working with shorter epochs (1 s intervals are classified). Time did not permit this work to be carried out; it could be a topic for future investigation.

Misclassified Test Data

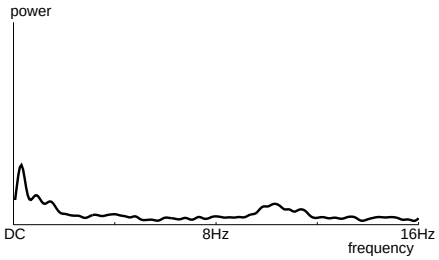
These are the examples which gave trouble with the classification. Firstly there is a section whose correct classification is `Wake`, but which was incorrectly labelled `REM` by the algorithm. Its time-series and spectrum are given in figure 3.6, together, for comparison, with copies of the example training spectra for `Wake` and `REM` states.



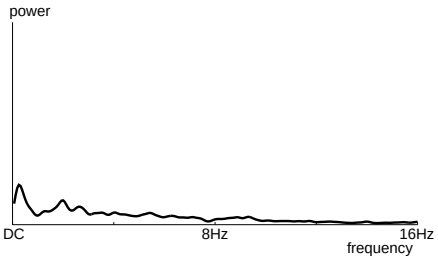
(a) Time series of `Wake` sequence incorrectly labelled as `REM`



(b) Discrete Fourier Transform of `Wake` sequence incorrectly labelled as `REM`



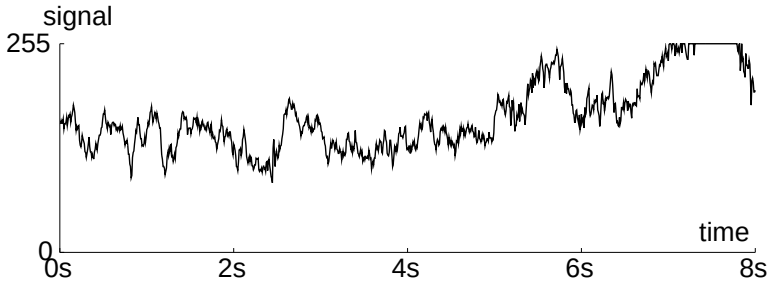
(c) Spectrum of `Wake` training data



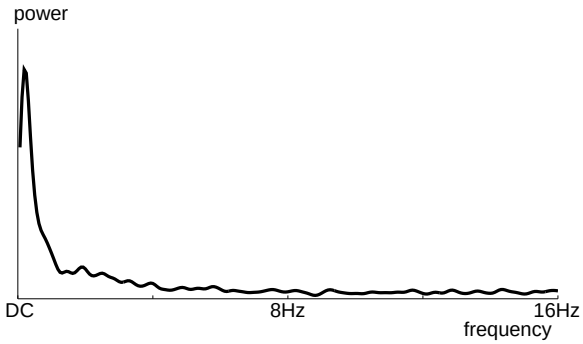
(d) Spectrum of `REM` training data

Figure 3.6: **Classification using dynamical models is not perfect — `Wake` test data is labelled `REM`.** *Classification based on likelihoods of dynamical models is not perfect; this section of* `Wake` *test data has the greatest likelihood of arising from the learnt* `REM` *model. Comparing the time-series to the* `REM` *test sequence of figure 3.4(b), a similarity is present.*

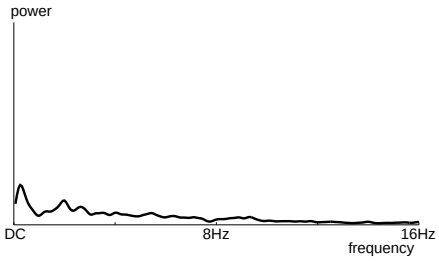
Secondly, there is a section of REM data which was incorrectly labelled as Wake, which is illustrated in figure 3.7.



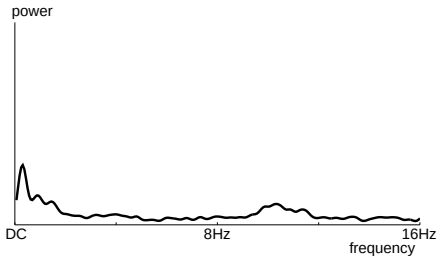
(a) Time series of `REM` sequence incorrectly labelled as `Wake`



(b) Discrete Fourier Transform of `REM` sequence incorrectly labelled as `Wake`



(c) Spectrum of `REM` training data



(d) Spectrum of `Wake` training data

Figure 3.7: **Second erroneous classification — `REM` test data labelled `Wake`.** *This* `REM` *sequence appears fairly unrepresentative of any state to a human observer, when compared to the time-series of* `REM` *and* `Wake` *states given in previous figures.*

It is understandable that this second example was misclassified — there is a large section at around 7–8 s which is extremely atypical. The sensor has exhibited 'clipping' or 'saturation' (returning a reading of 255 for a true value which is almost certainly higher). The phenomenon is effectively one of 'missing data'; this will be addressed in section 4.6. Time did not permit the implementation of the theory, and so no experimental evidence is available. It would be interesting to see whether explicitly handling the clipping could correct the problematic classifications.

Remarks

A feature of the data considered here is that their amplitude can not truly be treated as bearing any information — factors such as the gain setting on the instrument and the quality of the electrode/patient contact have an influence on the overall scale of the readings. Therefore, using the driving noise B as part of a classification system is perhaps disingenuous, as its only effect in a scalar system is to scale the whole sequence. Thought should be given to calculating likelihoods in a scale-independent manner.

Another interesting point is that auto-regressive model fitting can be used as a spectrum-estimation technique (Pardey et al., 1996), and therefore it may well be possible to consider the likelihood comparison as an operation in the frequency domain. A human observer can certainly perform such a comparison. Investigation of this idea could be an area for future work.

Chapter 4

Learning Dynamical Models using EM

4.1 Introduction

Section 3.3 described how specifying a dynamical model for a Kalman-filter-based tracker can be achieved by learning from a training sequence. In the absence of true states, the approximation was made that the filtered estimates given by tracking the training sequence using a hand-set model were close enough to the true states to treat them as such.

Consider, however, the results of applying the filtered-learning algorithm to a synthetic first-order scalar system with different levels of measurement noise, shown in table 4.1. For the true values of a and b shown, a state sequence was generated, as were measurements according to the standard measurement model $z = hx + v$. These measurements were then filtered with a dynamical model close to, but not exactly equal to, the true one, and the resulting estimates used to produce the 'filtered-learnt' system. This learnt system is shown for various values of the relative importance of the process and measurement noises.

σ^2/h^2b^2	**Estimate of** a ($a = 0.8$)	**Estimate of** b ($b = 0.2$)
0.01	0.803	0.184
0.25	0.780	0.100
1.00	0.720	0.055
4.00	0.666	0.032

Table 4.1: **Filtered learning becomes increasingly inaccurate as measurement noise increases.** *The model resulting from applying filtered learning to a synthetic first-order scalar system is shown against the level of the measurement noise in the system. It can be seen that for low measurement noise, the estimate is reasonable, but as measurement noise increases, the estimate of the process noise in particular becomes extremely inaccurate.*

The filtered learning approach, then, works well when there is low measurement noise. In such circumstances, the filtered state estimates are good compared with the true states. However, when measurement noise is higher (in a visual tracking system, this situation arises in the presence of image clutter or motion blur, for example), the accuracy of the filtered estimates drops, and a dynamical model learnt from these estimates can be grossly incorrect, particularly with respect to the process noise learnt. This chapter addresses the problem of learning the dynamical model directly from a sequence of measurements, using the statistical technique of Expectation-Maximisation (EM).

The EM algorithm, as introduced by Dempster et al. (1977), is a general iterative technique for performing maximum likelihood estimation of parameters in problems which can be regarded as having 'missing data'. It has been applied in various situations, for example, learning the parameters describing a neural network for pattern recognition (Bishop, 1995; Ripley, 1996), and learning Hidden Markov Models for use in speech recognition problems (Huang et al., 1990; Rabiner and Juang, 1993). It has been used for computer vision research in areas including optic flow computation (Jepson and Black, 1993) and the processing of noisy images (Tanaka and Katayama, 1989), and its convergence properties have been studied (Wu, 1983; Boyles, 1983).

The technique of EM is a natural approach to apply to the present problem: the missing data are the true states describing the configuration of the contour, and the observed data are the image-based measurements. This chapter develops the theory to solve the problem using EM, and demonstrates its use on synthetic data. The following chapter shows the improvement gained by using models learnt in this way as the prediction component of a real-time dynamic contour tracker. The chapters contain material originally published by North and Blake (1997; 1998).

4.2 The Expectation-Maximisation Algorithm

The derivation of the fundamental EM theory in this section follows (Dempster et al., 1977). Suppose a problem can be represented in terms of two measure spaces: Z, a space of observable data, and X, one of unobservable data. Suppose also that there is a parameter vector φ on which the distributions of Z and X depend. The aim is to find that φ which maximises $L(z, \varphi) = \log[p(z \,|\, \varphi)]$ for a given set z of observed data. For example, z might be a set of samples from a Gaussian mixture; here φ would consist of the means and variances together with the mixture probabilities. In general, finding this φ is not possible analytically, and so some approximate algorithm must be used instead. The Expectation-

Maximisation algorithm produces a series of estimates for φ, each one producing a greater value for L. The procedure can then be run to convergence of φ.

Given an estimate φ_i for φ, the question is how to produce another one, φ_{i+1}, which is better in the sense that $L(z, \varphi_{i+1}) > L(z, \varphi_i)$. Now

$$p(z, x \mid \varphi) = p(z \mid \varphi) p(x \mid z, \varphi),$$

and so

$$L(z, \varphi) = \log[p(z, x \mid \varphi)] - \log[p(x \mid z, \varphi)]. \tag{4.1}$$

Now consider the following:

$$\begin{aligned}
\log[p(z \mid \varphi_{i+1})] &= \log[p(z \mid \varphi_{i+1})] \int p(x \mid z, \varphi_i)\, \mathrm{d}x \\
&= \int \log[p(z \mid \varphi_{i+1})] p(x \mid z, \varphi_i)\, \mathrm{d}x \\
&= \mathcal{E}[\log[p(z \mid \varphi_{i+1})] \mid z, \varphi_i] \\
&= \mathcal{E}[L(z, \varphi_{i+1}) \mid z, \varphi_i],
\end{aligned}$$

where $\mathcal{E}$ is the expectation operator over the distribution for x.

Using (4.1) this becomes

$$L(z, \varphi_{i+1}) = Q(\varphi_i, \varphi_{i+1}) - H(\varphi_i, \varphi_{i+1}),$$

where

$$Q(\varphi_i, \varphi_{i+1}) = \mathcal{E}[\log[p(z, x \mid \varphi_{i+1})] \mid z, \varphi_i];$$

$$H(\varphi_i, \varphi_{i+1}) = \mathcal{E}[\log[p(x \mid z, \varphi_{i+1})] \mid z, \varphi_i].$$

Now, $H(\varphi_i, \varphi_{i+1}) \leq H(\varphi_i, \varphi_i)$ for any φ_{i+1}:

$$\begin{aligned}
H(\varphi_i, \varphi_{i+1}) - H(\varphi_i, \varphi_i) &= \int \log \left[\frac{p(x \mid z, \varphi_{i+1})}{p(x \mid z, \varphi_i)} \right] p(x \mid z, \varphi_i)\, \mathrm{d}x \\
&\leq \int \left[\frac{p(x \mid z, \varphi_{i+1})}{p(x \mid z, \varphi_i)} - 1 \right] p(x \mid z, \varphi_i)\, \mathrm{d}x \\
&= \int p(x \mid z, \varphi_{i+1})\, \mathrm{d}x - \int p(x \mid z, \varphi_i)\, \mathrm{d}x \\
&= 0,
\end{aligned}$$

where $\log a \leq a - 1$ has been used to give the inequality, and the last line arises because both integrals evaluate to 1. (The case of a discrete distribution is similar to that of a continuous one.)

Therefore, picking any φ_{i+1} such that $Q(\varphi_i, \varphi_{i+1}) > Q(\varphi_i, \varphi_i)$ will increase the value of L; in particular, finding the φ_{i+1} which maximises Q will produce a better estimate for the parameters. This procedure can then be iterated until a satisfactory estimate of φ is produced.

Explicitly, choosing φ_{i+1} to maximise

$$Q(\varphi_i, \varphi_{i+1}) = \int \log[p(z, x \mid \varphi_{i+1})] p(x \mid z, \varphi_i) \, \mathrm{d}x$$

in the continuous case, or

$$Q(\varphi_i, \varphi_{i+1}) = \sum_x \log[p(z, x \mid \varphi_{i+1})] P(x \mid z, \varphi_i)$$

in the discrete case, will produce a better estimate for φ.

Gaussian Mixture Example

Consider as an illustrative example the case where z is a set of N independent data points drawn from a Gaussian mixture of M components. The case where each component is constrained to be isotropic is presented as an example in (Bishop, 1995). The set φ of parameters then comprises the means μ_j and variances Σ_j as well as the distribution $P(j)$ of the components. For ease of notation, write superscript 0 to indicate distributions given $\varphi = \varphi_i$, and similarly for superscript 1. In this case, x is a vector such that x^n is the mixture component from which z^n came.

Then

$$\begin{aligned} \log[p^1(z, x)] &= \sum_n \log[P^1(x^n) p^1(z^n \mid x^n)] \\ &= \sum_n \sum_j \delta_{j x^n} \log[P^1(j) p^1(z^n \mid j)], \end{aligned}$$

giving

$$Q = \sum_x \sum_n \sum_j \delta_{j x^n} \log[P^1(j) p^1(z^n \mid j)] P^0(x \mid z).$$

Now

$$P^0(x \mid z) = \prod_m P^0(x^m \mid z^m),$$

so

$$Q = \sum_{x^1} \cdots \sum_{x^N} \sum_n \sum_j \delta_{j x^n} \log[P^1(j) p^1(z^n \mid j)] \prod_m P^0(x^m \mid z^m).$$

Now reorder the summations, take the logarithm term outside the summations over x^i and consider the following.

$$\begin{aligned}\sum_{x^1}\cdots\sum_{x^N}\delta_{jx^n}\prod_m P^0(x^m\,|\,z^m) &= \sum_{x^1}\cdots\sum_{x^{n-1}}\sum_{x^{n+1}}\cdots\sum_{x^N}P^0(j\,|\,z^n)\prod_{m\neq n}P^0(x^m\,|\,z^m)\\ &= P^0(j\,|\,z^n)\sum_{x^1}\cdots\sum_{x^{n-1}}\sum_{x^{n+1}}\cdots\sum_{x^N}\prod_{m\neq n}P^0(x^m\,|\,z^m)\end{aligned}$$

and, for ease of notation writing $(x^1,\ldots,x^{n-1},x^{n+1},\ldots,x^N)=(u^1,\ldots,u^{N-1})$,

$$\begin{aligned}\sum_{u^1}\cdots\sum_{u^{N-1}}\prod_m P^0(u^m\,|\,z^m) &= \sum_{u^1}\left(P^0(u^1\,|\,z^1)\left(\cdots\left(\sum_{u^{N-1}}P^0(u^{N-1}\,|\,z^{N-1})\right)\cdots\right)\right)\\ &= 1\end{aligned}$$

by substituting from the innermost sum outwards. Therefore,

$$Q=\sum_n\sum_j P^0(j\,|\,z^n)\log[P^1(j)p^1(z^n\,|\,j)].$$

In the case of a Gaussian mixture,

$$p^1(z^n\,|\,j)=(2\pi)^{-d/2}|\Sigma_j^1|^{-1/2}\exp\left[-\frac{1}{2}(z^n-\mu_j^1)^\top(\Sigma_j^1)^{-1}(z^n-\mu_j^1)\right],$$

so

$$\log[p^1(z^n\,|\,j)]=-\frac{d}{2}\log(2\pi)-\frac{1}{2}\log|\Sigma_j^1|-\frac{1}{2}(z^n-\mu_j^1)^\top(\Sigma_j^1)^{-1}(z^n-\mu_j^1),$$

where d is the dimension of each z^n. Thus, to within an additive constant,

$$\begin{aligned}Q=\sum_n\sum_j P^0(j\,|\,z^n)\Bigg[&\log P^1(j)-\frac{1}{2}\log|\Sigma_j^1|\\ &-\frac{1}{2}(z^n-\mu_j^1)^\top(\Sigma_j^1)^{-1}(z^n-\mu_j^1)\Bigg]\end{aligned}$$

and it is this which must be maximised with respect to the $P^1(j)$, the μ_j^1 and the Σ_j^1. There is the slight complication that there is a constraint on the $P^1(j)$, namely that their sum must be 1. Therefore, introduce a Lagrange multiplier λ and maximise the quantity

$$\hat{Q}=Q+\lambda\left(\sum_j P^1(j)-1\right), \tag{4.2}$$

including λ in the set of variables over which the maximisation takes place.

Setting $\partial\hat{Q}/\partial[P^1(j)]=0$ gives

$$\sum_n\left[\frac{P^0(j\,|\,z^n)}{P^1(j)}\right]+\lambda=0; \tag{4.3}$$

now multiply by $P^1(j)$ and sum over j to get

$$\sum_j \sum_n \left[P^0(j \mid z^n)\right] + \lambda = 0.$$

Exchanging the order of summation then gives $\lambda = -N$. Substituting this back into eqn (4.3) gives

$$P^1(j) = \frac{1}{N} \sum_n P^0(j \mid z^n).$$

Setting $\partial \hat{Q}/\partial \mu_j^1 = 0$ gives

$$-\frac{1}{2} \sum_n P^0(j \mid z^n) \frac{\partial}{\partial \mu_j^1} \left[(z^n - \mu_j^1)^\top (\Sigma_j^1)^{-1} (z^n - \mu_j^1)\right] = 0;$$

and

$$\frac{\partial}{\partial \mu_j^1} \left[(z^n - \mu_j^1)^\top (\Sigma_j^1)^{-1} (z^n - \mu_j^1)\right] = -2(z^n - \mu_j^1)^\top (\Sigma_j^1)^{-1},$$

so substituting this, taking the transpose and rearranging gives

$$\mu_j^1 = \frac{\sum_n P^0(j \mid z^n) z^n}{\sum_n P^0(j \mid z^n)}.$$

Finding $\partial \hat{Q}/\partial \Sigma_j^1$ is unpleasant; therefore solve instead for $R_j = (\Sigma_j^1)^{-1}$. Differentiating eqn (4.2) with respect to R_j gives

$$\frac{\partial \hat{Q}}{\partial R_j} = \sum_n P^0(j \mid z^n) \left[\frac{1}{2} \frac{\partial}{\partial R_j} \log |R_j| - \frac{1}{2} \frac{\partial}{\partial R_j} \left[(z^n - \mu_j^1)^\top R_j (z^n - \mu_j^1)\right]\right]$$

and, using

$$\frac{\partial}{\partial R_j} \log |R_j| = R^{-\top}; \quad \frac{\partial}{\partial R_j} \left[(z^n - \mu_j^1)^\top R_j (z^n - \mu_j^1)\right] = (z^n - \mu_j^1)(z^n - \mu_j^1)^\top$$

and the fact that R_j is symmetrical,

$$\Sigma_j^1 = \frac{\sum_n P^0(j \mid z^n)(z^n - \mu_j^1)(z^n - \mu_j^1)^\top}{\sum_n P^0(j \mid z^n)}$$

results.

Experiments The algorithm, for the case of a Gaussian mixture in two dimensions, was implemented, and a Gaussian mixture of 5 components was fitted to a set of 1000 data points drawn from a distribution uniform over an annulus. The initial value for φ was chosen by picking the first five data points as the means, setting the variance of each component to be

isotropic with variance equal to the distance to the nearest mean of the other components, and setting the mixture distribution to be uniform over the five components.

The results are shown in figure 4.1; 96 iterations of the algorithm were performed, and a sample of frames shown in the figure. Increasing iteration count goes in raster order. It can be seen that the algorithm seems to converge to what would intuitively be expected as a best fit to the data.

4.3 Learning Dynamical Systems from Data Sequences with Missing Points

In this section, the Expectation-Maximisation algorithm is applied to the problem of learning a dynamical model from a training sequence where some of the data points are missing.

4.3.1 First-order Scalar System

The first, simplest, case to be considered is the case where the data are scalar, and the dynamical system is first-order, i.e., of the form

$$x_t - \bar{x} = a(x_{t-1} - \bar{x}) + bw_t,$$

where $\bar{x}$ is the mean of the system, a describes the deterministic part of the model, w_t are independent noise terms, each of which is unit normal, and b describes the coupling of this noise into the system.

Maximum Likelihood Estimation of Parameters First recall from section 3.2 the procedure for producing the Maximum Likelihood estimate of the system parameters when there are no missing data.

Assume that the mean of the data set is 0. Then the dynamical model is

$$x_t = ax_{t-1} + bw_t, \tag{4.4}$$

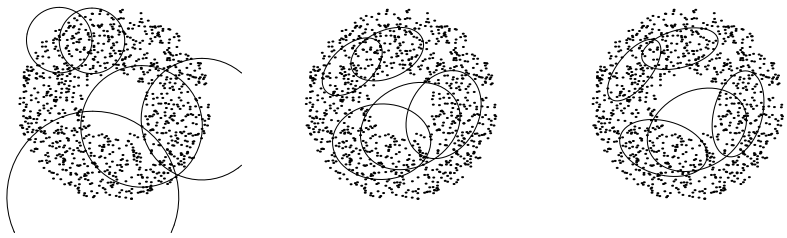
and the parameter vector φ is then the pair (a, b).

The important results from section 3.2 are the actual maximum-likelihood estimates:

$$a = \frac{r_{01}}{r_{11}}; \qquad b^2 = \frac{1}{T-1}(r_{00} - 2ar_{01} + a^2 r_{11}), \tag{4.5}$$

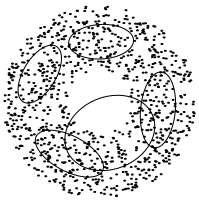
and the expression for the log-likelihood:

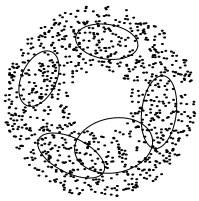
$$L = -(T-1)\log b - \frac{1}{2b^2}(r_{00} - 2ar_{01} + a^2 r_{11}), \tag{4.6}$$

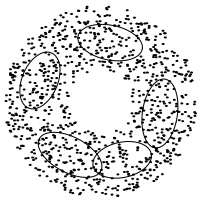

Iteration 1

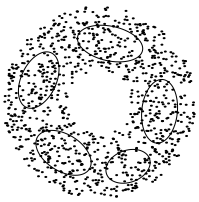
Iteration 2

Iteration 4


Iteration 8


Iteration 13

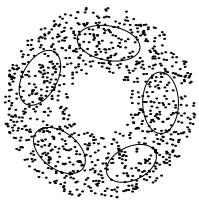

Iteration 20


Iteration 29


Iteration 39


Iteration 51


Iteration 64


Iteration 79


Iteration 96

Figure 4.1: **The Expectation Maximisation algorithm calculates a best-fit Gaussian mixture to a uniform distribution on an annulus.** *The 1-σ uncertainty ellipses for the components are shown; as further iterations are performed, the mixture converges towards what is intuitively expected as a best fit (there may be no unique best fit owing to the rotational symmetry of the initial distribution) to the supplied data set.*

Missing Data Point Consider a data sequence with one of the points, say x^*_τ, missing, and the problem of estimating the system parameters from these flawed data. The observed data are $x^*_1, x^*_2, \ldots, x^*_{\tau-1}, x^*_{\tau+1}, \ldots, x^*_T$, and the hidden datum is x^*_τ. From eqn (4.6) above, the log-likelihood L is a linear function of the moments r_{ij}, so $\mathcal{E}[L \mid \varphi]$ may be computed from the expectation of these moments (over a distribution for the missing datum).

Note that the only terms in r_{01} which involve x^*_τ are $x^*_{\tau-1}x^*_\tau$ and $x^*_\tau x^*_{\tau+1}$, which are both linear in x^*_τ, and so

$$\mathcal{E}[r_{01}(x^*_\tau)] = r_{01}(\mathcal{E}[x^*_\tau]) \tag{4.7}$$

(treating r_{01} as a function of x^*_τ in the intuitive way). However, r_{00} and r_{11} both include the term $(x^*_\tau)^2$ (assuming $\tau \neq N$ — this makes sense since if the last datum is missing, it is ignored completely, and the sequence $x^*_1, x^*_2, \ldots, x^*_{T-1}$ considered instead; similarly, assume x^*_1 is not missing), and

$$\mathcal{E}[(x^*_\tau)^2] = \mathcal{E}[x^*_\tau]^2 + \mathrm{Var}[x^*_\tau]. \tag{4.8}$$

Therefore,

$$\mathcal{E}[r_{11}(x^*_\tau)] = r_{11}(\mathcal{E}[x^*_\tau]) + \mathrm{Var}[x^*_\tau],$$

and finding the values of $\mathcal{E}[x^*_\tau]$ and $\mathrm{Var}[x^*_\tau]$ will therefore give the expected values of the moments r_{ij} and hence $\mathcal{E}[L \mid \varphi]$.

Distribution of the Missing Data It will be shown that

$$\mathcal{E}[x^*_\tau \mid a, b, x^*_1, \ldots, x^*_{\tau-1}, x^*_{\tau+1}, \ldots, x^*_T] = \frac{a}{1+a^2}\left(x^*_{\tau-1} + x^*_{\tau+1}\right); \tag{4.9}$$

$$\mathrm{Var}[x^*_\tau \mid a, b, x^*_1, \ldots, x^*_{\tau-1}, x^*_{\tau+1}, \ldots, x^*_T] = \frac{b^2}{1+a^2} \tag{4.10}$$

by finding the distribution for x^*_τ given a previous estimate (a, b) of the parameters and the observed data. (Henceforth, the conditioning on $a, b, x^*_1, \ldots, x^*_{\tau-1}, x^*_{\tau+1}, \ldots, x^*_N$ will be omitted where this omission does not introduce ambiguity.) The missing value x^*_τ occurs in two time-steps of the dynamical system, eqn (4.4):

$$\begin{aligned} x^*_\tau &= a x^*_{\tau-1} + b w_\tau; \\ x^*_{\tau+1} &= a x^*_\tau + b w_{\tau+1} \end{aligned}$$

so, eliminating the unknown x^*_τ,

$$x^*_{\tau+1} - a^2 x^*_{\tau-1} = ab w_\tau + b w_{\tau+1},$$

and this condition (on the random variable $abw_\tau + bw_{\tau+1}$) must be incorporated. In the notation of section 4.2 (z is the observed data and x the hidden data), then,

$$p(x \mid z, \varphi) = p_{bw_\tau}(x^*_\tau - ax^*_{\tau-1} \mid abw_\tau + bw_{\tau+1} = x^*_{\tau+1} - a^2 x^*_{\tau-1}), \tag{4.11}$$

where the condition is, essentially, that both the values of the unknown random variables w_τ and $w_{\tau-1}$ produce the same value for x^*_τ. To calculate this probability density, put

$$A = bw_\tau; \tag{4.12}$$

$$B = bw_{\tau+1}; \tag{4.13}$$

$$C = aA; \tag{4.14}$$

$$D = B + C. \tag{4.15}$$

Then, writing

$$u = x^*_{\tau+1} - a^2 x^*_{\tau-1}, \tag{4.16}$$

the required probability is

$$p(y \mid x, \varphi) = p_A(z \mid D = u)|_{z = x^*_\tau - ax^*_{\tau-1}}. \tag{4.17}$$

Now, by Bayes' rule,

$$p_A(z \mid D = u)p_D(u) = p_D(u \mid A = z)p_A(z), \tag{4.18}$$

and

$$p_A(x) = p_B(x) = \frac{1}{b\sqrt{2\pi}} \exp\left(-\frac{x^2}{2b^2}\right); \tag{4.19}$$

$$p_C(x) = \frac{1}{|a|b\sqrt{2\pi}} \exp\left(-\frac{x^2}{2a^2b^2}\right); \tag{4.20}$$

$$p_D(x \mid A = z) = p_B(x - az). \tag{4.21}$$

Also, since $D = B + C$, and both B and C are Gaussian.

$$p_D(u) = \frac{1}{(1+a^2)^{1/2} b\sqrt{2\pi}} \exp\left(-\frac{u^2}{2(1+a^2)b^2}\right). \tag{4.22}$$

Therefore, the distribution of the missing datum is as follows.

$$p_A(z \mid D = u) = \frac{(1+a^2)^{1/2}}{b\sqrt{2\pi}} \exp\left[-\frac{(1+a^2)}{2b^2}\left(z - \frac{a}{1+a^2}u\right)^2\right]. \tag{4.23}$$

Therefore, the random variable A conditional on $D = u$, denoted $(A \mid D = u)$, is given by

$$(A \mid D = u) \sim N\left(\frac{a}{1+a^2}, \frac{b^2}{1+a^2}\right),$$

from which can be deduced the values for $\mathcal{E}[x^*_\tau]$ and $\mathrm{Var}[x^*_\tau]$ given at the start of the section.

This expected value can then be used to find the expected values of the moments of the (complete) data, using eqns (4.7) and (4.8).

Extension to Multiple Missing Data Note that the above derivation applies, since the model is first-order only, to any number of missing data, provided that no two missing data points occur consecutively in the sequence. Suppose, therefore, that M data points $x^*_{\tau_1}, x^*_{\tau_2}, \ldots, x^*_{\tau_M}$ are missing (with $\tau_{i+1} - \tau_i \neq 1$). Note also that the variance of a missing data point depends only on the previous estimate for the parameters; therefore, the expected values of the moments are as follows.

$$\mathcal{E}[r_{00}] = r_{00}(\mathcal{E}[x^*_{\tau_1}], \mathcal{E}[x^*_{\tau_2}], \ldots, \mathcal{E}[x^*_{\tau_M}]) + M\sigma^2; \tag{4.24}$$

$$\mathcal{E}[r_{01}] = r_{01}(\mathcal{E}[x^*_{\tau_1}], \mathcal{E}[x^*_{\tau_2}], \ldots, \mathcal{E}[x^*_{\tau_M}]); \tag{4.25}$$

$$\mathcal{E}[r_{11}] = r_{11}(\mathcal{E}[x^*_{\tau_1}], \mathcal{E}[x^*_{\tau_2}], \ldots, \mathcal{E}[x^*_{\tau_M}]) + M\sigma^2 \tag{4.26}$$

(where $\sigma^2 = b^2/(1+a^2)$ is the variance of any one of the missing data points — they are all the same).

Update Equations for EM The update equations for one step of the algorithm therefore consist of finding the expected values of the mean and variance of the missing data points (using equations (4.9) and (4.10)), finding from these the expected values of the moments r_{ij} (using equations (4.24)–(4.26)), and then calculating a new estimate (a', b') of the parameters:

$$a' = \frac{\mathcal{E}[r_{01}]}{\mathcal{E}[r_{11}]}; \qquad (b')^2 = \frac{1}{N-1}(\mathcal{E}[r_{00}] - 2a'\mathcal{E}[r_{01}] + (a')^2\mathcal{E}[r_{11}]). \tag{4.27}$$

The EM algorithm then proceeds as usual by setting $a = a'$ and $b = b'$ and iterating this procedure until convergence of the parameters a and b.

Convergence Any numerical, iterative optimisation procedure, such as the EM algorithm and its application to the problems considered within this thesis, requires one to decide when to stop the procedure. Ideally, some automatic decision process is used, typically stopping the algorithm either when an iteration produces a change in the parameters of interest less

than some threshold value, or when the change in the value to be optimised (in this case, it is the likelihood which is to be maximised) is below some other threshold value. These tolerances are often related to the floating-point precision of the architecture in use (Press et al., 1988).

In this case, the problem is exacerbated by the fact that the convergence of the EM algorithm is often slow. (This is known from other work (Wu, 1983), and will also be seen in experiments throughout the remainder of this thesis.) Furthermore, several of the later applications (for example those in chapter 7) have the property that each individual iteration is very computationally expensive, sometimes taking several hours of CPU time on the hardware used.

Therefore, in the experiments described in this thesis, the convergence of the estimates was judged by inspection. In some cases, a decision was made to terminate the algorithm even though it was apparent that the system was close to convergence without having reached a final converged value. Even though the computational budget for an off-line process (such as the learning problems considered) is significantly larger than for an on-line process, it is still desirable to keep the burden within reasonable bounds, and so it may not be feasible to run to true convergence.

An area for future work could be an investigation of how one could specify a criterion for automatically stopping the process. This could perhaps be done using a tolerance on the parameter estimates, expressed in terms of natural units, such as the time scales of the modes of the system. Comparing the performance of models learnt in this fashion (using computational resources which could not realistically be used within a practical system) with that of models learnt as in this thesis — using manual (and sometimes looser) termination conditions — would also be of interest.

Experiments with Synthetic Data The EM algorithm in this case was implemented, creating a data sequence by simulating the model of eqn (4.4). A number of these data points were then marked as 'missing' and treated accordingly in the algorithm. Their simulated values were therefore not used.

To initialise the algorithm, starting estimates for the parameters a and b are required. These were obtained by 'faking' the missing data as follows: each missing point was set to the mean of the one before it and the one after it. (This is always possible since it was ensured that no two consecutive points are missing.) The parameters a and b were then estimated from this (partially faked) data set as in eqn (4.5). The EM algorithm was then

run until convergence of the estimates of a and b was satisfactory.

Data sets for various true values of a and b were treated in this way; the results are summarised in table 4.2. Also shown in the table are the initial estimates of the dynamics, in other words, the results of learning a dynamical model from the 'faked' data described above. Two examples of the convergence behaviour of the algorithm are given in figure 4.2.

True a	**Estimated** a **(EM)**	**Estimated** a **('faked')**	**True** b	**Estimated** b **(EM)**	**Estimated** b **('faked')**
0.5	0.5026	0.5218	0.1	0.1006	0.0829
0.5	0.5045	0.5488	0.2	0.2003	0.1867
0.8	0.7965	0.8157	0.1	0.1013	0.0932
0.8	0.8082	0.8185	0.2	0.2003	0.1858

Table 4.2: **Using EM to learn a dynamical system from a sequence with missing points produces good estimates, whereas using 'faked' data gives inaccurate dynamics.** *Average of five runs of the EM algorithm with synthetic data sequences of length 1024, of which 128 states were treated as missing, generated using various dynamical models. 16 steps of the algorithm were performed (enough to give convergence; see also figure 4.2) — the final estimate is shown. The systems learnt using 'faked' data consistently have a too large and b too small.*

The estimates produced by the algorithm are within 2% of their true values; also, the convergence of the estimates is rapid. Note also that the initial estimates (i.e., the system learnt using 'faked' data) have noise parameters which are too low — this is a result of forming the initial estimate by replacing each missing datum with the mean of its neighbours, a procedure which will tend to reduce the apparent noise.

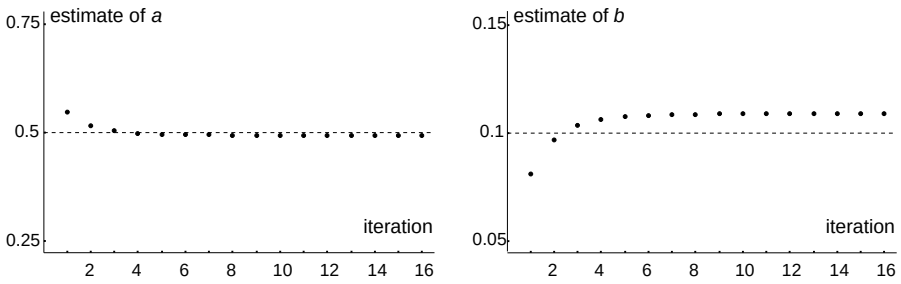
The technique of EM should therefore be a useful tool to apply to this problem. The extension of these ideas to vector systems is now considered.

4.3.2 First-order Vector System

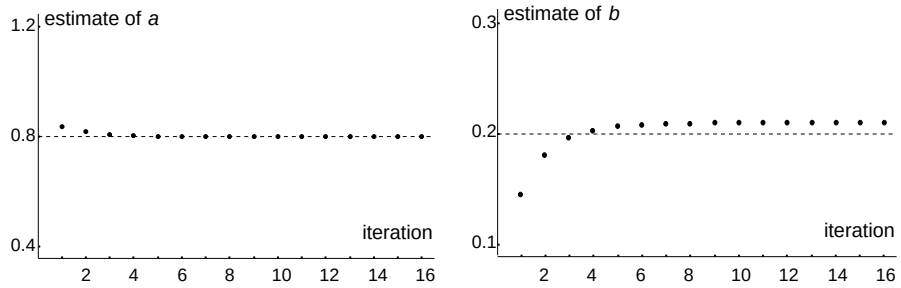
The next case considered is that where the dynamical system is first-order, but the data are vector-valued. In this case the system equation is of the form

$$\mathbf{x}_t - \overline{\mathbf{x}} = A(\mathbf{x}_{t-1} - \overline{\mathbf{x}}) + Bw_t, \tag{4.28}$$

where $\overline{\mathbf{x}}$ is the mean of the system, A is the matrix giving the dynamical component of the system, w_t are independent vectors of independent unit normal random variables, and B is the matrix coupling the random variables into the system. The development in this section will follow that in section 4.3.1.



Synthetic data created with dynamical model $a = 0.5$ and $b = 0.1$.



Synthetic data created with dynamical model $a = 0.8$ and $b = 0.2$.

Figure 4.2: **The EM algorithm converges rapidly to the Maximum Likelihood estimate of the system parameters.** *Convergence behaviour of the EM algorithm for a synthesised data sequence of 1024 points of which 448 were treated as missing. The algorithm converges to the ML value of the parameters, producing an estimate which can be considered a final converged value within 6 iterations. The true value of the parameters, used to synthesise the data, are shown by the dotted lines.*

Maximum-likelihood Estimation of Parameters For a sequence $\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_T^*$ of T vector-valued data points, the maximum-likelihood estimate of the system parameters A and B for a zero-mean system is given by the following results.

$$A = R_{10}R_{11}^{-1}; \tag{4.29}$$

$$C = \frac{1}{T-1}\left(R_{00} - AR_{10} - R_{01}A^\top + AR_{11}A^\top\right), \tag{4.30}$$

where $C = BB^\top$ and the moments R_{ij} are defined as

$$R_{ij} = \sum_{t=2}^{T} \mathbf{x}_{t-i}^*(\mathbf{x}_{t-j}^*)^\top. \tag{4.31}$$

Once more, there is no unique solution for B, since any orthogonal transformation of the noise vector will be identically distributed. A standardised matrix square root operation (Barnett, 1990) can be used to extract B from C.

Missing Data Point As for the scalar case, suppose a data sequence is given from which one of the points is missing. Exactly as in the scalar case, finding the expectation of the moments R_{ij} will allow $\mathcal{E}[L]$ to be computed, and to find these moments it is sufficient to find the mean and variance of the missing data point $\mathbf{x}_\tau^*$, for then, as before,

$$\begin{aligned}
\mathcal{E}[R_{00}] &= R_{00}(\mathcal{E}[\mathbf{x}_\tau^*]) + \mathrm{Var}[\mathbf{x}_\tau^*];\\
\mathcal{E}[R_{01}] &= R_{01}(\mathcal{E}[\mathbf{x}_\tau^*]);\\
\mathcal{E}[R_{10}] &= R_{10}(\mathcal{E}[\mathbf{x}_\tau^*]);\\
\mathcal{E}[R_{11}] &= R_{11}(\mathcal{E}[\mathbf{x}_\tau^*]) + \mathrm{Var}[\mathbf{x}_\tau^*].
\end{aligned}$$

Distribution of the Missing Data It will be shown, in a similar fashion to the scalar case, that

$$\mathcal{E}[\mathbf{x}_\tau^*] = A\mathbf{x}_{\tau-1}^* + H(\mathbf{x}_{\tau+1}^* - A^2\mathbf{x}_{\tau-1}^*); \tag{4.32}$$

$$\mathrm{Var}[\mathbf{x}_\tau^*] = (B^{-\top}B^{-1} + A^\top B^{-\top}B^{-1}A)^{-1}, \tag{4.33}$$

where

$$H = (A + BB^\top A^{-\top}B^{-\top}B^{-1})^{-1}. \tag{4.34}$$

To do this, consider the random variables

$$\alpha = Bw_\tau; \tag{4.35}$$

$$\beta = Bw_{\tau+1}; \tag{4.36}$$

$$\gamma = A\alpha; \tag{4.37}$$

$$\delta = \beta + \gamma. \tag{4.38}$$

As before, note that the distribution for the missing data point is given by

$$p(y \,|\, x, \varphi) = p_\alpha(\mathbf{x}^*_\tau - A\mathbf{x}^*_{\tau-1} \,|\, \delta = u), \tag{4.39}$$

where

$$u = \mathbf{x}^*_{\tau+1} - A^2\mathbf{x}^*_{\tau-1}. \tag{4.40}$$

Again, using Bayes' rules, expressions for $p_\delta(u \,|\, \alpha = z)$, $p_\alpha(z)$, and $p_\delta(u)$ are sought, to be used (with $z = \mathbf{x}^*_\tau - A\mathbf{x}^*_{\tau-1}$) to find the required quantity $p_\alpha(z \,|\, \delta = u)$. Most straightforwardly, $p_\delta(u \,|\, \alpha = z) = p_\beta(u - Az)$, and the distributions of α and β are known (they are the same):

$$p_\alpha(\mathbf{x}) = p_\beta(\mathbf{x}) = \frac{1}{\rho|B|} \exp\left[-\frac{1}{2}\left(\mathbf{x}^\top B^{-\top} B^{-1}\mathbf{x}\right)\right],$$

where $\rho = (2\pi)^{d/2}$ and d is the number of components in each vector $\mathbf{x}_i$. The only problem remaining, then, is to find $p_\delta(u)$. Since $\delta = \beta + \gamma$, and β and γ are zero-mean Gaussians,

$$p_\delta(u) = \frac{1}{\rho|D|} \exp\left[-\frac{1}{2}u^\top D^{-\top} D^{-1}u\right], \tag{4.41}$$

where

$$DD^\top = (AB)(AB)^\top + BB^\top. \tag{4.42}$$

Now the required quantity $p_\alpha(z \,|\, \delta = u)$ may be computed:

$$p_\alpha(z \,|\, \delta = u) = \frac{p_\delta(u \,|\, \alpha = z)p_\alpha(z)}{p_\delta(u)} \tag{4.43}$$

$$= \frac{|D|}{\rho|B|^2} \exp\left[-\frac{1}{2}\Big((u - Az)^\top B^{-\top} B^{-1}(u - Az) + z^\top B^{-\top} B^{-1} - u^\top D^{-\top} D^{-1}u\Big)\right] \tag{4.44}$$

and, after considerable rearranging, this can be written as

$$p_\alpha(z \,|\, \delta = u) = \frac{|D|}{\rho|B|^2} \exp\left[-\frac{1}{2}(z - Hu)^\top K^{-\top} K^{-1}(z - Hu)\right],$$

where

$$K^{-\top}K^{-1} = A^{\top}B^{-\top}B^{-1}A + B^{-\top}B^{-1}, \tag{4.45}$$

and H is as defined in eqn (4.34). The mean and variance of the missing data point can now be deduced as claimed at the start of the section.

Extension to Multiple Missing Data As in the scalar case, the above derivation applies to the case where there are multiple missing data points, provided no two are consecutive in the sequence. In this case, if M data points $\mathbf{x}^*_{\tau_1}, \mathbf{x}^*_{\tau_2}, \ldots, \mathbf{x}^*_{\tau_M}$ are missing, the expected value of the moments can be calculated in an analogous fashion to that in equations (4.24)–(4.26):

$$R^1_{00} = R_{00}(\mathcal{E}[\mathbf{x}^*_{\tau_1}], \mathcal{E}[\mathbf{x}^*_{\tau_2}], \ldots, \mathcal{E}[\mathbf{x}^*_{\tau_M}]) + M\Sigma; \tag{4.46}$$

$$R^1_{01} = R_{01}(\mathcal{E}[\mathbf{x}^*_{\tau_1}], \mathcal{E}[\mathbf{x}^*_{\tau_2}], \ldots, \mathcal{E}[\mathbf{x}^*_{\tau_M}]); \tag{4.47}$$

$$R^1_{11} = R_{11}(\mathcal{E}[\mathbf{x}^*_{\tau_1}], \mathcal{E}[\mathbf{x}^*_{\tau_2}], \ldots, \mathcal{E}[\mathbf{x}^*_{\tau_M}]) + M\Sigma, \tag{4.48}$$

where Σ is the variance matrix for each of the missing data (again, they are all identical):

$$\Sigma = (A^{\top}B^{-\top}B^{-1}A + B^{-\top}B^{-1})^{-1}. \tag{4.49}$$

Update Equations for EM One step of the EM algorithm in this case consists of finding the means and variances of the missing data points using equations (4.32) and (4.33), then finding the expected values of the moments R_{ij} using equations (4.46)–(4.48), and finally producing a new estimate for the parameters A and B by means of equations (4.29) and (4.30). This procedure is repeated until convergence of A and B.

4.3.3 Conclusions

Expectation-Maximisation has been shown to be a suitable technique to apply to the problem of system identification from a data sequence where some of the points are missing. This might seem to be the situation in the tracker, where a training sequence with missing fields is not an uncommon situation. However, section 4.4 will develop a more appropriate solution to the problem of system identification from a *measurement sequence*, rather than from a sequence of what are, in fact, filtered estimates of the state. Therefore, although there are situations where the results of this section are directly applicable, the training of the tracker will be handled by the theory to be developed below, which will generalise and extend the results described above.

4.4 Learning Dynamical Systems from a Measurement Sequence

The problem of learning a dynamical model given a time-sequence of measurements of the system is now addressed. Compared to the approach in section 3.3 (of treating the filtered estimates as true state values), this is a more realistic formulation of the underlying problem of learning a dynamical system for the tracker and, more generally, the problem of system identification for a Kalman filter. In terms of the 'observed' and 'hidden' variables of the EM algorithm, the measurements are the observed data, and the actual states at each time-step are the hidden data.

Ljung (1987) mentions one approach to the problem of system identification from a measurement sequence. It is given in appendix A.1. However, while the method is perfectly general, it is also computationally very expensive in this case. It involves the inversion of large matrices (with sizes of the order of the length of the training sequence multiplied by the state-vector size, typically a few thousand), and so a more specific solution is developed. The theory presented here was developed independently, but should be compared with treatments given by Shumway and Stoffer (1982), and by Ghahramani (1996). Their methods are described in appendix A.2.

The approach used will be as follows. Each step in the EM algorithm involves maximising $\mathcal{E}[L]$ for the log-likelihood L. Therefore, the problem of maximising L is first solved, and then that when $\mathcal{E}[L]$ is instead the subject of the maximisation.

4.4.1 Maximum Likelihood Estimation of Parameters

Recall from section 4.3.2 that, for a model of the form

$$x_t = Ax_{t-1} + Bw_t$$

and a given sequence of states $x_1^*, x_2^*, \ldots, x_T^*$, the maximum-likelihood estimate of the system parameters A and B is given by

$$A = R_{01}R_{11}^{-1}; \quad BB^\top = \frac{1}{T-1}\left(R_{00} - AR_{10} - R_{01}A^\top + AR_{11}A^\top\right), \tag{4.50}$$

where

$$R_{ij} = \sum_{t=2}^{T} x_{t-i}^*(x_{t-j}^*)^\top$$

are auto-correlation coefficients. (This model has a process mean of zero — the extension of the learning algorithm to allow estimation of the process mean is covered in section 4.5.2.)

Recall also that L is a linear function of the moments R_{ij} (see eqn (4.6) on page 58), so finding the expected values of these moments will provide the expected value of L, and hence allow its maximisation. $\mathcal{E}[R_{ij}]$ is therefore sought, given a previous estimate for the dynamical model and the measurements.

4.4.2 The Augmented-state Smoothing Filter

The procedure adopted for finding the expected values of the auto-correlation coefficients R_{ij} is an adaptation of 'fixed-interval smoothing' — the problem of finding the distribution of states x_t given a set $\{z_1, z_2, \ldots, z_T\}$ of measurements, where $t \leq T$. Consider an augmented state variable X_t defined as follows.

$$X_t = \begin{pmatrix} x_t \\ x_{t-1} \end{pmatrix},$$

and consider $\sum_t X_t X_t^\top$:

$$\begin{aligned} R &= \sum_{t=2}^{T} X_t X_t^\top \\ &= \sum_{t=2}^{T} \begin{pmatrix} x_t \\ x_{t-1} \end{pmatrix} \begin{pmatrix} x_t^\top & x_{t-1}^\top \end{pmatrix} \\ &= \sum_{t=2}^{T} \begin{pmatrix} x_t x_t^\top & x_t x_{t-1}^\top \\ x_{t-1} x_t^\top & x_{t-1} x_{t-1}^\top \end{pmatrix} \\ &= \begin{pmatrix} R_{00} & R_{01} \\ R_{10} & R_{11} \end{pmatrix} \end{aligned}$$

Finding $\mathcal{E}[R]$ therefore provides the values of all the individual $\mathcal{E}[R_{ij}]$, needed to find the expected log-likelihood $\mathcal{E}[L]$, and

$$\begin{aligned} \mathcal{E}[R] &= \sum_t \mathcal{E}[X_t X_t^\top] \\ &= \sum_t \Big(\mathcal{E}[X_t]\mathcal{E}[X_t]^\top + \mathrm{Var}[X_t] \Big) \\ &= \sum_t \mathcal{E}[X_t]\mathcal{E}[X_t]^\top + \sum_t \mathrm{Var}[X_t]. \end{aligned}$$

Thus finding the expected values and variances of the X_t completes the formulation of the EM algorithm in this case.

Dynamic and Measurement Models Consider, therefore, the dynamic and measurement models for the new state vectors X_t. The original dynamical model

$$x_t = A_0 x_{t-1} + B_0 w_t$$

becomes

$$X_t = \begin{pmatrix} A_0 & 0 \\ I & 0 \end{pmatrix} X_{t-1} + \begin{pmatrix} B_0 \\ 0 \end{pmatrix} w_t. \tag{4.51}$$

(The alternative choice,

$$X_t = \begin{pmatrix} A_0 & 0 \\ 0 & A_0 \end{pmatrix} X_{t-1} + \begin{pmatrix} B_0 & 0 \\ 0 & B_0 \end{pmatrix} \begin{pmatrix} w_t \\ w_{t-1} \end{pmatrix},$$

gives process noise which is correlated from time-step to time-step. This is not covered by the standard Kalman filter, but can be accounted for. The choice given by (4.51) is simpler, so it is used here.)

The dynamic model for the augmented state X is therefore given by

$$A = \begin{pmatrix} A_0 & 0 \\ I & 0 \end{pmatrix}; \quad B = \begin{pmatrix} B_0 \\ 0 \end{pmatrix}; \quad W_t = w_t$$

for $t = 2, \ldots, T$.

The measurement model for the original state x is given by

$$z_t = H_0 x_t + v_t,$$

where v_t are zero-mean uncorrelated Gaussian noise terms, with (known) variance Σ_0; this becomes, in terms of the augmented state X,

$$Z_t = H X_t + V_t,$$

where

$$Z_t = z_t; \quad H = \begin{pmatrix} H_0 & 0 \end{pmatrix}; \quad V_t = v_t,$$

with the noise terms uncorrelated and with variance $\Sigma = \Sigma_0$.

Smoothing Algorithm Assuming that the *a priori* distribution for the initial state of the system is Gaussian, all distributions remain Gaussian as time proceeds, and the solution for the mean and variances of the system states $X_2, X_3, \ldots, X_T$ given the complete set $\{z_1, z_2, \ldots, z_T\}$ of measurements is developed by, for example, Sage and Melsa (1971) and Jazwinski (1970). Let

$$\begin{aligned} \hat{X}_{t_0}^{t_1} &= \mathcal{E}[X_{t_0} \mid Z_1, \ldots, Z_{t_1}]; \\ P_{t_0}^{t_1} &= \mathrm{Var}[X_{t_0} \mid Z_1, \ldots, Z_{t_1}]. \end{aligned}$$

The solution to the smoothing problem is then given by the recurrence relations

$$\begin{aligned}\hat{X}_t^T &= \hat{X}_t^t + F_t(\hat{X}_{t+1}^t - A\hat{X}_t^t);\\ P_t^T &= P_t^t + F_t(P_{t+1}^t - P_{t+1}^t)F_t^\top,\end{aligned}$$

where

$$F_t = P_t^t A^\top (P_{t+1}^t)^{-1}. \tag{4.52}$$

Here the terms $\hat{X}_t^t$, P_{t+1}^t and P_t^t are known from the solution of the filtering problem; the recurrence relations are solved using $\hat{X}_T^T$ and P_T^T (also known from the filtered estimates) as boundary conditions.

The smoothing algorithm here relies on the availability of the solution to the filtering problem — the estimates $\hat{X}_t^t$ and the variances P_t^t and P_{t+1}^t; these are obtained by running the Kalman filter on the measurement sequence, using a suitable value for the *a priori* distribution of the first state in the sequence.

Initialising the Kalman Filter The question therefore arises as to how $\hat{X}_2^0 = \mathcal{E}[X_2]$ (the *a priori* mean) and $P_2^0 = \mathrm{Var}[X_2]$ (the *a priori* variance) can be specified to provide a starting state for the Kalman filter, the output of which is required for the smoothing algorithm. In doing so, the measurement z_1, which is not otherwise used, may be used.

When this procedure is considered within the framework of the EM algorithm, it is seen that the X_2^T and P_2^T of one iteration of the algorithm may be treated as the X_2^0 and P_2^0 of the next; in this way, the influence of the choice of the initial X_2^0 and P_2^0 will be decreased further.

There are two approaches often used in Kalman filtering. One is to start the system from a known state. In that case, $\hat{X}_2^0$ is this known state, and $P_2^0 = 0$ (since the state is known exactly). The alternative is to take the *a priori* variance of the system, which is the solution to the equation

$$P = APA^\top + BB^\top$$

(obtained by taking the variance of the dynamical equation). Which is used will depend on the situation. In fact, setting $P_2^0 = 0$ causes problems with smoothing — the matrix P_{t+1}^t to be inverted will be singular. Some small value of P_2^0 will be substituted in practice.

(1) Initialise: Choose starting distribution of the state X_2 (mean and variance), and an initial estimate for the dynamics A and B (what precisely this means will depend on the situation);

(2) Filter: Run the Kalman filter on the measurement set to produce the filtered estimates $\hat{X}_t^t$;

(3) Smooth: Run the smoothing algorithm on the measurement set (using the filtered estimates) to produce the smoothed estimates $\hat{X}_t^T$;

(4) Finish 'E' Step: Using the results of the smoothing, find the expected values $\mathcal{E}[R_{ij}]$ of the moments;

(5) 'M' Step: From the expected values of the moments, estimate the system parameters A and B;

(6) Iterate: Using the system derived in the previous step, and using the smoothed estimate of X_2 as initial condition, go back to step (2).

Figure 4.3: **The EM algorithm for the problem of estimating system parameters from a sequence of measurements.**

4.4.3 Summary of Algorithm

The steps of the EM algorithm for this case are therefore as shown in figure 4.3.

The algorithm should be run, as is usual for EM applications, until convergence of the parameters of interest, namely A and C.

(The approaches of Shumway and Stoffer (1982), and of Ghahramani (1996), described in appendix A.2, use a different method to obtain the required variances P_t^T.)

4.4.4 Experiments with Synthetic Data

The algorithm shown in figure 4.3 was implemented, and test runs performed for a scalar system (for ease of presentation of results); the experiments are the original work of the author. A sequence of system states was generated from the model $x_t = ax_{t-1} + bw_t$, and then a measurement was simulated of each state using the measurement model $z_t = hx_t + v_t$. These measurements were then processed to produce an estimate of the parameters a and b.

For the initialisation, the first method described above, that of starting the system in a known state, was used. The known state was that used as x_1 in producing the synthetic

sequence.

The performance of another learning algorithm, very similar to filtered learning, is also investigated briefly. Once the estimates have been smoothed, better estimates for the state at each time-step are available. Treating these estimates as the true data should produce a better estimate of the dynamical model than treating the unsmoothed estimates as true states.

The performance of the filtered learning algorithm will, it is expected, depend on the relative magnitudes of the process noise (which drives the system) and the measurement noise. In a situation where the measurement noise is low compared with the process noise, the filtered estimates will be good, and therefore the filtered learning algorithm will produce a reasonable estimate. The relative importance of the two noise terms can be measured by the ratio of their variances, σ^2/h^2b^2. The algorithms were compared for four different values of this ratio. Data sequences of length 1024 were simulated for the system given by $a = 0.8$ and $b = 0.2$ and measurements synthesised using $h = 2$ for the four values 0.04, 0.2, 0.4 and 0.8 of σ, making the ratio of the variances 0.01, 0.25, 1.0 and 4.0 respectively. The estimates produced by the three algorithms are shown in table 4.3, where the improved performance of EM-based learning can be seen. The difference between filtered and smoothed learning is small, and smoothed learning is not considered further.

$\frac{\sigma^2}{h^2b^2}$	**Estimate of a** ($a = 0.8$)			**Estimate of b** ($b = 0.2$)		
	Filtered	**Smoothed**	**EM**	**Filtered**	**Smoothed**	**EM**
0.01	0.803	0.807	0.801	0.184	0.185	0.193
0.25	0.780	0.834	0.803	0.100	0.112	0.190
1.00	0.720	0.786	0.806	0.055	0.066	0.181
4.00	0.666	0.704	0.798	0.032	0.038	0.197

Table 4.3: **EM-based learning produces better estimates than either filtered or smoothed learning.** *Comparison of the estimates produced by the 'filtered' learning algorithm, the 'smoothed' learning algorithm, and the EM-based learning algorithm. A data sequence of 1024 points was synthesised, and measurements taken. The filtered learning algorithm fares reasonably well for low values of the ratio σ^2/h^2b^2 (the case where the measurements are good), but produces estimates of the process noise b which are far too low when this ratio becomes larger. The EM algorithm performs well in all cases. (Filtered learning estimates are the same as shown in table 4.1.)*

Intuitively, it is expected that another very important factor in the performance of the filtered learning algorithm will be the dynamical model of the filter used to gather the training data. To investigate this effect experimentally, the case where $\sigma^2/h^2b^2 = 4.0$ was

studied for different values of the initial dynamics. These results are presented in table 4.4, and the observed behaviour can be explained as follows.

Qualitatively, if the filter's process noise is much lower than the true value, the filter will 'over-smooth' the measurements, and that therefore filtered learning will produce a model with too little process noise. Conversely, if the filter's process noise is much higher than the true value, too much of the measurement noise will be incorporated into the estimated states, and therefore the resulting learnt model will have too high a process noise.

b_I	**Estimate of** a ($a = 0.8$)		**Estimate of** b ($b = 0.2$)	
	Filtered	**EM**	**Filtered**	**EM**
0.80	0.581	0.827	0.380	0.195
0.40	0.700	0.790	0.222	0.212
0.20	0.731	0.797	0.087	0.195
0.10	0.684	0.821	0.037	0.203
0.05	0.661	0.813	0.023	0.192

Table 4.4: **The initial estimate of the dynamical model has a marked effect on filtered learning, but little effect on EM learning.** *Comparison of the learning methods for different values of the process noise of the dynamical model initially used to filter the measurements (shown in the table as b_I). As expected, initially over-estimating the process noise results in a learnt model with too high process noise, and similarly for an initial under-estimate. A sequence of 1024 time-steps was used, and the EM algorithm ran for 64 iterations, enough for convergence. Here $\sigma^2/h^2b^2 = 4.0$*

An example of the convergence behaviour of the algorithm is shown in figure 4.4, from which it can be seen that, since there is a significant amount of missing data, the convergence is not particularly rapid.

4.5 Second-order Systems

As noted in section 3.2.3, a first-order system is of limited use within a tracking system. The techniques of the previous section are therefore extended to perform learning of a second-order system. The model under consideration is now

$$x_t = A_1 x_{t-1} + A_2 x_{t-2} + B_0 w_t.$$

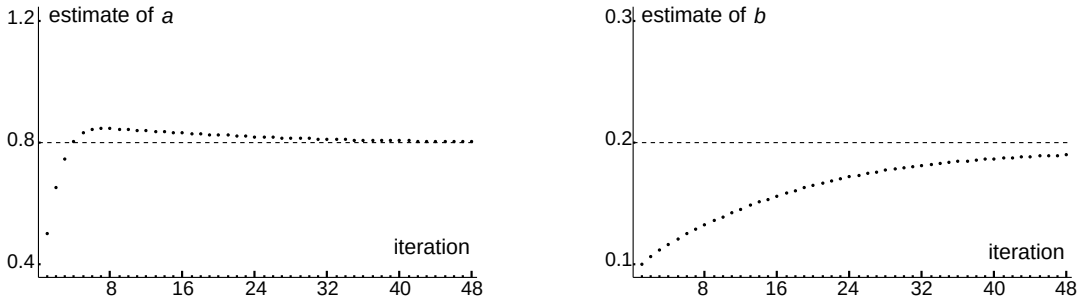


Figure 4.4: **More missing data results in slower convergence.** *The convergence of the EM algorithm is not as rapid as in the missing-data problem of figure 4.2, possibly even requiring slightly more iterations than depicted here (48) for the estimate of b in particular. The data and measurement sequences used were of length 1024; system parameters used for this example were* $a = 0.8$ *and* $b = 0.2$*. (These values are indicated on the graphs by dotted lines.) The measurement factor was* $h = 2.0$ *and the measurement noise had* $\sigma = 0.75$*.*

From section 3.2, the important results are the maximum-likelihood estimate itself, satisfying

$$\begin{aligned} A_1 R_{11} + A_2 R_{21} &= R_{01}; \\ A_1 R_{12} + A_2 R_{22} &= R_{02}; \\ C &= \frac{1}{T-2}\left(R_{00} - A_1 R_{01} - A_2 R_{02}\right), \end{aligned}$$

and the expression for the log-likelihood:

$$L = -\frac{1}{2}\mathrm{tr}(ZC^{-1}) - \frac{1}{2}(T-2)\log|C|,$$

where

$$\begin{aligned} Z = R_{00} &- R_{01}A_1^\top - R_{02}A_2^\top \\ &- A_1 R_{10} + A_1 R_{11} A_1^\top + A_1 R_{12} A_2^\top \\ &- A_2 R_{20} + A_2 R_{21} A_1^\top + A_2 R_{22} A_2^\top, \end{aligned}$$

Again, the log-likelihood is linearly dependent on the moments R_{ij} and therefore their expected values are sought.

4.5.1 Augmented-state Smoothing

Analogously to the first-order case, consider an augmented state variable X_t:

$$X_t = \begin{pmatrix} x_t \\ x_{t-1} \\ x_{t-2} \end{pmatrix}.$$

Then

$$R = \sum_t X_t X_t^\top,$$

as before, consists of all the moments matrices of the original state sequence x_t:

$$R = \begin{pmatrix} R_{00} & R_{01} & R_{02} \\ R_{10} & R_{11} & R_{12} \\ R_{20} & R_{21} & R_{22} \end{pmatrix},$$

and

$$\mathcal{E}[R] = \sum_t \mathcal{E}[X_t]\mathcal{E}[X_t]^\top + \sum_t \mathrm{Var}[X_t].$$

Dynamic and Measurement Models As for the first-order case, dynamic and measurement models for the new state vectors X_n are constructed. The second-order dynamical model,

$$x_t = A_1 x_{t-1} + A_2 x_{t-2} + B_0 w_t.$$

becomes

$$X_t = A X_{t-1} + B w_t,$$

where

$$A = \begin{pmatrix} A_1 & A_2 & 0 \\ I & 0 & 0 \\ 0 & I & 0 \end{pmatrix}; \quad B = \begin{pmatrix} B_0 \\ 0 \\ 0 \end{pmatrix}; \quad W_t = w_t$$

for $t = 3, 4, \ldots, T$.

(This is chosen in preference to

$$X_t = \begin{pmatrix} A_1 & A_2 & 0 \\ 0 & A_1 & A_2 \\ 0 & I & 0 \end{pmatrix} X_{t-1} + \begin{pmatrix} B_0 & 0 \\ 0 & B_0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} w_t \\ w_{t-1} \end{pmatrix}$$

as before.)

The measurement model,

$$z_t = H_0 x_t + v_t,$$

where v_t are zero-mean uncorrelated Gaussian noise terms, with variance Σ_0, similarly becomes, on setting

$$Z_t = z_t; \quad H = \begin{pmatrix} H_0 & 0 & 0 \end{pmatrix}; \quad V_t = v_t,$$

a measurement model of the form

$$Z_t = HX_t + V_t$$

with the noise terms uncorrelated and with variance $\Sigma = \Sigma_0$.

The rest of the development of the algorithm follows as in the first-order case.

Experiments with Synthetic Data

The EM algorithm for the case of learning scalar second-order systems was implemented, and tested on synthetic data. A more natural representation of the two deterministic coefficients a_1 and a_2 in the dynamical model is to give the characteristics of the equivalent continuous model, in terms of its frequency ω and damping constant β. For a given sampling interval Δ, these parameters are related as follows. If λ is one eigenvalue of the matrix

$$\begin{pmatrix} a_1 & a_2 \\ 1 & 0 \end{pmatrix}$$

(for an oscillatory system, therefore, $\bar{\lambda}$ is the other eigenvalue), then β and ω are given by

$$-\beta + i\omega = \frac{1}{\Delta} \log \lambda.$$

In the following results, $\Delta = 0.02\,\mathrm{s}$ will be used (corresponding to a frequency of $50\,\mathrm{Hz}$), a typical value for the tracking applications in mind.

For comparison, the performance of the filtered learning algorithm is also examined, using constant velocity dynamics (i.e., $a_1 = 2$ and $a_2 = -1$) to initially filter the measurements, with process noise whose standard deviation was set to a factor of 5 too high. This is typical of a tracking situation, where the order of magnitude of the process noise is all that can realistically be set manually. The results of these experiments, in terms of the mode of the equivalent continuous system, are presented in table 4.5.

It can be seen that the EM learning algorithm produces more accurate estimates, although the estimate of the damping constant is not as good as the other parameters. Filtered learning, in each case, produces a system which is significantly overdamped — this effect will be noted again in section 5.2.

An example of the convergence behaviour of the EM algorithm in this case is given in figure 4.5. The frequency estimate converges almost instantly; the estimate of the damping constant displays satisfactory convergence behaviour, as does the noise term's estimate. The algorithm has produced a good estimate of all components of the model within approximately 16 iterations.

True system			**Filtered estimate**			**EM estimate**		
β	ω	b	β	ω	b	β	ω	b
0.05	3.14	0.01	0.1242	2.996	0.02053	0.0358	3.146	0.01024
0.05	6.28	0.01	0.1928	6.031	0.02012	0.0609	6.321	0.01003
0.10	3.14	0.01	0.3305	2.995	0.02002	0.0917	3.151	0.01001
0.10	6.28	0.01	0.3791	5.997	0.02005	0.1021	6.279	0.00983

Table 4.5: **EM learning produces much better estimates for second-order dynamics than filtered learning..** *Comparison of the estimates produced by filtered and EM learning, for various scalar second-order systems. The EM algorithm was run for 32 iterations, which was enough to give convergence (see figure 4.5). A synthetic sequence of length 8192 was used, with the measurement system given by* $h = 1$ *and* $\sigma = 0.01$*. The initial estimate for the dynamics was given by* $a_0 = -1$*,* $a_1 = 2$*, and* $b = 0.05$*; this system was also used to produce the filtered estimates. (The angular frequencies of* 3.14 *and* 6.28 *correspond to frequencies of* 0.5 Hz *and* 1 Hz *respectively, corresponding to fairly typical time-scales for tracking.)*

4.5.2 Learning the Process Mean

Reynard et al. (1996) describe how the process mean, $\bar{x}$ in the full dynamical model

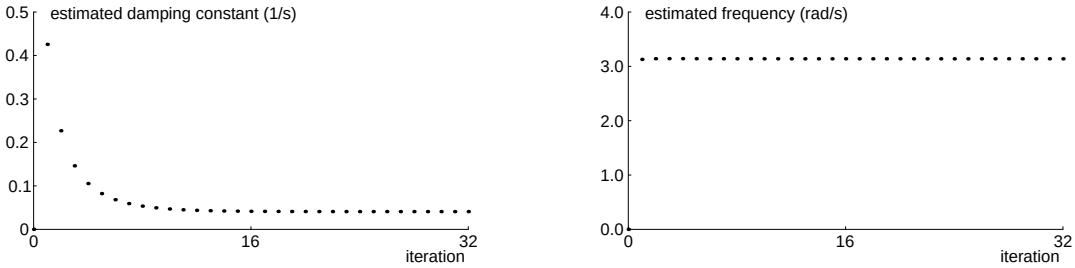
$$(x_t - \bar{x}) = A_1(x_{t-1} - \bar{x}) + A_2(x_{t-2} - \bar{x}) + Bw_t,$$

may be estimated, together with its variability within a class of the same type of object. Here a simplification of the result is sufficient — the solution to the problem of estimating, from a training sequence, the process mean for a single object.

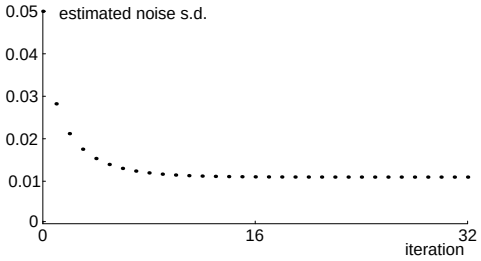
The maximum-likelihood estimates of the model are derived in section 3.2, where it is noted that the log-likelihood in this case is linear in the moments R_{ij} and R_i. The Expectation step therefore reduces, as before, to finding the expected values of these moments. The $\mathcal{E}[R_{ij}]$ are found as in previous sections, and the calculation of the $\mathcal{E}[R_i]$ is trivial:

$$\begin{aligned}\mathcal{E}[R_i] &= \mathcal{E}\left[\sum_{t=3}^{T} x_{t-i}\right] \\ &= \sum_{t=3}^{T} \mathcal{E}[x_{t-i}].\end{aligned}$$

Finally, $\mathcal{E}[x_{t-i}]$ is the top third of $\mathcal{E}[X_{t-i}]$ (where X_t is the augmented state vector defined in section 4.5).



(a) Deterministic part



(b) Stochastic part

Figure 4.5: **Convergence of the estimate of the dynamical model is good.** *Shown are graphs of (a) the estimates of the damping constant and frequency, and (b) the estimate of the process noise for the model against number of iterations of the EM algorithm. The initial dynamical model is constant-velocity, giving zero damping constant and frequency. The values of* β, ω *and* b *used to simulate the system were* 0.05, 3.14 *and* 0.01 *respectively; the sequence was of length 8192. Convergence of the estimates is good within the 32 iterations shown here.*

4.6 Missing Data as an Infinitely Noisy Measurement Process

The analysis given in section 4.3 for learning a dynamical model from a sequence with missing points dealt only with the case of non-consecutive missing data. In this section, the problem of consecutive missing data is addressed, with particular regard to the sleep-data example application presented in section 3.4.

Examining, for example, figure 3.2(c), it can be seen that the value returned by the sensor is often clipped at the top or bottom of its range. The result is that when a reading of 255 is returned, the actual information is that the true value is *at least* 255; it may be more. Similarly, a reading of 0 indicates that the true value is *at most* 0. Particularly for extended periods of such clipping, a significant effect may be produced on the dynamical model learnt for the process.

Dealing with Clipped Data A reasonable approximation to the situation is to treat the sensor as a measurement process which, when it returns 255 or 0, is effectively signalling 'missing'. As noted, the problem of missing data within a time-sequence was addressed to some degree in section 4.3, but here a different technique will be applied, bringing the problem within the scope of the framework of sections 4.4 and 4.5.

The sensor clipping can be cast as a noisy measurement process, by writing, as usual, x_t for the true value at time-step t, and z_t for the sensor reading at time t. Then the measurement process can be modelled as follows.

$$z_t = x_t + v_t,$$

where the measurement noise v_t has variance r_t, with

$$r_t = \begin{cases} r_0, & \text{if } z_t \notin \{0, 255\}; \\ \infty, & \text{if } z_t \in \{0, 255\}. \end{cases}$$

Note that this formulation does not take into account the fact that when a reading $z_t = 255$ is returned, the inequality $x_t \geq 255$ can be deduced, or the corresponding piece of information for $z_t = 0$. It will, however, be a considerable improvement on a model which ignores the phenomenon entirely.

Note also that the measurement noise for a non-clipped sensor reading has been denoted r_0. One choice would be to set $r_0 = 0$, indicating that non-clipped readings are exact, but this will be seen to cause problems later. Fortunately, there is a good reason for setting $r_0 \neq 0$: quantisation error. The assumption is that there is a true, continuous signal being measured to only 8-bit accuracy. Therefore, the sensor reading is not noise-free.

The gain K_t can be written as follows for the two cases $r_t = r_0$ and $r_t = \infty$.

$$K_t = \begin{cases} P_t^{t-1} H^\top (H P_t^{t-1} H^\top + r_0 I)^{-1}, & \text{if } r_t = r_0; \\ 0, & \text{if } r_t = \infty. \end{cases}$$

(The problem of specifying exactly what is meant by a measurement process with infinite variance is therefore avoided.) Note that although the problem in this case is essentially scalar, the matrix notation has been retained. This is to accommodate the augmentation of the state vector.

Smoothing, i.e., obtaining $\hat{x}_t^T$ and P_t^T, can then proceed as usual, as it does not depend explicitly on the measurement variances, only the filtered variances P_t^t and predicted variances P_t^{t-1}.

The smoothing is where having $r_0 = 0$ would cause trouble: after several non-clipped readings, the predicted variance matrix P_t^{t-1} would be of the form

$$P_t^{t-1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & C \end{pmatrix},$$

which is singular. Since the smoothing gain defined in eqn (4.52) involves a term $(P_t^{t-1})^{-1}$, the expressions for the smoothed estimates would have to be re-derived for this special case. Ensuring $r_0 \neq 0$ avoids this.

The remainder of the EM learning algorithm for this particular problem then proceeds as in section 4.4.

Effect on Classification

It has been seen in experiments on synthetic data that EM learning in general provides good estimates for the underlying model. It is reasonable to expect, then, that applying the above techniques to the classification problem of section 3.4.2 would produce an improvement in classification accuracy, even though the existing results were around 90%. The learnt models should more accurately represent the behaviour of the data.

Evaluating the Likelihood There is one further detail which must be worked out. The classification procedure involves evaluating the log-likelihood of the sequence given each candidate model, and choosing the model with the largest result. The question therefore arises of how the log-likelihood is to be calculated when there are missing data in the sequence. Time did not permit the full development of this theory, or indeed any experimental investigation; this area would be an interesting one for future research.

Chapter 5

EM-Learning for a Kalman-filter-based Tracker

5.1 Applying EM Learning to Dynamic Contour Tracking

Chapter 4 developed the theory of maximum-likelihood estimation of a dynamical model for a Kalman filter from a sequence of noisy measurements. The application of particular interest in this thesis is the problem of dynamical models for a visual tracker. Chapter 3 described how, in the case of tracking with good measurements, significant improvements in tracking can be obtained by making the approximation $\hat{x}_t = x_t$. This chapter examines learning in the case where this approximation is no longer valid, which will arise when measurements are of poorer quality. It describes the implementation of the algorithm, and experiments, both of which are the original work of the author (except for some set-up details as noted in section 5.1.3).

One situation where this can occur is when tracking an object moving across a cluttered background. This problem should be one in which the EM learning algorithm enjoys two (related) advantages: firstly, in a situation with poor measurements (as will be the case for tracking over a cluttered background), having a good dynamical model is important, more so than when the measurements are reliable, and secondly, chapter 4 showed that, at least for the synthetic systems considered there, the differences between EM learning and filtered learning are greater when measurement noise is higher.

It is worth mentioning that if one has control over the environment, one would choose to train the tracker from a sequence gathered in a situation with good measurements, i.e., with little clutter; also, if the background is static and known ahead of time, background subtraction (for example the dynamic background model used by Koller et al. (1994) for traffic monitoring, the median-filtered background subtraction used by Baumberg and Hogg (1994)

for person-tracking, or the statistical background models developed by Rowe (1996)) is a natural technique to apply to the problem. However, such control may not always be available, and therefore this chapter addresses the problem of learning a dynamical model for the tracker from a training sequence gathered in the presence of significant clutter, incorporating no knowledge of the background.

5.1.1 Implementation of EM Learning in the Tracker

To apply the learning algorithm described to the tracker, it is necessary to include a small number of refinements. In the tracker, the measurement matrix H is not constant over time — it is calculated at each time-step based on the image normals at measurement points around the spline, as described in section 2.1.4. This affects the filtering process, but not the smoothing process, and so it is fairly straightforward to include this extension. Another complication to the measurement process is that the tracker makes multiple measurements, typically 3–5 per span of the spline, at each time-step and not all of them find a feature. (The tracked object may, for example, be over an area of similar greylevel or colour to itself, and there may therefore be little contrast across the boundary of the object.) Therefore, the measurement vector can vary in length from time-step to time-step. This effect can also be included in a straightforward fashion, since it is part of the more general phenomenon of having a time-varying H.

While the algorithm is easily extended to the case where H varies from time-step to time-step, for any given time-step, it must be the same from iteration to iteration of the algorithm. The values used must be the ones used while the training sequence was being gathered. There are at least two ways of achieving this. One would be to store the values of H used by the tracker used to gather the training data; the other would be to re-run the untrained tracker for each iteration, thereby regenerating the same series of H matrices (since the measurements, and everything else controlling the tracking, is the same). The trade-off is between the storage required for the H_t and the calculations required to re-compute them. Arbitrarily, the latter method was chosen for these experiments. Note also that the capability of the algorithm to cope with a time-varying measurement process allows it to accommodate the case where video fields are dropped — this can happen, for instance, on a multi-tasking machine when another process competes for CPU time with the tracker. A dropped field can be considered a measurement with infinite measurement noise or, equivalently, with zero information (see section 4.6). Once again, this only affects the filtering; the smoothing and other steps of the algorithm are unchanged.

5.1.2 Example Application: Hand Tracking

As an example of tracking an object in clutter, consider the problem of tracking a hand moving across news-sheet. This background was chosen as being representative of the type of clutter which would be expected in the general problem of hand-tracking across a cluttered background, for example an untidy desktop. A tracker was therefore set up with a contour matching the outline of the hand.

Estimating the Measurement Noise The first issue to deal with is that of determining the variance Σ of the measurement process. As will be described in section 5.3.1, it is possible to learn this variance from a measurement sequence, at the same time as learning the dynamical model, but, in practice, convergence of the EM algorithm in this case is rather slow (see section 5.3.1). Therefore, an alternative and more appropriate approach to the estimation of Σ was used.

The measurement process in the tracker (see figure 2.1 on 21) consists of casting normals at certain points around the predicted position of the contour and searching for features along these normals. Often, especially in clutter, many features will be found. One must then be chosen; the distance ν_i along the normal between it and the contour is used as one component of the innovation. In the experiments which are described here, the strongest edge feature was used.

The complete innovation vector is then built up from the ν_i from all measurement points. The noise of each measurement is assumed to be Gaussian, with all errors independent. The variance of the measurement vector is therefore $\sigma^2 I$, and it is σ^2 which is estimated, as follows.

Consider the situation where the contour is positioned exactly over the outline of the hand. In this case, if there were no noise, every ν_i would be zero. In the presence of noise, the sample variance of the ν_i can then be used as an estimate of the measurement process variance σ^2.

An image taken of the hand in a typical position for the tracking problem is given in figure 5.1. The contour has been positioned manually over the hand. Also shown in this figure are the image normals, and the strongest feature on each normal. The above process of calculating the sample variance of ν_i was performed on this image, with the result that the measurement process variance was estimated as $62\,\text{pixels}^2$. This value was used for the experiments which follow. Note that this value of the measurement noise gives a standard deviation of around 8 pixels, which is considerably greater than the typical value resulting

from image noise alone. It is the clutter in the image which produces the large measurement noise.

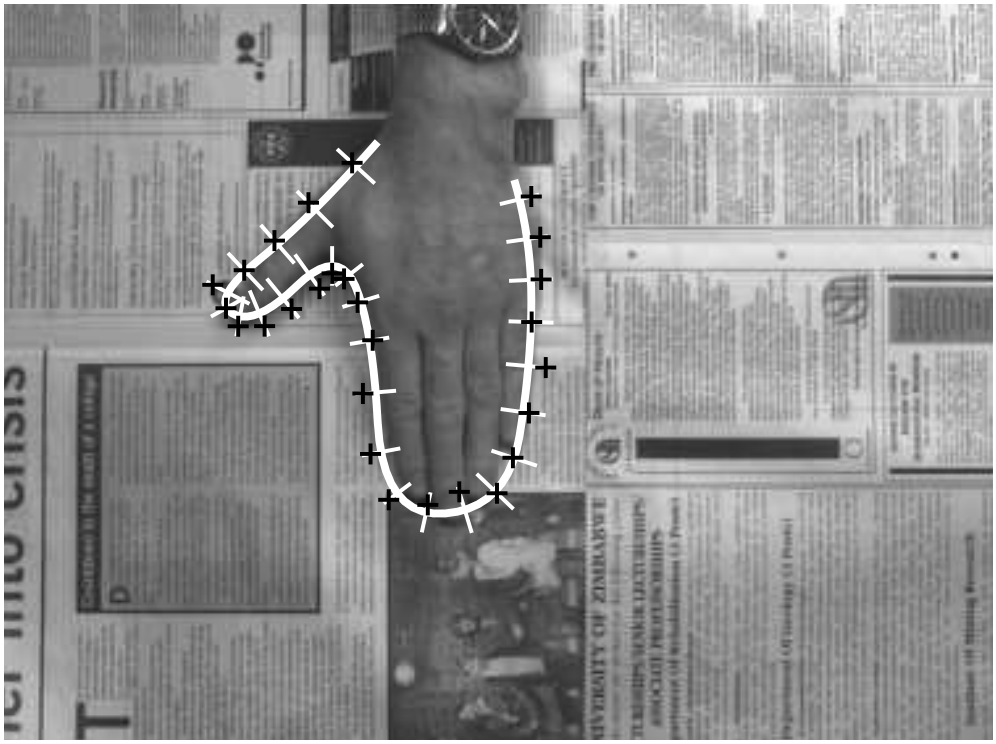


Figure 5.1: **A direct method is used to estimate measurement noise for a hand on a cluttered background.** *The position of the contour corresponding to the location of the hand is shown (thick white line), as are the image normals along which features are sought (thin white lines). The black crosses show the locations of the chosen feature along each search line. Note that many of the crosses are on clutter features rather than on the image edge at the outline of the hand. The sample variance of the normal distances between the contour and the chosen features is used as an estimate of the measurement noise — in this case,* $62\,\text{pixels}^2$.

It must be acknowledged that the Gaussian model used for the measurement process is a significant simplification of the underlying random processes, and this becomes more apparent here, when estimating its variance. Many factors — the length of the search normal; the strength of the true edge; the position, number and strength of clutter features — influence the innovation measured along a particular contour normal, and these, within the present framework, are all conflated into the Gaussian model. The variance used must necessarily, then, reflect innovations arising from normals where a clutter feature was chosen. Addressing the limitations of the present model, MacCormick and Blake (1998) present a much more thorough analysis of the stochastic processes by which image features

arise.

In particular, the length of the search line will be highly significant in the estimate of the measurement variance, and its choice is currently slightly ad hoc. The matter is further complicated by the fact that during tracking, the length of each search line varies in proportion to the variance in that direction at that point of the contour, as calculated from the state variance P. (This is done to provide a basic form of 'validation gate' for the tracker (Blake et al., 1995).) However, if the length used in the measurement variance estimate is typical of the length used in tracking, as was the case here, the resulting value for σ will be a reasonable estimate of the noise, within the Gaussian approximation used.

Simple Motion — Gathering a Training Sequence As a basic test, the tracker was given a model space allowing translation in the x and y directions, and a set of 'untrained' dynamics representing damped simple harmonic motion. A short (128-field) training sequence of the hand executing motion in the x-coordinate direction was used as data for the purposes of comparing the two learning methods — the untrained dynamics did not allow a sequence of much greater length to be obtained. The x-coordinate of the hand during the sequence, as estimated by the tracker with untrained dynamics, is shown in figure 5.2. For the purposes of performing EM learning on this sequence, the measurements (i.e., the chosen features) were recorded at each time step, as well as the filtered estimates. In this case, no fields were dropped, although, as noted, it would have been possible to use a sequence with missing fields.

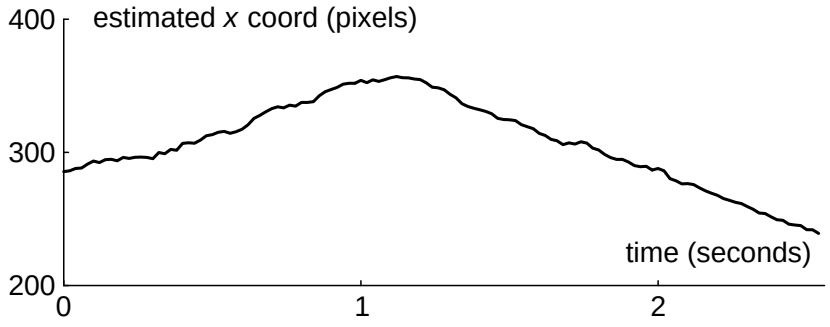


Figure 5.2: **A simple sequence consisting of horizontal translation is used as an initial test of the learning algorithm.** *This sequence was used in both learning algorithms.*

Learning the Dynamical Model To compare the two learning methods, each was used with the sequence shown in figure 5.2. The filtered learning algorithm used the estimates,

whereas the EM learning algorithm used the measurements. The EM algorithm was initialised with the model learnt by filtered learning, and run for 96 iterations, enough for reasonable convergence. (See below for a summary of the convergence behaviour.)

Recall that the model used is

$$(x_t - \bar{x}) = A(x_{t-1} - \bar{x}) + Bw_t,$$

where the mean $\bar{x}$, the deterministic part A and the stochastic part B are all to be learnt. Note that, as usual, A is given in terms of the second-order coefficients A_1 and A_2 by

$$A = \begin{pmatrix} A_1 & A_2 \\ 1 & 0 \end{pmatrix}.$$

The results of learning the mean $\bar{x}$ by the two methods are as follows:

$$\bar{x}_F = (308, 193); \quad \bar{x}_{EM} = (309, 182),$$

where $\bar{x}_F$ is the mean as learnt by filtered learning, and $\bar{x}_{EM}$ is the mean as learnt by EM learning, and these coordinates are in pixels.

The deterministic part A of the dynamical model produced by each learning algorithm is shown in table 5.1. The eigenvalues of the corresponding continuous system ($\dot{x} = Fx$) are used to give the damping constant and frequency of the mode. (These eigenvalues are of the form $-\beta + i\omega$ for damping constant β and frequency ω.) Note that the frequencies shown in the table are in Hz, whereas ω is in rad/s.

Damping constant (s^{-1})	**Frequency** (Hz)
0.23	0
5.26	0
18.81	2.35

(a) Modes learnt using filtered learning

Damping constant (s^{-1})	**Frequency** (Hz)
−0.27	0.35
10.24	1.80

(b) Modes learnt using EM learning

Table 5.1: **EM learning produces a model with a more plausible deterministic part.** *The modes of the system learnt by each method are shown, in terms of the damping constant and frequency of the corresponding mode of the continuous system. A frequency of zero indicates a non-oscillatory mode; see text for comments on the unstable mode learnt by EM, indicated by the negative damping constant. (Note that since this is a second-order, two-dimensional system, there are four modes per model. Each oscillatory mode provides two (conjugate) eigenvalues.) The first mode of each model lies within* 8° *of the x axis, as expected.*

The stochastic parts of the two models are as follows, in terms of the eigenvalues of the noise variance matrix $BB^\top$. For the model learnt using filtered learning,

$$\sigma_1^2 = 3.15; \qquad \sigma_2^2 = 0.62$$

(in pixels^2), whereas for the model learnt using EM learning:

$$\sigma_1^2 = 0.36; \qquad \sigma_2^2 = 0.06$$

(again, in pixels^2).

The estimates for the mean produced by the two algorithms are very similar, and match what one might intuitively expect to be reasonable values by looking at the training sequence in figure 5.2.

Both algorithms produce a dynamical model whose deterministic part is dominated by a mode almost along the x axis with a time constant of a few seconds, in line with expectations. Curiously, the model learnt by the EM algorithm is technically unstable. However, in the presence of good enough measurements, the resulting tracker will be stable. The training sequence does indicate some growth (see figure 5.2), especially when viewed as an oscillation around the learnt mean, so an unstable system is perhaps a more plausible model of the behaviour than the stable one learnt using filtered learning.

Both models have a time-scale (reciprocal of the damping constant) of roughly four seconds, which, again, fits expectations from the training sequence. The frequency of the dominant mode as learnt using EM corresponds to a period of around 2.9 s, a value consistent with the time-series of figure 5.2.

The difference between the stochastic parts of the models is much greater — the first principal variance is smaller by a factor of about 10 for the EM-learnt model. This can be explained qualitatively as follows. While gathering the training sequence, the process noise was such that the measurement noise was incorporated into the tracked estimates (cf section 4.4.4). Therefore, filtered learning includes this noise in the model. EM learning, however, avoids this problem, by learning from the measurements directly. It is not easy to say whether the smaller, EM-learnt noise is more correct; the tracking performance on test sequences described below suggest that it is, however.

Tracking Results The two sets of dynamics (one from the existing learning algorithm and one from the EM learning algorithm described here) were used to configure two trackers. All other parameters controlling the trackers' behaviour were identical in each tracker.

A sequence of 4 s in length was recorded to test the two trackers. In this sequence, the hand performed motion of broadly the same type, i.e., motion in the x-coordinate direction, as was presented to the training algorithms.

Both trackers were run on this sequence — the results are shown in figure 5.3 for the tracker using dynamics learnt with the existing algorithm and in figure 5.4 for the tracker using EM-learnt dynamics. For comparison purposes, an untrained tracker was also run on the sequence; its performance is indicated in figure 5.5.

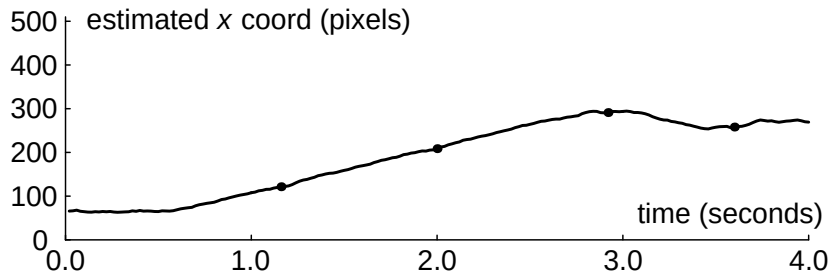
Firstly, the untrained tracker is distracted early on in the sequence by the background clutter, and loses lock. It does not recover, and the estimated hand contour moves around randomly for the rest of the sequence (not shown).

The tracker trained using filtered learning fares better, as is to be expected. For approximately the first three-quarters of the sequence, tracking is good, if a little noisy. This noise in the tracked estimate is a direct result of the higher process noise used — the noise of the measurements is included in the tracked estimate. However, at approximately still (c) of figure 5.3, the contour latches onto a clutter feature; it does not return to the outline of the hand for the rest of the sequence. Note, however, that although the estimate in still (d) is not correct, it is at least consistent with the broad character of the training sequence in that the y coordinate is within the fairly tight bounds expected — contrast this with the behaviour of the untrained tracker in figure 5.5, whose estimate is in error vertically.

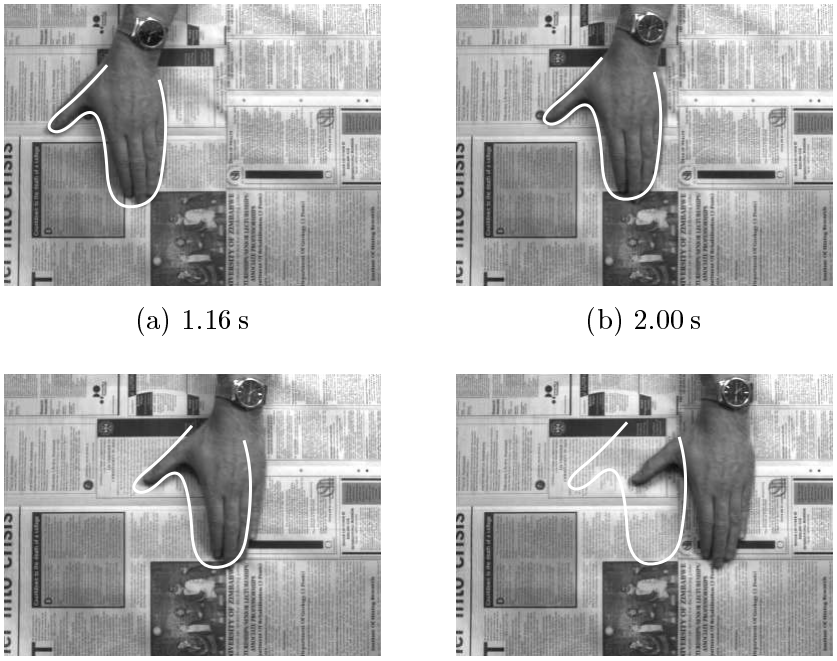
Finally, the tracker whose dynamical model was learnt using the EM algorithm presented here tracks successfully throughout the sequence, with good estimates and low noise.

Convergence of the EM Algorithm The convergence behaviour of the EM learning algorithm in this application is now briefly examined. Figures 5.6 and 5.7 show the convergence of the various parts of the dynamical model.

As might be expected from the experiments on synthetic data described in section 4.4.4, the convergence of the model mean is almost instantaneous, and so is not shown. The frequency of the principal mode (shown in figure 5.6) also converges rapidly. The damping constant (same figure) converges rather more slowly, and it might even be the case that a few more iterations would result in a slightly different value, but the effect on the resulting model would be small. The variances of the stochastic part also converge within a reasonable number of iterations.



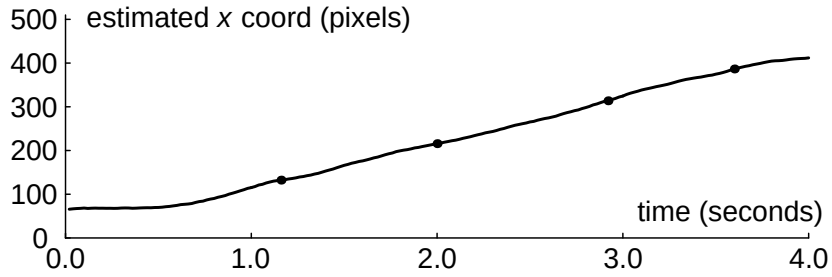
Estimate of the hand's x coordinate against time — tracker trained using filtered learning. Note loss of lock around 3.0 s.



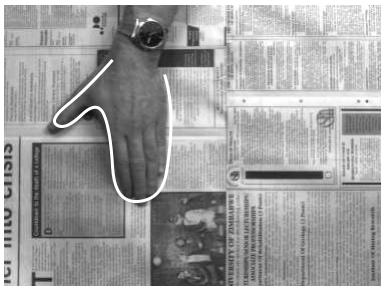
(a) 1.16 s (b) 2.00 s

(c) 2.92 s (d) 3.60 s

Figure 5.3: **Filtered learning produces a tracker which is distracted by clutter.** *Graph of the tracker's estimate of the hand's x coordinate (in pixels) against time for a* 4 s *sequence. Stills at selected points during the tracking are shown underneath, with the tracker's estimate of the contour superimposed. (The points at which the stills were recorded are shown as beads on the graph.) Using a dynamical model obtained from filtered learning, tracking is successful during the initial motion (stills (a) and (b)). However, at still (c), the tracker is distracted by some background clutter, and begins to lose lock. By still (d), lock is lost completely — the tracker follows clutter and image noise for the rest of the sequence.*



Estimate of the hand's x coordinate against time — tracker trained using EM learning. Note retention of lock throughout.



(a) 1.16 s

(b) 2.00 s

(c) 2.92 s

(d) 3.60 s

Figure 5.4: **EM learning produces successful tracking throughout the sequence.** *The sequence in figure 5.3 is successfully tracked — the only difference between the configurations of the two trackers is that here the dynamical model used was one learnt (from the same training sequence as the one used to train the tracker in figure 5.3) using the EM algorithm described.*

Figure 5.5: **An untrained tracker is rapidly distracted by clutter.** *A still from 0.32 s into the sequence, by which time an untrained tracker has already become distracted by the background clutter.*

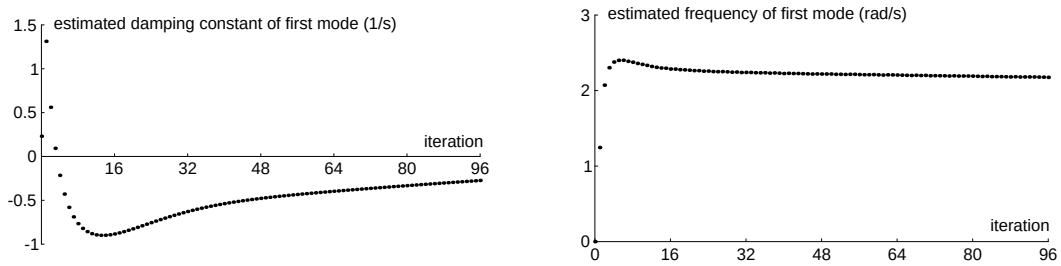


Figure 5.6: **The estimate of the deterministic part of the dynamical model converges to a value consistent with inspection of the training sequence.** *Shown are graphs of the estimates of the damping constant and frequency for the first (most persistent) mode of the model against number of iterations of the EM algorithm.*

More Complex Motion — Gathering a Training Sequence To investigate the behaviour of the EM-based training algorithm when the desired motion is more complex than that described above, another training sequence (again in the hand-tracking application) was gathered. To gather a longer training sequence, some initial 'bootstrap' training was required, since, as noted in the description of the previous experiment, the untrained dynamics were incapable of following the hand's motion for more than two or three seconds. ('Bootstrapping' consists of iteratively learning a dynamical model, using an estimated model to filter the measurements, and then learning the next estimate of the dynamics from the filtered state estimates.)

For this sequence, the hand was moved in a rough ellipse filling most of the field of view of the camera, in an anti-clockwise direction. The motion was tracked using dynamics learnt from a shorter sequence, as described above. Plots of the tracker's estimates of the x- and y-coordinate against time are shown in figure 5.8. As for the experiments with simpler motion, the measurements made by the tracker were recorded in this training sequence, so

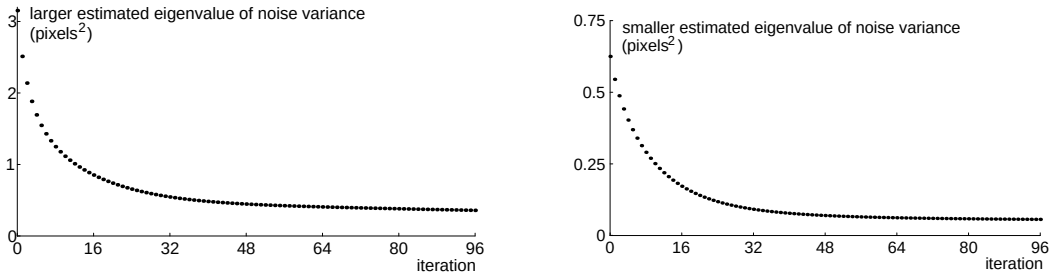


Figure 5.7: **The estimate of the stochastic part of the dynamical model converges to a lower value than that learnt using filtered learning.** *Shown are graphs of the eigenvalues of the noise variance matrix against number of iterations of the EM algorithm.*

that the sequence could be used by both learning algorithms.

Learning the Dynamical Model Both learning algorithms were run using the data gathered — the models learnt using each method are now summarised.

The mean position of the hand as learnt by each method is as follows. For filtered learning,

$$\bar{x}_F = (307, 282),$$

and for EM learning,

$$\bar{x}_{EM} = (305, 286).$$

These figures (all in pixels) are in line with what would be expected from a brief examination of the trajectory in figure 5.8(a), and, as before, there is little difference between the models learnt by the two methods.

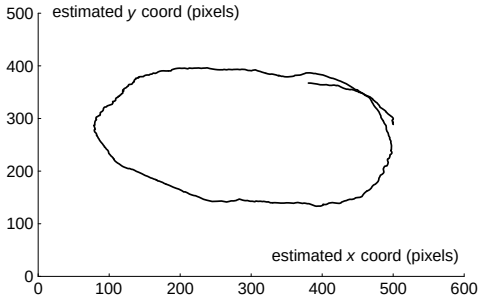
The deterministic parts of the models are shown in table 5.2, in terms of the modes of the equivalent continuous model (as described for the experiments with simple motion).

The stochastic parts of the two models, in terms of the eigenvalues σ_1^2 and σ_2^2 of the noise variance $BB^\top$, are as follows (all in pixels^2). For the system learnt with filtered learning,

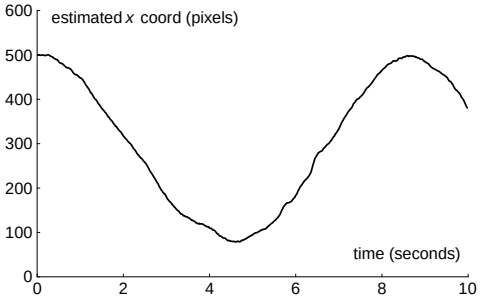
$$\sigma_1^2 = 1.30; \qquad \sigma_2^2 = 0.20,$$

whereas for the model learnt using EM learning,

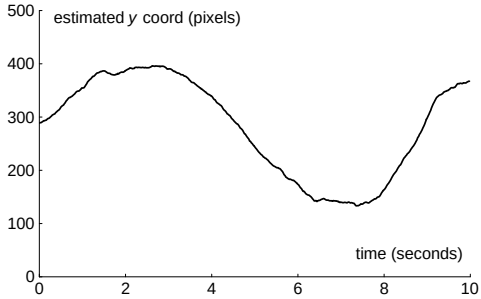
$$\sigma_1^2 = 0.33; \qquad \sigma_2^2 = 0.22.$$



(a) Trajectory of training sequence.



(b) Time sequence of estimated x-coordinate.



(c) Time sequence of estimated y-coordinate.

Figure 5.8: **A more complex sequence, of roughly elliptical motion, is used as a second test of the algorithm.** *A training sequence gathered using dynamics learnt (using filtered learning) from a 2-second sequence, as described in the text. This sequence was used in both learning algorithms.*

Damping constant (s^{-1})	**Frequency** (Hz)
0.01	0.12
16.17	0
30.58	0

(a) Modes learnt using filtered learning

Damping constant (s^{-1})	**Frequency** (Hz)
0.02	0.11
13.58	0
32.31	0

(b) Modes learnt using EM learning

Table 5.2: **Filtered and EM learning produce models with very similar deterministic parts.** *The modes of each system are shown, in terms of the damping constant and frequency of the corresponding mode of the continuous system. A frequency of zero indicates a non-oscillatory mode. For both learning methods, there is one dominant mode, with a frequency matching what would be estimated from the training data of figure 5.8 (i.e., a period of around* 9 s*). The other modes have very short decay times and are non-oscillatory.*

Although the effect is less marked for this sequence, filtered learning has again produced a model with higher process noise. The deterministic parts are very similar; the estimated frequency corresponds to a period of around 9 s for both models, matching the time-series in figure 5.8 well. With greater process noise, the tracker will rely more on the measurements, with two (related) results: firstly, the estimates produced will be noisier, and secondly, the tracker may be more easily distracted by clutter to the extent of losing lock. The performance of the two trackers on test sequences is next examined.

Tracking Results To test the first prediction, that the tracked estimate will be noisier for the system produced by filtered learning, each tracker was run on a sequence in which the hand was held entirely still. The tracker was initialised by hand, and then run for five seconds. Ideally, the resulting estimate of the hand's position would therefore be constant over the sequence, but image noise will cause some jitter — this is unavoidable. The actual tracked estimates against time are shown in figure 5.9. Note that the first ten fields are not shown — during this time the tracker adjusted slightly from the hand-chosen starting location — and that the coordinates have been shifted so that they are zero-mean.

The variances of these sequences are as follows. For the dynamics learnt with filtered learning,

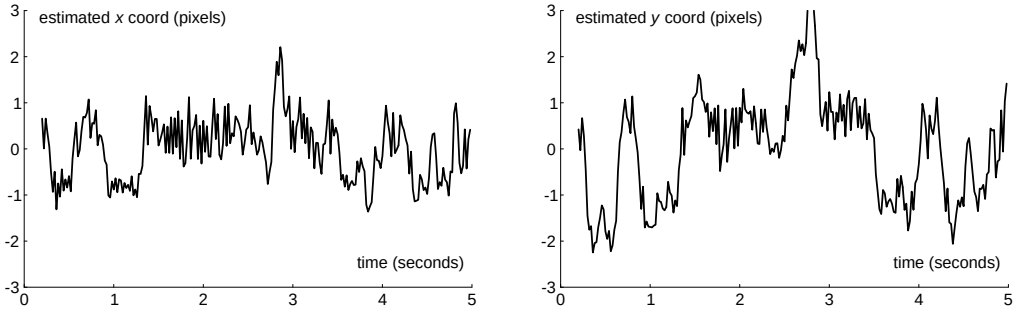
$$p_x = 0.44; \qquad p_y = 1.35,$$

and for the EM-learnt dynamics,

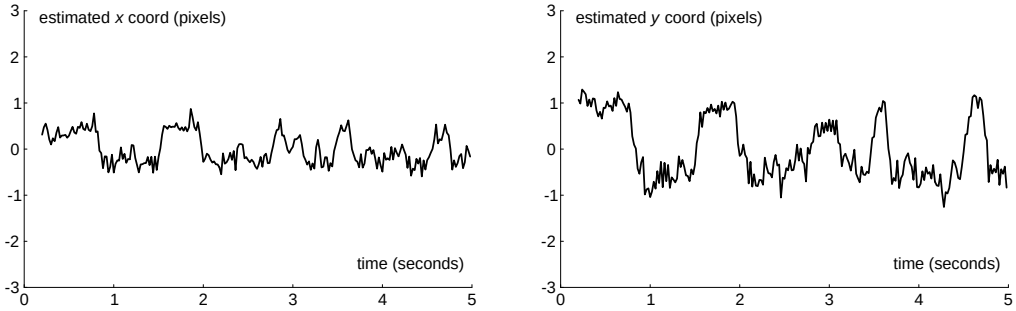
$$p_x = 0.11; \qquad p_y = 0.45$$

(all in pixels^2). In both cases, the standard deviation of the noise in the y-coordinate direction is larger by a factor of approximately 2. This is almost certainly caused by interlace — the camera produces images with double the resolution in the x-coordinate direction. The variances for the EM-trained tracker are 3–4 times lower than those for the filtered-trained tracker, although all are small (with standard deviations of order a pixel). The difference may be significant in, for example, a 'mousing' application, where precise control of a pointer is required.

The trackers were also tested on longer test sequences in which the hand executes motion of broadly the same type (namely, elliptical motion in an anti-clockwise direction) as in the training sequence. Two test sequences were recorded, and both trackers used on them. The first 10 s sequence was recorded of the hand performing two complete anti-clockwise



(a) Estimated location with filtered-learnt dynamics



(b) Estimated location with EM-learnt dynamics

Figure 5.9: **A tracker trained using EM learning produces a much less noisy estimate of the position of a stationary hand.** *Estimated x- and y-coordinates of the tracked estimate of the location of a stationary hand against time. (a) Tracker trained using filtered learning. (b) Tracker trained using EM learning. The EM-learnt dynamics have lower process noise and therefore, as expected, perform more smoothing of the noisy measurements.*

rough ellipses — this test sequence therefore contained motion of a more agile nature than that in the training sequence, which consisted of just over one cycle. A tracker using each dynamical model was then run on this sequence. As before, the only difference between the parameters controlling the behaviour of the trackers was the dynamical model. Each tracker was initialised by hand, then run on the sequence.

The performance of the tracker trained using filtered learning is indicated in figure 5.10. For all of the first ellipse and half of the second, it tracks satisfactorily, but then the presence of a clutter feature causes it to lose lock — it then remains positioned over this feature for the rest of the sequence. The corresponding results for the tracker trained using EM learning are presented in figure 5.11. Although there is some noise in the estimates, as is to be expected for a sequence with noisy measurements, lock is maintained throughout — tracking is successful.

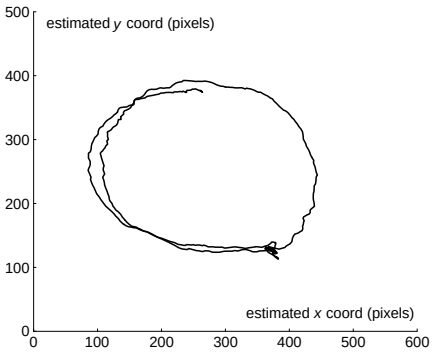
A second 10 s test sequence was recorded, this time of the hand describing most of an ellipse, more slowly than in the training sequence. (Neither of the test sequences, therefore, contained motion exactly the same as the training sequence, providing an indication of the capability of the model to generalise the training motion.) The results are similar to those for the previous test sequence, and are shown in figures 5.12 and 5.13. As before, the tracker trained using filtered learning is distracted by a clutter feature, whereas that trained with EM learning tracks the whole sequence successfully.

Convergence of the EM Algorithm The convergence behaviour in this case is shown in figures 5.14 and 5.15. Note that the convergence of the estimated process mean was almost instantaneous (within three iterations), and so it is not shown.

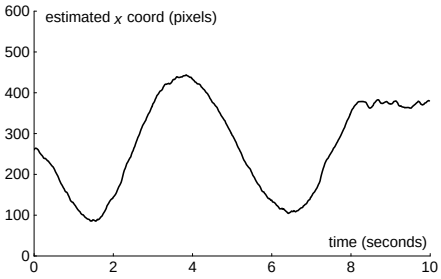
As in the case of simpler motion, the convergence of the damping constant is the least rapid, but the results are adequate within the 64 iterations which were performed. It is unlikely that iterating further would produce significantly different dynamics.

5.1.3 Example Application: Lip Tracking

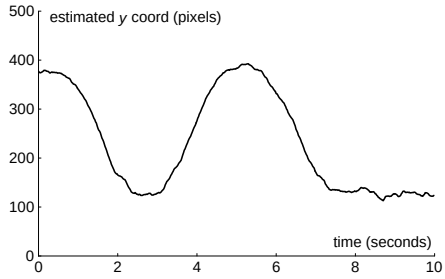
As an example of the application of the learning technique to another tracking domain, consider the problem of tracking the outline of a user's lips for speech-reading purposes (Kaucic, 1997). The information gained from such a tracker can produce significant improvements in the accuracy of speech recognition when compared to audio-only systems, especially in the presence of audio noise.



(a) Trajectory

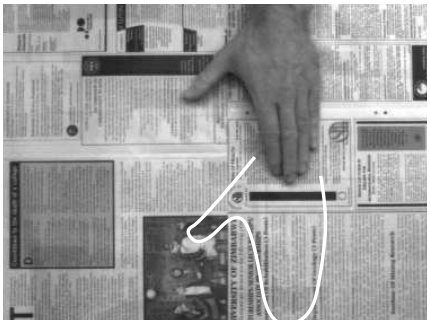


(b) Time series of x-coordinate



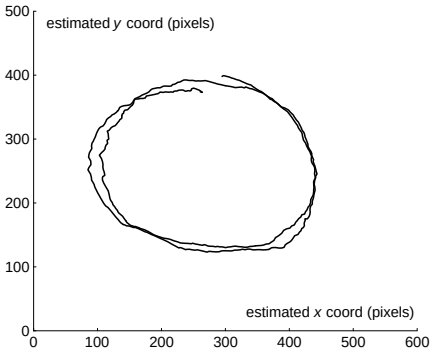
(c) Time series of y-coordinate

(d) 8.20 s

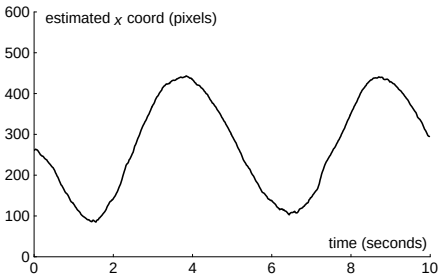


(e) 9.60 s

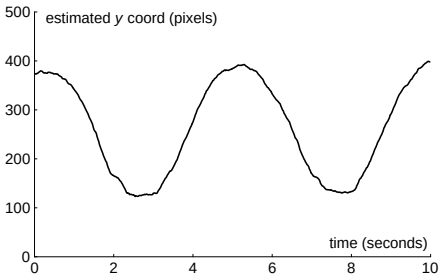
Figure 5.10: **First test sequence: A tracker trained using filtered learning is distracted by a clutter feature.** *In (a), the trajectory of the tracked estimate is shown, together with the time-series of the estimated x- and y-coordinates. In (d) and (e), representative stills from towards the end of the sequence are shown. Initially, tracking is satisfactory, but shortly after (d), the contour locks onto a clutter feature (losing the hand), where it remains for the rest of the sequence, as seen in (e).*



(a) Trajectory

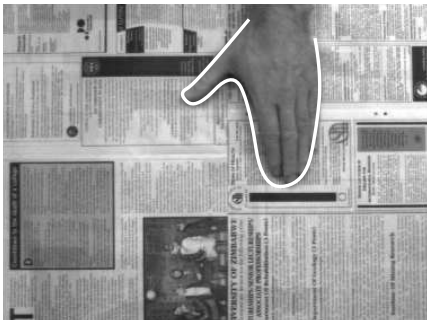


(b) Time series of x-coordinate



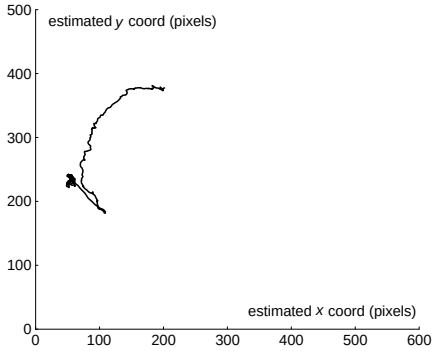
(c) Time series of y-coordinate

(d) 8.20 s

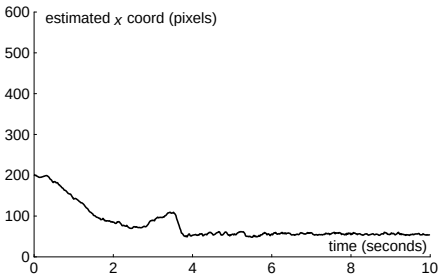


(e) 9.60 s

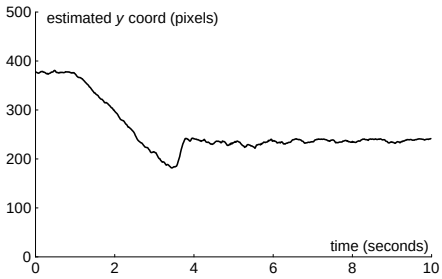
Figure 5.11: **First test sequence: A tracker trained using EM learning successfully tracks the entire sequence.** *The same sequence as used to produce figure 5.10 is tracked using dynamics learnt with EM learning — the trajectory is shown in (a), with the estimated coordinates plotted in (b) and (c). Although there is some unavoidable noise from the measurements incorporated into the estimates, tracking is successful throughout the sequence. Stills from the sequence (taken at the same time as the corresponding stills in figure 5.10) are shown in (d) and (e), where it can be seen that the tracked estimate is good.*



(a) Trajectory



(b) Time series of x-coordinate

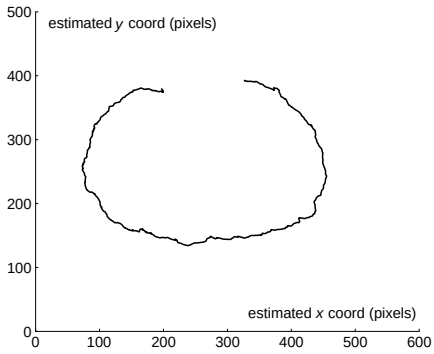


(c) Time series of y-coordinate

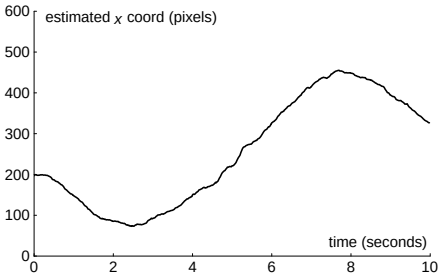
(d) 3.12 s

(e) 6.00 s

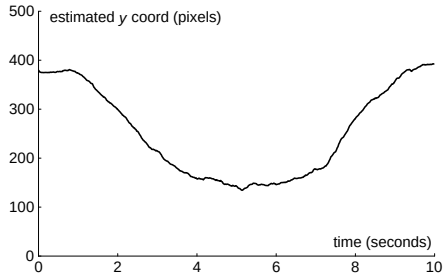
Figure 5.12: **Second test sequence: The tracker trained using filtered learning locks onto a clutter feature early in the sequence.** *In (a), the estimated trajectory of the hand is shown, with coordinate estimates in (b) and (c). Shortly after the still shown in (c), the tracker is attracted to a clutter feature, where it remains. Still (e) shows the estimate at a later stage.*



(a) Trajectory



(b) Time series of x-coordinate



(c) Time series of y-coordinate

(d) 3.12 s

(e) 6.00 s

Figure 5.13: **Second test sequence: Successful tracking using dynamics learnt using EM.** *The whole sequence is tracked correctly — example stills (at times as in figure 5.12) are shown.*

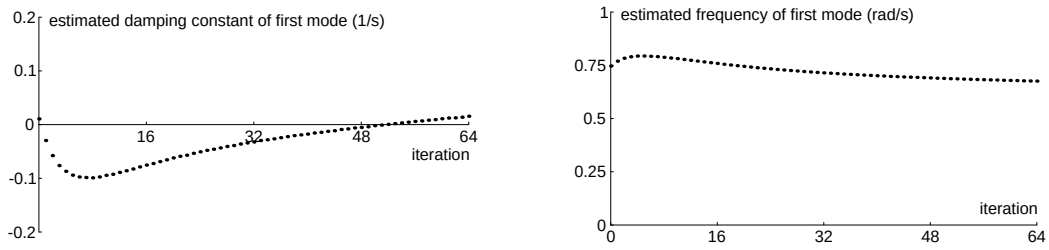


Figure 5.14: **Convergence: deterministic part of the dynamical model for hand motion.** *Shown are graphs of the estimates of the damping constant and frequency for the first (most persistent) mode of the model against number of iterations of the EM algorithm.*

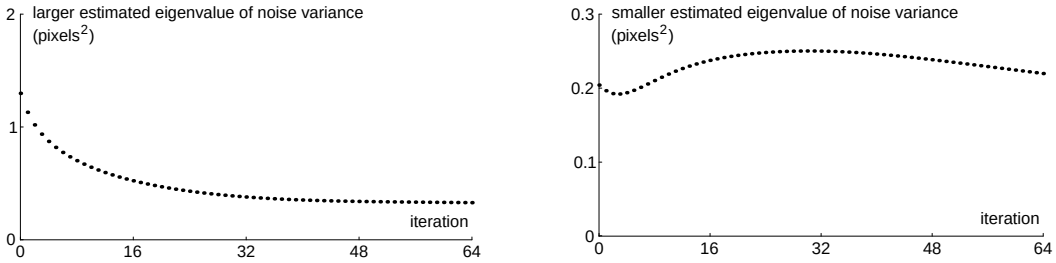


Figure 5.15: **Convergence: stochastic part of the dynamical model for hand motion.** *Shown are graphs of the eigenvalues of the noise variance matrix against number of iterations of the EM algorithm.*

Problem Set-Up

(The data and some of the basic tracking set-up used in this section are used by kind courtesy of Major Dr Robert Kaucic.) The usual grey-level tracking system does not work particularly well in this problem, especially for detection of the location of the lower lip. Kaucic (1998) develops a system tailored to the lip-tracking problem.

A Fisher discriminant is used in RGB-space to locate the outer lip contour, choosing the strongest edge feature as in the hand-tracking example. To locate the inner-lip contour (between the lips and the interior of the mouth), a Gaussian distribution is learnt for lip-coloured pixels, and a mixture of Gaussians learnt for pixels corresponding to the interior of the mouth. (A mixture is appropriate here since teeth, tongue, and very dark mouth-interior pixels may arise.) An edge feature is then located by thresholding the posterior probability that each pixel along the search line arises from the 'lip' or 'mouth' distributions.

It should be emphasised that these enhancements to the basic tracking system do not cause any conceptual changes in the filtering or learning algorithms — all that is changed is the method whereby the measurement feature is located. Within this specialised tracking

framework, a 9-dimensional PCA-based model space was used, covering the deformations undergone by the lips' contours during typical speech.

Learning the Model

The training sequence used was 25 s long, and consisted of the speaker counting from 17 to 25 with short pauses between the numbers. To provide slightly better measurements for the learning process, the speaker wore coloured lipstick, and the tracker's Fisher axes modified appropriately. To gather the training data, dynamics modelling a simple damped harmonic oscillator were employed. The resulting tracking was adequate for gathering training measurements.

The EM algorithm was then employed on these data, with 96 iterations being performed. This was enough to produce convergence within acceptable limits.

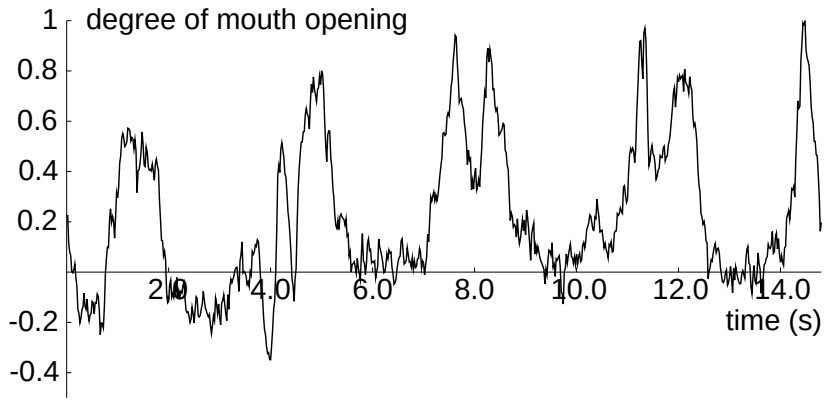
Tracking Results

A sequence consisting of an unadorned (without any lipstick) subject speaking the numbers 14 to 17 was used to test the learnt model.

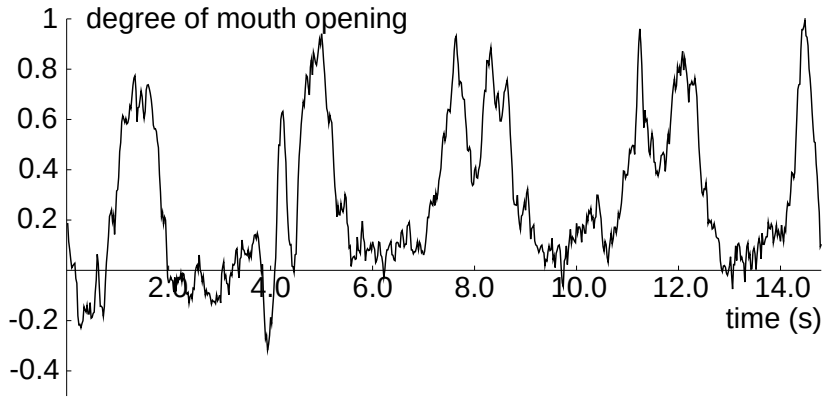
In figure 5.16(a), the degree of mouth opening is shown for the sequence tracked using the model learnt with filtered learning. The corresponding results for the model learnt using EM learning are given in figure 5.16(b). Note that there was no single model-space component corresponding to mouth opening, but a straightforward projection onto a suitable one-dimensional subspace of the full control-point space allows this parameter to be measured. The value of this parameter can be negative, indicating that the mouth is more closed than in its template configuration.

It will be appreciated that the two graphs are qualitatively very similar, implying that the tracked estimates arising from the use of the two models are close. To better examine the differences in behaviour, the 'distance' between the contours at each time-step was produced, in accordance with the metric defined in section 2.1.6. This graph is shown in figure 5.17.

The distance between the two splines is mostly around 5–10 pixels. However, there are a few points during the sequence when the distance becomes greater, and these are examined in more detail now. Observation of figure 5.17 reveals that the four biggest peaks in this distance function occur at around fields 80, 240, 440, and 570. Inspection of the raw data produces more accurate values for these field numbers: 80, 239, 439, and 566. The estimates produced using each model at these fields are shown in figure 5.18.



(a) Model produced by filtered learning



(b) Model produced by EM learning

Figure 5.16: **Filtered learning and EM learning produce models giving similar estimates of mouth-opening parameter.** *The two time-series are similar, but there are areas where greater differences are apparent. See text and figures 5.17 and 5.18.*

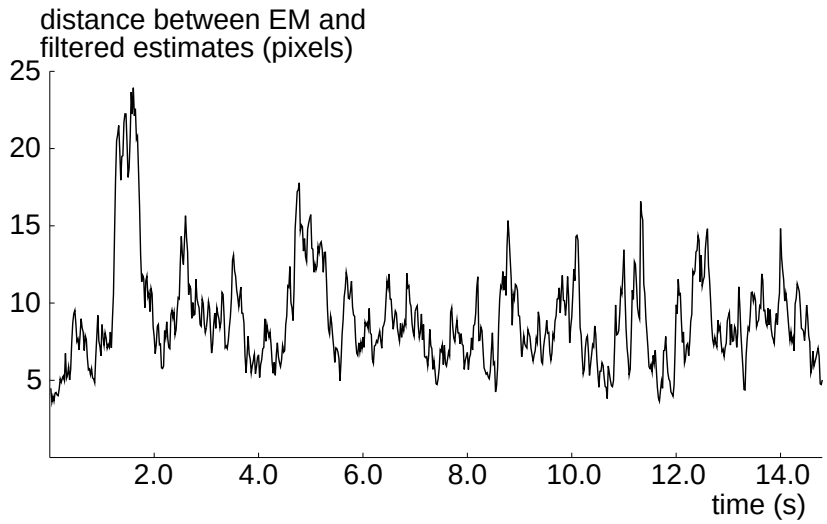


Figure 5.17: **At some times during the lip-tracking, the difference between the two estimates becomes significant.** *The difference between the splines estimated by the two trackers is calculated by means of the spline metric (see section 2.1.6).*

The differences are chiefly in the configuration of the inner contour, and more particularly in its horizontal position. There is little information to be gleaned from the image in this direction — the contour normals deviate little from the vertical. However, there are some differences in the outer contour's configuration, and the EM-trained estimates are in better agreement with the image.

Both trackers have produced extremely inaccurate estimates at field 566. The mouth is almost completely closed, and yet both estimated contours are quite open. This field occurs during the 'v' of 'seventeen', a point of very high acceleration, as the lower lip rises sharply from the first 'e'.

Discussion

The difference between the old and new learning methods in this case is much less than has been noted in the hand-tracking examples — neither tracker loses lock, and, for the most part, both trackers produce good estimates of the lips' configuration. Even at the points of the sequence when the difference between the estimates is large, the outlines are very similarly sited, and although one can claim that the estimates produced by the tracker using the EM-learnt model are slightly better, particularly in terms of the shape of the inner-lip contour, the improvement is small.

The worst performance, from both trackers, was seen at the 'v' of 'seventeen' (around

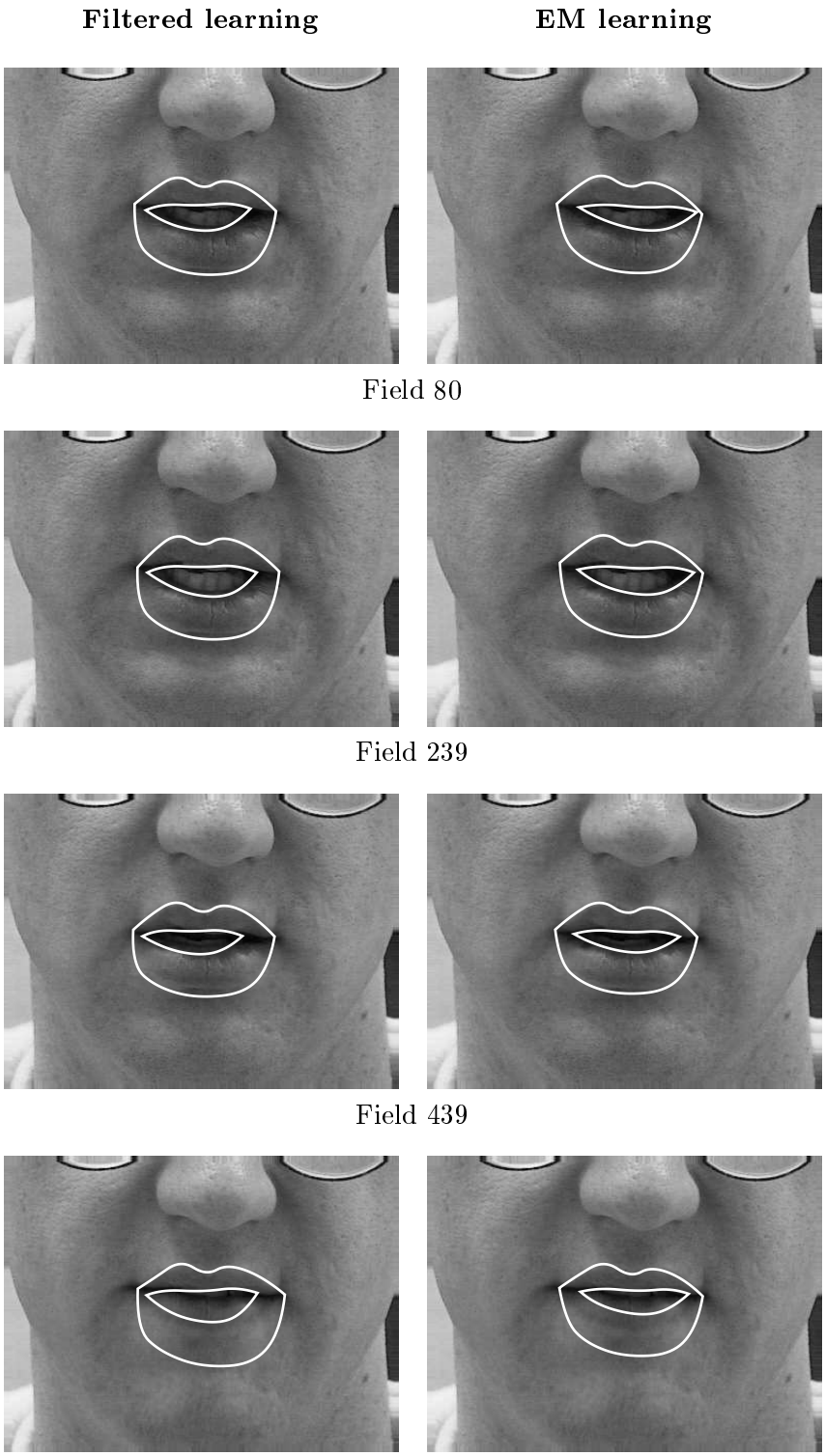


Figure 5.18: **Tracking snapshots at time-steps of greatest difference between the estimates produced by the two models.** *The inner-lip contour is more accurate when the EM-learnt model is used; see text for discussion.*

field 566). The motion undergone by the lips when speaking this word is significantly different to that seen in the rest of the sequence. The tracker is thus attempting to estimate atypical motion; although the word 'seventeen' was part of the training sequence, the rapid '–eve–' motion occupies a very small fraction of the overall sequence.

This is perhaps an unavoidable restriction of trying to use a single dynamical model for the whole range of lip motion produced by speech. A possibly better approach would be to use a filtering framework allowing more than one dynamical model, with switching between them. More specific models for the various 'visemes' could then be employed, and the switching times could also be informative from a speech-recognition point of view. Such a filtering framework is investigated in chapters 6 and 7.

5.2 Dynamics with Stationary Noise

Consider a dynamical process with stationary noise, i.e.,

$$\begin{aligned} x_t &= Ax_{t-1} + Bw_t; \\ y_t &= x_t + B_1 u_t, \end{aligned} \tag{5.1}$$

where only the y_t are observed. Suppose also that the ML estimate of A and B were to be found, treating the observed y_t as if they were the actual x_t. This section examines the effect which this (obviously erroneous) assumption has on the learnt system, and how the techniques already developed for learning systems from measurements can be applied to this problem.

5.2.1 Second-order Systems

Note that eqn (5.1) can be viewed as a standard measurement process with $H = 1$ and where the variance of the noise is $B_1 B_1^\top$. Therefore, the results of section 4.4 apply, and it is easy to apply EM to find the correct maximum-likelihood estimate of the dynamics A, B, and B_1. This estimate is to be compared, not with the 'filtered' estimate of previous sections, but with the estimate produced by treating the y_n as if they were the x_n.

Experiments with Synthetic Data For a synthetic second-order scalar system, the model becomes

$$\begin{aligned} x_t &= a_1 x_{t-1} + a_2 x_{t-2} + bw_t; \\ y_t &= x_t + b_1 u_t. \end{aligned}$$

For the following experiments, the driving noise of the underlying AR process will be fixed at $b = 0.01$, and a system (a_1, a_2) constructed which corresponds to a continuous system (when sampled at 50 Hz) with a damping constant β of 0.05 and an angular frequency ω of π (giving a frequency of 0.5 Hz). The parameter to be altered is the standard deviation of the stationary noise, b_1.

The results of running each algorithm a small number of times for various levels of stationary noise b_1 are given in table 5.3. Note that the naïve estimation procedure, that of treating the y_n as the x_n, only produces an estimate for b, since the presence of the stationary noise is not acknowledged.

True	**Estimate of system**		
b_1	β **(true** $\beta = 0.05$**)**	ω **(true** $\omega = 3.14$**)**	b **(true** $b = 0.01$**)**
0.001	0.0542 ± 0.01	3.1448 ± 0.02	0.0103 ± 0.00
0.003	0.0808 ± 0.02	3.1444 ± 0.01	0.0124 ± 0.00
0.010	0.3761 ± 0.08	3.1318 ± 0.01	0.0263 ± 0.00
0.030	2.5641 ± 0.37	1.5868 ± 0.87	0.0721 ± 0.00

(a) Naïve estimate of system

True	**Estimate of system**			
b_1	β **(true** $\beta = 0.05$**)**	ω **(true** $\omega = 3.14$**)**	b **(true** $b = 0.01$**)**	b_1
0.001	0.0500 ± 0.01	3.1447 ± 0.02	0.0099 ± 0.00	0.0012 ± 0.00
0.003	0.0519 ± 0.01	3.1438 ± 0.01	0.0100 ± 0.00	0.0030 ± 0.00
0.010	0.0517 ± 0.01	3.1414 ± 0.01	0.0100 ± 0.00	0.0100 ± 0.00
0.030	0.0497 ± 0.01	3.1474 ± 0.01	0.0100 ± 0.00	0.0300 ± 0.00

(b) EM estimate of system

Table 5.3: **EM accurately estimates a dynamical system with stationary noise; a more naïve method over-estimates the damping constant and driving noise.** *(a) The naïve estimate of the system parameters for various levels of additive noise, and (b) the EM estimate of the system parameters for the same sequences. In both cases, several runs were performed, and the (sample) mean and s.d. are given as '$\mu \pm \sigma$'. The true values of the parameters used to simulate the data were $\beta = 0.05$, $\omega = 3.14$, and $b = 0.01$. Each run used a sequence of length 16384, and 16 runs of each experiments were performed. The EM algorithm was run for 256 iterations, which was enough to produce convergence.*

It can be seen, considering the EM estimates first, that the estimates are consistently good, although there is much higher variance in the estimate of the damping constant than in the estimates of the other parameters. When the additive noise is low, ignoring it does not affect the naïve estimate too badly — the results are as good as the EM estimates. However, as the level of additive noise increases, the estimate of the damping constant in particular

becomes wildly inaccurate — too high by a factor of nearly 50 in the worst case shown here. This over-estimation of the damping constant was also observed in section 4.5.1.

5.2.2 Example with Real Data — Heartbeat Data

As an example of the application of the learning technique to real data, consider the time-series presented in figure 5.19. (These data were provided by Gary Jacob.) An ultrasound video sequence of a beating heart was tracked, and the second principal component is shown. (The first component contained mostly reparametrisation of the spline curve used for tracking.)

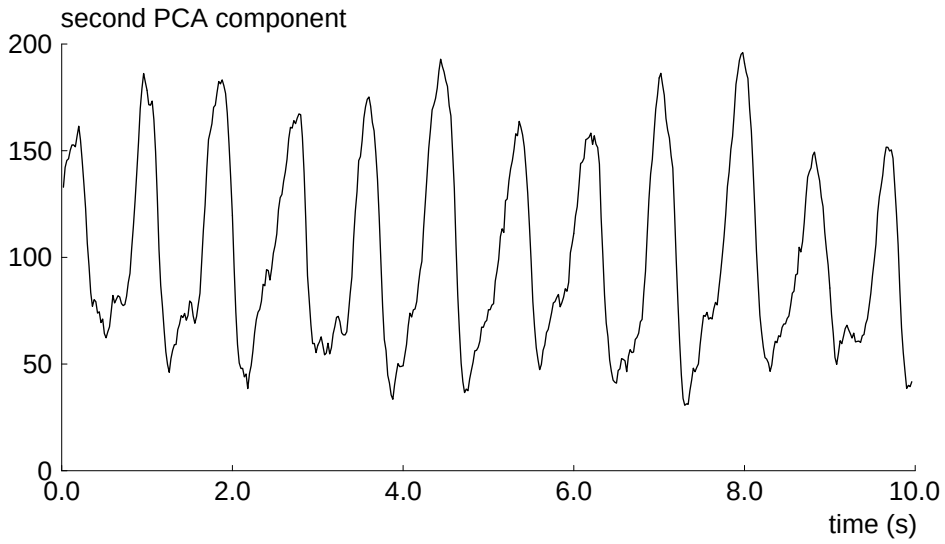


Figure 5.19: **Time-series of tracked heart data, for which a dynamical model with stationary noise will be learnt.** *A beating heart is tracked, and the second component of the PCA basis used is plotted here against time-step. (The first PCA component shows less periodic behaviour; it seems that it represents mostly reparametrisation of the contour.) It can be seen that the data are highly periodic, although not perfectly so.*

The Learnt Systems The naïve method of learning a dynamical system from these data was compared with the method based on EM presented here. The system learnt by the naïve method was as follows:

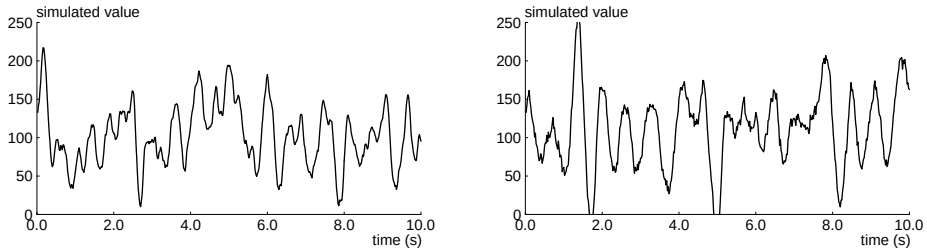
$$\bar{x} = 99.43; \qquad \beta = 4.646; \qquad \omega = 8.332; \qquad b = 4.680,$$

whereas that learnt using EM, allowing for additive stationary noise, was

$$\bar{x} = 100.0; \qquad \beta = 1.676; \qquad \omega = 8.924; \qquad b = 2.824; \qquad b_1 = 3.846.$$

As has been observed in other situations, the damping constant of the model learnt using EM is much smaller. Examination of the training sequence suggests that the true motion is indeed not particularly damped. The EM-learnt model is therefore a more accurate description of the behaviour of the system.

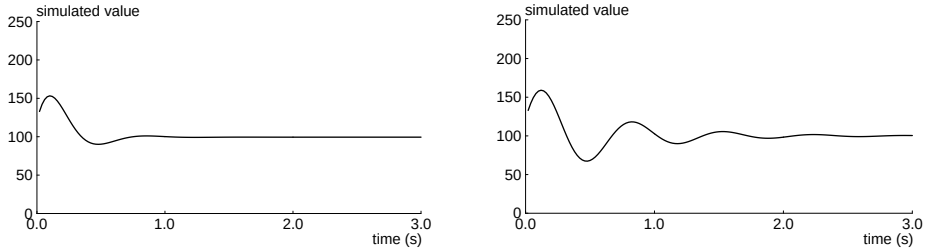
Simulating the Systems The stochastic models learnt can be simulated to help visualise the motion which each represents. Example simulations are shown in figure 5.20.



(a) Naïve learning (b) EM learning

Figure 5.20: **Simulations of learnt models show that the EM-learnt model more closely resembles the training data.** *The dynamical models learnt by the two methods are simulated using Gaussian driving noise. In (a), the system learnt by the naïve method; in (b), that learnt using the EM method. It can be seen that (b) resembles the original training data (in figure 5.19) more closely.*

It is also useful to look at a simulation of the learnt systems in the absence of driving noise — i.e., with b and b_1 set to zero. The resulting sequences indicate the predictive behaviour of a tracker using such a model. These graphs are shown in figure 5.21.



(a) Naïve learning (b) EM learning

Figure 5.21: **EM-learnt model shows better predictive behaviour.** *Without any driving noise, the predictive behaviour of the learn models can be seen. The effect of the larger damping constant of the naïvely learnt model in (a) can clearly be seen.*

The simulation using the EM-learnt model is perhaps slightly more convincing than that using the filtered-learnt model, but the difference is small. The predictive behaviour

displays a much more noticeable improvement when using the EM-learnt model. This would become important when fields are dropped during tracking; the accuracy of the prediction would then allow recovery of the tracked object after the dropped fields.

5.3 Discussion

The technique of Expectation-Maximisation has been successfully applied to the estimation problem of system identification, from a sequence of measurements, in a Kalman filter framework. Specifically, the method has been used to learn a dynamical model for an active-contour tracker. A general method for learning measurement noise was also described, although a more suitable method of estimating this parameter was used for the experiments. Experiments show that a system learnt using this EM-based method is superior to one learnt using existing methods.

This work suggests other applications of EM to problems related to learning models for contour-tracking. For example, the space of allowable deformations of the contour is presently specified independently of the dynamical model. A promising area for investigation is the possibility of learning this component of a contour tracker directly from the same sequence as that used to learn the dynamical model.

5.3.1 Learning Measurement Noise

In this section, the question is addressed of how the maximum-likelihood estimate of the measurement noise can be obtained, using EM. The theory was developed independently by the author, but is equivalent to (Shumway and Stoffer, 1982) as described in appendix A.2.

Maximum Likelihood Estimate of Parameters

Suppose a complete set of data is available — the true states $(x_1, x_2, \ldots, x_T)$ and the measurements $(z_1, z_2, \ldots, z_T)$. The problem of forming the maximum-likelihood estimate of the dynamical model parameters A and B decouples from the problem of estimating Σ, the measurement noise variance. These estimates are then as follows. The case of a first-order model is considered for simplicity; the extension to other model orders is straightforward.

$$\begin{aligned} A &= R_{10}R_{11}^{-1}; \\ BB^\top &= \frac{1}{T-1}\left(R_{00} - AR_{10} - R_{01}A^\top + AS_{11}A^\top\right); \\ \Sigma &= \frac{1}{T}\sum_{t=1}^{T}(z_t - Hx_t)(z_t - Hx_t)^\top. \end{aligned}$$

To derive these expressions, note that the problem is to maximise $p(X, Z \mid A, B, \Sigma)$, where $X = (x_1, x_2, \ldots, x_T)$ and $Z = (z_1, z_2, \ldots, z_T)$. Now

$$p(X, Z \mid A, B, \Sigma) = p(Z \mid X, A, B, \Sigma)\; p(X \mid A, B, \Sigma)$$

and

$$p(Z \mid X, A, B, \Sigma) = p(Z \mid X, \Sigma)$$

since, if X is given, then A and B provide no further information about Z (i.e., conditioned on X, Z is independent of A and B). Similarly,

$$p(X \mid A, B, \Sigma) = p(X \mid A, B)$$

as X is independent of Σ. Therefore,

$$p(X, Z \mid A, B, \Sigma) = p(Z \mid X, \Sigma)\; p(X \mid A, B)$$

and so

$$L(X, Z \mid A, B, \Sigma) = L(Z \mid X, \Sigma) + L(X \mid A, B).$$

The maximisation of $L(X, Z \mid A, B, \Sigma)$ can therefore be achieved by separately maximising the two likelihoods $L(Z \mid X, \Sigma)$ and $L(X \mid A, B)$. The solution to the second of these maximisations has already been described in previous sections; the derivation of the solution to the first is as follows.

$$\begin{aligned} p(Z \mid X, \Sigma) &= p(z_1, z_2, \ldots, z_T \mid x_1, x_2, \ldots x_T, \Sigma) \\ &= p_{v_1, v_2, \ldots, v_T; \Sigma}(z_1 - Hx_1, z_2 - Hx_2, \ldots, z_T - Hx_T) \\ &= \prod_{t=1}^{T} p_{v_t;\Sigma}(z_t - Hx_t) \end{aligned}$$

since the v_t are independent. Then

$$p_{v_t;\Sigma}(u) = \frac{1}{\rho}|\Sigma|^{-1/2} \exp\left[-\frac{1}{2}u^\top \Sigma^{-1} u\right]$$

and therefore (up to an additive constant)

$$L(Z \mid X, \Sigma) = -\frac{T}{2}\log|\Sigma| - \frac{1}{2}\sum_{t=1}^{T} u_i^\top \Sigma^{-1} u_i, \tag{5.2}$$

where $u_t = z_t - Hx_t$. Writing $u^\top \Sigma^{-1} u$ as $\mathrm{tr}(uu^\top \Sigma^{-1})$, this becomes

$$L = \frac{T}{2} \log |\Sigma^{-1}| - \frac{1}{2} \mathrm{tr} \left[\left(\sum_{t=1}^{T} u_i u_t^\top \right) \Sigma^{-1} \right].$$

Differentiating with respect to Σ^{-1} and setting the result equal to zero gives

$$T\Sigma^\top - \left(\sum_{t=1}^{T} u_t u_t^\top \right)^\top = 0$$

and the claimed result for Σ follows.

Finding the Expected Log-Likelihood

Next consider the expectation of this term of L given a previous estimate for the parameters. Only the second term of eqn (5.2) contains the random variables x_t:

$$\mathcal{E}\left[\mathrm{tr} \left(\left[\sum_{t=1}^{T} u_t u_t^\top \right] \Sigma^{-1} \right) \right] = \mathrm{tr} \left(\mathcal{E}\left[\sum_{t=1}^{T} u_t u_t^\top \right] \Sigma^{-1} \right)$$

and

$$\begin{aligned} \mathcal{E}\left[\sum_{t=1}^{T} u_t u_t^\top \right] = \sum_{t=1}^{T} z_t z_t^\top - H \left[\sum_{t=1}^{T} \mathcal{E}[x_t] z_t^\top \right] - \left[\sum_{t=1}^{T} z_t \mathcal{E}[x_t]^\top \right] H^\top \\ + H \left[\sum_{t=1}^{T} \left(\mathcal{E}[x_t]\mathcal{E}[x_t]^\top + \mathrm{Var}[x_t] \right) \right] H^\top. \end{aligned} \tag{5.3}$$

$\mathcal{E}[x_t]$ and $\mathrm{Var}[x_t]$ have already been calculated (they are the solutions to the smoothing problem), and so the updated estimate for Σ is given by

$$\begin{aligned} \Sigma = \frac{1}{T} \Bigg(\sum_{t=1}^{T} z_t z_t^\top - H \left[\sum_{t=1}^{T} \mathcal{E}[x_t] z_t^\top \right] - \left[\sum_{t=1}^{T} z_t \mathcal{E}[x_t]^\top \right] H^\top \\ + H \left[\sum_{t=1}^{T} \left(\mathcal{E}[x_t]\mathcal{E}[x_t]^\top + \mathrm{Var}[x_t] \right) \right] H^\top \Bigg). \end{aligned}$$

Therefore, the results of section 4.4.2 can be used to estimate the measurement noise as well as the dynamical model from a training measurement sequence.

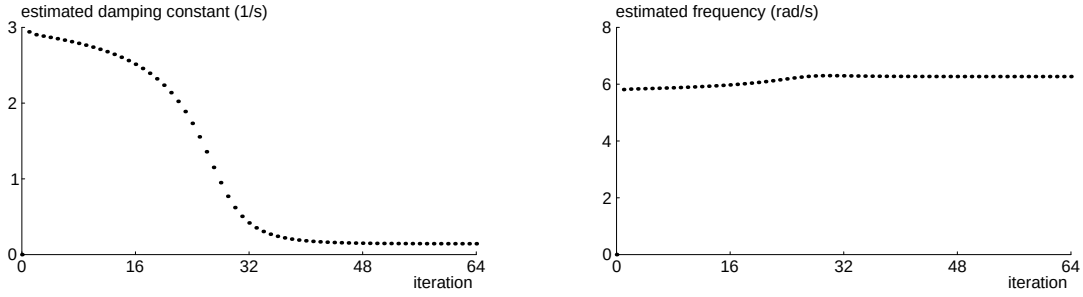
Experiments with Synthetic Data

The behaviour of this algorithm with synthetic data is now briefly examined. A typical scalar second-order discrete system, whose mode corresponds to a continuous one with $\beta = 0.1$ and $\omega = 6.28$, was simulated with various levels of measurement noise. The results of performing the algorithm described on these data sets are summarised in table 5.4.

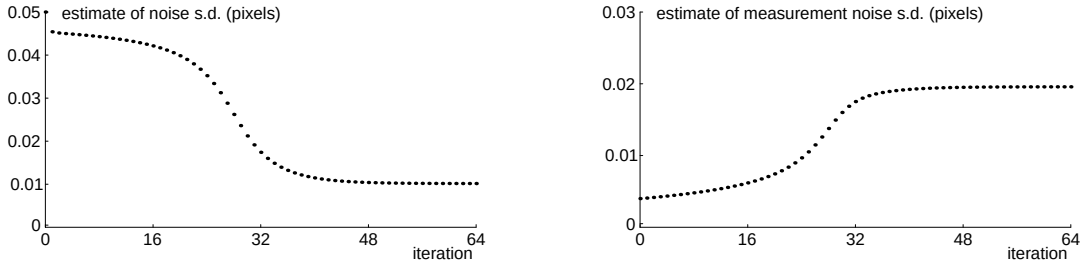
True	**EM estimate of system**			
σ	β	ω	b	σ
0.005	0.1203	6.290	0.00977	0.00518
0.010	0.0859	6.286	0.00997	0.00986
0.020	0.1445	6.267	0.01019	0.01957
0.050	0.1410	6.308	0.01014	0.05037

Table 5.4: **The EM algorithm can be extended to simultaneously learn measurement noise.** *The learning algorithm presented was exercised on synthetic systems with* $\beta = 0.1$, $\omega = 6.28$ *and* $b = 0.01$ *and various values of measurement noise* σ. *The estimates are mostly good, although the damping constant* β *is, in most cases, over-estimated. The sequence was of length 8192, and 256 iterations of the algorithm were performed — the convergence is slower in this case than the others presented.*

The convergence behaviour for one of these examples is shown in figure 5.22. When compared with figure 4.5, the convergence is much more computation intensive, especially considering that the sequence used to produce figure 5.22 was much longer than that used to produce figure 4.5. A qualitative explanation for this phenomenon is that the effects of process noise and measurement noise on the measurement sequence are similar, so distinguishing between them is a slow process.



(a) Deterministic part of dynamical model



(b) Process noise

(c) Measurement noise

Figure 5.22: **Convergence of system estimates when learning measurement noise.** *The graphs show the convergence behaviour of the estimates of (a) the deterministic part of the dynamical model, (b) the stochastic part of the dynamical model, and (c) the measurement noise. The synthetic sequence had* $\beta = 0.1$, $\omega = 6.28$, $b = 0.01$, *and* $\sigma = 0.02$, *and was of length 8192.*

Chapter 6

Dynamical Models for Condensation

6.1 Introduction

The Condensation (standing for 'conditional density propagation') algorithm of Isard and Blake (1996; 1998a) is a filtering (and smoothing) algorithm which addresses and overcomes some of the limitations of the Kalman filter. The chief problem with the application of a Kalman filter to visual tracking is that the assumption that the conditional observation density $p(z \mid x)$ is Gaussian is rarely valid. In clutter, there will be many competing features, causing multi-modality of $p(z \mid x)$. The recursive Bayesian filtering problem referred to in chapter 1 then ceases to be analytically tractable, and various approximate techniques have been developed.

Bucy (1969) proposes numerical integration of Bayes' rule, which is very general but suffers from prohibitive computational cost once the dimensionality of the problem rises above 1 or 2. Mixtures of Gaussians are used in pattern recognition, as described by Duda and Hart (1973) or Bishop (1995). This idea is extended by Sorenson and Alspach (1971) to propagate densities through time, fusing similar Gaussians and discarding negligible ones in order to keep the total number of mixture components tractable. When the dynamical model or the measurement model ceases to be linear, the Extended Kalman Filter (Jacobs, 1993) can be employed. It operates by forming linear approximations, and ultimately approximates the state distribution as a Gaussian. It has successfully been used for filtering applications in vision, e.g., (Harris, 1992).

The problem of static image analysis has been addressed using various random sampling techniques, including the work of Geman and Geman (1984) on pixel-based methods for image restoration using Markov random fields. The method of 'factored sampling' has been

applied, again to static images, by Grenander et al. (1991); the images considered here are of hands. An encoding of hand shapes is used, as opposed to a pixel-based system. Both of these approaches use iterative simulation, as does the work of, for example, Ripley and Sutherland (1990) on images of galaxies.

The idea of factored sampling has been extended to the representation and propagation of densities over time. This stochastic Bayesian filtering framework was developed by Gordon et al. (1993), who termed it a 'bootstrap filter' and Kitagawa (1996), who called it a 'Monte-Carlo filter' and looked only at the one-dimensional case. The Condensation algorithm discussed in this chapter is effectively the application of these ideas to the field of visual tracking.

6.1.1 The Condensation Tracking Algorithm

The Kalman filter makes several assumptions about the dynamical system and observation process. The dynamical model must be linear, with Gaussian process noise, and the measurement model must also be linear with Gaussian noise. Within these constraints, the filter produces an analytical solution to the problem of finding the posterior densities $p(\mathbf{x}_t \mid \mathcal{Z}_1^t)$ given a sequence $\mathcal{Z}_1^t = (z_1, \ldots, z_t)$ of measurements in terms of its mean and variance — enough to completely specify the Gaussian density which results.

In some tracking applications, these assumptions are reasonable, and the Kalman filter can be used to good effect, as seen in previous chapters. However, in the presence of *clutter*, the assumption that the posterior remains Gaussian fails. In particular, the true posterior may easily have more than one peak, representing competing plausible configurations of the tracked object. The Condensation algorithm addresses these issues by adopting a more general representation of the various distributions involved (including the desired posterior). Other benefits arise almost incidentally — the requirements that the dynamical model be linear and contain only Gaussian noise are both discarded; they are replaced with much less severe restrictions.

The essence is to represent a distribution, such as $p(\mathbf{x}_t \mid \mathcal{Z}_1^t)$, by means of a *weighted sample set*. This is a set $\{(\mathbf{x}^{(n)}, \pi^{(n)}), 1 \leq n \leq N\}$ of pairs which can be said to represent a distribution $p(\mathbf{x})$ in the following sense. The process of selecting one of the $\mathbf{x}^{(n)}$ with probability proportional to $\pi^{(n)}$ is 'a good approximation' to drawing from $p(\mathbf{x})$, in a sense which can be made precise. In the limit as $N \to \infty$, the approximation becomes exact.

The Condensation algorithm then propagates such a sample set forwards through time. It is a recursive algorithm, constructing a sample-set representation of $p(\mathbf{x}_t \mid \mathcal{Z}_1^t)$ in terms

of that of $p(\mathbf{x}_{t-1} \mid \mathcal{Z}_1^{t-1})$. The algorithm is described in figure 6.1.

Iterate for $t = 1, 2, \ldots, T$.

Construct the sample-set

$$\{(\mathbf{x}_t^{(n)}, \pi_t^{(n)}), n = 1, 2, \ldots, N\}$$

for time t.

For each n:

1. **Choose** (with replacement) a 'base' sample $m \in \{1, 2, \ldots, N\}$ with probability $\pi_{t-1}^{(m)}$.
2. **Predict** by sampling from
$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(m)})$$
to choose $\mathbf{x}_t^{(n)}$.
3. **Compute observation weights** $\pi_t^{(n)}$ are computed from the observation density, evaluated for the current observations $\mathbf{z}_t$:
$$\pi_t^{(n)} = p(\mathbf{z}_t \mid \mathbf{x}_t = \mathbf{x}_t^{(n)}),$$
then normalise multiplicatively so that $\sum_n \pi_t^{(n)} = 1$.

Figure 6.1: **The Condensation algorithm for forward propagation.** *A sample-set representation of the conditional density $p(\mathbf{x}_t \mid \mathcal{Z}_1^t)$ is propagated through time t. This presentation includes the simplifying assumption that the prediction density is first order (so that $p(\mathbf{x}_t \mid \mathcal{X}_1^{t-1}) = p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$), which need not be the case in general. The generalisation to higher model orders will be made explicit below.*

Initialisation

The algorithm presented in figure 6.1 requires an initial sample-set to represent the prior distribution $p(\mathbf{x}_0)$. There are (at least) two ways of specifying this distribution in practice, depending on how much information is available for the problem in question.

Known Starting Configuration If the configuration $\mathbf{x}_0$ of the target object is known at $t = 0$, this certainty can be easily represented in the sample set:

$$\begin{aligned} \mathbf{x}_0^{(n)} &= \mathbf{x}_0 \qquad \text{for } 1 \leq n \leq N; \\ \pi_0^{(n)} &= \frac{1}{N} \qquad \text{for } 1 \leq n \leq N. \end{aligned}$$

(The choice $\pi_0^{(1)} = 1$ and $\pi_0^{(n)} = 0$ for $n > 1$ gives the same result.)

Steady-state Distribution If the only prior knowledge of the starting configuration of the object comes from the dynamical model, this can be used as follows. Iterating the dynamical model will result in a steady-state distribution for the state, provided that the model is stable. Drawing N samples from this distribution and giving each a weight of $1/N$ will represent such a prior.

6.1.2 Higher-order AR Models

The algorithm shown in figure 6.1 used a first-order auto-regressive process for the prediction model for presentational simplicity, but this is not an inherent restriction of Condensation. In this section, the (fairly simple) extension to higher-order AR models is presented.

Suppose the dynamical model is of order K, so that

$$\mathbf{x}_t = \sum_{k=1}^{K} A_k \mathbf{x}_{t-k} + \mathbf{d} + B\mathbf{w}_t,$$

for (matrix) coefficients A_k, offset $\mathbf{d}$, and zero-mean, unit Gaussian noise vectors $\mathbf{w}_t$ shaped by the matrix B. To incorporate this into Condensation, each member $(\pi_t^{(n)}, \mathbf{x}_t^{(n)})$ of the sample set is replaced with an element carrying enough history to perform prediction. The sample site $\mathbf{x}_t^{(n)}$ becomes the ordered K-tuple $(\mathbf{x}_{(t-K+1)\,|\,t}^{(n)}, \ldots, \mathbf{x}_{t\,|\,t}^{(n)})$.

The prediction step from a chosen base sample m then consists of drawing a sample $\mathbf{x}_{t\,|\,t}^{(n)}$ from the distribution with density

$$p(\mathbf{x}_t \,|\, \mathbf{x}_{(t-K)\,|\,(t-1)}^{(m)}, \ldots, \mathbf{x}_{(t-1)\,|\,(t-1)}^{(m)}),$$

which in the case of the Gaussian AR process above, is accomplished by drawing a $\mathbf{w}$ from a suitably-dimensioned $N(0,1)$ and computing

$$\mathbf{x}_{t\,|\,t}^{(n)} = \sum_{k=1}^{K} A_k \mathbf{x}_{(t-k)|(t-1)}^{(m)} + \mathbf{d} + B\mathbf{w}.$$

The other addition to the algorithm is the computation of the history for the newly-chosen sample. This is a matter of copying the relevant portion of the history from the base sample m:

$$\mathbf{x}_{(t-k)\,|\,t}^{(n)} = \mathbf{x}_{(t-k)\,|\,(t-1)}^{(m)}, \qquad \text{for } 1 \le k \le K-1.$$

6.1.3 Multi-mode Dynamics

The Kalman filter requires that the predictive model for the object's motion and deformation be governed by a linear process with Gaussian noise. For many types of motion, this is adequate, but in some cases it is too restrictive. For instance, the hand-tracking example of (Isard and Blake, 1998c) requires three modes to model (and classify) the distinct motion-types of the hand, and the results of section 5.1.3 suggested that a multi-mode model might better represent the different types of lip motion used during speech.

With this extension, the state consists of both a continuous part, $\mathbf{x}_t$, and a label for the discrete state, y_t. These are combined into a single state vector $\mathbf{X}_t$:

$$\mathbf{X}_t = \begin{pmatrix} \mathbf{x}_t \\ y_t \end{pmatrix}.$$

Each discrete state y is assumed to represent an auto-regressive model of order K; if $y_t = y$, then

$$\mathbf{x}_t = \sum_{k=1}^{K} A_k^y \mathbf{x}_{t-k} + \mathbf{d}^y + B^y \mathbf{w}_t. \tag{6.1}$$

(The orders for different models can be different with the only increased difficulty being largely notational.) The transitions between states are assumed to be a first-order Markov process with transition matrix M:

$$P(y_t = y' \,|\, y_{t-1} = y) = M_{y,y'},$$

meaning that the joint prediction distribution can be written as

$$p(\mathbf{x}_t, y_t \,|\, \mathbf{x}_{t-1}, y_{t-1}, \ldots, \mathbf{x}_{t-K}, y_{t-K}) = p_{y_t}(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-K}) M_{y_{t-1} y_t}.$$

The distribution $p_y(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-K})$ is derived from eqn (6.1), giving the following explicit form for $p(\mathbf{X}_t \,|\, \mathcal{X}_0^{t-1})$.

$$p(\mathbf{X}_t | \mathcal{X}_0^{t-1}) = (2\pi |B^{y_t}|)^{-D/2} \exp\left[-\frac{1}{2} \left| (B^{y_t})^{-1} \left(\mathbf{x}_t - \sum_{k=1}^{K} A_k^{y_t} \mathbf{x}_{t-k} - \mathbf{d}^{y_t} \right) \right|^2 \right] M_{y_{t-1}, y_t}. \tag{6.2}$$

6.2 EM Learning for Condensation

Again, the learning problem within the framework of Condensation is an example of a parameter-estimation problem with incomplete data, so a natural approach is to use the EM algorithm. The details of the application of EM to this problem are now developed.

6.2.1 Maximisation Step

The 'M' step of the EM algorithm requires the calculation of the ML estimate of the model parameters based on complete data. In this case, this reduces to knowledge of a given sequence $\mathbf{X}_1^*, \ldots, \mathbf{X}_T^*$ of known state vectors. (In general, 'complete data' will include the observed data $\mathbf{z}$, but in this case, the measurements add nothing to the information provided by the true states.)

This section develops the results which will be used to make these ML calculations, first for a single arbitrary-order AR process, and then for a multi-model process of the type introduced in section 6.1.3.

Single-mode Models

Here the state sequence has only a continuous part $\mathbf{x}_1^*, \ldots, \mathbf{x}_T^*$. In the case of a single dynamical model, finding the maximum-likelihood estimate of the model parameters is a fairly simple extension of the derivation in section 4.5 for second-order systems.

The predictive system now in question is

$$\mathbf{x}_t = \sum_{k=1}^{K} A_k \mathbf{x}_{t-k} + \mathbf{d} + B w_t.$$

The full derivation is given in appendix A.3; the result is that the prediction matrix

$$A = \begin{pmatrix} A_1 & A_2 & \cdots & A_K \end{pmatrix}$$

has a maximum-likelihood estimate satisfying

$$AR^0 = \mathbf{R}_0^0,$$

where the matrices R^0 and $\mathbf{R}_0^0$ are defined as

$$R^0 = \begin{pmatrix} R_{11}^0 & R_{12}^0 & \cdots & R_{1,K}^0 \\ R_{21}^0 & R_{22}^0 & \cdots & R_{2,K}^0 \\ \vdots & \vdots & \ddots & \vdots \\ R_{K,1}^0 & R_{K,2}^0 & \cdots & R_{K,K}^0 \end{pmatrix}; \qquad \mathbf{R}_0^0 = \begin{pmatrix} R_{01}^0 & R_{02}^0 & \cdots & R_{0,K}^0 \end{pmatrix}$$

and the modified second-order moments R_{ij}^0 are

$$R_{ij}^0 = R_{ij} - \frac{1}{T-K} R_i R_j^\top,$$

where

$$R_{ij} = \sum_{t=K+1}^{T} x_{t-i}^* (x_{t-j}^*)^\top; \qquad R_i = \sum_{t=K+1}^{T} x_{t-i}^*.$$

The offset $\mathbf{d}$ is given by

$$d = \frac{1}{T-K}(R_0 - A\mathbf{R}),$$

where

$$\mathbf{R} = \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_K \end{pmatrix}.$$

Finally, the variance C of the noise process is given by

$$C = \frac{1}{T-K}\left(R_{00}^0 - A(\mathbf{R}_0^0)^\top\right).$$

Multi-mode Models

If the dynamical system in question has more than one discrete state, the full state vector $\mathbf{X}_t$ must be considered. Again, the details of the derivation are deferred, to appendix A.5. The result is that each mode y has a set of parameters estimated exactly as for a single-mode system, but with the moments restricted to those time-steps during which the system was in mode y.

Specifically, parameters A^y, $\mathbf{d}^y$ and C^y are learnt from block matrices $(R^0)^y$, $(\mathbf{R}_0^0)^y$ and $(\mathbf{R}_0)^y$ formed from moments matrices $(R_{ij}^0)^y$ and R_i^y given by

$$(R_{ij}^0)^y = \sum_{y_t=y} \mathbf{x}_{t-i}^*(\mathbf{x}_{t-j}^*)^\top - \frac{1}{T_y - K} R_i^y (R_j^y)^\top,$$

where

$$R_i^y = \sum_{y_t^*=y} \mathbf{x}_{t-i}^*; \tag{6.3}$$

$$T_y = \sharp\{t : y_t^* = y\},$$

and the set-size operator $\sharp\{\}$ is defined such that for a finite set A, $\sharp\{A\}$ is the number of elements in A. In this case, then, T_y can also be written using

$$T_y = \sum_{t:y_t^*=y} 1.$$

The maximum-likelihood estimate for the transition matrix M is given by

$$M_{yy'} = \frac{T_{yy'}}{\sum_{y'} T_{yy'}},$$

where $T_{yy'}$ are the transition counts:

$$T_{yy'} = \sharp\{t : y_{t-1}^* = y, y_t^* = y'\}.$$

6.2.2 Expectation Step

The expected value of the log-likelihood L is sought; in full,

$$\mathcal{E}[L] = \mathcal{E}_x[L(z, x \,|\, \varphi) \,|\, z, \varphi_i].$$

In appendix A.5 it is shown that L is linearly dependent on the moments $(R_{ij})^y$ and $(R_i)^y$, and the transition counts $T_{yy'}$. Therefore, in a similar fashion to the derivation given in section 4.4 for the case of the Kalman filter, their expected values are sought, for example

$$\mathcal{E}[T_{yy'}] = \mathcal{E}_{\mathcal{X}_1^T}[T_{yy'} \,|\, \mathcal{Z}_1^T, \varphi_i],$$

where $\mathcal{X}$ is the complete sequence of states:

$$\mathcal{X}_1^T = (\mathbf{X}_1, \ldots, \mathbf{X}_T).$$

Therefore, information about the distribution $p(\mathcal{X}_1^T \,|\, \mathcal{Z}_1^T)$ is sought.

Smoothing using Condensation

This problem of finding the distribution $p(\mathcal{X}_1^T \,|\, \mathcal{Z}_1^T)$ is the same one as was encountered in section 4.4 — smoothing. A solution to the smoothing problem within the Condensation framework was presented by Isard (1998d) for the case of a single-mode system, with the primary aim of obtaining better state estimates. The extension to multi-mode systems and the application to learning a dynamical model using EM given below is original joint work of the author with Andrew Blake. The implementation and experiments are the original work of the author.

In a similar fashion to the procedure for Kalman smoothing, a second, backward, pass is taken over the sequence. This replaces the sample weights $\pi_t^{(n)}$ with 'smoothing' weights $\psi_t^{(n)}$, such that the weighted sample set, using the original sample sites with new weights,

$$\{(\mathbf{x}_{(t-K)\,|\,t}^{(n)}, \ldots, \mathbf{x}_{t\,|\,t}^{(n)}), \psi_t^{(n)}\}$$

represents the required distribution of the states given all the data. Note that the history tuple has been extended backwards one further time-step — $\mathbf{x}_{(t-K)\,|\,t}^{(n)}$ is stored in addition. The reason for this will become apparent shortly.

The extension to the multi-mode case, producing a sample set representation

$$\{(\mathbf{X}_{(t-K)\,|\,t}^{(n)}, \ldots, \mathbf{X}_{t\,|\,t}^{(n)}), \psi_t^{(n)}\}$$

of the distribution for the combined continuous/discrete state vector sequence $\mathbf{X}_t$, is conceptually reasonably simple and is shown in figure 6.2.

Initialise at $t = T$:

$$\psi_T^{(n)} = \pi_T^{(n)}, \qquad \text{for } n = 1, \ldots, N.$$

Compute smoothing-weights at time t, in terms of those at time $t+1$, as follows:

1. **Prediction probabilities**

$$\alpha_t^{(m,n)} = p\big[\mathbf{X}_{t+1} = \mathbf{X}_{(t+1)\,|\,(t+1)}^{(m)} \,|\, (\mathbf{X}_{(t-k)\,|\,t} = \mathbf{X}_{(t-k)\,|\,t}^{(n)},\ k = 1, 2, \ldots, K)\big],$$

   calculated for a multi-modal ARP using eqn (6.2).

2. **Correction factors:**

$$\gamma_t^{(m)} = \sum_{n=1}^{N} \pi_t^{(n)} \alpha_t^{(m,n)} \qquad \text{for } m = 1, \ldots, N.$$

3. **Backward variables:**

$$\delta_t^{(n)} = \sum_{m=1}^{N} \psi_{t+1}^{(m)} \alpha_t^{(m,n)} / \gamma_t^{(m)} \qquad \text{for } n = 1, \ldots, N.$$

4. **Smoothing weights:**

$$\psi_t^{(n)} = \pi_t^{(n)} \delta_t^{(n)} \qquad \text{for } n = 1, \ldots, N,$$

   then normalise multiplicatively so that $\sum_n \psi_t^{(n)} = 1$.

Figure 6.2: **The Condensation smoothing algorithm.** *From a sample-set representation of the filtered distributions, a recursive method (starting from the end of the sequence) is used to compute a new set of weights $\psi_t^{(n)}$ which form a sample-set representation of the smoothed distributions.*

Moments and Transition Counts

After smoothing, the sample set

$$\{(\mathbf{X}^{(n)}_{(t-K)\,|\,t}, \ldots, \mathbf{X}^{(n)}_{t\,|\,t}), \psi^{(n)}_t\}$$

approximates (with the approximation becoming exact in the limit that $N \to \infty$) the distribution

$$p(\mathbf{X}_{t-K}, \ldots, \mathbf{X}_t \,|\, \mathcal{Z}_1^T),$$

which will allow the calculation of the required expected values of the moments. Recall that for any function f of the state $\mathbf{X}_t$, the expectation of $f(\mathbf{X}_t)$ can be approximated using the sample-set by

$$\mathcal{E}[f(\mathbf{X}_t)] \approx \sum_n \psi^{(n)}_t f(\mathbf{X}^{(n)}_t).$$

This information is what is required to calculate (an approximation to) the expected values of the moments. Take, for instance, $\mathcal{E}[R^y_i]$. From eqn (6.3):

$$\begin{aligned}
\mathcal{E}[R^y_i] &= \mathcal{E}\left[\sum_{y^*_t = y} \mathbf{x}^*_{t-i}\right] \\
&= \mathcal{E}\left[\sum_{t=K+1}^{T} \delta_{y^*_t, y} \mathbf{x}^*_{t-i}\right] \\
&= \sum_{t=K+1}^{T} \mathcal{E}\left[\delta_{y^*_t, y} \mathbf{x}^*_{t-i}\right],
\end{aligned}$$

where δ_{uv} is the Kronecker delta function:

$$\delta_{uv} = \begin{cases} 0, & \text{if } u \neq v; \\ 1, & \text{if } u = v. \end{cases}$$

The individual terms in the sum are expected values of functions of the combined discrete/continuous state vector $\mathbf{X}$, so can be approximated by means of the sample set:

$$\begin{aligned}
\mathcal{E}\left[\delta_{y^*_t, y} \mathbf{x}^*_{t-i}\right] &\approx \sum_{n=1}^{N} \psi^{(n)}_t \delta_{y^{(n)}_{t\,|\,t}, y} \mathbf{x}^{(n)}_{(t-i)\,|\,t} \\
&= \sum_{n=1}^{N} \psi^{(n)}_t \chi_y\left(y^{(n)}_{t\,|\,t}\right) \mathbf{x}^{(n)}_{(t-i)\,|\,t},
\end{aligned}$$

defining the indicator function

$$\chi_u(y) = \begin{cases} 0, & \text{if } y \neq u; \\ 1, & \text{if } y = u. \end{cases}$$

Expected values of the second-order moments and the transition counts can be computed in the same fashion, leading to the following expressions.

$$\mathcal{E}[R_{ij}^y] = \sum_{t=K+1}^{T} \sum_{n=1}^{N} \psi_t^{(n)} \chi_y \left(y_{t\,|\,t}^{(n)} \right) \mathbf{x}_{t-i\,|\,t}^{(n)} \left(\mathbf{x}_{t-j\,|\,t}^{(n)} \right)^\top ; \tag{6.4}$$

$$\mathcal{E}[R_i^y] = \sum_{t=K+1}^{T} \sum_{n=1}^{N} \psi_t^{(n)} \chi_y \left(y_{t\,|\,t}^{(n)} \right) \mathbf{x}_{t-i\,|\,t}^{(n)}; \tag{6.5}$$

$$\mathcal{E}[T_y] = \sum_{t=K+1}^{T} \sum_{n=1}^{N} \psi_t^{(n)} \chi_y \left(y_{t\,|\,t}^{(n)} \right); \tag{6.6}$$

$$\mathcal{E}[T_{yy'}] = \sum_{t=2}^{T} \sum_{n=1}^{N} \psi_t^{(n)} \chi_{y,y'} \left(y_{(t-1)\,|\,t}^{(n)}, y_{t\,|\,t}^{(n)} \right), \tag{6.7}$$

where

$$\chi_{uv}(y, y') = \begin{cases} 0, & \text{if } y \neq u \text{ or } y' \neq v; \\ 1, & \text{if } y = u \text{ and } y' = v. \end{cases}$$

As noted in appendix A.5, the log-likelihood is linear in these quantities, so the above equations allow the completion of the Expectation step of the EM algorithm. Note that it is here that the need to store $\mathbf{X}_{(t-K)\,|\,t}$ becomes apparent — the expected value of the moment R_{0K} requires it. For prediction and filtering, histories are only required for the previous $K-1$ timesteps, i.e., as far back as $\mathbf{X}_{(t-K+1)\,|\,t}$.

6.2.3 Summary of Algorithm

The essence of the learning algorithm is then similar to that for learning within the framework of Kalman filtering. It is summarised in figure 6.3.

6.3 Condensation-EM in Practice

This section describes some experiments which were performed to gain some intuition about the behaviour of the Condensation-EM algorithm. They are the original work of the author. Since an approximation is involved in the iterative step, some experimental indications of the effect this has on the algorithm are desired.

6.3.1 Simple Motion — a Juggled Ball

The example used will be of learning the parameters for a constant-acceleration model describing the motion of a ball being juggled. Typical frames from the sequence in question are shown in figure 6.4.

There are several different phases of motion for the ball, which may be named as follows.

(1) Choose suitable values for the starting dynamics; constant-velocity has proved quite versatile if the noise is set high enough to allow tracking. Also set up initial conditions for the first time-step;

(2) Run the Condensation tracker on the training image sequence, producing a sample-set representation of the distributions $p(X_t \mid \mathcal{Z}_1^t)$;

(3) Run the Condensation smoothing algorithm, producing modified sample weights to give a representation of the smoothed distributions $p(X_t \mid \mathcal{Z}_1^T)$;

(4) Using the results of the smoothing, find the expected values of the moments and transition counts;

(5) From these expected values, estimate the dynamical system;

(6) Using the system derived in the previous step, go back to step (2).

Figure 6.3: **The EM algorithm for finding the ML estimate of a dynamical model within the framework of Condensation.** *This should be compared to the algorithm given in figure 4.3 for the Kalman filtering case. The basic steps are the same; the chief difference is in the 'E' step — the calculation of expected values of the statistics on which the log-likelihood depends.*

Figure 6.4: **Frames from the juggling sequence.** *The sequences from which these frames are taken will be used in EM-based learning experiments within the Condensation framework. Some clutter can be seen in the background, although the balls are orange and so colour information can be used to improve measurements.*

- Ballistic — motion under gravity, when the ball is in the air.
- Catch — the ball is undergoing rapid deceleration as it is caught.
- Carry — the ball is being carried towards the centre of the body in preparation for being launched into the air.
- Throw — the ball undergoes rapid upwards acceleration as it is thrown to the other hand.

The state of the object can be described by a two-dimensional space, that of translation. (The small deformations caused by catching and throwing the ball are ignored; also, it is a reasonable approximation that the ball moves in a plane perpendicular to the line of sight of the camera, so the image of the ball does not significantly change size.)

The 'ballistic' motion will be examined here. It has the advantage of being known — the 'ground truth' model for a ball moving under gravity is constant downwards acceleration of amplitude very close to g (the balls were quite light, and so air resistance may have an effect).

Other Tracking Details The image sequence used for tracking was recorded in colour, so advantage was taken of the additional information available. A Fisher axis (Duda and Hart, 1973) was learnt from example images, and the image colours along the search lines projected onto this axis. After this projection, a one-dimensional feature search was performed on the resulting data, and the strongest edge feature used to calculate the observation density. The noise of this process was estimated using the same method as described in section 5.1.2. This measurement process was used for all of the following experiments on the juggling sequences.

The tracker requires initial conditions; these were specified manually for these experiments, although work has been done (MacCormick and Blake, 1998) on automatically locating objects in static images with a view to using the output as a starting point for tracking.

Units for Dynamical Parameters For the sake of interpretation of results, it is important to consider the units for the various parameters of the dynamical model. For a constant-acceleration model, the interpretation of the offset parameter $\mathbf{d}$ is straightforward; an object moving with continuous dynamics

$$\ddot{\mathbf{x}} = \mathbf{a}$$

obeys the recurrence relation

$$\mathbf{x}(t) = 2\mathbf{x}(t-\Delta) - \mathbf{x}(t-2\Delta) + \mathbf{d},$$

for a time-step of size Δ, i.e.,

$$\mathbf{x}_t = 2\mathbf{x}_{t-1} - \mathbf{x}_{t-2} + \mathbf{d},$$

where the relation between $\mathbf{a}$ and $\mathbf{d}$ is found to be

$$\mathbf{a} = \frac{\mathbf{d}}{\Delta^2}.$$

The conversion between pixels and metres must be included in the final expression, which is fairly simple in this case as the ball is moving approximately in a plane perpendicular to the line of sight of the camera. Measurements yield a value of $0.00227\,\mathrm{m/pixel}$, which, together with the time-step value of $\Delta = 0.02\,\mathrm{s}$, gives the conversion to be

$$\frac{\mathbf{a}}{\mathrm{ms}^{-2}} = 5.68\frac{\mathbf{d}}{\mathrm{pixels}}.$$

For the noise parameter C, some interpretation of the eigenvalues (expressed in pixels^2) is required. For a system with a steady-state variance P_∞, this provides one natural interpretation (after converting from pixels to metres). However, a constant-acceleration system has no steady-state variance, so this approach cannot be used. For the precise problem under consideration, an alternative is to evaluate the system's prior variance (i.e., without measurements) after an appropriate number of time-steps, with a zero starting variance. Here 'appropriate' will be a typical duration of the motion in question.

The system's prior variance behaves as follows. Augment the state in the usual fashion:

$$X_t = \begin{pmatrix} x_t \\ x_{t-1} \end{pmatrix}; \qquad A = \begin{pmatrix} 2 & -1 \\ 0 & 1 \end{pmatrix};$$

then the augmented variance matrix P_t obeys

$$P_t = AP_{t-1}A^\top + \begin{pmatrix} C & 0 \\ 0 & 0 \end{pmatrix}; \qquad P_0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

This can simply be iterated to find the upper-left corner of P_τ (where τ is the 'appropriate' time mentioned), which is the variance of the (non-augmented) state x_τ.

Also, results will usually be presented in terms of the square roots of the eigenvalues, to give a final value measured in metres (after multiplication by the pixels-to-metres factor described above).

Behaviour of EM Algorithm — Learning a Single Model

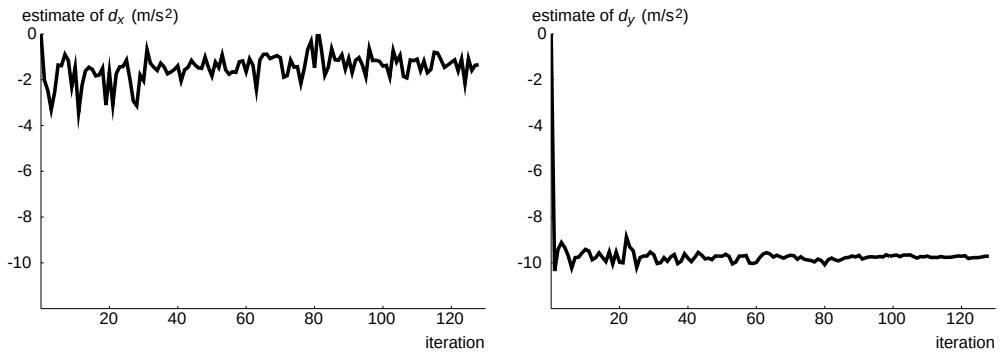
Consider first the problem of learning a model for the ballistic section of the motion. Learning just one model allows some simplification of equations (6.4)–(6.7). For example, the second-order moments can be estimated by means of

$$\mathcal{E}[R_{ij}] \approx \sum_{n=1}^{N} \sum_{t=K+1}^{T} \psi_t^{(n)} \mathbf{x}_{(t-i)\,|\,t}^{(n)} \left(\mathbf{x}_{(t-j)|t}^{(n)}\right)^\top. \tag{6.8}$$

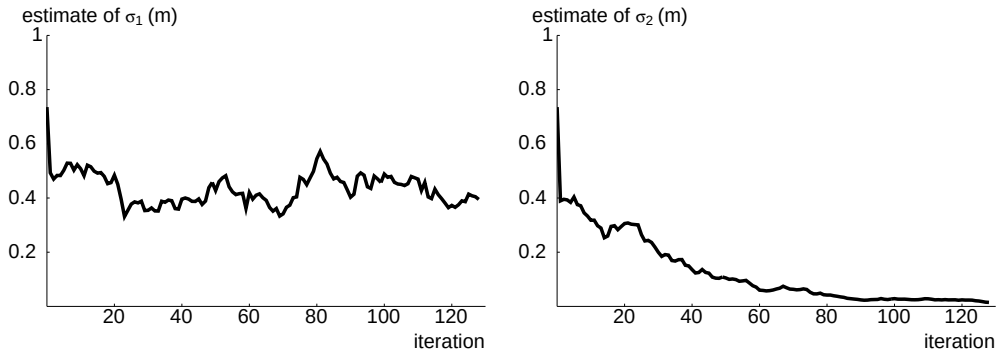
Here, the ψ are the smoothing weights whose calculation was described in section 6.2.2. Similar expressions for the expected values of the first-order moments R_i apply.

One approach to this learning problem is to set the number of samples N to a value just high enough to give reasonable tracking, and run the algorithm, using the approximations given in eqn (6.8) in the Expectation step. The results of following such a procedure are shown in figure 6.5, where the 'suitable timescale' for the process has been set to the length of the sequence: $\tau = 27$.

It can be seen that such behaviour cannot really be called 'convergence' — there is a large amount of random fluctuation, and it is difficult to tell what the final learnt model should be, or even if the estimates have in fact converged.



Estimate of offset parameter



Estimate of noise parameter

Figure 6.5: **Using only enough samples to allow tracking produces erratic behaviour of the EM iterative process.** *Results of applying the EM learning algorithm to a sequence tracked using $N = 100$ samples. Shown are the x and y components of the offset parameter of the constant-acceleration dynamics, and the two eigenvalues of the noise variance matrix.*

Moments are Random Variables

Returning to eqn (6.8), the reason for the randomness observed in the previous section becomes apparent. A sample-set approximation to the various probability densities is used — this means that the values used for the expected values of the moments are in fact random variables, dependent in a complex fashion on the random variables used in the operation of the Condensation filter. Therefore, given one estimate for the dynamical parameters, the value calculated as the next estimate is also a random variable. Without any knowledge of the behaviour of this random variable, it will be difficult or impossible to produce sensible results from the Condensation-EM learning algorithm.

First note, though, that the approximation of eqn (6.8) becomes exact in the limit as $N \to \infty$. However, taking an extremely large value for N is not practical, as the smoothing process involved in the algorithm is $O(N^2)$.

Behaviour of a Single Step

Therefore, the behaviour of a single step of the Condensation-EM estimation procedure was examined for practical values of N. In the case of estimation of a constant-acceleration model in 2 dimensions, the parameters of interest are the 2 components of the acceleration, and the noise variance matrix.

Starting from the same value of φ_i (in this case, constant velocity with noise set to allow tracking), one step of the Condensation-based EM algorithm was run several times, and the mean and variance of the several values of φ_{i+1} computed. Values of N used ranged from 128 (below this, tracking was unreliable) to 2048 (beyond this, the computational cost of the smoothing operation becomes unacceptably high) in factors of 2. For each value of N, the entire learning procedure was performed 32 times. The data resulting from this experiment are summarised, in terms of the mean and variance of the values, in tables 6.1 and 6.2 and figures 6.6 and 6.7.

It is clear from these statistics that, as expected, the variance of the estimates decreases with increasing N. (As previously noted, in the limit of $N \to \infty$, the variance would become zero.) There is a slight anomaly in that the variances for 128 samples are in most cases lower than those for 256 samples. A qualitatively plausible explanation for this is that 128 samples is too low to produce any sensible approximations whatsoever. Examining the results, considering in particular the behaviour of the noise estimator, $N = 2048$ samples must be considered a minimum number for use in the iterative EM procedure.

N	$\hat{\mu}(d_x)$	$V(d_x)$	$\hat{\mu}(d_y)$	$V(d_y)$
128	0.127	0.282	-11.2	4.75
256	0.164	0.141	-10.9	3.70
512	0.059	0.060	-10.3	0.80
1024	0.042	0.056	-9.97	0.24
2048	0.037	0.030	-9.81	0.08

Table 6.1: **Using a larger sample-set produces an estimator of the offset parameter with lower variance.** *The sample mean $\hat{\mu}$ and sample variance V for each component of the estimate of the offset parameter of the dynamical model are shown.*

N	$\hat{\mu}(\sigma_1)$	$V(\sigma_1)$	$\hat{\mu}(\sigma_2)$	$V(\sigma_2)$
128	1.56	0.771	0.564	0.00139
256	1.50	1.322	0.563	0.00066
512	1.21	0.419	0.558	0.00040
1024	0.89	0.167	0.557	0.00017
2048	0.77	0.050	0.555	0.00005

Table 6.2: **A larger sample-set produces a lower-variance estimator of the noise parameter.** *The sample mean $\hat{\mu}$ and sample variance V for each component of the estimate of the noise parameter, given in terms of the two eigenvalues of the variance matrix, of the dynamical model are shown. Apart from the $N = 128$ entry, a higher sample-set size gives a better approximation to the noise estimator.*

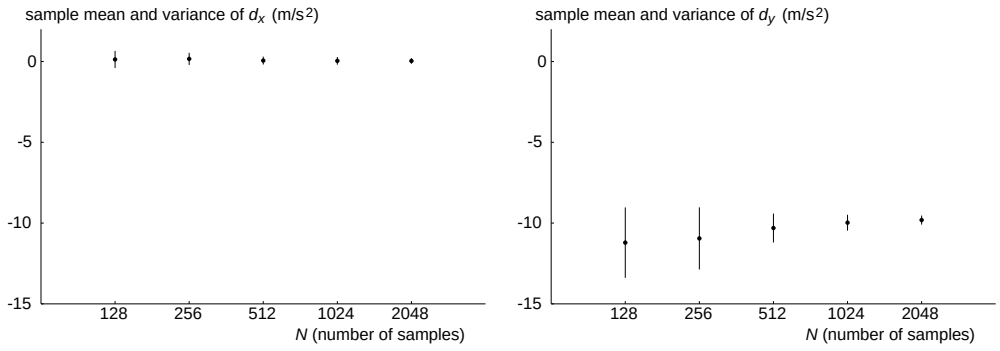


Figure 6.6: **Larger sample-sets produce better estimates for the offset parameter.** *Shown are the means of the 32 estimates with a vertical bar extending one (sample) standard deviation above and below the mean. As more samples are used, the variance of the estimate decreases.*

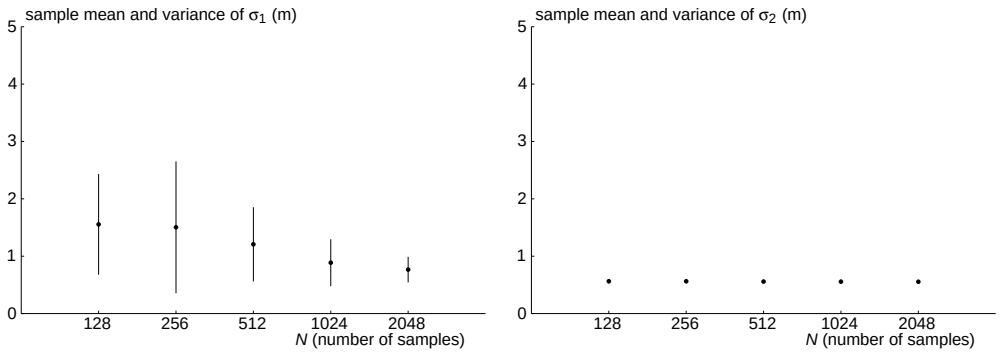


Figure 6.7: **Except for very low N, variance of noise estimate decreases with increasing sample-set size.** *Shown are the means of the 32 estimates with a vertical bar extending one standard deviation above and below the mean.*

The Cost of More Samples As a rough indication of the computational cost of raising N, the times taken for computation of one step are given below. The sequence in question is quite short — 27 time-steps, the length of the section of ballistic motion under consideration. Note that the tracking time given here can be improved on considerably by various programming and algorithmic methods, but since the dominant cost in these calculations is that of the smoothing, this is less of an issue. Times here were measured on an SGI Octane with a 175 MHz R10k processor.

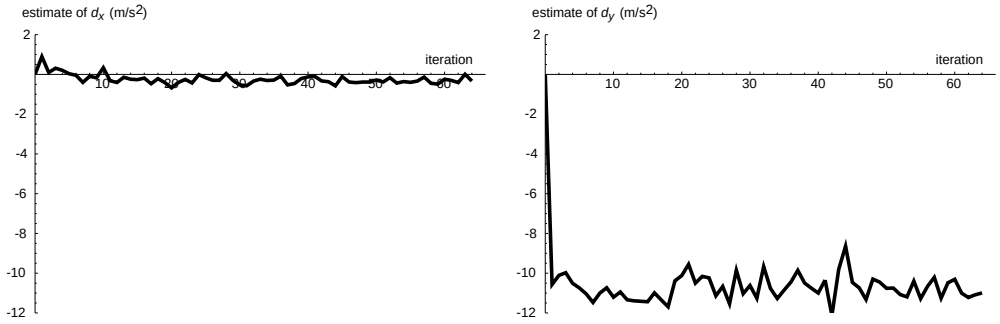
N	**Tracking time (s)**	**Smoothing time (s)**	**Estimation time (s)**	**Total time (s)**
128	12	3	0	15
256	19	9	0	28
512	33	35	0	68
1024	61	138	1	200
2048	117	528	2	647

Table 6.3: **Increasing N incurs a significant computational cost.** *Using a larger number of samples N produces a decrease in the variance of the estimate of the next EM iteration's parameter values, but at the cost of considerably increased computational time. (The tracking (filtering, i.e., the forward Condensation pass) operation is $O(N)$, as is the estimation step, but the smoothing operation is $O(N^2)$. These complexities can be seen in the times shown.)*

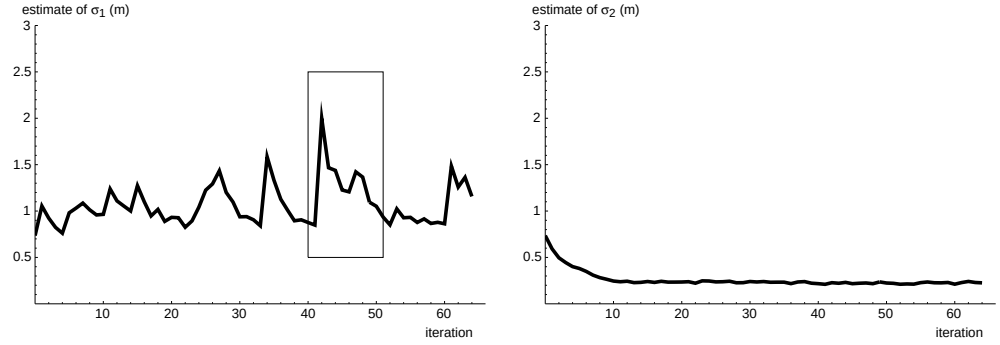
EM Behaviour with More Samples

Having investigated the influence of the sample-set approximation to the expectation equation, the experiment described in section 6.3.1 was repeated with $N = 2048$. The results

are presented in figure 6.8 below.



Estimate of offset parameter



Estimate of noise parameter

Figure 6.8: **Even a very large sample set ($N = 2048$) does not produce smooth convergence.** *Although the variances of the estimators are reduced by using a much larger N, enough variability in the estimates remains to prevent convergence from being acceptable. See text for the meaning of boxed region in bottom-left graph.*

These results indicate that even this much larger value of N, with the lower variance it produces, does not result in smooth convergence of the dynamical model.

Considering in particular the behaviour of the noise parameter, the chief cause of the random nature of the iterations seems to be large spikes which then take some time to decay again. It seems that the 'converged' value ought to be around 1, looking at the area highlighted in the bottom-left graph of figure 6.8. However, as was seen in the previous section, occasionally the estimate produced will have a very high noise component, and this then takes some time to decay away again.

6.3.2 Robustness in the EM Step

The behaviour noted in the previous section is undesirable, in that an occasional outlying output from the Condensation-EM procedure can throw off the convergence of the algorithm for some considerable number of steps, and, furthermore, can prevent convergence entirely. This makes it difficult if not impossible to tell when the algorithm has reached its final converged value. Some robustness in the calculation of φ_{i+1} is therefore required.

Once this robustness has been achieved, figure 6.8 leads us to suspect that convergence will be quite rapid — possibly even within half a dozen steps or so. The increased cost of gaining robustness in each step could, it is to be expected, be offset by a reduction in the number of steps required.

Re-examination of a Single Step

Consider the behaviour of a single step when $N = 2048$ more closely, in particular the noise parameter. From the results, this is the most troublesome of the parameters. The raw data for this part of the experiment are shown in figure 6.9.

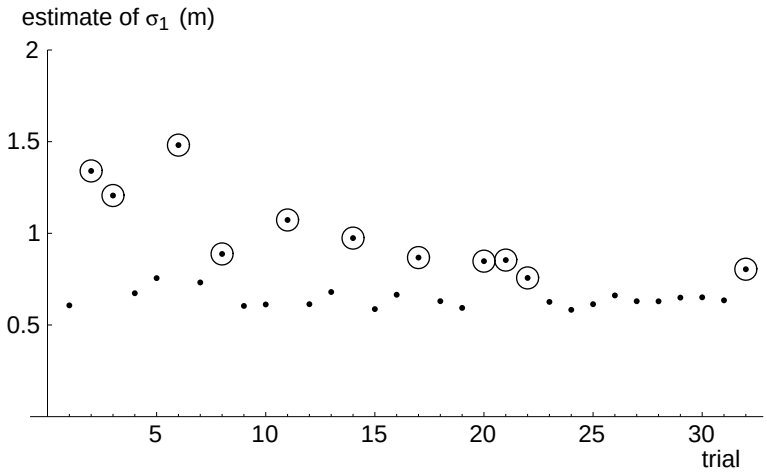


Figure 6.9: **Output from the $N = 2048$ estimator consists of values near the 'correct' one and outliers (ringed).** *Shown here are the results for σ_1 of repeated trials of one iteration of the estimation algorithm. It can be seen that some method of making the EM step robust is required, since even with a large value of N, sometimes estimates are produced which are significantly in error.*

Examining these data, a reasonable hypothesis is that the 'correct' value of this parameter is around the $\sigma_1 = 0.7$ mark, with quite a lot of outliers, shown ringed in the figure. Some method of ignoring those outliers would then help. From this sample, the probability of an outlier occurring can be estimated as around 11/32.

Consider the operation of repeating the estimation procedure n times (for n odd), and taking the median of the resulting parameter estimates. Assuming that outliers are all too large (this is certainly a reasonable assumption given the data in figure 6.9), the median will be an outlier iff at least half of the n samples are outliers, i.e., with probability

$$P_{\text{out}}(n;p) = \sum_{i=\lceil n/2 \rceil}^{n} \binom{n}{i} p^i (1-p)^{n-i},$$

where p is the probability of an outlier occurring. In this case, p can be set to 11/32.

A balance must be struck between the added robustness of using an ever-larger n in this procedure, and the the increased cost of producing more estimates — recall that for $N = 2048$, each estimate takes around ten minutes. Consider also that even if an outlier *is* chosen, it will be a small one and therefore an improvement in the approximation will have been gained nonetheless. A value of $n = 7$ with $N = 2048$ was therefore chosen, giving $P_{\text{out}} = 0.18$.

The meaning of 'median' must first be elaborated slightly: the median of a set of offset vectors can easily be defined component-wise, i.e., the nth component of the median is set to the median of the set of nth components. The noise parameter requires slightly more thought; the procedure adopted was to take that matrix which had the median determinant.

Behaviour of Robustified EM

The procedure described in the previous section was implemented, and applied to the learning of the ballistic model. At the same time, it was observed that the starting value chosen for the noise matrix was coincidentally very close to the final converged value, which does not provide a very stringent test of the method's performance. Therefore, for the following experiment, the starting value of C was set a factor of 4 larger than previously. The resulting behaviour of the algorithm is shown in figure 6.10.

The convergence (from this less accurate initial noise value), is now much more convincing, although there is still some entirely unavoidable fluctuation in the parameters. Even so, given the expected variance in the estimation process, it is now reasonable to conclude with some certainty that the EM algorithm has in fact converged, and the jitter in the estimates is caused by the inherent randomness in the process. The learnt model for ballistic motion is therefore, in the original pixel-based units used by the tracker,

$$\mathbf{d} = \begin{pmatrix} 0.03 \\ -1.71 \end{pmatrix}; \quad C = \begin{pmatrix} 0.38 & 0.01 \\ 0.01 & 7.31 \end{pmatrix}.$$

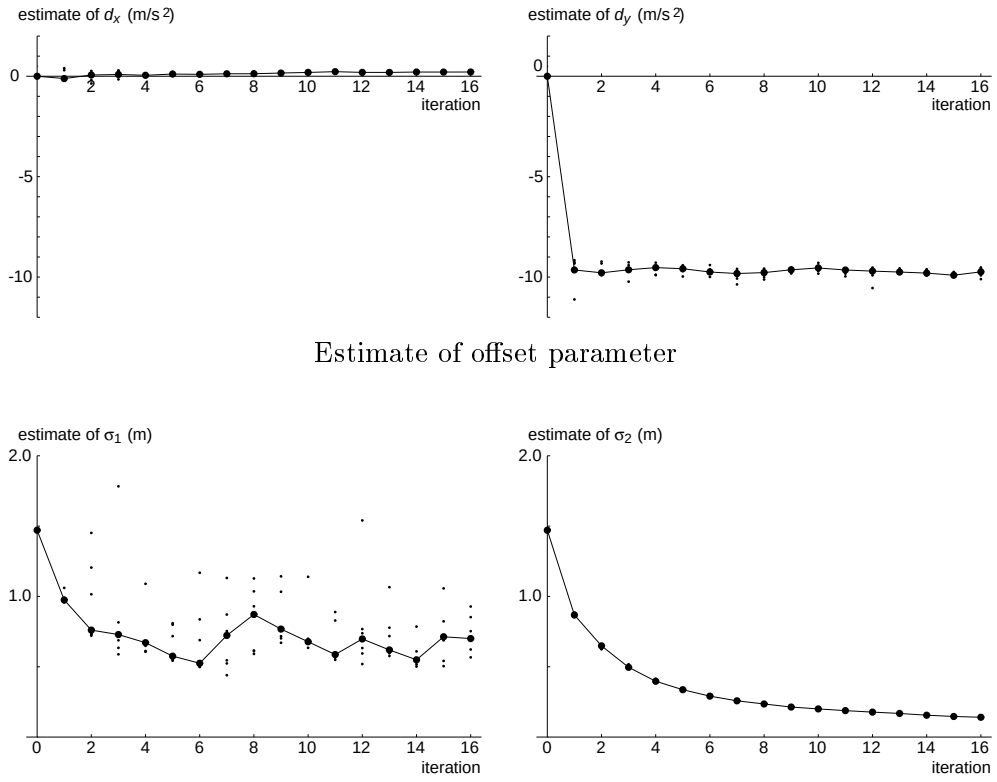


Figure 6.10: **Robustifying the Condensation-EM algorithm produces better convergence.** *Each iteration consists of producing 7 samples of the 'next estimate', and choosing the median. The individual samples are shown using small dots, and the chosen median with a large dot.*

This is equivalent to a downwards acceleration for the ball of around $9.7\,\mathrm{ms}^{-2}$, close to $g = 9.81$, the correct value for acceleration due to gravity (Howatson et al., 1972).

6.3.3 Longer Example Sequences

The experiments described above were all performed on a relatively short training sequence: 27 time-steps. This was the length of time the ball was in the air (just over 0.5 s), and so is the longest possible contiguous sequence of training data for this problem.

It is possible to combine data from several disjoint sequences, by adding the moments (or, in the case of EM, the expected values of the moments) from each sequence — see appendix A.4. Experiments were performed on a set of training sequences, each of 27 time-steps, and the effect of combining up to six such sets for various sizes N of the sample set investigated.

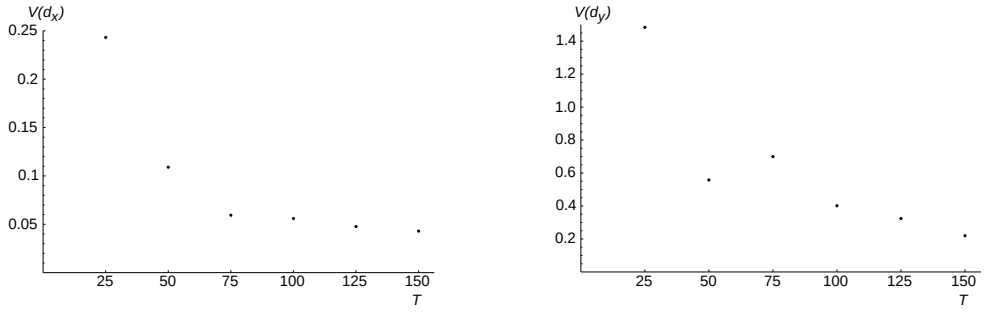
Experimental Results

Thirty-two trials using each combination of sequence length and sample-set size were performed. Typical results are those for $N = 512$; the variance of the estimator is the quantity of interest, and it is shown in figure 6.11. Note that system noise σ_1 and σ_2 parameters have been expressed as the predicted noise after $\tau = 27$ timesteps; although the sequences are not all of this length, comparison between the models learnt from sequences of different lengths requires that τ be the same for each one. For the graphs, the more relevant quantity $\sum(T_m - K)$ has been used on the horizontal axis — see appendix A.4 — here $K = 2$ so multiples of 25 result from sequences of length 27.

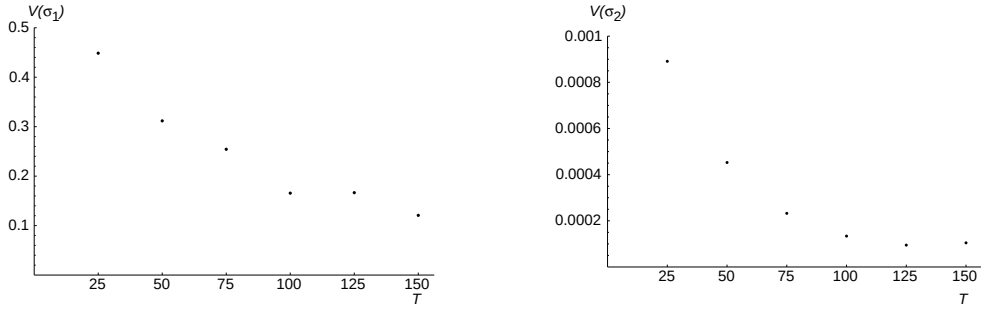
It is already known (Blake and Isard, 1998) that a longer training sequence will produce more reliable estimates for the dynamical model, even when the true states themselves are known. The current experiments indicate that this requirement is stronger when learning is performed within the Condensation-EM framework.

6.3.4 Robustifying 'Across Iterations'

Although the procedure of section 6.3.2 did produce an improvement in the convergence behaviour of the parameters, it also increased the computational cost considerably; by a factor of 7 in the experiments described. As an attempt to maintain the robustness of this 'within-iteration' smoothing while decreasing the cost, the following 'across-iterations' procedure was devised. Suppose an average of m estimates is desired. Then for iteration i of the Condensation-EM algorithm, use model parameters φ_i in the filter/smoother to



(a) Offset parameter



(b) Noise parameter

Figure 6.11: **Longer training sequences result in lower variance in the estimators.** *Sample variances of the model parameters from repeated runs of a single step in the EM algorithm using $N = 512$ samples per time-step, and various total length T of training sequence. On the whole, as T increases, the variance decreases. More repetitions of each experiment (32 were performed here) would give more accurate estimates of these variances.*

produce an estimate φ_i'. Then define φ_i by

$$\varphi_i = \texttt{Robust}(\varphi_i', \varphi_{i-1}', \ldots, \varphi_{i-m+1}'), \tag{6.9}$$

where the function `Robust()` can be a median or mean (although the mean does not provide much robustness as such, it will still reduce the variance of the estimate in situations where a median is difficult to define) as appropriate. The aim of this procedure is to keep the advantage of robustness while allowing the parameters φ_i to converge more quickly.

Note that defining φ_i by

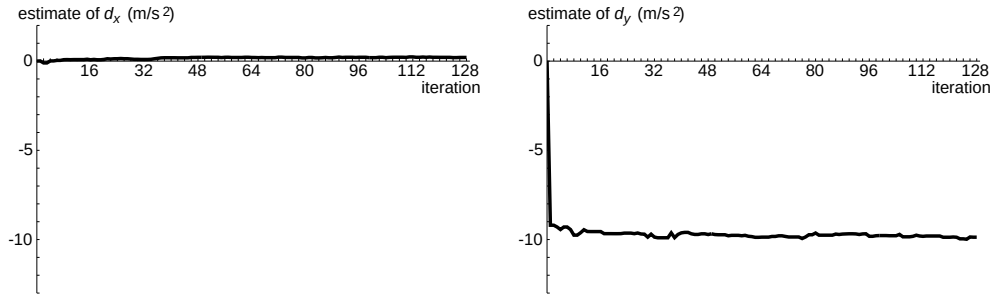
$$\varphi_i = \texttt{Robust}(\varphi_i', \varphi_{i-1}, \ldots, \varphi_{i-m+1}),$$

while at first sight appealing as it uses already-averaged values, results in problems when the median is used as `Robust()`. In this case, m must be odd. Suppose that at some iteration, $\lceil m/2 \rceil$ of the arguments to `Robust()` are the same. This is likely, as a median filter often produces consecutive equal values. Thenceforth, all φ_i will be this value, as the median of a set of numbers over half of which are identical is that value. Therefore, the choice given in eqn (6.9) was used.

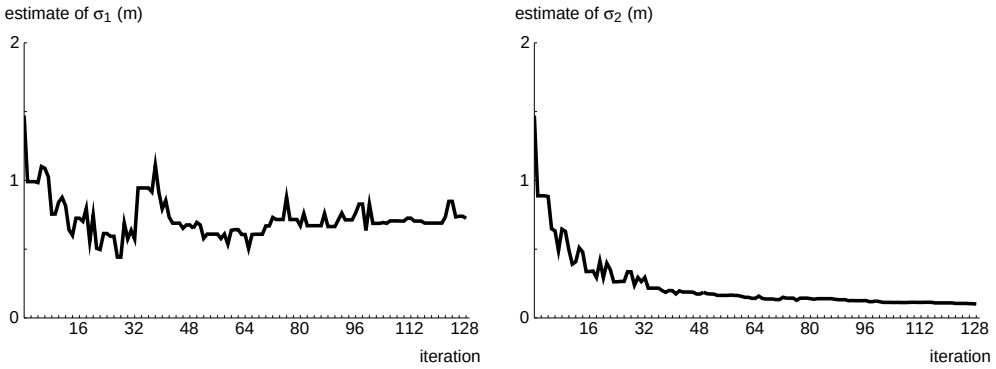
Experiments

The learning experiment for the juggling ball undergoing ballistic motion was repeated with this across-iterations method of averaging, using $m = 7$ and the median operator for `Robust`. The results are shown in figure 6.12.

From the graphs, a reasonable claim is that the estimates have converged after around 80 iterations, at a cost therefore of around 90 estimate-generations. The experiment shown in figure 6.10 converged after around 12 iterations, at a cost of around 84 estimation-generations. Both sets of graphs (figures 6.12 and 6.10) exhibit similar levels of noise in the 'converged' sections of the final iterations performed. Also, it is reassuring that the converged values are the same (to within the accuracy allowed by the noise). Therefore, there is little to choose between the two methods; the within-iteration method will be used in the experiments to be described in chapter 7.



Estimate of offset parameter



Estimate of noise parameter

Figure 6.12: **Across-iterations robustifying produces similar results to the within-iteration method, at a similar cost.** *Instead of repeating the estimation procedure a number of times at each iteration of the EM algorithm and taking an average, an average of previous estimates together with a newly-produced one can be taken. Smoother behaviour results, at a similar cost to the within-iteration averaging.*

Chapter 7

Applications of Condensation-EM Learning

Chapter 6 demonstrated that EM provides a useful methodology for learning dynamical models for a Condensation-based tracker. This chapter investigates the performance of such a learning algorithm in two situations where the strengths of Condensation are most apparent, namely, a system where a multi-mode dynamical model is to be learnt, and also a situation where the robustness of Condensation to cluttered measurements is required. The experiments described are the original work of the author.

7.1 Multi-model Learning

The first area of interest is that of learning a mixed discrete/continuous model. As an example application of the Condensation-based EM learning algorithm in this case, consider again the juggling sequence. During juggling, each ball undergoes the sequence 'ballistic', 'catch', 'carry', 'throw', 'ballistic', continuing with its being caught in the other hand. This section describes experiments to determine the performance of multi-mode learning on one such juggling cycle.

7.1.1 Starting Conditions

If the technique is to be fully general, the learning system should be given as little prior knowledge as possible about the dynamical system it is to learn. Within the current framework, certain parameters of the model must be specified, however.

The number of discrete models must be known, together with starting values for all parameters. In this example, three states are specified, corresponding to the ideas of 'ballistic', 'catch/throw', and 'carry'. The 'catch' and 'throw' phases have been combined

as initial informal investigations indicated that they were very similar, involving sharp upwards acceleration.

All three will be constrained to constant-acceleration models, with process noises fixed at reasonable values determined from the previous learning experiments on this sequence. Although in theory the process noise can be learnt simultaneously, the experiments in section 6.3 showed that it is the least reliable parameter. To concentrate on the multi-model learning, then, the noises were fixed.

The initial transition matrix used was

$$M_0 = \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{pmatrix},$$

embodying some notion that the states are expected to be reasonably persistent, but not specifying the asymmetry of the transitions (e.g., the transition 'ballistic' to 'carry' never happens).

Two different sets of starting conditions for the accelerations of the three models were used, to provide the system with different levels of prior knowledge. One set gave all three models initial values of zero acceleration, and the other provided some symmetry-breaking by specifying

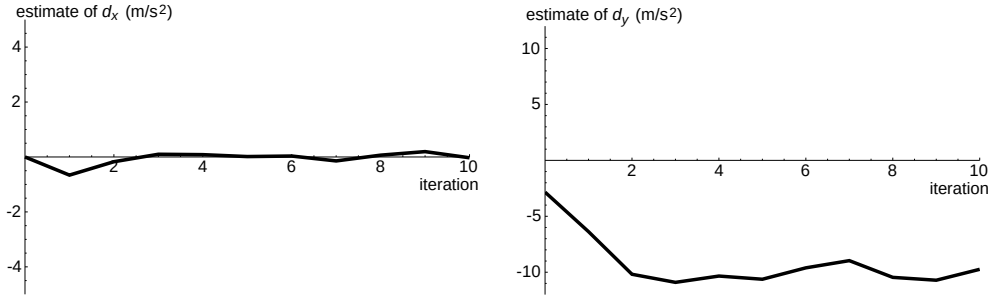
$$\mathbf{a}_0^1 = \begin{pmatrix} 0.0 \\ -2.8 \end{pmatrix}; \quad \mathbf{a}_0^2 = \begin{pmatrix} 0.0 \\ 2.8 \end{pmatrix}; \quad \mathbf{a}_0^3 = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix}$$

(in ms^{-2}). The expectation that the models will be chiefly differentiated by their vertical accelerations is therefore supplied. (The apparently odd values arise from specifying the model in terms of pixels for a time-step of 0.02 s, where the value is ± 0.5.)

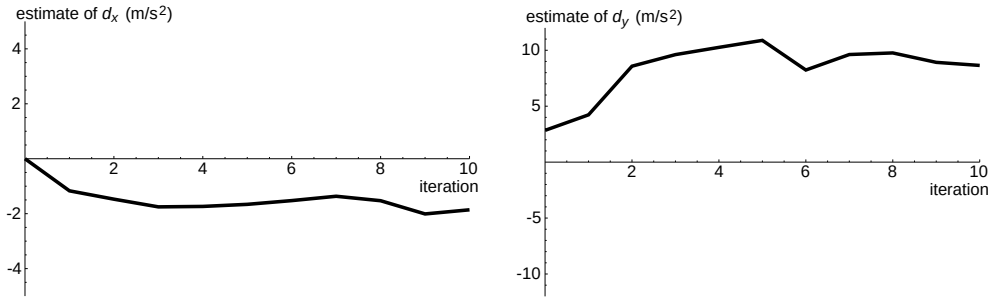
Other Experimental Factors Using the techniques developed in previous sections for adding robustness to the inherently random Condensation-EM learning algorithm, 7 estimates were produced per EM step and the mean used in the next iteration. The sequence used was 65 time-steps long, which although quite short, contains one complete cycle of the model — 'ballistic', 'catch', 'carry', 'throw', 'ballistic'. All the important behaviour of the model is thus present in the sequence.

7.1.2 Results — Asymmetric Prior

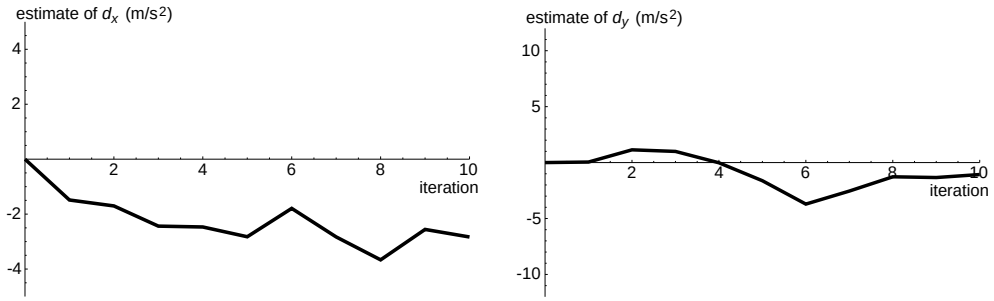
The behaviour of the system when given slightly more prior knowledge (the fact that models possess up/down asymmetry) is presented first. Figure 7.1 shows the behaviour of the accelerations of the learnt models as the EM algorithm progresses.



(a) First model (ballistic)



(b) Second model (catch/throw)



(c) Third model (carry)

Figure 7.1: **From a non-symmetric starting estimate, the algorithm learns the three types of motion involved in juggling.** *A small amount of symmetry-breaking results in models which represent the three types of motion involved in the juggling cycle — ballistic, catch/throw, and carry.*

It can be seen that the model which was pushed towards 'downwards', i.e., the first model, has converged (within the bounds of the inherent noisiness) to a value of around -9.9ms^{-2} (taking the mean of the last five estimates), which is close to $-g = -9.81\text{ms}^{-2}$. Its horizontal component has remained close to zero. A reasonable conclusion, then, is that this first model has converged to one accurately describing motion under gravity.

The second model, whose initial conditions favour an upwards-accelerating model, has converged to an acceleration of roughly g upwards, with some leftwards acceleration. This is consistent with the motion of the ball, as while in flight it is moving rightwards, and must be moving leftwards by the time it re-enters the ballistic state. These characteristics are consistent with this model describing the 'throw' and 'catch' states of the ball's motion.

The third model has very little vertical acceleration and slightly more leftwards acceleration. This matches the 'carry' phase of the motion.

The other part of the learnt model is the transition matrix, the progress of which is shown in figure 7.2. Cumulative transition probabilities out of each mode are shown.

The final value of the transition matrix is

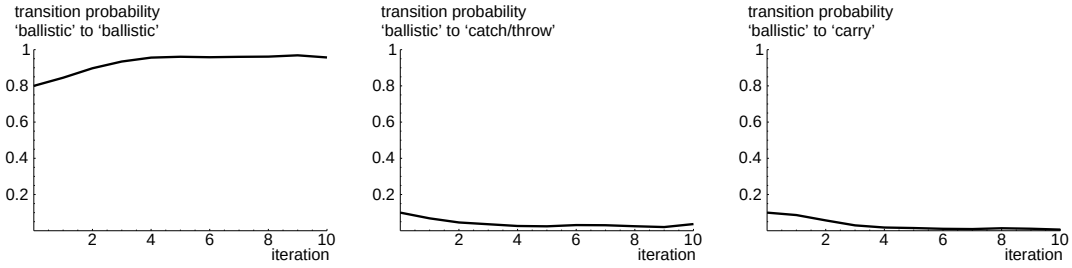
$$M_{\text{final}} = \begin{pmatrix} 0.957 & 0.037 & 0.006 \\ 0.024 & 0.934 & 0.042 \\ 0.155 & 0.266 & 0.579 \end{pmatrix}, \tag{7.1}$$

which is illustrated as a transition diagram in figure 7.3.

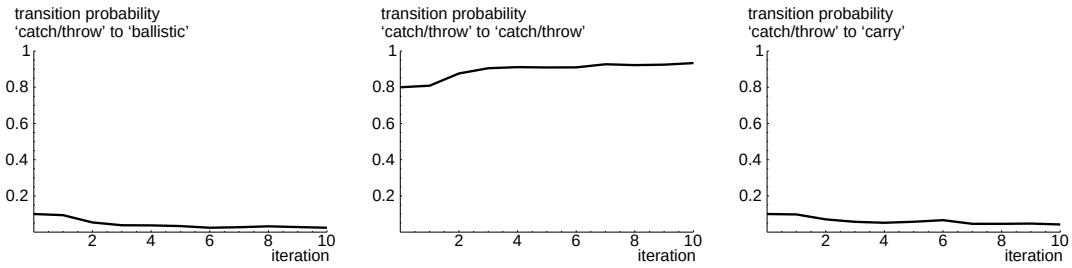
The transition behaviour is qualitatively correct. 'Ballistic' motion has been found to be extremely unlikely to lead into 'carry' motion; ideally, this entry in the matrix would be zero, but the value 0.006 is certainly very low compared with the 'correct' transition into 'catch/throw', whose probability has been estimated as 0.037. Transitions out of the other states are approximately in line with expectations: 'catch/throw' shows similar probabilities of leading to 'ballistic' and 'carry'. The probabilities out of 'carry' are the least convincing, since there is no transition 'carry' to 'ballistic', yet this has an estimated probability of 0.155. This can to some degree be explained by the shortness of the 'carry' phase of the training sequence meaning that few data are available for estimating this row of the matrix.

Final Classification The final learnt system can be used to classify the training sequence into the three learnt states; this is shown in figure 7.4, with snapshots of the estimated transitions shown in figures 7.5 to 7.8.

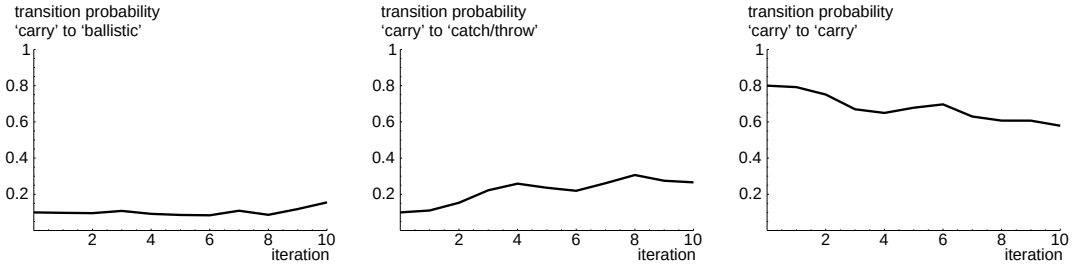
The classifications are approximately where a human observer would place them, although perhaps a person would not say that 'throw' starts as early as the system has labelled it. From the point of view of the estimated acceleration of the ball, however, the



Transition probabilities out of first (ballistic) model



Transition probabilities out of second (catch/throw) model



Transition probabilities out of third (carry) model

Figure 7.2: **Evolution of transition probabilities from non-symmetric starting position.** *The convergence of the transition matrix shows qualitatively the correct trends. The noise in the transition probabilities out of the third model can be explained by the relatively short duration of this mode.*

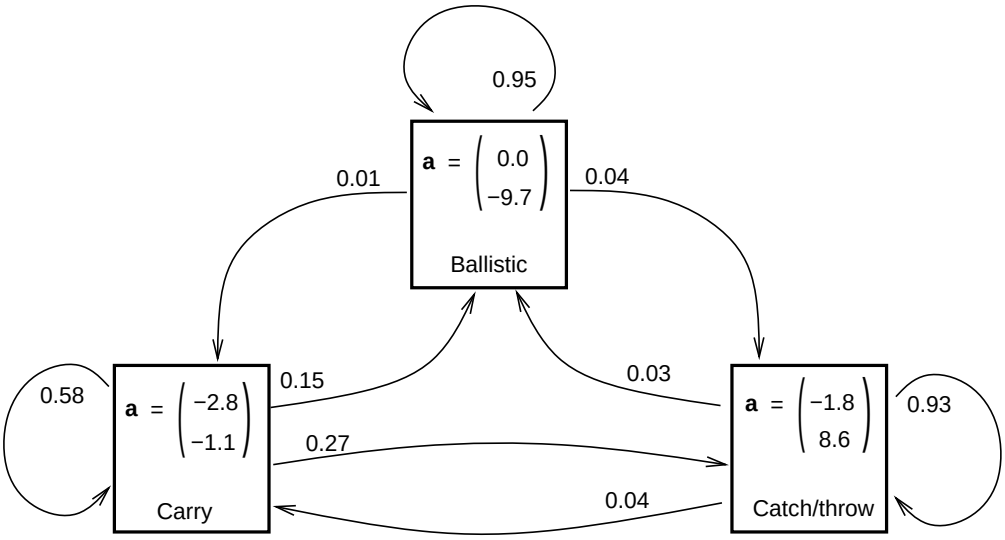


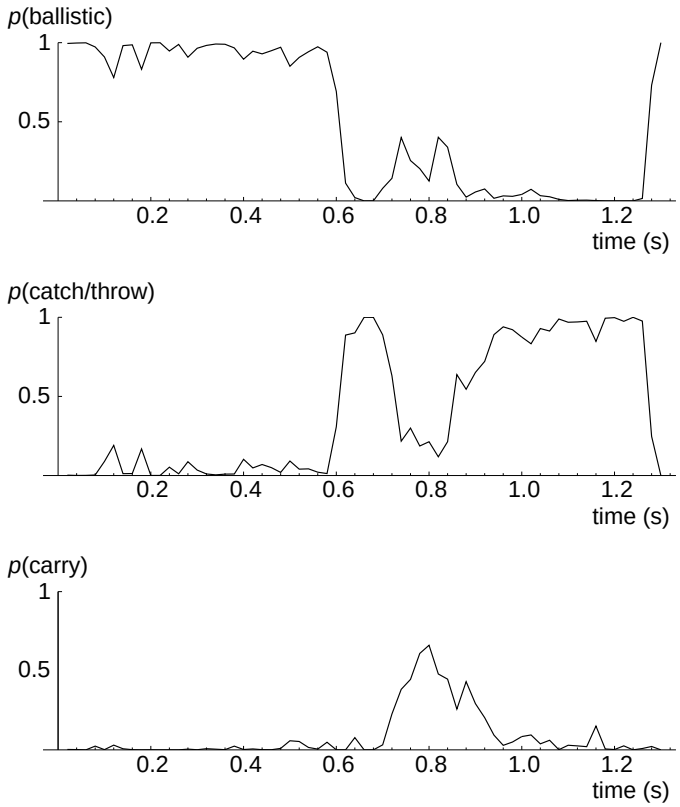
Figure 7.3: **The final learnt transition diagram shows the correct qualitative characteristics for the juggled ball.** *The numerical transition probabilities in the graph are from the final learnt transition matrix shown in eqn (7.1). The acceleration values are the mean of the last three values of figure 7.1. The states have been labelled with their interpretation, and show transition probabilities corresponding approximately to what would intuitively be expected.*

placement is very good. This is illustrated in figure 7.9, where the system's classification is seen to fit well with the different types of motion. One factor in the misalignment is that, particularly at the times of the transitions into and out of 'ballistic' motion, the ball is moving rapidly, producing significant motion blur in the images. This effect can be seen for the ball in the juggler's right hand in figures 7.6 and 7.7.

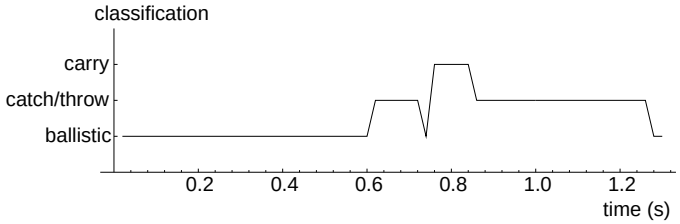
(One point of peripheral interest raised by the acceleration graph of figure 7.9(c) is that during the 'throw' phase, the acceleration undergone by the ball increases fairly smoothly from zero up to $20\,\mathrm{ms}^{-2}$ or so. This is therefore motion of approximately constant 'jerk', the term given to the derivative of acceleration. This quantity has been shown to be of significance for human motor control — Flash and Hogan (1985) present results which indicate that the integral over time of the squared magnitude of the jerk is minimised in voluntary human motion.)

7.1.3 Results — Symmetric Prior

The results of trying to reduce the amount of initial information given to the system are now investigated, in particular the effect of removing the starting asymmetry from the initial accelerations, in order to test the extent to which model structure may be purely



(a) Estimated probabilities of the three states



(b) Classification output

Figure 7.4: **Classification of training sequence is mostly successful.** *The automatically-learnt models have largely correctly segmented the sequence into its phases, with the progression 'ballistic', 'catch', 'carry', 'throw', 'ballistic' emerging fairly cleanly. There is a short glitch where the carried motion resembles ballistic motion, but on the whole the classification is correct.*

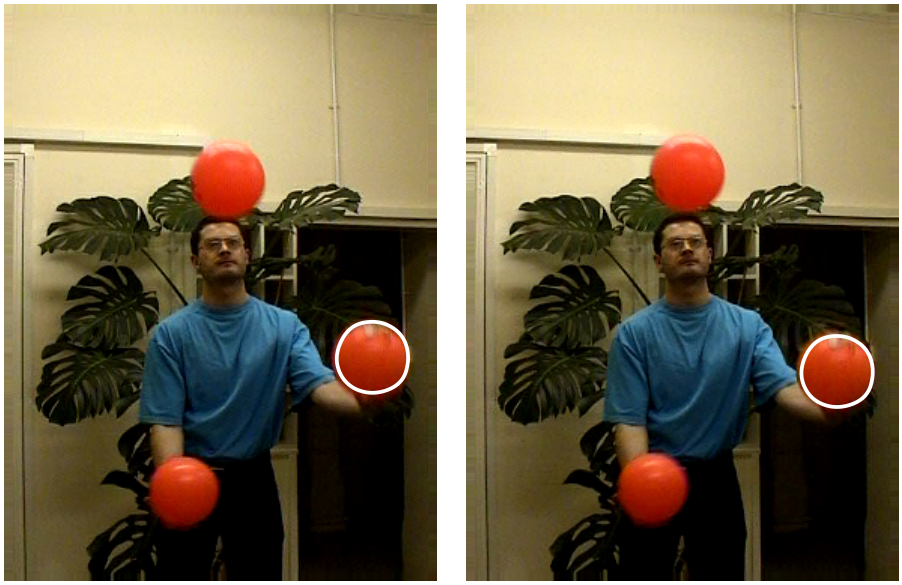


Figure 7.5: **Snapshots of detected 'ballistic' to 'catch' transition.**

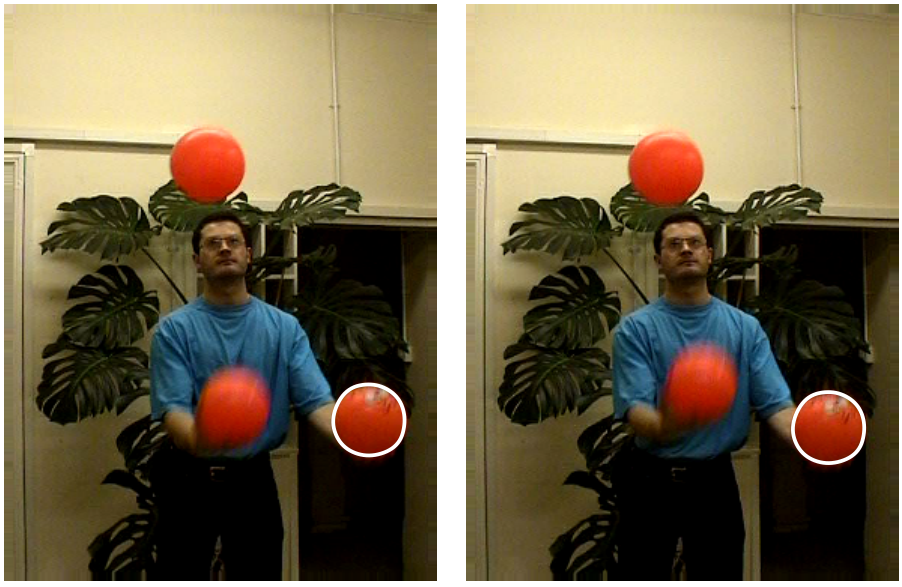


Figure 7.6: **Snapshots of detected 'catch' to 'carry' transition.**

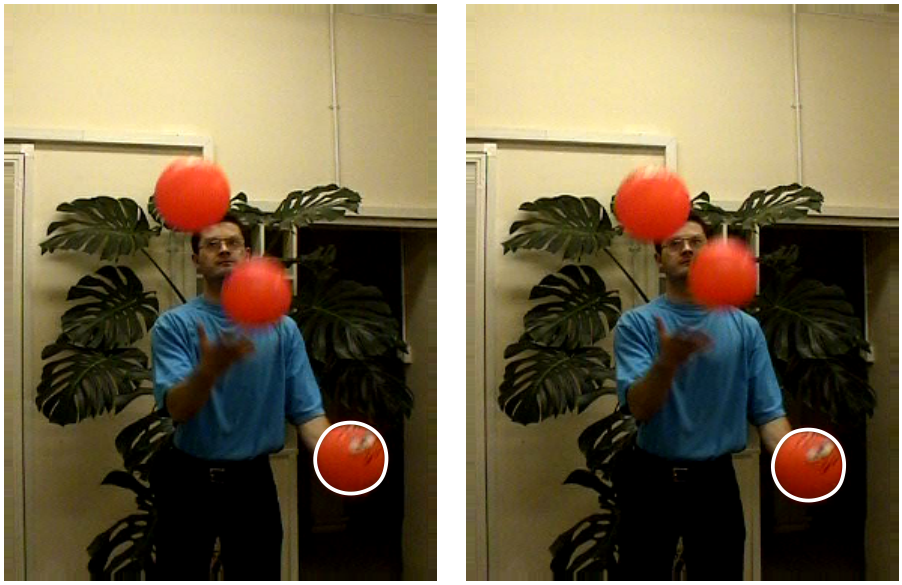


Figure 7.7: **Snapshots of detected 'carry' to 'throw' transition.**

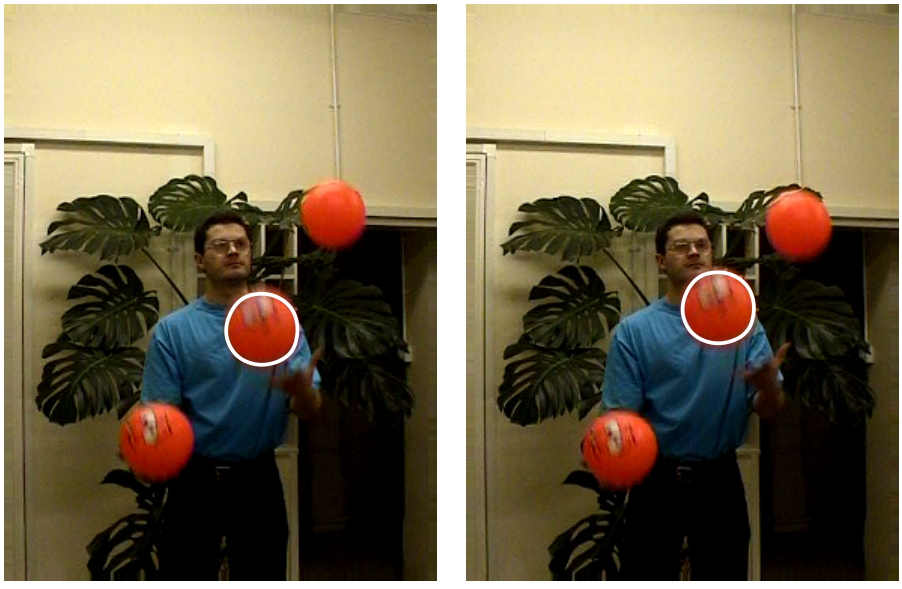


Figure 7.8: **Snapshots of detected 'throw' to 'ballistic' transition.**

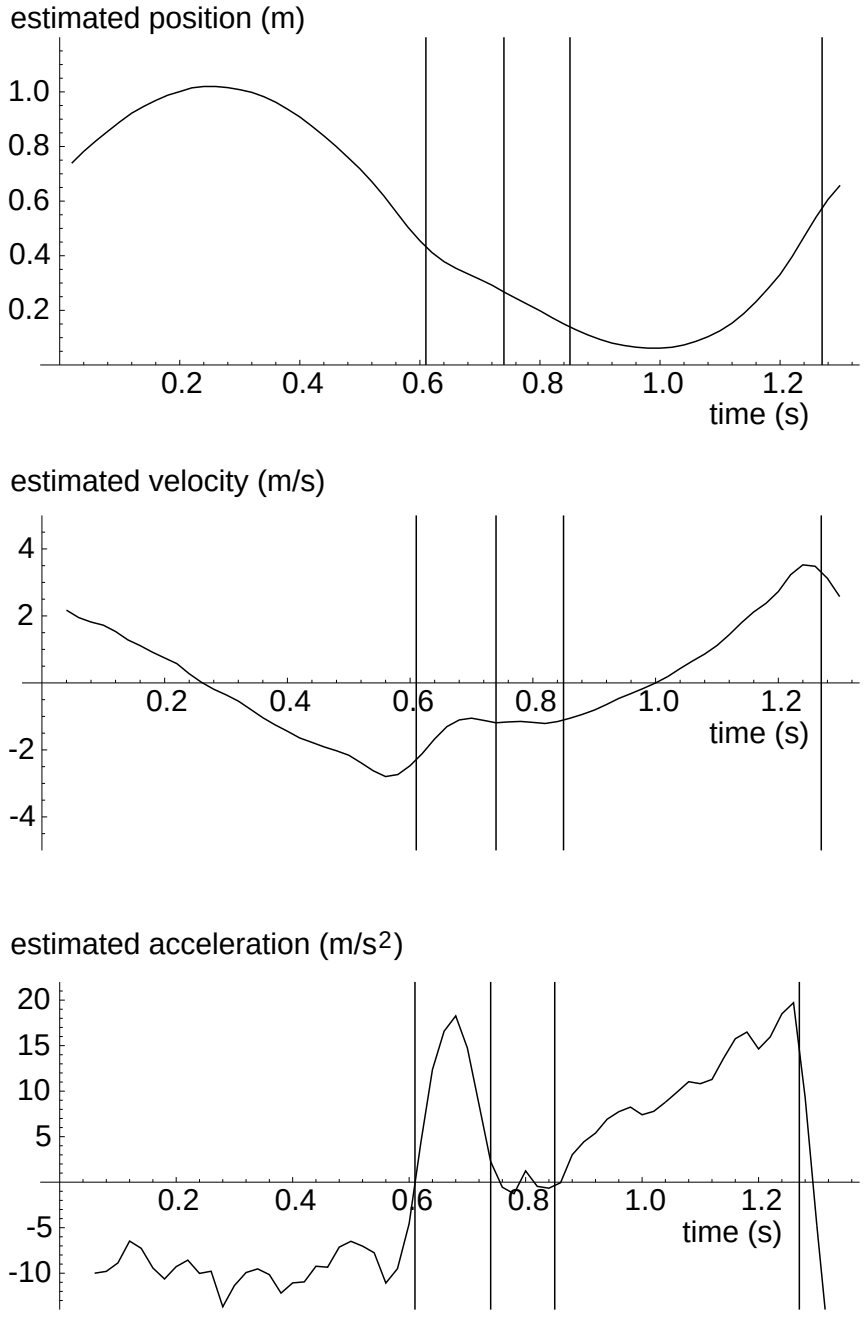


Figure 7.9: **Estimated vertical position, velocity, and acceleration of tracked ball, with detected transitions marked.** *It is clear, especially from the acceleration graph, that the automatic segmentation is in good agreement with the different classes of motion undergone by the ball.*

emergent. Therefore, the experiment above was repeated with the only change being that

$$\mathbf{a}_0^1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}; \quad \mathbf{a}_0^2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}; \quad \mathbf{a}_0^3 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The evolution of the accelerations from these initial values is shown in figure 7.10, where the final values of the first and second models can be seen to be very similar. Both have accelerations corresponding to motion under gravity. The third model therefore must encompass all motion which does not take place in free-fall; its value reflects this, being between the two non-ballistic learnt models described in the previous section.

The transition matrix behaves as shown in figure 7.11; the final numerical value is

$$M_{\text{final}} = \begin{pmatrix} 0.794 & 0.160 & 0.046 \\ 0.136 & 0.839 & 0.025 \\ 0.014 & 0.000 & 0.986 \end{pmatrix},$$

which is illustrated in the state transition diagram of figure 7.12.

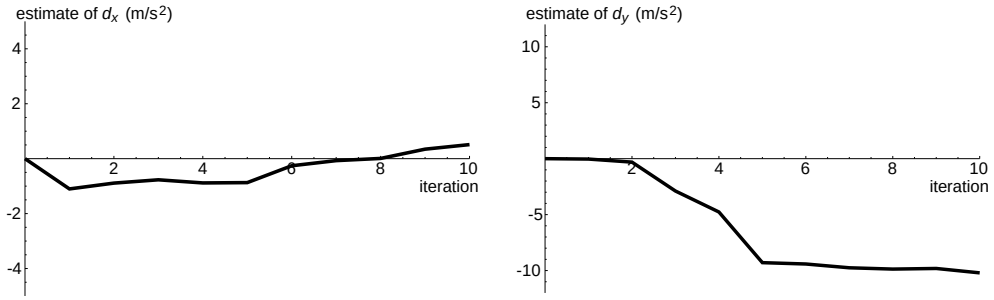
The third row of the matrix, the probabilities of transition out of the third (non-ballistic) model, shows that the tracker will never predict a 'ballistic 2' sample from a 'non-ballistic' one. Instead, 'non-ballistic' always leads to 'ballistic 1'. This is consistent with the first two models being indistinguishable — a slight imbalance in the transition probabilities from 'non-ballistic' will become greater as the EM algorithm proceeds, eventually resulting in the situation observed where one of them is zero.

The persistence of this third model is perhaps longer than would have been predicted, having an expected lifetime of around 74 timesteps, whereas figure 7.9 indicates the correct lifetime for the conflated non-ballistic models is more like 40 timesteps. One reason for the inaccuracy in these estimates of transition probabilities is discussed in section 7.3.5.

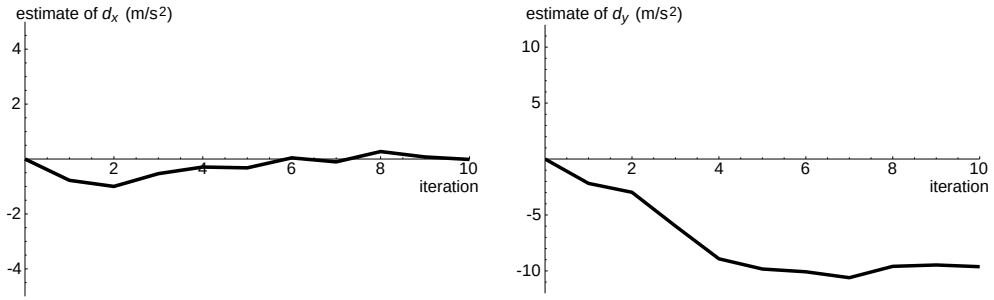
7.1.4 Classification of Test Sequences

Section 3.4 described the use of dynamical models for classification of unseen data. A natural experiment, then, is to apply the model learnt in the experiments of the previous section to some unseen sequences of the same type of motion, i.e., a 'ballistic', 'catch', 'carry', 'throw', 'ballistic' progression. Sixteen test sequences were recorded against the same background as the training sequence, and then tracked and classified using the learnt model and $N = 2048$ samples. Three example classification results are shown in figures 7.13 to 7.15.

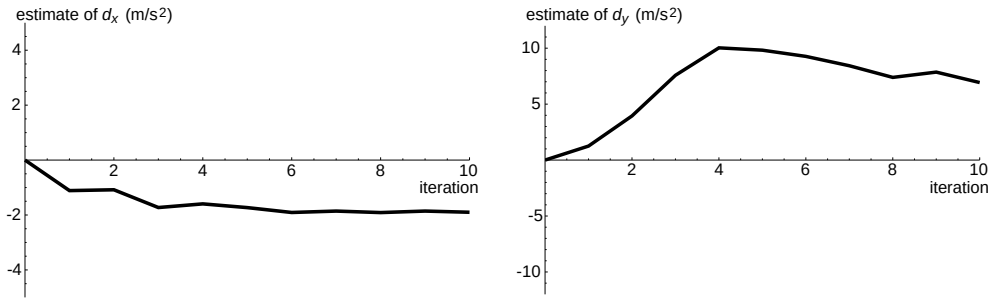
Various properties of the tracking were identified by manual inspection of the tracker's output: whether the outline of the ball was successfully tracked throughout the sequence;



First model (ballistic 1)

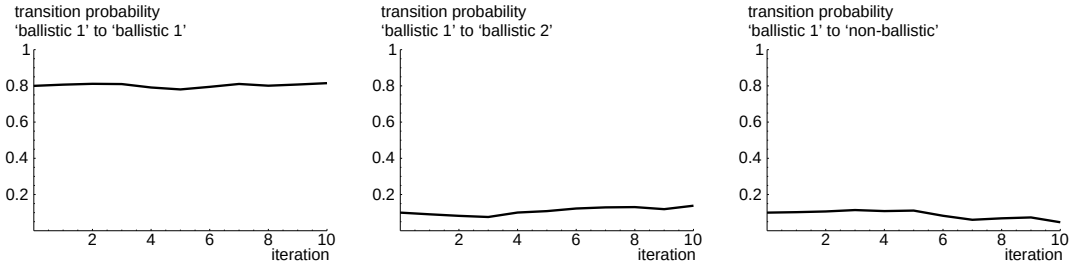


Second model (ballistic 2)

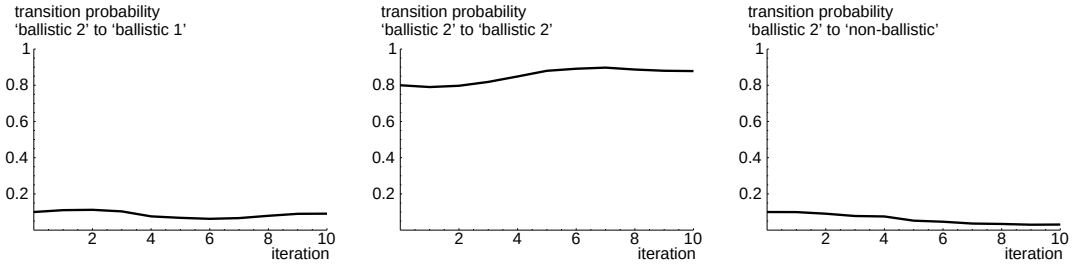


Third model (non-ballistic)

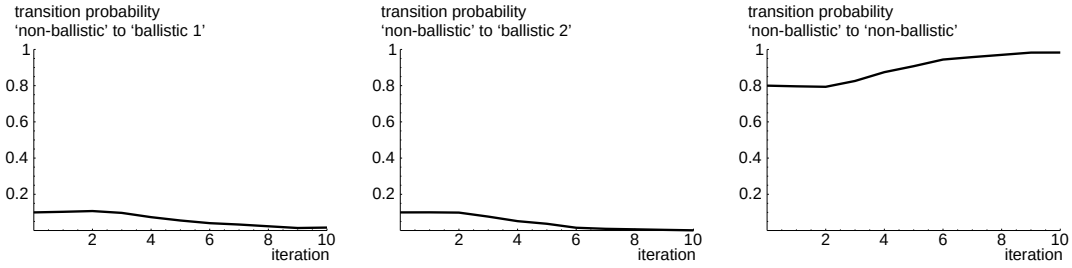
Figure 7.10: **Evolution of acceleration of models from symmetric starting position.** *The learning appears to be less successful than when a non-symmetric prior is supplied to the system. The hoped-for 'carry' phase does not appear as one of the models, whereas two models converge to a representation of ballistic motion.*



Transition probabilities out of first (ballistic 1) model



Transition probabilities out of second (ballistic 2) model



Transition probabilities out of third (non-ballistic) model

Figure 7.11: **Evolution of transition probabilities from symmetric starting position.** *The system evolves into a state where the non-ballistic model has zero probability of making a transition into the 'ballistic 2' model. The learnt model is degenerate, being effectively two-state.*

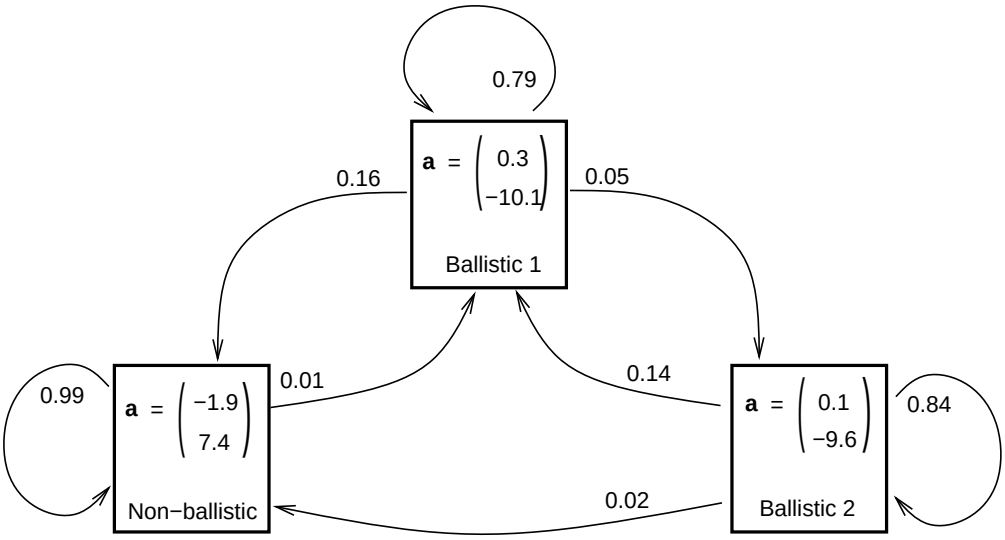


Figure 7.12: **An initial estimate with equal accelerations evolves into a two-state model.** *As can be seen from the acceleration values, two of the final learnt states describe ballistic motion, with the third having to encompass both types (catch/throw and carry) of non-ballistic motion. There is no probability of transition from 'non-ballistic' into 'ballistic 2'; as 'ballistic 1' and 'ballistic 2' are essentially identical, the situation where transitions are possible into both from 'non-ballistic' is unstable.*

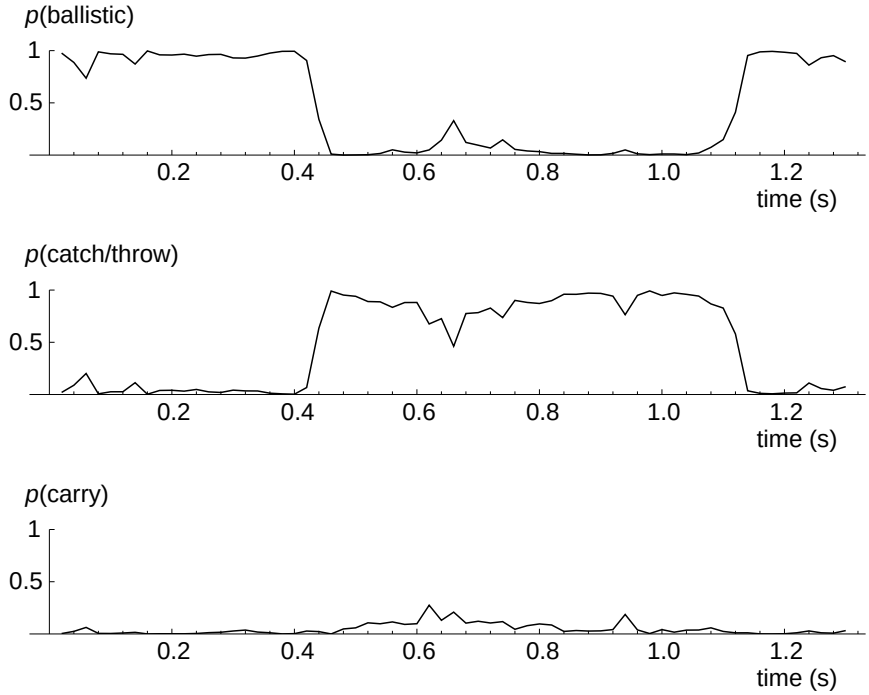
whether the system correctly separated the 'ballistic' motion from motion of the other two types; and whether the system isolated a 'carry' phase of the motion. The proportions of the test sequences exhibiting these properties are shown in table 7.1.

Property	**Proportion**
Tracking successful	0.9375
Ballistic/non-ballistic distinguished	0.9375
Carry phase identified	0.0625

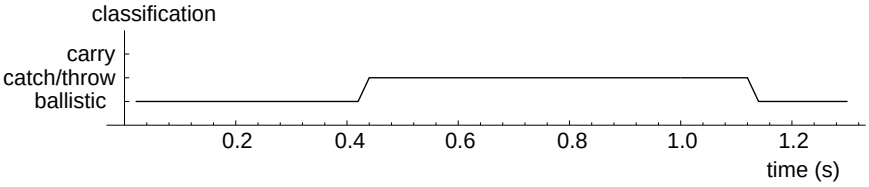
Table 7.1: **The system can distinguish between 'ballistic' and 'non-ballistic' motion, but rarely identifies a separate 'carry' phase.** *Out of sixteen test sequences, the proportion for which the tracking exhibited each given property is shown.*

It can be seen that the classification of the test sequences is not as good as that of the training sequence. In particular, for only one test sequence — the example shown in figure 7.15 — does the system correctly identify a 'carry' phase. The distinction between ballistic and non-ballistic motion of the ball is signalled in all cases but one.

Of the fifteen sequences without a clear 'carry' labelling, eleven exhibited rapid transitions between all three modes during the portion of the sequence where a human observer would place the 'carry' phase. This indicates the presence of uncertainty about the type

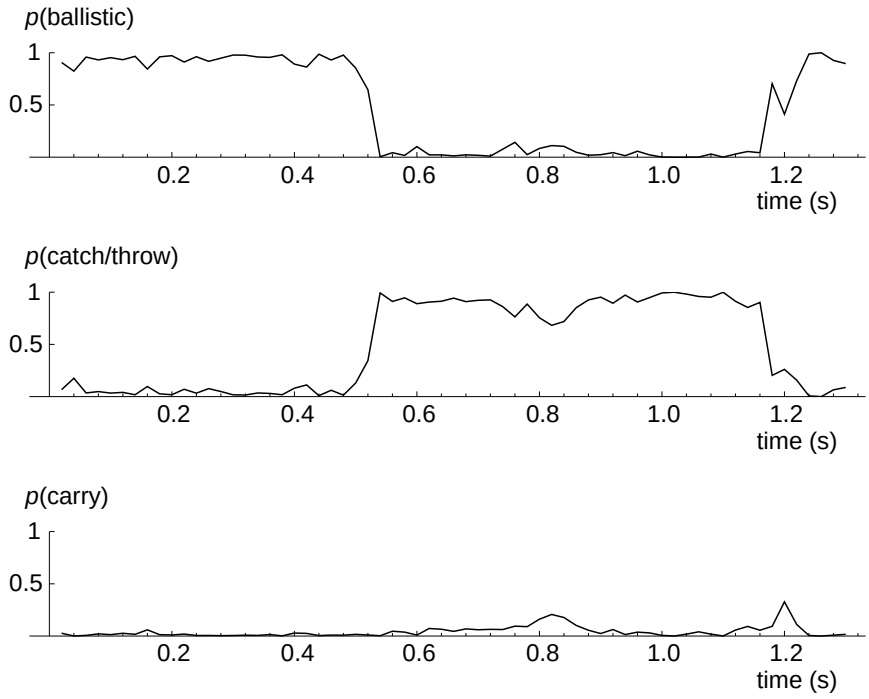


(a) Estimated probabilities of the three states

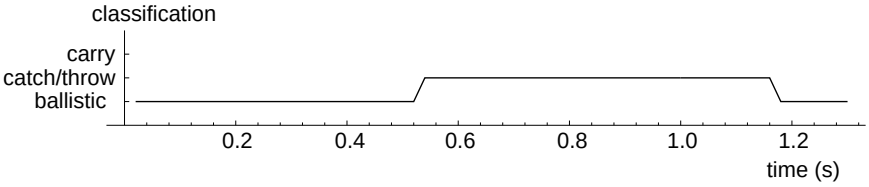


(b) Classification output

Figure 7.13: **Classification of first example test sequence detects ballistic and non-ballistic sections, but fails to signal a 'carry' phase.** *Although the transitions from 'ballistic' into 'catch' and from 'throw' into 'ballistic' are successfully marked, none of the sequence is labelled as 'carry'. There is a slight rise in the probability of 'carry' at the appropriate place in the sequence, but it is not high enough for a classification to result. One factor in this failure could be that the learnt 'carry' model is over-specific to the training sequence.*

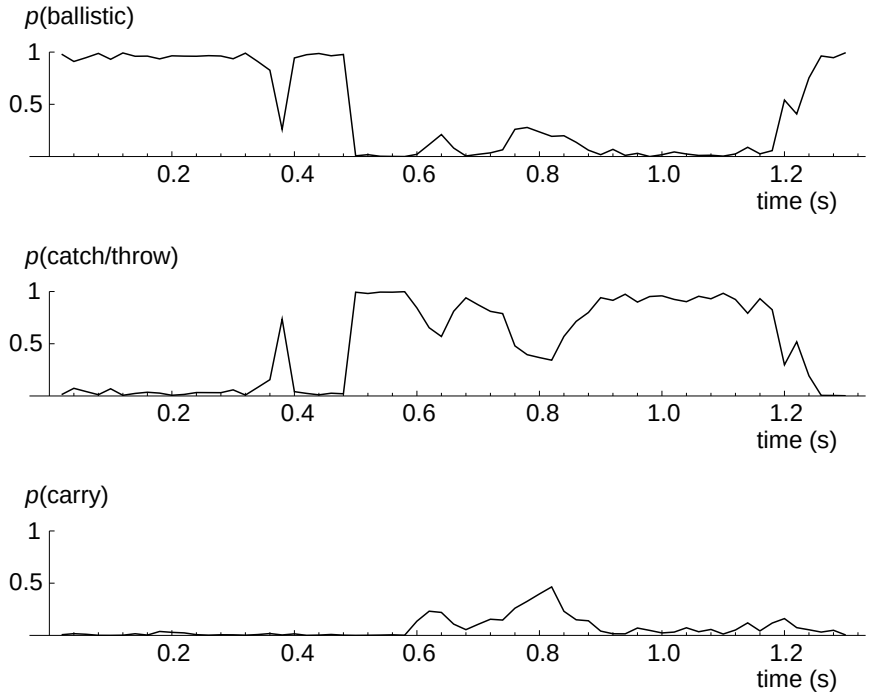


(a) Estimated probabilities of the three states

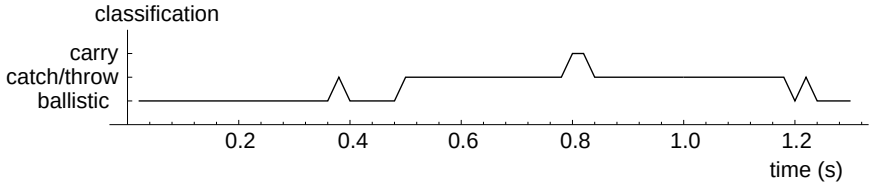


(b) Classification output

Figure 7.14: **Classification of second example test sequence shows similar behaviour to the first test sequence — no 'carry' phase is signalled.** *As for the test sequence shown in figure 7.13, ballistic and non-ballistic motions have been successfully marked, but no part of the sequence behaves sufficiently like 'carry' to be classified as such.*



(a) Estimated probabilities of the three states



(b) Classification output

Figure 7.15: **Classification of third example test sequence does flag a 'carry' phase, but is more noisy than the other two sequences.** *The sequence shows a short period of motion labelled as 'carry', but also displays two momentary tracking distractions at times* 0.38 s *and* 1.22 s*, where 'catch/throw' is signalled in the middle of ballistic motion.*

of motion being performed by the ball during this section of the sequences. Nevertheless, one must conclude that a 'carry' model of sufficient generalising power has not been learnt, whereas the model has successfully captured enough of the difference between 'ballistic' and 'non-ballistic' motion.

There are two possible causes for this performance of the classification system. One is that the training may have been too specific, producing a set of motion models which was too closely tuned for the training sequence. Attempting to classify other sequences using this model would then produce the results observed. The other is that the human division of the non-ballistic motion into 'catch', 'carry', and 'throw' may be rather artificial and not truly reflected in the motion of a juggled ball. Perhaps two states would be sufficient to model the ball's motion; this experiment, together with one where a system learnt from more than one training sequence was examined, could be the subject of future work.

7.1.5 Discussion

In general, the multi-model learning experiments described in this section have been successful. A model was learnt, in an unsupervised fashion, for a juggled ball as it underwent three different types of motion. Possible areas for improvement were revealed when an attempt was made to use the model for classification of unseen sequences. One reason for this could be that the learnt model was too specifically tuned to the (fairly short) training sequence presented to the system, so learning from more data would reduce this problem. This learning would take the form of providing the system with a collection of test sequences and using the results of appendix A.4.

Another experiment would be to attempt learning and classification of a longer contiguous sequence, for example of a ball undergoing juggling motion including both hands. Ideally, the difference between motion in the two hands would be detected.

7.2 More Complex Single-mode Motion — Hand-tracking

As well as being able to handle multi-mode systems, Condensation has the advantage of tracking objects over cluttered backgrounds which would cause a Kalman-filter-based tracker to fail. This section therefore describes some experiments on learning from sequences involving heavy clutter, heavier than that used in section 5.1.2 to test the Kalman-EM learning.

7.2.1 Preliminaries

A training sequence of 5 s length was recorded. Typical fields from the sequence are shown in figure 7.16, where the heavy nature of the clutter is evident. Features were detected by means of an edge-detection filter on the one-dimensional grey-level data, and the strongest one chosen for use in the calculation of the observation density.

In keeping with the aim of providing the learning system with as little prior information as possible, the starting dynamics for the learning experiment were constant-velocity, with the noise set more or less arbitrarily — a fairly high value must be used to allow the incorrect constant-velocity dynamics to track the motion.

The motion is adequately described by a 4-dimensional linear subspace of the complete spline space, that of Euclidean similarities. This consists of 2 dimensions covering translation, and two covering both scaling and rotating, in the form of the 'zoom' and 'rotated zoom' vectors first defined in section 2.2.3. The higher dimensionality of this tracking system provides a more rigorous test of the continuous learning aspect of the system.

The training sequence is considerably longer than those previously used — around ten times as long as the section of ballistic motion of the juggling ball. Therefore, bearing in mind the results of section 6.3.3, it was expected that, with 2048 samples, combining the moments from two runs would result in sufficiently low variance on the estimates.

7.2.2 Convergence of Model

With this length of sequence and combining the moments from two runs, the resulting behaviour of the EM algorithm is much more smooth than has been observed in previous cases, as can be seen from the graphs in figures 7.17 to figure 7.19. Most of the parameters exhibit fairly smooth convergence. Although the 'zoom' component of the process noise has some jitter, it is not so much that the final value can not be taken as a converged estimate. Note also that the final model is, for some parameters, quite different from the initial model used — the algorithm has converged from a distant starting configuration. This is particularly evident in the translation components of the process noise.

7.2.3 Simulation of Learnt Model

The final learnt model can be simulated by free-running the resulting system

$$\mathbf{x}_t = A_1\mathbf{x}_{t-1} + A_2\mathbf{x}_{t-2} + B\mathbf{w}_t$$

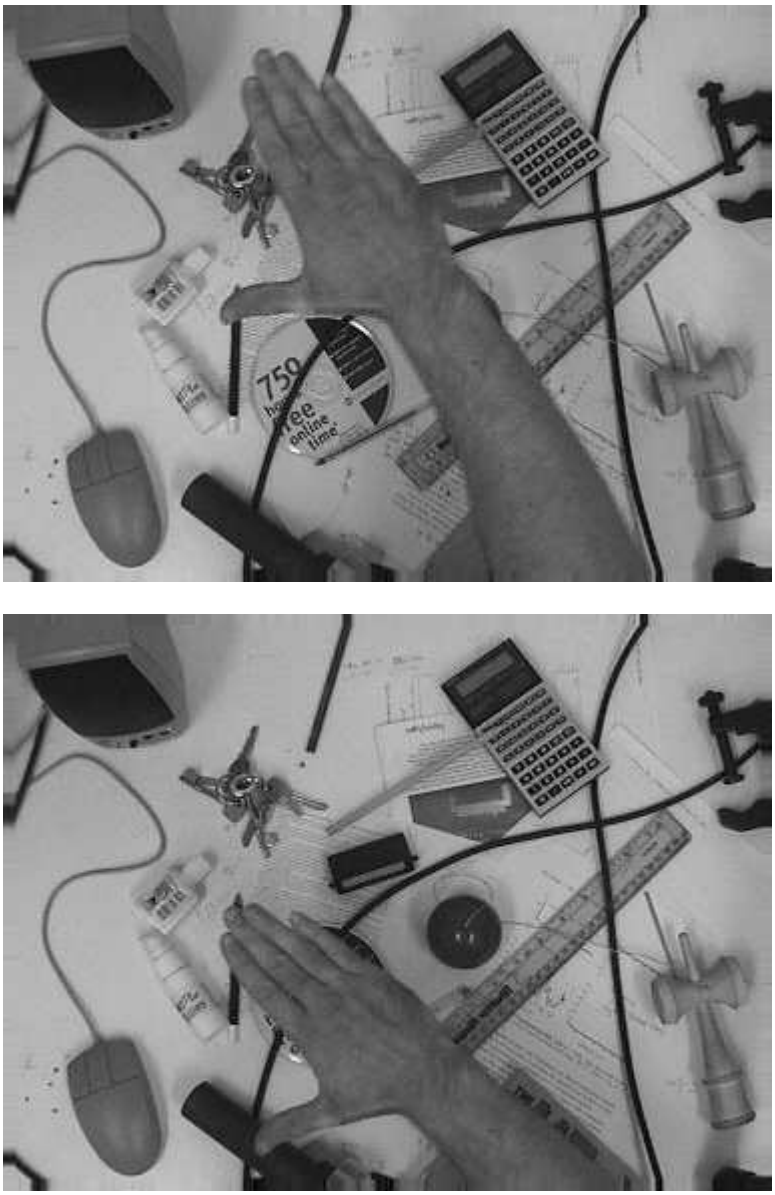


Figure 7.16: **Hand-tracking in heavy clutter requires a robust tracking framework.** *Learning will be performed on this sequence, which is of a hand translating and rotating, with some small amount of scaling.*

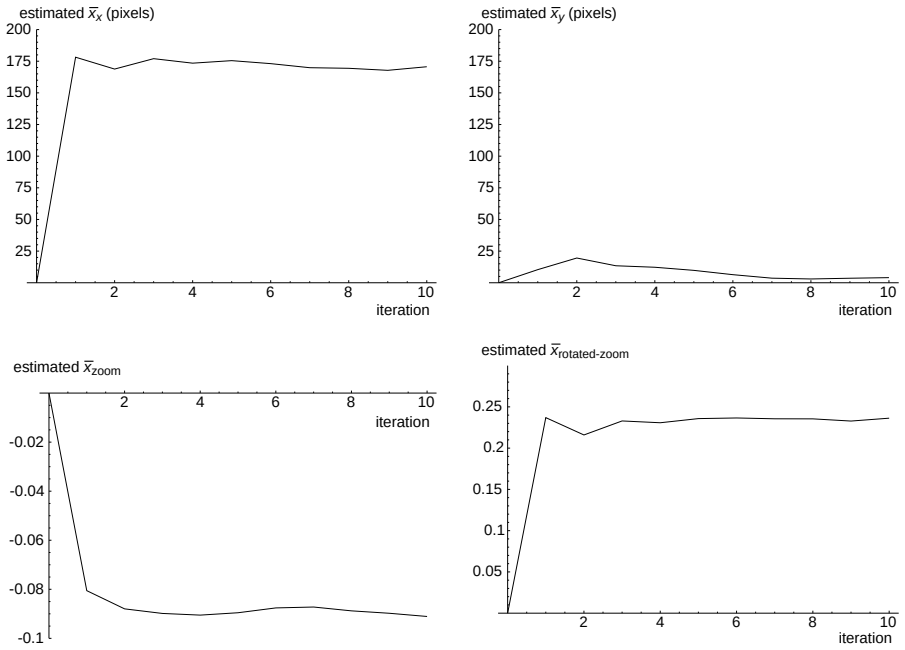
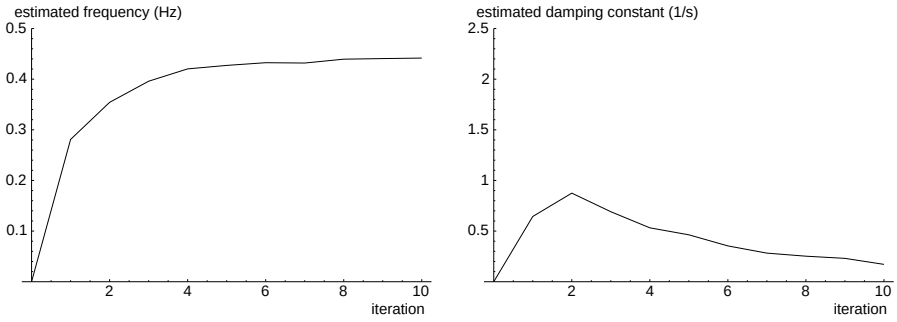
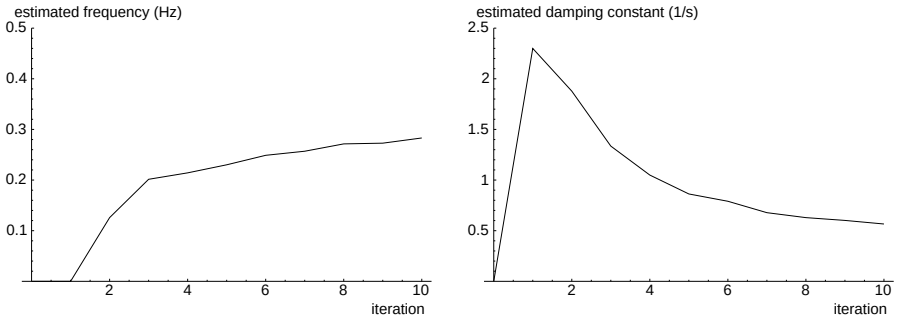


Figure 7.17: **Convergence of learnt model for hand tracked over clutter — system mean.** *The convergence of the system mean parameters of the dynamical model is quite swift, as has been observed with other learning experiments. Note that the 'zoom' and 'rotated zoom' components are effectively unitless, as they correspond to multiples of the template and rotated template respectively.*



Frequency and damping constant of first oscillatory mode



Frequency and damping constant of second oscillatory mode

Figure 7.18: **Convergence of learnt model for hand tracked over clutter — oscillatory modes.** *Frequency (Hz) and damping constants (s^{-1}) are shown for the two most persistent modes of the learnt system. The initial conditions of constant-velocity motion can be seen as having zero frequency and zero damping constant. Convergence is not as rapid as for the system mean, and in fact it is not entirely clear that the second mode would not continue to increase slightly in frequency were more iterations to be performed. However, the dominant mode appears very close to its asymptotic value.*

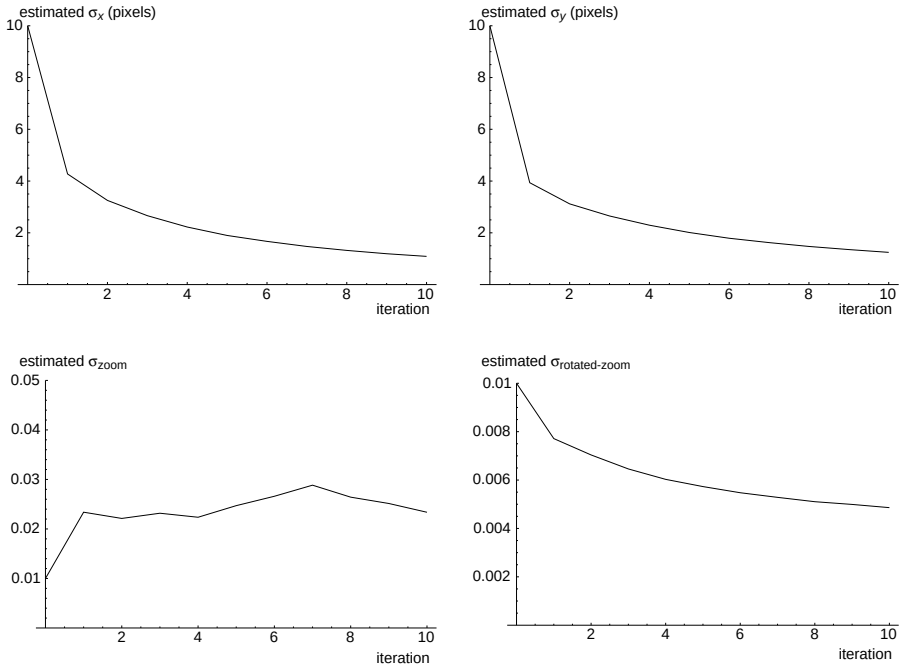


Figure 7.19: **Convergence of learnt model for hand tracked over clutter — system noise per time-step.** *As was expected from the behaviour of Condensation-EM learning in previous experiments, the noise parameter displays the most erratic behaviour. However, the longer training sequence and the* $2\times$ *oversampling used in this experiment has produced an acceptably low amount of jitter, and it is reasonable to claim that the final values represent a converged system.*

and qualitatively comparing the behaviour with the training sequence. A typical 5 s trajectory from such a simulation is shown in figure 7.20. Note that for these graphs, the zoom and rotated-zoom components of the state-space representation have been transformed to 'pure' scaling and rotation.

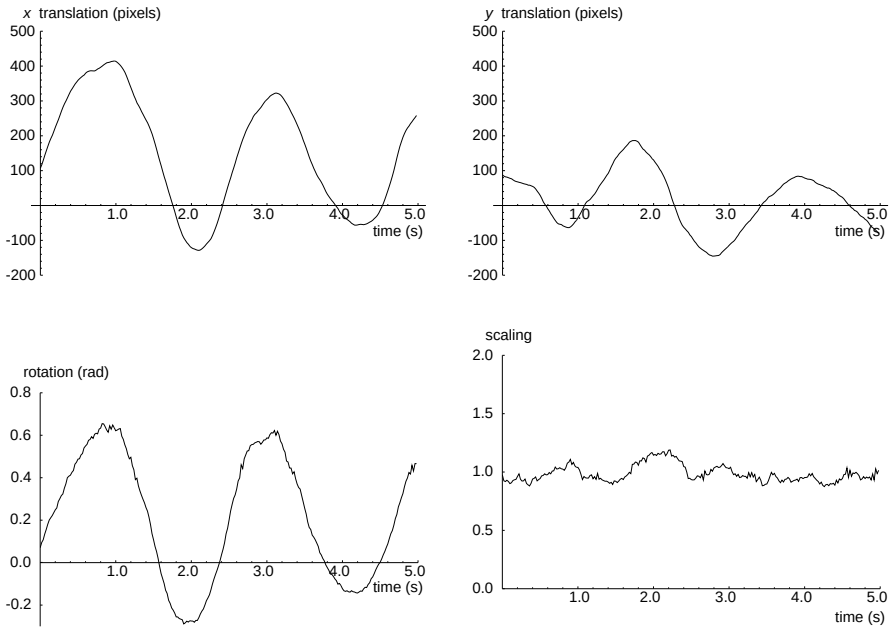


Figure 7.20: **Simulating the learnt system — a trajectory qualitatively similar to the training sequence is produced.** *Shown are the translation, rotation, and scaling components of the simulated motion. These graphs should be compared with those in figure 7.22 — the motion is similar. In particular, the periodicity (both frequency and amplitude) and very close correlation between the x translation and the rotation has been captured successfully.*

One observation to be made about figure 7.20 is that the amplitude of variation of the scaling parameter is higher than that in the training sequence. This is a limitation of the shape-space model used, and the linear dynamical model used. This is made clearer by figure 7.21.

Switching to a parametric representation which explicitly models rotation and scaling would avoid this restriction. The Condensation framework allows essentially arbitrary representations of state, so this would be possible.

7.2.4 Tracking Results

The behaviour of a Condensation tracker using the final learnt model on the training sequence is first presented. This will provide verification that the learnt model does indeed

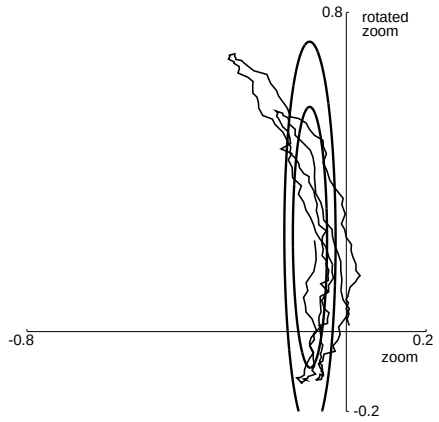


Figure 7.21: **A linear dynamical model can not accurately represent motion consisting mostly of rotation.** *The projection of the hand's trajectory during the training sequence onto the zoom/rotated-zoom plane is shown, together with* 2σ *and* 3σ *ellipses of the dynamical model's steady-state. While these are reasonable from the point of view of capturing most of the training trajectory, they also contain points representing significant pure scaling, as seen in the simulated system of figure 7.20.*

describe the motion observed in the training sequence.

Figure 7.22 shows the translation component of the hand's motion during successful tracking using very few samples — 32 as opposed to the 2048 used for learning purposes. Figure 7.23 shows snapshots, where it can be seen that the agreement between the tracked outline and the image is good.

Figure 7.22 also shows a marked (but not perfect) periodicity in the traces, with around 2.25 cycles occurring during the 5 s sequence. This corresponds to a frequency of around 0.45 Hz, matching closely the converged frequency of the dominant mode as shown in figure 7.18.

Taking these results in combination with those of the previous section, a fair conclusion is that the learnt model has captured the important features of the training sequence.

7.2.5 Performance on a Test Sequence

The learnt dynamical model was tested on a separately-recorded 5 s sequence, to evaluate its performance on unseen data. The test sequence contained motion of roughly the same type as the training sequence.

Performance on the test sequence was, unsurprisingly, not as good as on the training sequence, but good nonetheless. Tracking was not successful with $N = 32$ samples per time-step, but increasing this number to $N = 64$ produced successful tracking. This is

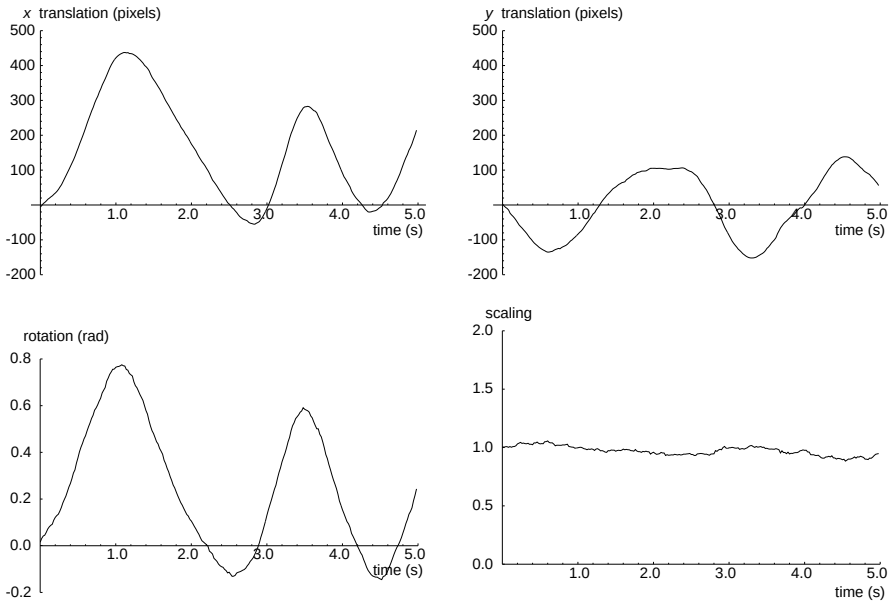


Figure 7.22: **The learnt dynamical model allows tracking of the training sequence using as few as $N = 32$ samples.** *Using the dynamical model learnt using the Condensation-EM algorithm, the training sequence can be tracked using a very small sample set. Estimated translation, rotation and scaling of the hand over time are shown. Figure 7.23 shows that these estimates are in good agreement with the images.*

still a very low number of samples for a Condensation tracker; more typical values are of the order of a thousand or even more for complex sequences. The tracked trajectory and snapshots from the tracking are shown in figures 7.24 and 7.25.

For comparison, the constant-velocity model used as a starting point for the learning was also employed on this test sequence, using the same number — $N = 64$ — of samples per time-step. Tracking with such small sample-sets and an untuned dynamical model was not successful; the estimate was approximately correct for the first 1 s, but thenceforth the estimated outline diverges significantly from the image of the hand, and lock is lost. This is illustrated in figure 7.26, where the remainder of the sequence is not shown — the estimated contour continues to rapidly spiral off the image.

7.2.6 Performance on Multiple Test Sequences

The performance of the learnt dynamical model on the single test sequence described in section 7.2.5 cannot, by itself, be used to draw quantitative conclusions concerning the benefit over a constant-velocity dynamical model. Therefore, 16 additional 5 s test sequences (of a similar type to the training and previous test sequence) were recorded, and their tracking

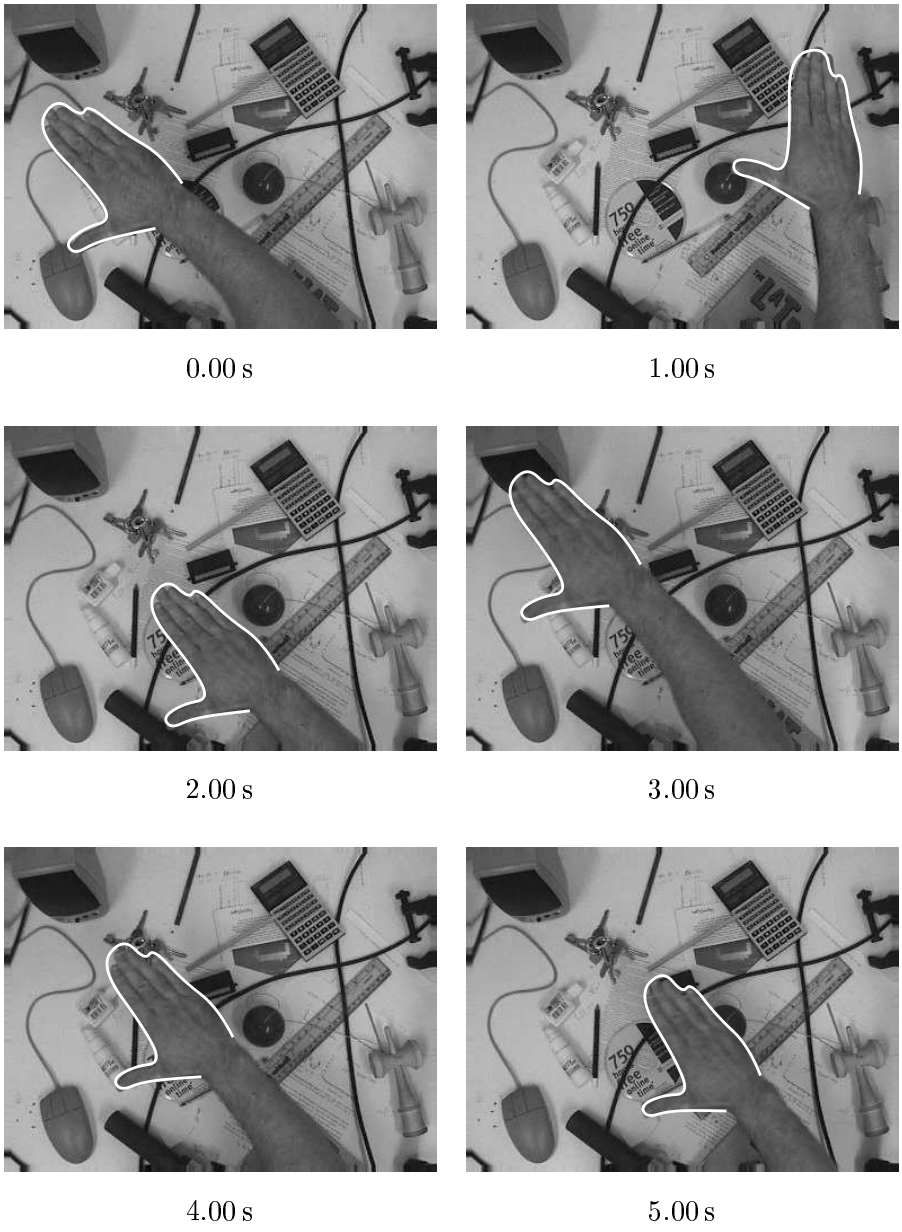


Figure 7.23: **Tracking the training sequence with 32 samples — agreement with image is good.** *Snapshots taken during the successful tracking described in figure 7.22 show that the estimates produced by the Condensation tracker are good.*

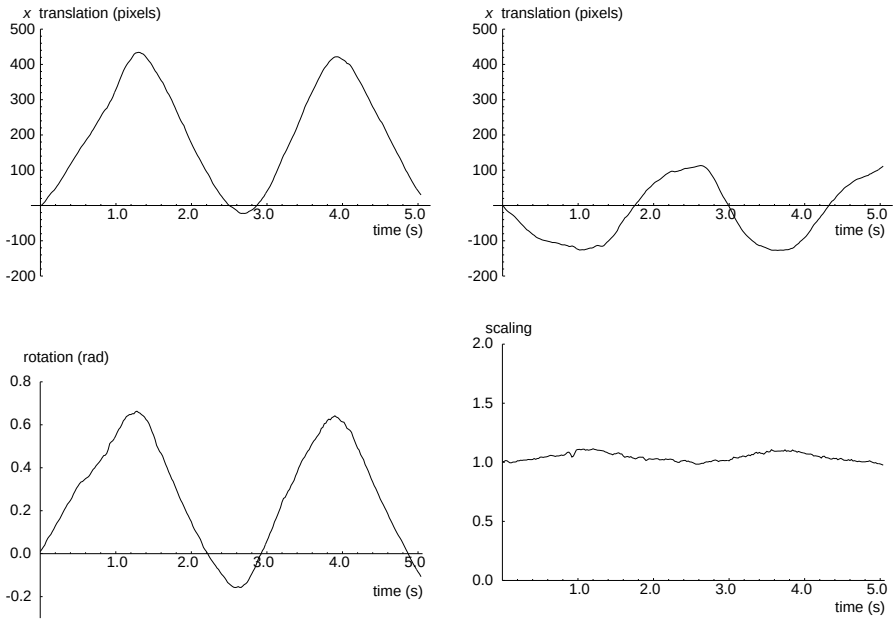


Figure 7.24: **The learnt dynamical model successfully tracks a test sequence using only $N = 64$ samples per time-step.** *The estimated x- and y-translation, rotation and scaling are plotted against time. See also figure 7.25, where the accuracy of the tracking is seen to be good.*

attempted using both dynamical models. $N = 64$ samples were used — as noted, this is quite low, but the experiments of the section 7.2.5 indicate that it should be sufficient for at least the learnt dynamical model. The sequences were labelled as 'successful' or 'failed' according to whether the tracked estimate remained locked onto the outline of the hand. The results of such a classification for each dynamical model are shown in table 7.2.

Dynamical model	**Proportion of test sequences successfully tracked**
Constant-velocity	0
Learnt	0.8125

Table 7.2: **The learnt dynamical model achieves a high success rate on multiple test sequences; a constant-velocity model consistently fails.** *Using $N = 64$ samples per time-step, 16 independent test sequences were tracked using both the learnt dynamical model and a constant-velocity one. The proportion of the test sequences which were tracked successfully is shown. It can be seen that a constant-velocity model is totally inadequate for this problem, whereas the learnt model performs well.*

Writing p_{EM} and p_{CV} for the (population) proportion of sequences of similar motion to the training sequence which are successfully tracked using the learnt model and the

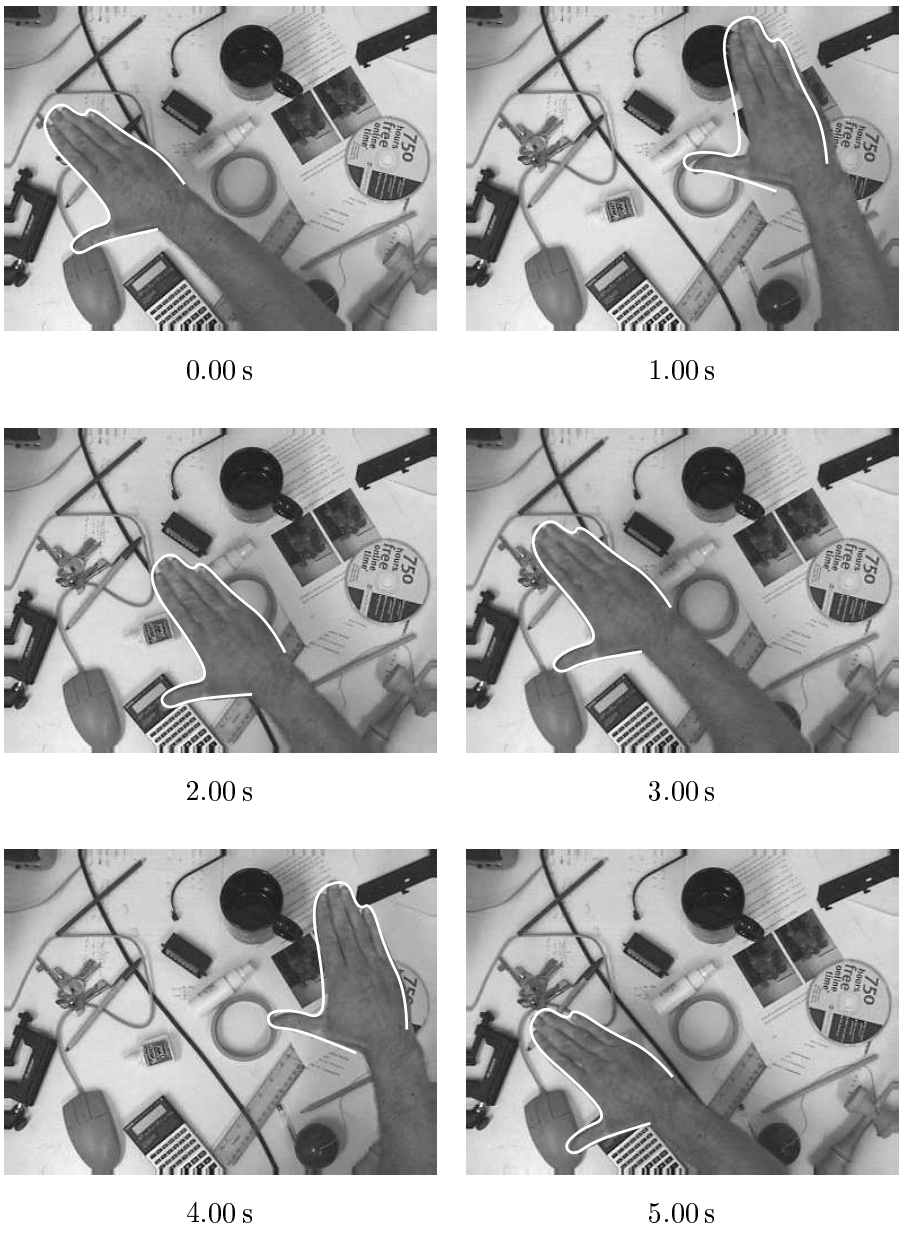


Figure 7.25: **Snapshots from successful tracking of test sequence.** *A test sequence is tracked using the model learnt using EM learning. $N = 64$ samples per time-step are used here, a very low number. That the tracking is successful with so few samples indicates the predictive accuracy of the dynamical model.*

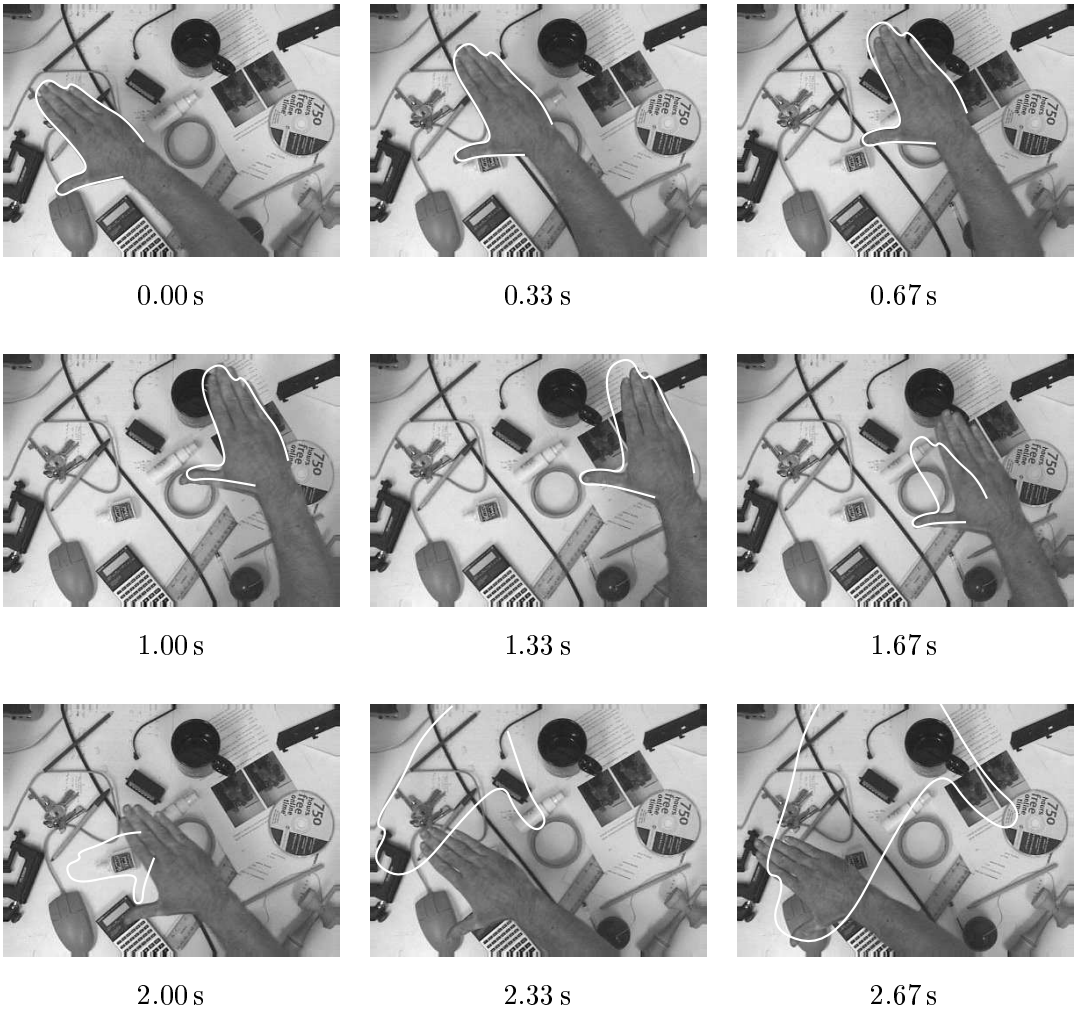


Figure 7.26: **Using a constant-velocity model on the test sequence fails.** *Using $N = 64$ samples per time-step, the same number as used for the successful tracking in figure 7.25, a tracker using a constant-velocity dynamical model fails to track the sequence for longer than* 1 s *or so. Even at* 0.67 s*, some divergence between the estimated position and the image can be observed. Beyond around* 1 s*, tracking fails, and the estimate essentially follows the constant-velocity prediction from around* 2 s *onwards.*

constant-velocity model respectively, the hypothesis

$$H_0 : p_{EM} = p_{CV}$$

can be tested as follows (Hoel, 1966). Write p'_{EM} and p'_{CV} for the sample proportions of successfully tracked sequences using the learnt model and constant-velocity model respectively, and n_{EM} and n_{CV} for the two sample sizes (which are the same here). Then

$$n_{EM} = 16; \qquad p'_{EM} = 0.8125; \qquad n_{CV} = 16; \qquad p'_{CV} = 0.$$

Under H_0, the difference $p'_{EM} - p'_{CV}$ may be considered to be approximately normally distributed with mean zero and variance

$$\sigma^2_{p'_{EM}-p'_{CV}} = pq\left(\frac{1}{n_{EM}} + \frac{1}{n_{CV}}\right),$$

where $p = p_{EM} = p_{CV}$ is the population proportion (since under H_0, the proportions p_{EM} and p_{CV} are the same), and $q = 1 - p$. The value of p is not known, so it must be approximated by the sample proportion p' of the combined data; here, $p' = 0.40625$. Then

$$\sigma^2_{p'_{EM}-p'_{CV}} = 0.03015.$$

A two-sided critical region of size 0.01 is therefore given by

$$\begin{aligned} |p'_{EM} - p'_{CV}| &> 2.576\sigma_{p'_{EM}-p'_{CV}} \\ &= 0.4473. \end{aligned}$$

(Although it is hoped that the learnt model will provide better tracking than the constant-velocity one, i.e., $p_{EM} > p_{CV}$, there is no justification for including this hope in the formulation of the alternative hypothesis. Therefore, the alternative hypothesis used is $H_1 : p_{EM} \neq p_{CV}$, meaning that a two-sided critical region is to be considered). The sample value of

$$|p'_{EM} - p'_{CV}| = 0.8125$$

falls inside the critical region, and so H_0 is rejected at the 1% level. One can therefore conclude with considerable confidence that the learnt model makes a difference to tracking performance, and it is clearly an improvement.

Note that the situation here is different to that in section 7.1.4, where the learnt model for the juggled ball was tested. For the hand tracking considered here, there are two models (the learnt one and the constant-velocity one), and a comparison between them is desired.

There is no sensible non-learnt model for the juggled ball of section 7.1.4 (the initial model's states are not meaningful), and so it is not possible to carry out a hypothesis test of this kind on the results of those experiments.

Typical snapshots from test sequences successfully tracked using the learnt model are shown in figures 7.27 and 7.28. Figure 7.29 shows an example of minor misalignment of the estimated hand outline which occurred in some of the test sequences, when tracking with the learnt dynamical model. Lock was not lost, however, and tracking continued successfully. Figures 7.30 and 7.31 show examples of tracking failure when the constant-velocity model is used.

7.2.7 Discussion

The experiments described in this section have shown the successful application of the Condensation-EM learning algorithm to a problem involving more complex motion in a higher-dimensional shape space. The benefit of learning from a longer training sequence is apparent from the point of view of the quality of the convergence of the learnt model. This comes at a considerable computational penalty, however — each iteration takes around eight hours. Tracking the sequence with 2048 samples takes around an hour, and smoothing takes around a further three hours. This is done twice per EM iteration, giving eight hours. (The final 'Maximisation' calculation takes negligible time compared to these tracking and smoothing times.) These times are, as before, using a 175 MHz R10k SGI Octane workstation.

7.3 Discussion

Some issues arising from the experiments on Condensation-EM learning in more complex situations are now discussed.

7.3.1 History-based Smoothing

It is apparent that the most costly part of the Condensation-EM learning algorithm is the smoothing phase, containing an $O(TN^2)$ calculation, where T is the length of the training sequence, and N the number of samples used. Since T and especially N are typically quite large, this can take a significant time.

Isard and Blake (1998d), following Kitagawa (1996), describes a second smoothing technique which is less costly in terms of computational time. Each sample-site, instead of containing just enough history to allow prediction (and one further time-step for learning

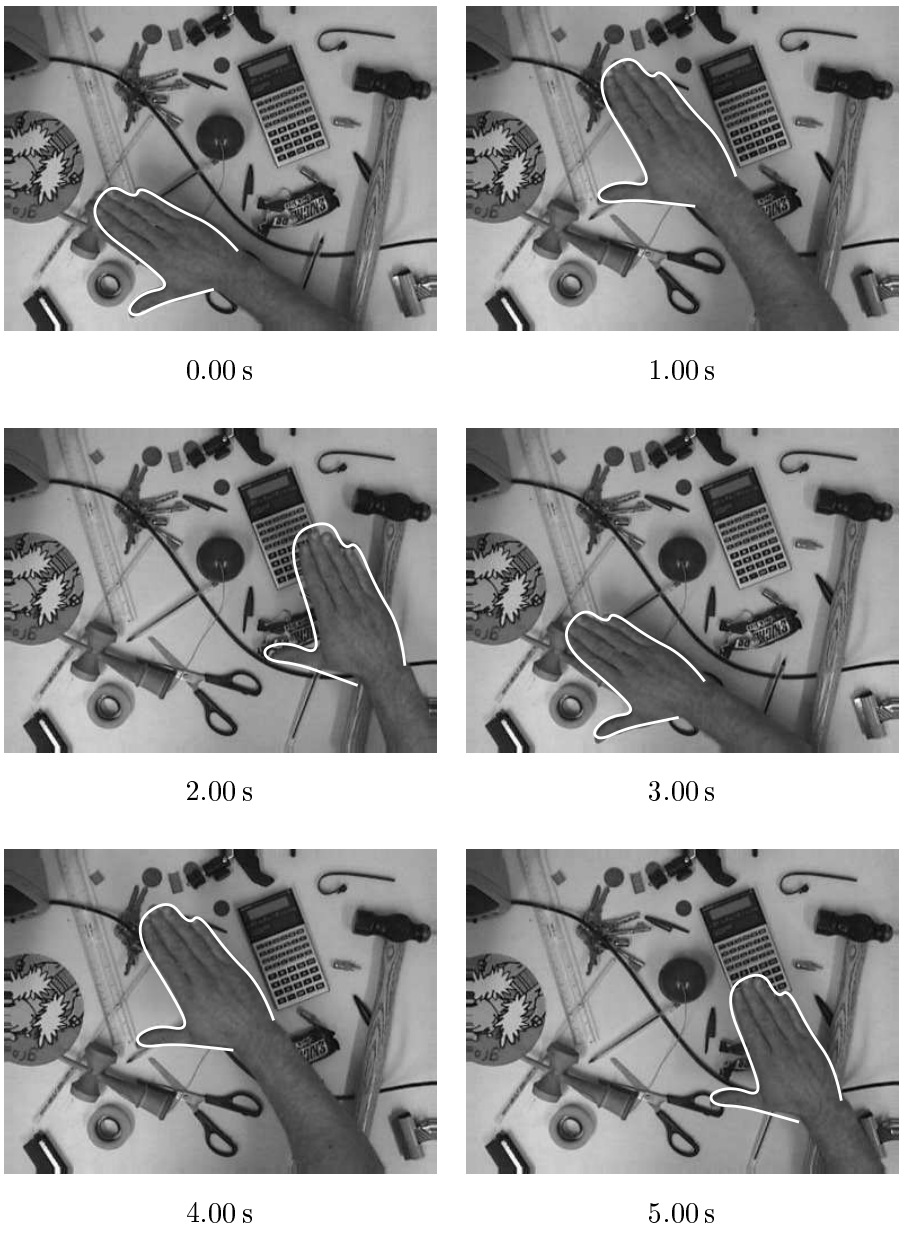


Figure 7.27: **Typical behaviour of learnt dynamical model on test sequence — first example.** *A successful tracking run on one of the test sequences used to examine the performance of the two dynamical models more qualitatively. A tracker incorporating the learnt dynamical model performs well with* $N = 64$ *samples.*

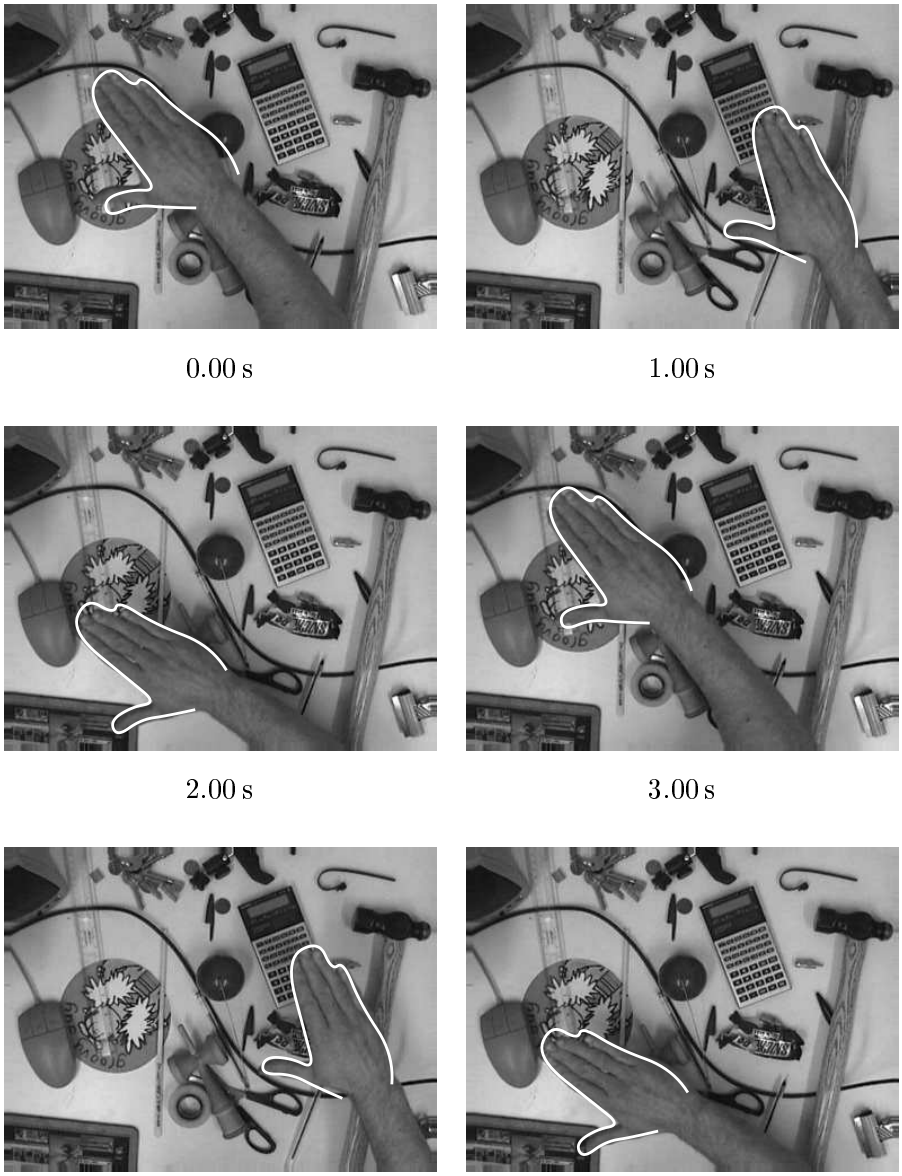


Figure 7.28: **Typical behaviour of learnt dynamical model on test sequence — second example.** *Another successful tracking run, similar to that shown in figure 7.27. Again, a tracker using the learnt dynamical model produces good estimates of the hand's position with* $N = 64$ *samples.*

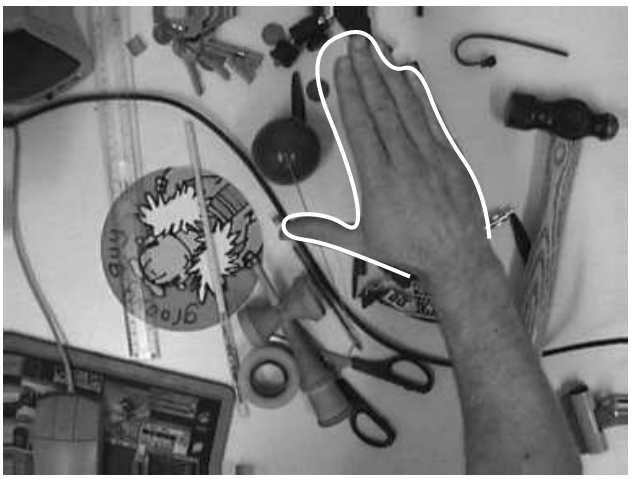


Figure 7.29: **Although tracking is not always perfect, lock is maintained.** *Some of the successfully tracked test sequences, using the learnt dynamical model, exhibited brief inaccuracies, as shown in this example. They were not severe enough to cause loss of lock, however, and the tracking continued.*

purposes), contains history all the way back to $t = 1$. The sample set at time t is then

$$\{\big((\mathbf{X}_{1\,|\,t}^{(n)}, \ldots, \mathbf{X}_{t\,|\,t}^{(n)}), \pi_t^{(n)}\big), n = 1, \ldots, N\},$$

representing $p(\mathcal{X}_1^t \,|\, \mathcal{Z}_1^t)$. The final sample set, at $t = T$,

$$\{\big((\mathbf{X}_{1\,|\,T}^{(n)}, \ldots, \mathbf{X}_{T\,|\,T}^{(n)}), \pi_T^{(n)}\big), n = 1, \ldots, N\},$$

then represents the distribution $p(\mathcal{X}_1^T \,|\, \mathcal{Z}_1^T)$, which is the joint distribution for all time-steps $\mathcal{X}_1^T$ of the state sequence given the entire measurement sequence $\mathcal{Z}_1^T$.

The drawback of this method is that in practice, the ancestries for the sample sites merge within relatively few time-steps back from t. The history components $\mathbf{X}_{t\,|\,\tau}$ for $\tau - t > 10$ or so have many fewer distinct values than the size N of the sample set. This is illustrated in figure 7.32, where the number of distinct locations of the sample sites is shown as a function of the difference $\tau - t$.

The chief consequence of this reduction in the number of distinct sample-sites further back in the history, from the point of view of learning dynamical models, is that the approximations to second-order statistics become very inaccurate, as seen in section 6.3. The statistics of particular interest for learning are the second-order moments R_{ij}, but the loss of accuracy effects quantities such as the variance of the state as well. This phenomenon was observed by Kitagawa (1996).

One reason for the reduction is the method used for choosing a base sample. Choosing a base sample is in essence a matter of choosing a number s in $(0, 1)$ and finding the lowest m

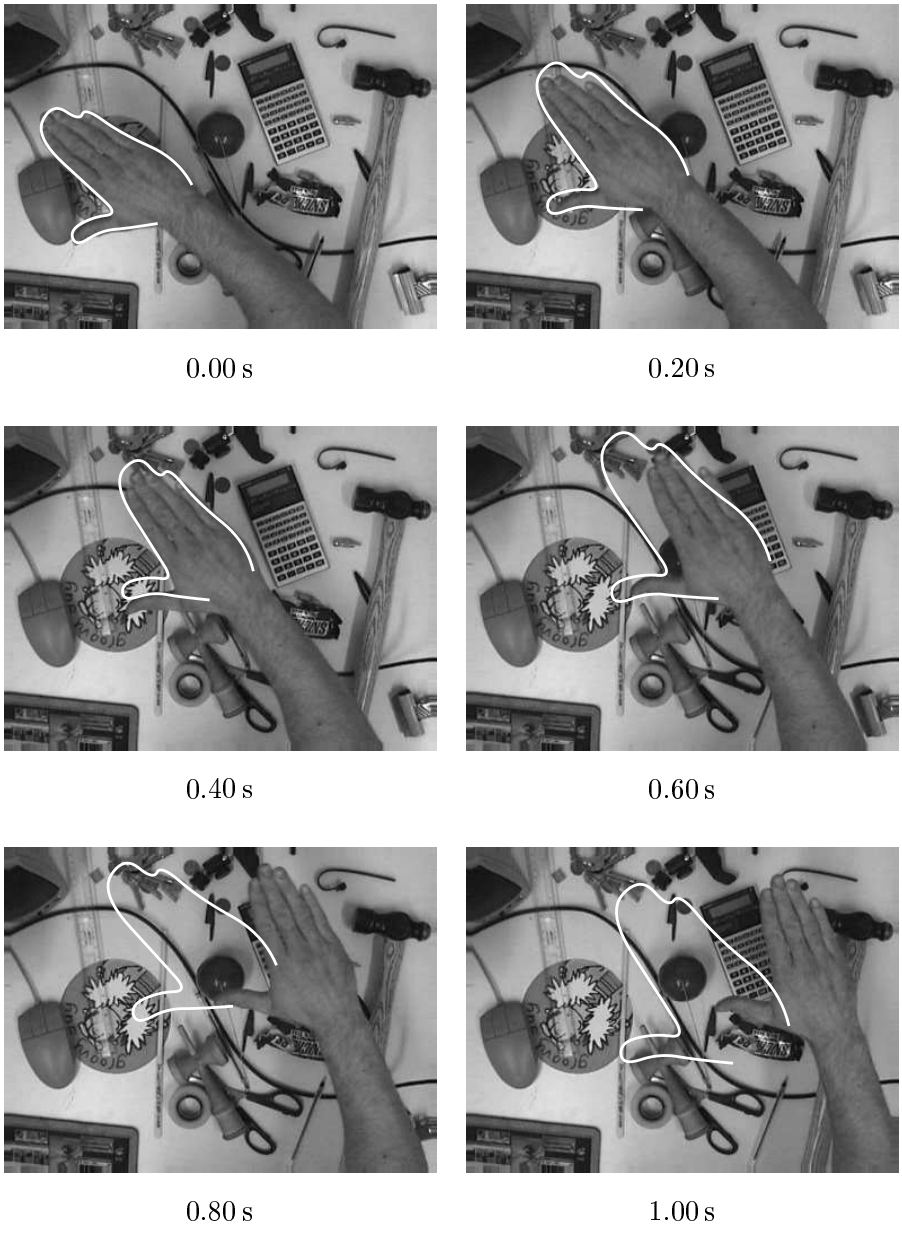


Figure 7.30: **Typical behaviour of constant-velocity dynamical model on test sequence — first example.** *A constant-velocity tracker rapidly loses lock.*

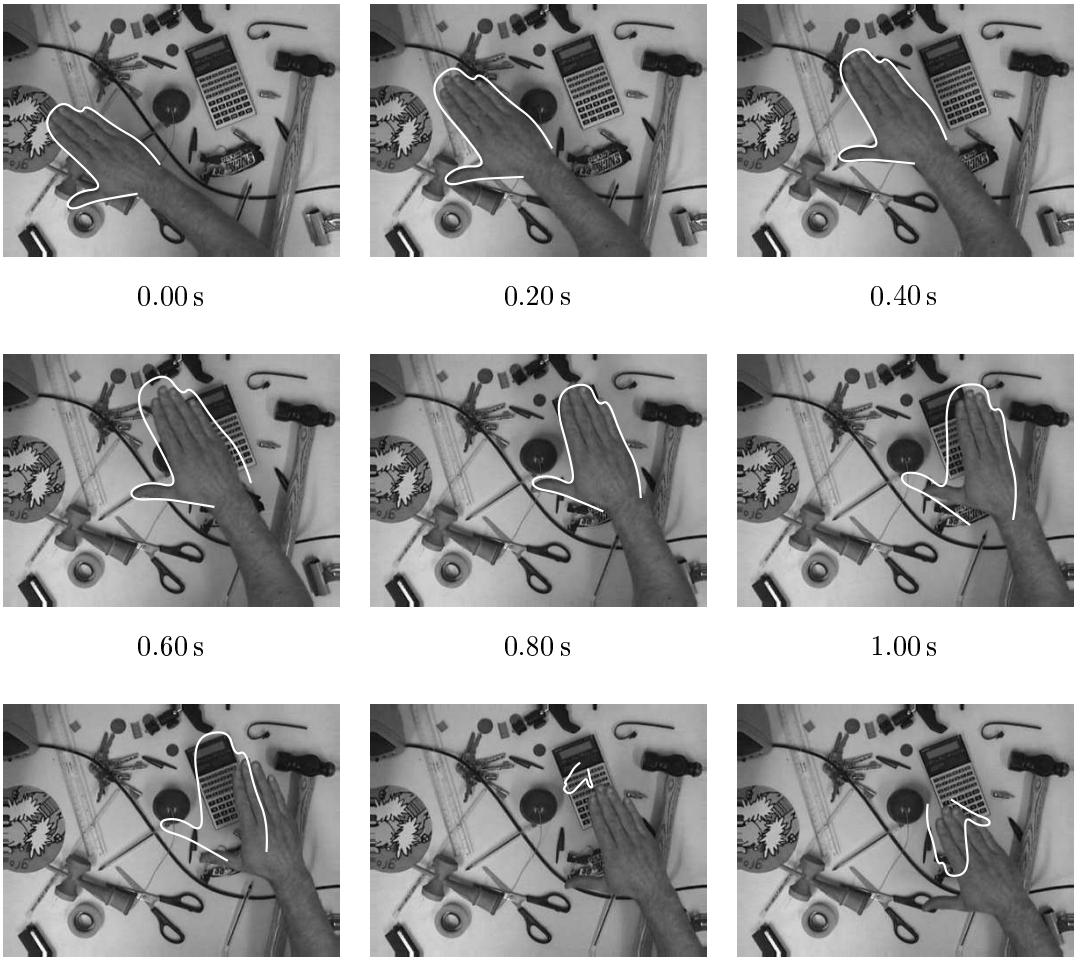


Figure 7.31: **Typical behaviour of constant-velocity dynamical model on test sequence — second example.** *A constant-velocity tracker maintains lock on the hand for slightly longer in this sequence, but tracking still fails.*

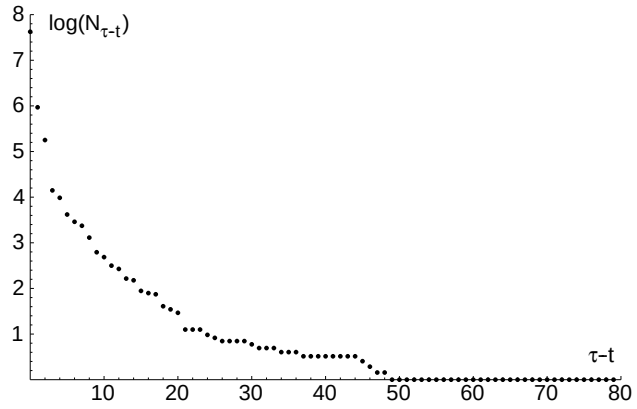


Figure 7.32: **The number $N_{\tau-t}$ of distinct sample sites rapidly dwindles as $\tau - t$ increases.** *A tracking run using the 'history smoothing' was performed on the juggling sequence, and the average number of distinct sample-sites computed over three runs. $N =$ 2048 samples were used.*

which satisfies $\sum_{i=1}^{m} \pi^{(i)} > s$. In the random sampling scheme used, s is chosen by drawing from $U(0, 1)$. A natural consequence of this is that base samples with large π get chosen more often, while those with smaller π may not be chosen at all.

Another sampling scheme would be to choose, for sample n, $s = (2n - 1)/2N$, thus ensuring that every base sample with $\pi > 1/N$ gets chosen at least once. The effect of this sampling scheme on the sample-set decay is shown in figure 7.33.

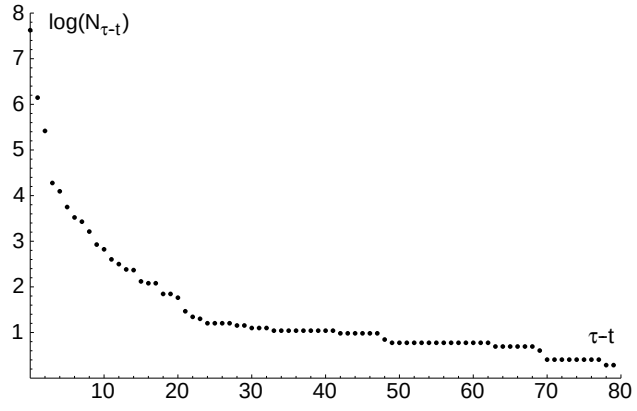


Figure 7.33: **Deterministic sampling produces only slightly slower decay of sample-set size than random sampling.** *Replacing the random sampling scheme with a deterministic one does ameliorate the sample-set-dwindling problem, but does not remove it.*

It can be seen that $N_{\tau-t}$ decreases less rapidly with increasing $\tau - t$ than for random sampling, but that it still becomes unacceptably small within a few tens of time-steps. Given

the importance of obtaining good estimates of $\mathcal{E}[R_{ij}]$ etc., then, the benefit of reduced computation time is outweighed by the much greater disadvantage of loss of accuracy. Unfortunately, then, using history smoothing does not seem a viable alternative to the more computationally expensive forward-backward smoothing algorithm for determining the posterior distribution $p(\mathcal{X}_1^T \,|\, \mathcal{Z}_1^T)$.

7.3.2 Limited-history Smoothing

A smoothing method which might combine the benefits of both the history-based and forward-backward techniques is as follows. Suppose one were to let the sample set at time t be

$$\{(\mathbf{X}^{(n)}_{(t-K')\,|\,t}, \ldots, \mathbf{X}^{(n)}_{t\,|\,t}), \pi_t^{(n)}\}, n = 1, \ldots, N\},$$

where $K' \geq K$ is a constant controlling how much of the history is kept. This sample set represents the distribution

$$p(\mathbf{X}_{t-K'}, \ldots, \mathbf{X}_t \,|\, \mathcal{Z}_1^t).$$

Now, this can be re-written

$$p(\mathbf{X}_t, \mathbf{X}_{t+1}, \ldots, \mathbf{X}_{t+K'} \,|\, \mathcal{Z}_1^{t+K'}),$$

and from this can be extracted, for example, the marginal distribution $p(\mathbf{X}_t \,|\, \mathcal{Z}_1^{t+K'})$. As K' becomes large, it is plausible that this distribution tends to the smoothed distribution $p(\mathbf{X}_t \,|\, \mathcal{Z}_1^T)$, if the influence of measurements at time $t' > t$ diminishes rapidly as $t' - t$ becomes large.

At the two extremes of the range of possible values for K' are the following choices. For $K' = K$, there is minimal smoothing, whereas if K' is allowed to depend on t and one sets $K'(t) = t - 1$, the entire history is stored and the case of history-based smoothing results.

It might be hoped that intermediate values of K', say $K' = 2K$, will produce a reasonable approximation to the smoothed density while avoiding the $O(N^2)$ step in the forward-backward smoothing algorithm. In fact, there would be no post-processing step at all, as in pure history-based smoothing.

Use of the resulting approximations to the smoothed distributions at each time step would be used in the calculation of the expected values of the moments by using the $\mathbf{X}_t$ as far back as possible within each time-step. Detailed investigation of this approach is deferred to future research.

7.3.3 Reduced Forward-backward Smoothing

The expensive step in the forward-backward smoothing algorithm of figure 6.2 is step 2, the calculation of the correction factors $\gamma_t^{(m)}$; there are N of them, and the calculation of each one requires the summation of N terms:

$$\gamma_t^{(m)} = \sum_{n=1}^{N} \pi_t^{(n)} \alpha_t^{(m,n)} \qquad \text{for } m = 1, \ldots, N.$$

In practice, the set of weights $\{\pi_t^{(n)}\}$ contains a relatively small number of significant values, and many small (with hopefully negligible contributions to the sum) values. This is illustrated in figure 7.34.

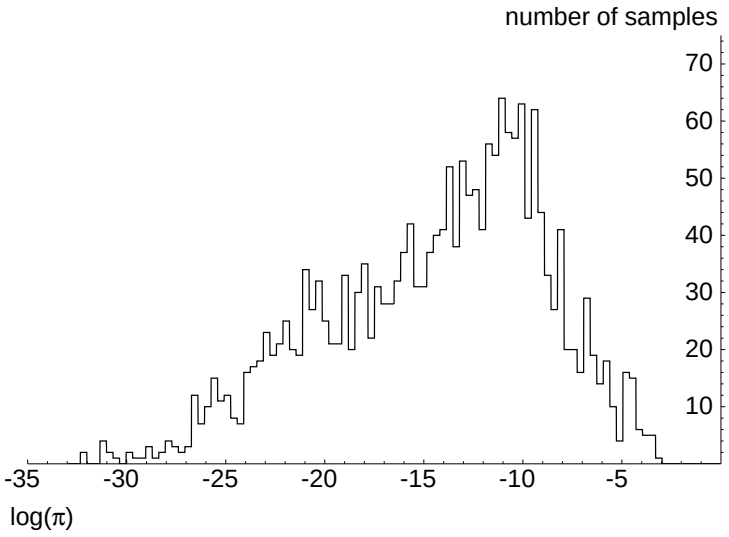


Figure 7.34: **Histogram showing distribution of $\log \pi^{(n)}$ for a typical time-step in a Condensation tracking run.** *A sample-set size of $N = 2048$ was used for this experiment. See also figure 7.35.*

Shown in figure 7.35 is the cumulative probability against fraction of sample set summed, after sorting into descending order of the $\pi^{(n)}$.

One obvious approximation, then, is to calculate the sum over only the largest N' of the $\pi_t^{(n)}$, for some $N' \ll N$. Determining which of the samples constitute the set of the largest N' is not free, of course, but it can be done efficiently — in $O(N \log N')$ time. The cost of computing the N factors $\gamma_t^{(m)}$ is then a one-off cost of $O(N \log N')$ for the selection plus a cost of $O(NN')$ for the summations. The overall cost is thus $O(NN')$. Setting N' to around $N/10$ still produces an $O(N^2)$ algorithm, but one which runs around 10 times as fast for any given N. This constitutes a significant improvement.

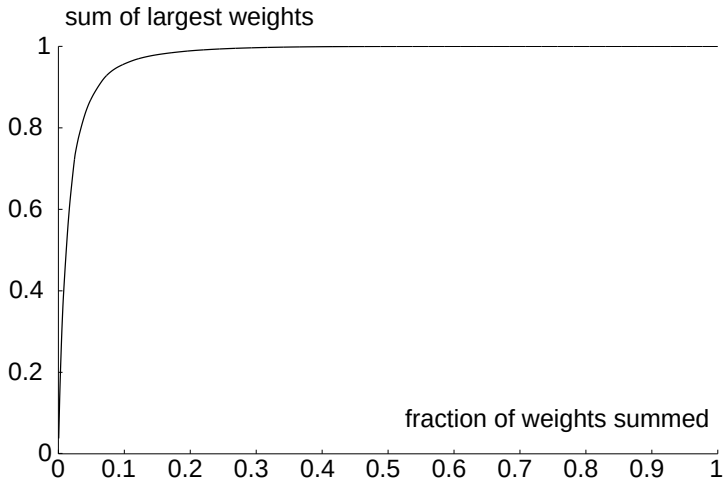


Figure 7.35: **The largest 10% of the samples typically represent over 95% of the distribution's weight.** *When sorted by size, a small proportion of the samples is enough to cover most of the weight of the distribution. A sample-set size of* $N = 2048$ *was used.*

Again, implementation and investigation of this idea is deferred to further research. In particular, generalisation of the phenomenon illustrated in figure 7.35 to other sample-set sizes would be necessary. The expectation would be that the great time savings would be at the cost of a very small decrease in the quality of the approximations.

7.3.4 Behaviour of One-dimensional Condensation

As a further investigation of the quality of the sample-set approximation to the expected values of the second-order moments, a much simpler situation was examined. Consider a 1-dimensional process consisting of a random walk driven by Gaussian noise. In this case, the Kalman filter provides an exact analytical solution to the problem of propagating the density forwards through time. Comparing the 'ground truth' values of the posterior mean and variance with the estimates obtained from the Condensation sample set will give insight into the quality of the approximation.

Consider, then, a 1-dimensional process of the form

$$x_t = Ax_{t-1} + Bw_t$$

where w_t is a Gaussian random variable together with a linear measurement process with Gaussian noise:

$$z_t = Hx_t + v_t.$$

In this case, the posterior distributions

$$p(x_t \mid z_1, z_2, \ldots, z_t)$$

for the filtered densities can be calculated analytically in a recursive fashion: the Kalman filter.

To facilitate comparisons with the Condensation filter, a particularly simple example system was chosen. Consider the one-dimensional random walk

$$x_t = x_{t-1} + w_t$$

where $w_t \sim N(0, 1)$. Suppose the measurement process is also simple:

$$z_t = x_t + v_t$$

where the measurement noise $v_t \sim N(0, r)$. Suppose the system is started in a known configuration, so that

$$\hat{x}_0^0 = 0; \qquad p_0^0 = 0.$$

Then the exact solution to the filtering problem is given by first calculating the predicted distribution:

$$\begin{aligned} \hat{x}_t^{t-1} &= \hat{x}_{t-1}^{t-1}; \\ p_t^{t-1} &= p_{t-1}^{t-1} + 1, \end{aligned}$$

and then incorporating the measurement by means of the innovation and Kalman gain:

$$\begin{aligned} \nu_t &= z_t - \hat{x}_t^{t-1}; \\ k_t &= \frac{p_t^{t-1}}{p_t^{t-1} + r}; \\ \hat{x}_t^t &= \hat{x}_t^{t-1} + k_t \nu_t; \\ p_t^t &= (1 - k_t) p_t^{t-1}. \end{aligned}$$

These exact expressions for the mean and variance of the posterior distribution $p(x_t \mid z_1, \ldots, z_t)$ can then be compared to the approximations the Condensation algorithm provides:

$$\hat{x}_t^t \approx \hat{\mu}_t = \sum_n \pi_t^{(n)} s_t^{(n)}; \tag{7.2}$$

$$p_t^t \approx \hat{\sigma}_t^2 = \sum_n \pi_t^{(n)} (\hat{\mu}_t - s_t^{(n)})^2. \tag{7.3}$$

Details of Comparison The free parameter in the experiment described above is the variance r of the measurement process. Several values of this parameter were used, and the quality of the approximation produced by the Condensation filter examined for each one. Note that the dynamical process variance has been fixed at 1 for these experiments; this does not result in any loss of generality as altering this parameter produces nothing more than a scaling of the whole process.

Note also that it is largely irrelevant how well the filters estimate the true value of the system state x_t. All that is of interest is how well the Condensation approximation matches the exact analytic expression for the posterior distribution $p(x_t \mid z_1, z_2, \ldots, z_t)$. Since this distribution is Gaussian, the approximation can be evaluated by examining the sample-set approximation to the mean and variance of the distribution.

Once the Kalman filter has converged (to some predefined accuracy) to its steady-state variance

$$p_\infty = \frac{\sqrt{1+4r}-1}{2},$$

the Condensation filter will produce estimates $\hat{\sigma}_t^2$ which should all be equal to this p_∞. This provides a convenient method of testing the accuracy of the variance approximation.

Evaluating the approximation to the mean $\hat{x}_t^t$ raises the following issue. The mean of the posterior $\hat{x}_t^t$ might very well legitimately be zero or very close to zero. Therefore, a simple relative measure such as $(\hat{\mu}_t - \hat{x}_t^t)/\hat{x}_t^t$ will not give meaningful results. The error relative to the RMS value over the 128 time-steps considered will therefore be given.

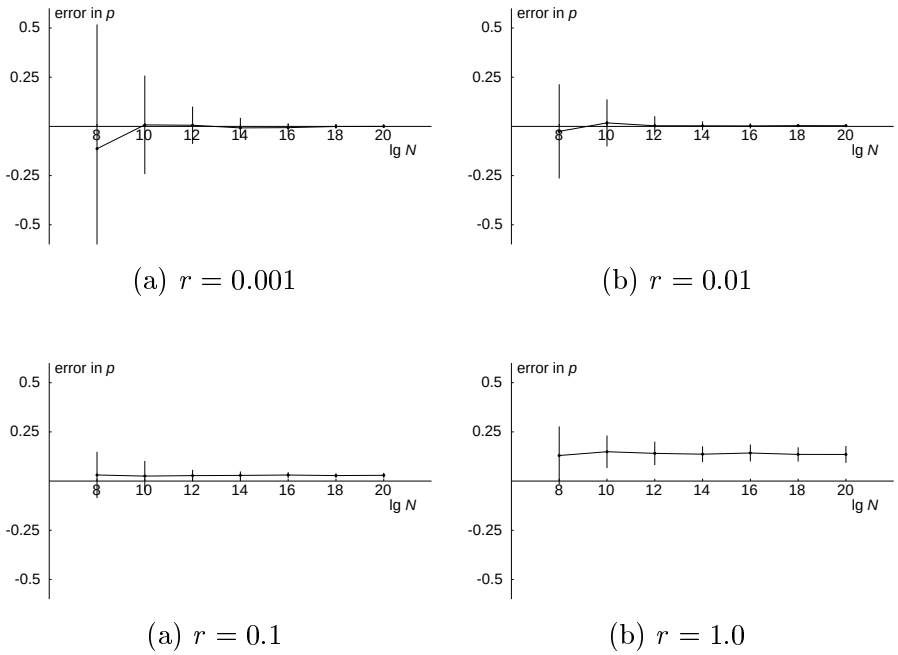
To evaluate the approximation of the approximation $\hat{\sigma}_t^2$ to the true posterior variance p_t^t, the relative error $(\hat{\sigma}_t^2 - p_t^t)/p_t^t$ will be used, as the variance rapidly converges to its steady-state value, which is non-zero.

Numerical Considerations A practical consideration when evaluating the approximations defined in equations 7.2 and 7.3 is that the computation involves the sum of a large number of quantities, and the results might therefore be subject to errors arising from the floating-point representation. This was guarded against by computing the sum in a fashion which recursively sums the numbers in pairs, thereby avoiding the error-inducing step of adding two numbers with widely different magnitudes.

In practice, the sum will be good to an adequate number of significant figures even though up to around 10^6 numbers are being added, since a double-precision floating-point number on the architecture in use provides 16 s.f. Therefore a reasonable estimate would be that even the naïve addition procedure would give a result good to 10 s.f.

Results The sample-set approximation to the posterior mean $\hat{x}_t^t$, (relative to its RMS value over the sample) was good for all values of measurement noise ($r \in \{0.001, 0.01, 0.1, 1.0\}$) and sample-set size ($N$ ranged from 2^8 to 2^{20} in multiples of 4) considered here; the largest error was 2% and most were of the order of 0.05%. This supports the hypothesis that mean-estimation can be performed accurately using sample sets.

The errors in the approximation to the posterior variance p_t^t are shown in figure 7.36. For large measurement noise ($r = 1.0$), even 2^{20} samples does not result in a particularly good approximation, with the Condensation approximation to the variance being consistently high. This corroborates the findings of previous sections, that estimating second-order moments cause difficulty.



(a) $r = 0.001$ (b) $r = 0.01$

(a) $r = 0.1$ (b) $r = 1.0$

Figure 7.36: **Increasing N reduces the variance of the estimator of p_t^t, but the estimator is biased.** *For a one-dimensional Condensation filter, the error in the approximation to p_t^t relative to its true value (calculated from the Kalman filter) is shown. A point indicates the mean estimate of 128 runs, with bars extending one (sample) standard deviation above and below. The sample-set size ranges from $N = 2^8$ up to $N = 2^{20}$; the measurement variance r varies from 0.001 up to 1.0, with the process variance fixed at 1.0.*

One curious result is that the estimator for p_t^t becomes biased for larger measurement noise, with a value which is higher than the true value. Intuition might suggest that Condensation should produce an underestimate of the variance, as the sample-set is likely to under-represent the 'tails' of the distribution. More detailed investigation of these effects would be a useful subject for further work.

7.3.5 Pedestal Condensation

One difficulty with the multi-model learning, and indeed with multi-mode tracking once a model has been learnt, is that a fairly common situation involves, say, two modes, each of which is persistent, so that the true probability of changing state is low. A multi-mode system with the time between transitions being of the order of 10 s is not unrealistic. This translates to a transition matrix resembling

$$M = \begin{pmatrix} 0.998 & 0.002 \\ 0.002 & 0.998 \end{pmatrix}.$$

When tracking with $N = 1000$ or so, a typical value, only around 2 samples, then, will be generated for motion corresponding to a state transition having happened. This will in general be insufficient to allow a transition to be noticed, especially if system noise is high and measurement noise low. Even if a sample corresponding to the transition does happen to fall over the object, the approximation to the transition counts will be inaccurate, leading to the highly variable behaviour of the estimate of $M_{yy'}$ as seen in section 7.1.

One work-around would be to artificially raise the probability of a transition, but a more rigorous approach would be desirable.

Isard's 'ICondensation' (Isard and Blake, 1998b) — Importance Condensation — introduces the notion of an 'importance function', which dictates where samples are placed. One application of the method is to efficiently use information from a secondary sensor, typically of different modality. These samples' weights are then modified accordingly, so that the resulting sample set is still a fair representation of the posterior distribution. The example given in (Isard and Blake, 1998b) is of using a colour-segmentation-based blob tracker to provide coarse information of the position of a hand for a user-input application.

In this hand-tracking application, the importance function is chosen such that more samples are placed at points in state-space judged, by dint of some independent information, to be more likely configurations of the tracked object. The samples' weights are then reduced to keep the final sample-set fair.

In the present problem, the reverse can be done: samples can be generated as if the transition matrix were 'more uniform', with the extreme being the case where M is a matrix all of whose entries are identical. This will ensure some minimum number of samples even for transitions which are very unlikely — a 'pedestal' level of samples is maintained.

Samples corresponding to a transition generated in this way then have their weights reduced, and those corresponding to a continuation of the current state would have their weights increased. In this way, the validity of the sample-set representation is maintained.

The increased quality of the approximation around transitions should produce a benefit in the quality of the learnt model. Detailed investigation of this approach is left as a subject for future research.

Chapter 8

Discussion

8.1 Relation to other Model-learning Frameworks

All the learning algorithms described in this thesis can be viewed within the wider context of the learning or estimation problem for a 'graphical model'. Graphical models are also known as 'probabilistic independence networks', and examples include Markov random fields, and hidden Markov models (HMMs).

The idea is that for a problem involving many random variables, a graph is constructed whose nodes (vertices) correspond to them. An edge between two nodes indicates dependence of the random variables corresponding to those nodes. More precisely, let V be the set of nodes in the graph (equivalent to the set of random variables), and for a node $\alpha \in V$, let $\mathrm{adj}(\alpha)$ be the set of nodes directly connected to α. For random variables X, Y, and Z, write $X \perp\!\!\!\perp Y \,|\, Z$ if X and Y are conditionally independent given Z. Then the graph represents independence in the sense that

$$\alpha \notin \mathrm{adj}(\beta) \implies \alpha \perp\!\!\!\perp \beta \,|\, (V \setminus \{\alpha, \beta\}). \tag{8.1}$$

In other words, pairs of variables corresponding to non-adjacent nodes are conditionally independent given the values of the remaining variables. This property of the graph and distribution of the random variables is referred to as the 'Markovian property'.

As a concrete example, the relationships between the random variables in the problem of learning a dynamical model can be expressed as a graph as shown in figure 8.1.

Typically, only local information is known, such as joint marginal distributions of adjacent pairs of variables, and the joint distribution of the entire set of variables is sought. One area of application of these methods is to 'expert systems'. An expert system is a model which is intended to give reasoned judgements in areas where human experts are scarce. In this setting, it is often the case that the values of some variables are known,

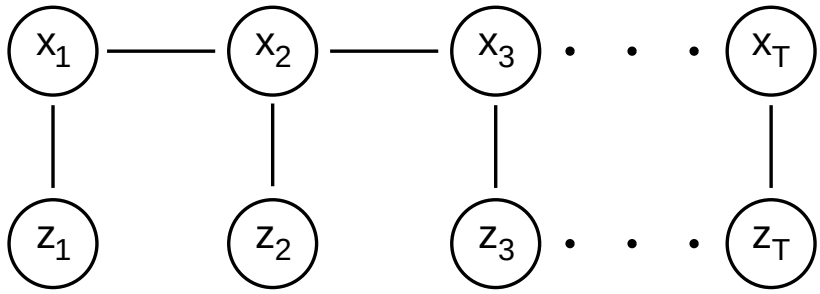


Figure 8.1: **A dynamical model can be represented as a graph.** *Edges in the graph represent the dependence of the variables in the nodes as expressed in eqn (8.1). The state x_t is independent of all other variables, once the preceding and succeeding states x_{t-1} and x_{t+1} and the measurement z_t are given. Similarly, the measurement z_t is independent of all other variables, given x_t. This can be seen in the configuration of the edges of the graph.*

and the problem is to find the distribution of the remaining variables, conditioned on the known ones. Lauritzen and Spiegelhalter (1988) describe a solution to this problem in the case that all random variables involved are discrete.

Darroch et al. (1980) perform maximum-likelihood learning of the individual discrete distributions given training data on such structures, and Lauritzen (1995) combines these two algorithms to apply EM to the learning problem in the case of training data with missing observations. This is the area most closely related to the learning described in this thesis.

Smyth et al. (1996) describe an extension to this algorithm — the so-called 'Jensen, Lauritzen, Oleson' (Jensen et al., 1990) ('JLO') algorithm. Dawid (1992) gives a more mathematical analysis of the algorithm, which propagates information through the graph by means of 'flows' between neighbouring cliques. An algorithm for finding the maximum likelihood value of the whole (discrete) distribution is also given.

Hidden Markov Models

The equivalence of the graph of figure 8.1 to that for a Hidden Markov Model problem will be appreciated, in that only the measurements z_t (corresponding to the emitted symbols of a HMM) are directly observable. The three key problems of HMMs are given by Rabiner and Juang (1986). They are 'classification', evaluating the likelihood of a symbol sequence given the model, 'perception', finding the most likely state sequence given a symbol sequence, and 'learning', finding the best model to describe a set of training symbol sequences. Although the classification problem has been touched on in this thesis in section 3.4, the problem most relevant is the learning one.

For a HMM, the Baum-Welch algorithm is the result of applying EM to the learning problem. The Expectation step is performed by means of the 'forward-backward' algorithm, very similar to the 'filtering-smoothing' algorithm employed in this thesis. It is the specialisation of the JLO algorithm to the HMM case. The Maximisation step takes place by means of a set of re-estimation formulae given by Rabiner. The Viterbi algorithm for finding the maximum-a-posteriori estimate of the state sequence within a HMM is a special case of an algorithm described by Dawid (1992).

Continuous Variables

All of the work described above applies only to discrete-valued random variables. Lauritzen and Wermuth (1989) do give an extension to models with a mixture of discrete and continuous variables, but the continuous variables are constrained to be Gaussian. The relations between this work and that on linear Gaussian dynamical models in chapters 4 and 5 of this thesis are apparent, but have not yet been explored in great detail.

The stochastic methods of chapter 6 successfully handle the problem of general continuous distributions on the type of graph in figure 8.1. They could, then, show promise as a solution to the problem of handling completely general continuous distributions within a graphical model setting. This could be the subject of future research.

8.2 Future Work

The work presented in previous chapters has suggested some areas where more research would be beneficial and interesting. Ideas for possible developments are now discussed. Learning within a Kalman-filtering framework is fairly well understood, and although it is a useful tool, concentrating future research on Condensation-based systems seems a more profitable direction.

Further Experiments

Experiments have been described in chapter 7 which show the usefulness of the learning algorithms in applications where a multi-state model is appropriate (the juggled ball of section 7.1) and in heavy clutter (the hand-tracking example of section 7.2). Time did not permit testing of the methods under the combination of these conditions, i.e., learning a multi-mode model in heavy clutter. Examining the performance of the learning methods in such circumstances would be an interesting area for further work.

A further obvious set of beneficial experiments would involve attempting to learn multi-mode models from longer sequences. The juggled-ball experiments in section 7.1, for example, used a rather short sequence, and so gave results which, while useful, had room for improvement. A better model (particularly in terms of its transition matrix) would almost certainly be produced from combining data from multiple juggling cycles.

Straightforward Improvements to Condensation-EM

Section 7.3 brought up some ideas for improving the performance of the EM learning algorithm when used within a Condensation framework. The main disadvantage of EM learning, when compared with the approximation of filtered learning, is its large additional cost, incurred by the smoothing step in the algorithm. Perhaps the most promising modification to this step is the 'reduced forward-backward' idea, which although it does not remove the $O(N^2)$ step, would produce a very significant speed increase. It would also be straightforward to implement.

The concept of 'pedestal condensation', introduced in section 7.3.5, would also give no implementation difficulties, and, it is hoped, would result in increased learning performance in the case of multi-state models. Section 7.1 showed that the learning of this aspect of the model was quite ill-behaved, so an improvement would be welcome.

Sampling Schemes

Work has been done on evaluating the quality of the sample-set approximations to distributions used in Condensation. For example, Carpenter et al. (1997) and Doucet (1998) both propose the use of an 'effective sample size' as a measure of how much information the sample-set is providing about the distribution compared to a genuinely fair sample drawn from it. Doucet (1998) discusses various sampling methods, in particular an 'importance sampling' framework, as is used by Isard and Blake (1998b) for a Condensation-based hand-tracker. Pitt and Shephard (1997) also consider various methods whereby the sample set can better approximate the true distribution.

Chapter 6 described the effort which was necessary to extract meaningful estimates of the second-order moments for use in learning. Increasing the effectiveness of the sample-set representation is one obvious area for improvement, and this would undoubtedly have a positive effect on the performance of the learning algorithms. Investigating this in more detail would be a fruitful area for future work.

On-line Learning

All the work described in this thesis is based on 'off-line' learning: a training phase is used, entirely separate from any tracking which is then performed using the learnt dynamical model. Experiments have shown that this is a useful approach to the problem, but an attractive idea is to perform 'on-line' learning, namely learning a dynamical model as tracking is proceeding. Using an appropriate dynamical model allows a considerable reduction in the number of Condensation samples required for successful tracking, as has been demonstrated in section 7.2. A system which could learn a dynamical model in the early stages of tracking a sequence could then track the remainder using a lower number of samples. Tracking would then be achieved with a lower computational cost (assuming the on-line learning is not prohibitive), and with no increase in the amount of initial knowledge required.

It is not entirely clear how this learning would be done, although there has been work in other areas on similar problems. For instance, in the neural network literature, de Freitas et al. (1997) describes on-line (or 'sequential') learning of weights in multi-layer perceptrons. Further investigation into this literature and its applicability to on-line learning for trackers is required.

Non-Euclidean State Spaces

An area of interest for a future application of Condensation is to tracking using a kinematic state space. For instance, the most natural representation of the state of a human body is in terms of joint angles, rather than in terms of a two-dimensional state based in the image.

This raises the question of whether a linear auto-regressive model is appropriate for a state space not topologically equivalent to $\mathbb{R}^n$. In some cases, a 'cut' may be possible to produce a reasonable approximation of the state space to a piece of $\mathbb{R}^n$ — taking again the person-tracking example, the rotation of the head does not cover the whole range of rotation. In fact, this is the case for most joint angles within the human body.

On the other hand, the space of orientations of a rigid body can not sensibly be cut in this fashion; it is topologically equivalent to a 3-sphere. (One representation of the space of orientations is as the set of unit quaternions.) In such a space, the typical motion model

$$x_t = \sum_{k=1}^{K} a_k x_{t-k} + b w_t$$

is no longer meaningful, as it requires a vector space structure on the states x (and the noises w). The space of rotations in two dimensions has no such structure; neither does

the space of three-dimensional rotations. Investigating a sensible formulation of dynamical models in such spaces could be a subject for future research.

Higher-order Models

The theories developed in this thesis apply to any model order K, if not explicitly, then by simple extension. For the tracking experiments, though, second-order models only were used. They produced successful results, but the classification experiments in section 3.4 suggest that higher-order models might bring benefits. It would be desirable to gather some information on the effect of higher-order models on tracking performance.

Observation Models

Although more directly an area for research into improving the performance of the Condensation tracking algorithm itself, the choice of observation density has influence on the learning algorithms — the assumptions made during the formulation of $p(z \mid x)$ and their validity will affect the final posterior distribution for the state sequence.

The observation density used for the experiments described in this paper is fairly naïve, and work has been done by MacCormick and Blake (1998) on developing a rigorous, probabilistically-based methodology for calculating the likelihood that a given contour configuration arose from clutter or from the tracking target. Although this has not yet been developed into an observation density for Condensation, doing so is the subject of current research.

Also the subject of current research is the use of region-based observation densities. This thesis has exclusively used contour-based measurement processes, which have the significant advantage of speed — very few image pixels must be examined. Contour-based measurements are also invariant to illumination and scale. Much information, for example texture information, however, is being ignored, and it would be interesting to investigate how such information could be incorporated into a Condensation tracker.

Initialisation

The learning described in this thesis, as well as the tracking used as test cases for the models, used hand-initialised sequences. In other words, the state of the system at $t = 0$ was provided. It would be preferable to remove the necessity of this step, allowing the system to learn the initial distribution of the state as part of the algorithm. This would fit within the EM framework by treating the initial state as one of the parameters of the

model, and so should be a fairly straightforward extension. For using a learnt model to track other sequences, of course, the initial position in the training sequence would not be used.

8.3 Conclusions

This thesis has developed mathematically correct techniques for finding best-fit dynamical models from noisy measurement sequences. This learning has taken place within a Kalman filtering framework, and also in the context of the more powerful Condensation algorithm. The resulting models are capable of generalisation, in that experiments have shown success in using them for tracking sequences similar to the test ones. The learnt models can also be successfully used for classification, as well as a source of information in their own right (for instance the acceleration results from the juggled ball).

One obvious shortcoming of the technique, particularly with regard to its use in the Condensation domain, is its large computational cost. The vastly cheaper (one-shot as opposed to the iterative EM algorithm, and no $O(N^2)$ cost) filtered learning approach often produces results which are nearly as good. Filtered learning, however, can not be immediately applied to the multi-mode models of section 7.1, especially without supplying additional prior information. Methods for reducing the computational burden of, in particular, the smoothing phase of the algorithm must be sought. Research into this, and the other areas described above, should produce more efficient methods for learning dynamical models.

Appendix A

Derivations and Proofs

A.1 Ljung's Method

Consider the simplified model structure

$$x_t = Ax_{t-1} + w_t;$$
$$z_t = Cx_t + v_t,$$

where as usual x_t is the state at time t, A is the dynamics matrix, C is the measurement matrix, and w_t and v_t are all independent Gaussian random variables with zero mean and unit variance. Ljung (1987) suggests forming

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{pmatrix}; \qquad Z = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_T \end{pmatrix};$$

then, assuming that $x_0 = 0$, defining

$$F = \begin{pmatrix} 1 & 0 & & \cdots & 0 \\ A & 1 & 0 & \cdots & 0 \\ A^2 & A & 1 & 0 \cdots & 0 \\ & & \vdots & & \\ A^{T-1} & A^T & & \cdots & 1 \end{pmatrix}; \qquad H = \begin{pmatrix} C & 0 & \cdots & & 0 \\ 0 & C & 0 & \cdots & 0 \\ \vdots & 0 & & & \vdots \\ & \vdots & & \ddots & \\ 0 & 0 & \cdots & 0 & C \end{pmatrix}$$

and

$$E = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_T \end{pmatrix}; \qquad W = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_T \end{pmatrix}$$

allows the system to be written as

$$X = FW; \qquad Z = HX + E.$$

This is a slight simplification of the general problem, as both the measurement and process noises are fixed at unity. However, this does not affect the presentation or nature of the argument.

First consider the 'E' step — the problem of finding $\mathcal{E}_X[\log p(X, Z \mid \varphi) \mid Z, \varphi = \alpha]$. Now

$$p(X, Z \mid \varphi) = p(X \mid \varphi)p(Z \mid X, \varphi)$$

and

$$p(X \mid \varphi) = g(X; 0, FF^\top); \qquad p(Z \mid X, \varphi) = g(Z; HX, I),$$

where $g(x; \mu, \Sigma)$ is the density function of a Gaussian with mean μ and variance Σ evaluated at x. Thus

$$\log p(X, Z \mid \varphi) = -\frac{1}{2}\left[X^\top (FF^\top)^{-1} X + \log|(FF^\top)| + (Z - HX)^\top (Z - HX) + \text{const}\right].$$

Taking the expected value of this with respect to a distribution over X given $\varphi = \alpha$ yields

$$\begin{aligned}\mathcal{E}_X[\log p(X, Z \mid \varphi) \mid Z, \varphi = \alpha] = &-\hat{X}^\top (FF^\top)\hat{X} - \operatorname{tr}(F^{-1}PF^{-\top}) \\ &- \log|(FF^\top)| + \text{const wrt } \varphi, \end{aligned} \tag{A.1}$$

where

$$\hat{X} = \mathcal{E}[X \mid Z, \varphi = \alpha]; \qquad P = \operatorname{Var}[X \mid Z, \varphi = \alpha].$$

The 'E' part is then reduced to the finding of this mean and variance. For two random variables X and Z with means μ_X and μ_Z, variances P_X and P_Z and covariance P_{XZ}, the distribution of X conditioned on Z is

$$(X \mid Z) \sim N(\mu_X + P_{XZ}P_Z^{-1}(Z - \mu_Z), P_X - P_{XZ}P_Z^{-1}P_{XZ}^\top).$$

In this case, then, it will suffice to find the means and variances of X and Z, and the covariance of X against Z. Now

$$W \sim N(0, I),$$

so

$$X \sim N(0, FF^\top),$$

and hence

$$Z \sim N(0, HFF^\top H^\top + I).$$

Also,

$$\begin{aligned} P_{XZ} &= \mathcal{E}[XZ^\top] \\ &= \mathcal{E}[FW(HFW+E)^\top] \\ &= F\mathcal{E}[WW^\top]F^\top H^\top + \mathcal{E}[E] \\ &= FF^\top H^\top. \end{aligned}$$

Therefore

$$\begin{aligned} \hat{X} &= FF^\top H^\top R^{-1} Z; \\ P &= FF^\top - FF^\top H^\top R^{-1} HFF^\top, \end{aligned}$$

where

$$R = I + HFF^\top H^\top.$$

This completes the 'E' step; note that it involves the inversion of the large matrix R.

For the 'M' part, the value of φ which maximises the RHS of eqn (A.1) is sought, i.e.,

$$\arg\max_{\varphi} \left[-\hat{X}^\top (FF^\top)\hat{X} - \mathrm{tr}(F^{-1}PF^{-\top}) - \log|(FF^\top)| \right], \tag{A.2}$$

since $\varphi = A$ and $F = F(A)$. There is no obvious explicit expression for this arg max.

In practice, one would exploit the structure in $F(A)$ to perform both the 'E' and 'M' steps — the 'E' step involving the inversion of R, and the 'M' step involving the maximisation of eqn (A.2). While the above formulation is elegant and general, then, in the case of interest, namely dynamical models for tracking, the exploitation mentioned would lead to a solution of the kind described in section 4.4.

A.2 Alternative Approaches to EM Learning for a Kalman Filter

Shumway and Stoffer (1982) and Ghahramani (1996) both describe a formulation of the EM algorithm for the case of learning a dynamical model within the Kalman filter framework, equivalent to that developed in chapter 4. The Maximisation step is exactly equivalent; a slightly different method of performing the Expectation step is given. Instead of using an augmented state vector, as was done in this thesis, a more direct recursive method of calculating the covariances (using the notation of section 4.4.2)

$$P^T_{t-1,t-2} = \mathrm{Cov}[x_{t-1}, x_{t-2} \,|\, z_1, \ldots, z_T]$$

is given. (The augmented state filter was introduced to calculate these covariances.) The recursion is initialised with

$$P^T_{T,T-1} = (I - K_T H) A P^{T-1}_{T-1}$$

and proceeds according to

$$P^T_{t-1,t-2} = P^{t-1}_{t-1} + F_{t-1}(P^T_t - P^{t-1}_t)F^\top_{t-1}.$$

Shumway and Stoffer described the derivation of these results as 'somewhat lengthy', and refer to a technical report of theirs (Shumway and Stoffer, 1981).

Both papers also describe the extension to learning the measurement variance, and (Ghahramani and Hinton, 1996) also extends the theory to estimating the measurement gain H ('output matrix' in his notation). For the case of a visual tracker, however, the measurement gain is known, and the noise can be estimated independently and directly, so these extensions were not required for the work of this thesis.

A.3 Multi-dimensional Yule-Walker

For completeness, the derivation of the maximum-likelihood estimates of a dynamical model from an example sequence is now presented in more detail.

A.3.1 Zero-mean case

First, the case of a system with known mean is considered. Without loss of generality, assume that it is zero by subtracting it from every element of the training sequence $x^*_1, x^*_2, \ldots, x^*_T$. The log-likelihood L can be written in terms of the individual residue terms

$$r_t = x^*_t - \sum_{k=1}^{K} A_k x^*_{t-k}$$

as

$$L = -\frac{1}{2}\sum_{t=K+1}^{T} r_t^\top C^{-1} r_t - \frac{T-K}{2}\log|C|.$$

Using $u^\top M v = \mathrm{tr}(uv^\top M^\top)$ and the fact that C is symmetric, this can be written

$$L = -\frac{1}{2}\mathrm{tr}\left[\left(\sum_{t=K+1}^{T} r_t r_t^\top\right) C^{-1}\right] - \frac{T-K}{2}\log|C|.$$

For notational convenience, define

$$A = \begin{pmatrix} A_1 & A_2 & \cdots & A_K \end{pmatrix}.$$

Writing $Z = \sum_{t=K+1}^{T} r_t r_t^\top$ and expanding the sum gives

$$\begin{aligned} Z &= \sum_{t=K+1}^{T} \left[x_t^* - A \begin{pmatrix} x_{t-1}^* \\ x_{t-2}^* \\ \vdots \\ x_{t-K}^* \end{pmatrix} \right] \left[(x_t^*)^\top - \left((x_{t-1}^*)^\top \; (x_{t-2}^*)^\top \; \cdots \; (x_{t-K}^*)^\top \right) A^\top \right] \\ &= R_{00} - \mathbf{R}_0 A^\top - A\mathbf{R}_0^\top + ARA^\top, \end{aligned} \tag{A.3}$$

where

$$R = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1,K} \\ R_{21} & R_{22} & \cdots & R_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ R_{K,1} & R_{K,2} & \cdots & R_{K,K} \end{pmatrix}; \qquad \mathbf{R}_0 = \begin{pmatrix} R_{01} & R_{02} & \cdots & R_{0,K} \end{pmatrix},$$

and the moments R_{ij} are defined as

$$R_{ij} = \sum_{t=K+1}^{T} x_{t-i}^* (x_{t-j}^*)^\top.$$

Therefore, L can now be written

$$L = -\frac{1}{2}\mathrm{tr}(ZC^{-1}) - \frac{T-K}{2}\log|C|.$$

Differentiating this with respect to A gives the following equation for the maximum likelihood estimate of A:

$$AR = \mathbf{R}_0, \tag{A.4}$$

and differentiating with respect to C gives

$$\begin{aligned} C &= \frac{1}{T-K} Z \\ &= \frac{1}{T-K}(R_{00} - A\mathbf{R}_0^\top) \end{aligned}$$

(using eqn (A.4) to simplify the expression for Z).

A.3.2 Unknown Mean

This section extends the analysis to the case where the mean is unknown. In this case, the individual residuals become

$$r_t = x_t^* - \sum_{k=1}^{K} A_k x_{t-k}^* - d,$$

where $d = (I - \sum_{k=1}^{K} A_k)\bar{x}$ is the offset. Specifying d in preference to $\bar{x}$ is a more general approach, allowing, for example, constant-velocity motion, which has no mean. The expression for Z then becomes

$$Z = R_{00} - \mathbf{R}_0 A^\top - R_0 d^\top - A\mathbf{R}_0^\top + ARA^\top + A\mathbf{R}d^\top - dR_0^\top + d\mathbf{R}^\top A^\top + (T-K)dd^\top, \tag{A.5}$$

where

$$\mathbf{R} = \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_K \end{pmatrix} \qquad \text{and} \qquad R_i = \sum_{t=K+1}^{T} x^*_{t-i}.$$

Differentiating eqn (A.5) with respect to A produces

$$AR + d\mathbf{R}^\top - \mathbf{R}_0 = 0, \tag{A.6}$$

(using $R = R^\top$) and with respect to d gives

$$d = \frac{1}{T-K}(R_0 - A\mathbf{R}).$$

Substituting this expression for d into eqn (A.6) gives

$$A\left(R - \frac{1}{T-K}\left(\mathbf{R}\mathbf{R}^\top\right)\right) - \left(\mathbf{R}_0 - \frac{1}{T-K}\left(R_0 R_0^\top\right)\right) = \mathbf{R}_0.$$

Introducing the modified second-order moments

$$R^0_{ij} = R_{ij} - \frac{1}{T-K} R_i R_j^\top.$$

and writing

$$R^0 = \begin{pmatrix} R^0_{11} & R^0_{12} & \cdots & R^0_{1,K} \\ R^0_{21} & R^0_{22} & \cdots & R^0_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ R^0_{K,1} & R^0_{K,2} & \cdots & R^0_{K,K} \end{pmatrix}; \qquad \mathbf{R}^0_0 = \begin{pmatrix} R^0_{01} & R^0_{02} & \cdots & R^0_{0,K} \end{pmatrix}$$

allows this to be simplified:

$$AR^0 = \mathbf{R}^0_0.$$

Finally, differentiating L with respect to C gives $C = Z/(T-K)$, which simplifies to

$$C = \frac{1}{T-K}\left(R^0_{00} - A(\mathbf{R}^0_0)^\top\right),$$

using the expressions for A and d.

A.4 Learning from Multiple Sequences

If several training sequences are available, and it is desired to find the maximum-likelihood estimate for all the sequences, it is clearly incorrect to simply concatenate the sequences and apply the results of section A.3. There will be jumps at the joins of the sequences, producing an inaccurate estimate.

Suppose that M training sequences are given:

$$(x^*_{11}, \ldots, x^*_{1T_1}), \ldots (x^*_{M1}, \ldots, x^*_{M,T_M}).$$

Assuming that these sequences are independent, the overall log likelihood will be given by

$$L = -\frac{1}{2} \sum_{m=1}^{M} \left[\sum_{t=K+1}^{T_m} r_{mt}^\top C^{-1} r_{mt} \right] - \sum_{m=1}^{M} \left[\frac{T_m - K}{2} \log |C| \right],$$

defining the residues r_{mt} in the obvious way.

Following the development through, it can be shown that the form of the maximum-likelihood estimate remains the same, with

$$R_{ij} = \sum_m R_{mij}; \qquad R_i = \sum_m R_{mi},$$

where

$$R_{mij} = \sum_{t=K+1}^{T_m} x^*_{m,(t-i)} (x^*_{m,(t-j)})^\top \qquad R_{mi} = \sum_{t=K+1}^{T_m} x^*_{m,(t-i)}.$$

The expression $T - K$ in the definition of the modified moments R^0_{ij} and in the estimates for d and C must be replaced by $\sum_m (T_m - K)$.

A.5 Maximum-likelihood Estimate for Multi-mode Models

A mixed-state model has a state vector

$$\mathbf{X}_t = \begin{pmatrix} \mathbf{x}_t \\ y_t \end{pmatrix},$$

where $\mathbf{x}_t$ is the continuous state and y_t is a label for the discrete state. It is assumed that the transition probabilities for the discrete state are context-insensitive, i.e., they do not depend on $\mathbf{x}_t$. This assumption produces the following expression for the joint model.

$$p(\mathbf{X}_t \mid \mathcal{X}_0^{t-1}) = p_{y_t}(\mathbf{x}_t \mid \mathbf{x}_0, \ldots, \mathbf{x}_{t-1}) M_{y_{t-1} y_t}.$$

The log-likelihood of a model for a given sequence $\mathbf{X}_t^*$ of known states therefore separates into two components: the continuous component, whose maximisation has already been dealt with in appendix A.3, and the discrete component

$$L_d = \sum_t \log M_{y_{t-1}y_t}. \tag{A.7}$$

This must be maximised subject to the constraint

$$\sum_b M_{ab} = 1 \qquad \forall a.$$

As usual, introduce the Lagrange multipliers λ_a and seek the maximum of

$$L' = \sum_t \log M_{y_{t-1}y_t} + \lambda_a(1 - \sum_b M_{ab}).$$

Now, the first term can be written

$$\begin{aligned}\sum_t \log M_{y_{t-1}y_t} &= \sum_t \sum_a \sum_b \delta_{ay_{t-1}}\delta_{by_t} \log M_{ab} \\ &= \sum_{a,b} \sum_t \delta_{ay_{t-1}}\delta_{by_t} \log M_{ab} \\ &= \sum_{a,b} T_{ab} \log M_{ab},\end{aligned}$$

where

$$\begin{aligned}T_{ab} &= \sum_t \delta_{ay_{t-1}}\delta_{by_t} \\ &= \sharp\{t : y^*_{t-1} = a, y^*_t = b\}.\end{aligned}$$

The expression for L' can now be written

$$L' = \sum_{a,b} T_{ab} \log M_{ab} + \lambda_a(1 - \sum_b M_{ab}).$$

Differentiating with respect to M_{ab} gives

$$\frac{\partial L'}{\partial M_{ab}} = \frac{T_{ab}}{M_{ab}} + \lambda_a$$

Multiply by M_{ab} to get the set of equations

$$M_{ab}\lambda_a = T_{ab}$$

(one for each pair (a, b) of states). Sum over b to get

$$\lambda_a \sum_b M_{ab} = \sum_b T_{ab}$$

and use the constraint to get

$$\lambda_a = \sum_b T_{ab},$$

whence the ML values

$$M_{ab} = \frac{T_{ab}}{\sum_b T_{ab}}$$

may be deduced.

Note also that the log-likelihood L_d of eqn (A.7) can be written in terms of the T_{ab} as

$$L_d = \sum_{a,b} T_{ab} \log M_{ab},$$

which is linear in the T_{ab}. This linearity is important to the validity of the claimed EM algorithm for this problem described in section 6.2.1.

Bibliography

Ayache, N., Cohen, I., and Herlin, I. (1992). Medical image tracking. In Blake, A. and Yuille, A., editors, *Active Vision*, 285–302. MIT.

Azarbayejani, A., Starner, T., Horowitz, B., and Pentland, A. (1993). Visually controlled graphics. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15, 6, 602–604.

Bar-Shalom, Y. and Fortmann, T. (1988). *Tracking and Data Association.* Academic Press.

Barnett, S. (1990). *Matrices: Methods and Applications.* Oxford University Press.

Bartels, R., Beatty, J., and Barsky, B. (1987). *An Introduction to Splines for use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann.

Baumberg, A. and Hogg, D. (1994). Learning flexible models from image sequences. In Eklundh, J.-O., editor, *Proc. 3rd European Conf. Computer Vision*, 299–308. Springer-Verlag.

Bennett, A. and Craw, I. (1991). Finding image features for deformable templates and detailed prior statistical knowledge. In *Proc. British Machine Vision Conf.*, 233–239.

Bishop, C. (1995). *Neural networks for pattern recognition.* Oxford University Press.

Black, M. and Anandan, P. (1993). A framework for the robust estimation of optical flow. In *Proc. 4th Int. Conf. on Computer Vision*, 231–236, Berlin, Germany.

Black, M., Yacoob, Y., Jepson, A., and Fleet, D. (1997). Learning parameterized models of image motion. In *Conference on Computer Vision and Pattern Recognition*, 561–567, San Juan, Puerto Rico.

Blake, A. and Isard, M. (1998). *Active contours.* Springer.

Blake, A., Isard, M., and Reynard, D. (1995). Learning to track the visual motion of contours. *J. Artificial Intelligence*, 78, 101–134.

Blake, A., Zisserman, A., and Cipolla, R. (1992). Visual exploration of freespace. In Blake, A. and Yuille, A., editors, *Active Vision*, 175–188. MIT.

Boyles, R. (1983). On the convergence of the EM algorithm. *J. Roy. Stat. Soc. B*, 45, 47–50.

Bregler, C. (1997). Learning and recognising human dynamics in video sequences. In *Proc. Conf. Computer Vision and Pattern Recognition.*

Bregler, C. and Malik, J. (1998). Tracking people with twists and exponential maps. In *Proc. CVPR.*

Bregler, C. and Omohundro, S. (1994). Surface learning with applications to lipreading. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, 6. Morgan Kaufmann Publishers.

Bucy, R. (1969). Bayes theorem and digital realizations for non-linear filters. *J. Astronautical Sciences*, 17, 2, 80–94.

Carpenter, J., Clifford, P., and Fearnhead, P. (1997). An improved particle filter for non-linear problems. Technical report, Dept. of Statistics, University of Oxford. Available from `www.stats.ox.ac.uk/~clifford/index.html`.

Clarke, J., Carlsson, S., and Zisserman, A. (1996). Detecting and tracking linear features efficiently. In *Proc. British Machine Vision Conference.*

Cootes, T. and Taylor, C. (1992). Active shape models. In *Proc. British Machine Vision Conf.*, 265–275.

Cootes, T., Taylor, C., Cooper, D., and Graham, J. (1992). Training models of shape from sets of examples. In *British Machine Vision Conference*, 9–18.

Curwen, R. and Blake, A. (1992). Dynamic contours: real-time active splines. In Blake, A. and Yuille, A., editors, *Active Vision*, 39–58. MIT.

Darroch, J., Lauritzen, S., and Speed, T. (1980). Markov fields and log-linear interaction models for contingency tables. *The Annals of Statistics*, 8, 3, 522–539.

Dawid, A. (1992). Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and computing*, 2, 1, 25–36.

DeGroot, M. (1985). *Probability and statistics.* Addison-Wesley, 2nd edition.

Dempster, A., Laird, M., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc. B.*, 39, 1–38.

Doucet, A. (1998). On sequential simulation-based methods for Bayesian filtering. Technical Report CUED/F-INFENG/TR310, Dept. of Engineering, University of Cambridge. Available from `www.stats.bris.ac.uk:81/MCMC/pages/list.html`.

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis.* John Wiley and Sons.

Essa, I. and Pentland, A. (1995). Facial expression recognition using a dynamic model and motion energy. In *Proc. 5th Int. Conf. on Computer Vision*, 360–367.

Fischler, M. and Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 24, 381–95.

Fisher, R. (1912). On an absolute criterion for fitting frequency curves. *The messenger of mathematics*, 41, 155.

Flash, T. and Hogan, N. (1985). The coordination of arm movements: An experimentally confirmed mathematical model. *J. Neuroscience*, 5, 7, 1688–1703.

Fornland, P. (1995). Direct obstacle detection and motion from spatio-temporal derivatives. In *Proc. 6th Int. Conf. Computer Analysis of Images and Patterns*, Prague.

Freitas, J. d., Niranjan, N., and Gee, A. (1997). Hierarchical Bayesian-Kalman models for regularisation and ARD in sequential learning. Technical Report CUED/F-INFENG/TR 307, Cambridge University Engineering Dept.

Gelb, A., editor (1974). *Applied Optimal Estimation.* MIT Press, Cambridge, MA.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6, 6, 721–741.

Ghahramani, Z. and Hinton, G. (1996). Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, Dept of Computer Science, University of Toronto.

Goodwin, C. and Sin, K. (1984). *Adaptive filtering prediction and control.* Prentice-Hall.

Gordon, N., Salmond, D., and Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. F*, 140, 2, 107–113.

Grenander, U., Chow, Y., and Keenan, D. (1991). *HANDS. A Pattern Theoretical Study of Biological Shapes.* Springer-Verlag. New York.

Harris, C. (1992). Tracking with rigid models. In Blake, A. and Yuille, A., editors, *Active Vision*, 59–74. MIT.

Heap, T. and Hogg, D. (1997). Improving specificity in PDMs using a hierarchical approach. In *British Machine Vision Conference*, 80–89.

Heap, T. and Hogg, D. (1998). Wormholes in shape space: Tracking through discontinuous changes in shape. In *Proc. 6th Int. Conf. on Computer Vision*, 344–349.

Hoel, P. (1966). *Elementary Statistics.* Wiley, 2nd ed edition.

Horn, B. and Schunk, B. (1981). Determining optical flow. *J. Artificial Intelligence*, 17, 185–203.

Howatson, A., Lund, P., and Todd, J. (1972). *Engineering tables and data.* Chapman and Hall, London.

Huang, X., Arika, Y., and Jack, M. (1990). *Hidden Markov Models for Speech Recognition.* Edinburgh University Press.

Isard, M. and Blake, A. (1996). Visual tracking by stochastic propagation of conditional density. In *Proc. 4th European Conf. Computer Vision*, 343–356, Cambridge, England.

Isard, M. and Blake, A. (1998a). Condensation — conditional density propagation for visual tracking. *Int. J. Computer Vision*, 28, 1, 5–28.

Isard, M. and Blake, A. (1998b). ICondensation: Unifying low-level and high-level tracking in a stochastic framework. In *Proc. 5th European Conf. Computer Vision*, 893–908.

Isard, M. and Blake, A. (1998c). A mixed-state Condensation tracker with automatic model switching. In *Proc. 6th Int. Conf. on Computer Vision*, 107–112.

Isard, M. and Blake, A. (1998d). A smoothing filter for Condensation. In *Proc. 5th European Conf. Computer Vision*, 1, 767–781, Freiburg, Germany. Springer Verlag.

Jacob, G., Noble, J., and Blake, A. (1998). Robust contour tracking in echocardiographic sequences. In *Proc. 6th Int. Conf. on Computer Vision*, 408–413.

Jacobs, O. (1993). *Introduction to control theory.* Oxford University Press.

Jazwinski, A. (1970). *Stochastic processes and filtering theory.* Academic Press.

Jensen, F., Lauritzen, S., and Olesen, K. (1990). Bayesian updating in recursive graphical models by local computations. *Computational Statistical Quarterly*, 4, 269–282.

Jepson, A. and Black, M. (1993). Mixture models for optical flow computation. *Proc. Conf. Computer Vision and Pattern Recognition*, 760–761.

Kass, M., Witkin, A., and Terzopoulos, D. (1987). Snakes: Active contour models. In *Proc. 1st Int. Conf. on Computer Vision*, 259–268.

Kaucic, R. and Blake, A. (1998). Accurate, real-time, unadorned lip tracking. In *Proc. 6th Int. Conf. on Computer Vision.* 370–375.

Kaucic, R. A. (1997). *Lip tracking for audio-visual speech recognition.* PhD thesis, University of Oxford.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5, 1, 1–25.

Koenderink, J. and van Doorn, A. (1975). Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer. *Optica Acta*, 22, 9, 773–791.

Koller, D., Weber, J., and Malik, J. (1994). Robust multiple car tracking with occlusion reasoning. In *Proc. 3rd European Conf. Computer Vision*, 189–196. Springer-Verlag.

Lanitis, A., Taylor, C., and Cootes, T. (1995). A unified approach to coding and interpreting face images. In *Proc. 5th Int. Conf. on Computer Vision*, 368–373.

Lauritzen, S. (1995). The EM algorithm for graphical association models with missing data. *Computational statistics and data analysis*, 19, 191–201.

Lauritzen, S. and Spiegelhalter, D. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *J. R. Stat. Soc. B*, 50, 2, 157–224.

Lauritzen, S. and Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17, 1, 31–58.

Ljung, L. (1987). *System identification: theory for the user*. Prentice-Hall.

Lowe, D. (1992). Robust model-based motion tracking through the integration of search and estimation. *Int. J. Computer Vision*, 8, 2, 113–122.

MacCormick, J. and Blake, A. (1998). A probabilistic contour discriminant for object localisation. In *Proc. 6th Int. Conf. on Computer Vision*, 390–395.

Murray, R., Li, Z., and Sastry, S. (1994). *A mathematical introduction to robotic manipulation*. CRC Press.

North, B. and Blake, A. (1997). Using expectation-maximisation to learn dynamical models from visual data. In *Proc. British Machine Vision Conf.*, 669–678.

North, B. and Blake, A. (1998). Learning dynamical models using expectation-maximisation. In *Proc. 6th Int. Conf. on Computer Vision*, 384–389.

Pardey, J., Roberts, S., and Tarassenko, L. (1995). A review of parametric modelling techniques for EEG analysis. *Medical Engineering Physics*, 18, 1, 2–11.

Pardey, J., Roberts, S., Tarassenko, L., and Stradling, J. (1996). A new approach to the analysis of the human sleep/wakefulness continuum. *J. Sleep Res.*, 5, 201–210.

Pitt, M. and Shepherd, N. (1997). Filtering via simulation and auxiliary particle filters. Technical report, Nuffield College, University of Oxford. Available from `www.nuff.ox.ac.uk/economics/people/shephard.htm`.

Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1988). *Numerical Recipes in C*. Cambridge University Press.

Rabiner, L. and Juang, B. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine*, 4–16.

Rabiner, L. and Juang, B. (1993). *Fundamentals of speech recognition*. Prentice-Hall.

Rao, B. (1992). Data association methods for tracking systems. In Blake, A. and Yuille, A., editors, *Active Vision*, 91–105. MIT.

Reynard, D., Wildenberg, A., Blake, A., and Marchant, J. (1996). Learning dynamics of complex motions from image sequences. In *Proc. 4th European Conf. Computer Vision*, 357–368, Cambridge, England.

Ripley, B. (1996). *Pattern recognition and neural networks.* Cambridge University Press.

Ripley, B. and Sutherland, A. (1990). Finding spiral structures in images of galaxies. *Phil. Trans. R. Soc. Lond. A.*, 332, 1627, 477–485.

Rowe, S. (1996). *Robust feature search for active tracking.* PhD thesis, University of Oxford.

Sage, A. P. and Melsa, J. L. (1971). *Estimation theory with applications to communications and control.* McGraw-Hill.

Shumway, R. and Stoffer, D. (1981). Time series smoothing and forecasting using the EM algorithm. Technical Report 27, Division of Statistics, University of California, Davis.

Shumway, R. and Stoffer, D. (1982). An approach to time series smoothing and forecasting using the EM algorithm. *J. Time Series Analysis*, 3, 253–264.

Smyth, P., Heckerman, D., and Jordan, M. (1996). Probabilistic independence networks for hidden Markov probability models. Technical Report AI Memo No. 1565, MIT.

Sorenson, H. and Alspach, D. (1971). Recursive Bayesian estimation using Gaussian sums. *Automatica*, 7, 465–479.

Sullivan, G. (1992). Visual interpretation of known objects in constrained scenes. *Phil. Trans. R. Soc. Lond. B.*, 337, 361–370.

Tanaka, M. and Katayama, T. (1989). Edge detection and restoration of noisy images by the expectation-maximization algorithm. *Signal Processing*, 17, 213–226.

Terzopoulos, D. and Metaxas, D. (1991). Dynamic 3D models with local and global deformations: deformable superquadrics. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13, 7, 703–714.

Terzopoulos, D. and Metaxas, D. (1992). Tracking nonrigid 3D objects. In Blake, A. and Yuille, A., editors, *Active Vision*, 75–89. MIT.

Terzopoulos, D. and Waters, K. (1990). Analysis of facial images using physical and anatomical models. In *Proc. 3rd Int. Conf. on Computer Vision*, 727–732.

Torr, P. and Murray, D. (1993). Outlier detection and motion segmentation. In Schenker, P., editor, *Sensor Fusion VI*, 432–443, Boston. SPIE volume 2059.

Wachter, S. and Nagel, H. (1997). Tracking of persons in monocular image sequences. In *Workshop on motion of non-rigid and articulated objects*, 2–9. IEEE.

Walker, G. (1931). On periodicity in series of related terms. *Proc. Royal Soc. London A*, 131, 518–532.

Williams, P. (1989). *Gray's anatomy.* Churchill Livingstone, Edinburgh, 37th edition.

Wu, C. (1983). On the convergence properties of the EM algorithm. *The annals of statistics*, 11, 95–103.

Yuille, A. and Hallinan, P. (1992). Deformable templates. In Blake, A. and Yuille, A., editors, *Active Vision*, 20–38. MIT.

Yule, G. (1927). On a method for investigating periodicities in disturbed series with special reference to Wolfer's sunspot numbers. *Philos. Trans. Roy. Soc. London A*, 226, 267–298.