Visual Motion Analysis by Probabilistic Propagation of Conditional Density

Michael Acheson Isard
Robotics Research Group
Department of Engineering Science
University of Oxford
September, 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.

Michael Acheson Isard
Magdalen College

Doctor of Philosophy
Trinity Term, 1998

Visual Motion Analysis by Probabilistic Propagation of Conditional Density

Abstract

This thesis establishes a stochastic framework for tracking curves in visual clutter, using a Bayesian random-sampling algorithm. The approach is rooted in ideas from statistics, control theory and computer vision. The problem is to track outlines and features of foreground objects, modelled as curves, as they move in *substantial* clutter, and to do it at, or close to, video frame-rate. The algorithm, named CONDENSATION, for CONditional DENSity propagATION, has recently been derived independently by several researchers, and is generating significant interest in the statistics and signal processing communities. This thesis contributes to the literature on CONDENSATION-like filters by presenting some novel applications of and extensions to the basic algorithm, and contributes to the visual motion estimation literature by demonstrating high tracking performance in cluttered environments. Despite its power the CONDENSATION algorithm has a remarkably simple form and this allows the use of non-linear motion models which combine characteristics of discrete Hidden Markov Models with the continuous Auto-Regressive Process motion models traditionally used in Kalman filters. These mixed discrete-continuous models have promising applications to the emerging field of perception of action. This thesis also implements two algorithms to smooth the output of the CONDENSATION filter which improves the accuracy of motion estimation in a batch-mode procedure after tracking is complete.

Acknowledgements

I would like to thank my supervisor, Professor Andrew Blake, for invaluable direction and assistance in the development of this work. Thank you also to Simon Rowe, David Reynard, Andrew Wildenberg, Bob Kaucic, Colin Davidson, Ben North, Benedicte Bascle, John MacCormick, Josephine Sullivan and Jens Rittscher for many fruitful discussions, and for writing much essential and non-essential software. Thanks especially to Simon for setting me on the true path towards the optimal Xemacs configuration, Ben for continuing the Xemacs struggle and believing in the power of TEX, and Colin for supplying a broad range of general discussions and celebrity movie gossip.

Contents

1

Introduction and literature review

Over the last decade, visual motion analysis has emerged as one of the principal areas of research within the computer vision community. The increasing interest is due in part to the falling cost of computing power and, perhaps as important, the storage necessary to process extended image sequences. A sequence of images collected at or near video rate typically does not change radically from frame to frame, and this redundancy of information over multiple images can be extremely helpful in disambiguating the visual input, whether to track individual objects or to perform a more general motion segmentation. This should not be surprising given the experience of human observers watching, for example a camouflaged animal which cannot be seen against its background until it begins to move. The problem of taking full advantage of the redundancy in an image sequence is challenging. Our approach is to build probabilistic models to describe the likely motion and appearance of an object of interest. Together with a sequence of input images, these models define a probability density function (p.d.f.) which encodes all the available information about the positions and velocities of the object in the sequence. The remaining task is to design algorithms for filtering the input images in order to compute an approximation to this p.d.f.

The purpose of this thesis is to establish a stochastic framework for tracking curves in visual clutter, using a sampling algorithm. The approach is rooted in ideas from statistics, control theory and computer vision. The problem is to track outlines and features of foreground objects, modelled as curves, as they move in *substantial* clutter, and to do it at, or close to, video frame-rate. The algorithm developed here to address this tracking

problem, called CONDENSATION for CONditional DENSity propagATION, has recently been derived independently by several researchers, and is generating significant interest in the statistics and signal processing communities. This thesis therefore aims both to contribute to the literature on CONDENSATION-like filters, by presenting some novel applications and extensions, and to contribute to the visual motion estimation literature by demonstrating high tracking performance in cluttered environments. We also develop non-linear motion models which are easily incorporated in the CONDENSATION framework and have promising applications to the emerging field of perception of action.

1.1 Overview

This chapter presents a brief review of literature relevant to this thesis. Visual tracking papers from the computer vision community are covered as well as Bayesian statistical methods for analysing single images and a variety of nonlinear filtering approaches from control theory, statistics and signal-processing.

Chapter 2 outlines an "active contour" framework for visual tracking and provides background detail on the methods used for representing curves using B-splines and learning statistical models of shape and motion around these curve representations. There is also a brief discussion of simple image-processing considerations.

Chapter 3 includes a more technical discussion of Bayesian nonlinear filtering techniques and then describes the random-sampling approach which we adopt, and sets out the CONDENSATION algorithm to implement these ideas. The algorithm is described in its generality and observation and motion models are outlined which allow the filter to be tested on visual tracking problems in heavy clutter. A series of these experiments are then presented in chapter 4.

Chapter 5 presents a more complex motion model which allows discrete switching between multiple simpler models. The simplicity of incorporating such a model in the CONDENSATION algorithm is demonstrated, and the ideas are illustrated with experiments tracking a bouncing ball and a hand drawing with a pen.

Chapter 6 describes a modification to the sampling scheme used in CONDENSATION which allows the filter to take advantage of information from multiple measurement sources, and demonstrates the applicability of the method using a hand-tracking application which combines a simple colour-based blob-tracker with the contour models used in previous chap-

ters. Due to the increased efficiency of the sampling, the algorithm is able to run in real time, and as a side-effect a re-initialisation procedure can be implemented, bringing the hand-tracker close to a practical automatic system.

Chapter 7 presents algorithms to "smooth" the output of the CONDENSATION algorithm, performing a batch-mode procedure after tracking is complete which allows refined estimates of the object's state to be computed in the light of later measurements. The smoothing algorithms are applied to results from earlier chapters, showing increased accuracy of the estimates.

Finally, in chapter 8 there is further discussion of points raised in earlier chapters, as well as an outline of some promising areas of future research. A fuller analysis is also possible, in the light of the preceding chapters, of related literature which describes work similar to that presented in this thesis.

A web page at `http://www.robots.ox.ac.uk/~misard/condensation.html` describes current research relating to the CONDENSATION algorithm as well as showing MPEG movies of tracking performance.

1.2 Visual tracking

Tracking has been studied extensively in the computer vision literature, both because of its intrinsic interest and because of the large number of applications. For example, autonomous robots may need to be able to follow objects in their environment (Reid and Murray, 1996; Pahlavan et al., 1993; Davison and Murray, 1998; Murray et al., 1993; Espiau et al., 1992); one commonly studied special case of this concerns autonomous guided vehicles for driving on roads, which must track the features of the road (Dickmanns, 1992; Crisman, 1992) and also other moving vehicles (Smith, 1995). Static systems may also be used to track vehicles, either to collect traffic data from highway scenes (Ferrier et al., 1994; Koller et al., 1994) or to analyse complex environments such as airports (Sullivan, 1992). Tracking may also be used in robot arm applications to capture multiple views of an object from a moving camera and thus compute trajectories for exploring freespace (Blake et al., 1992) or to select an optimal grasp to pick up the object (Taylor, 1995). There is increasing interest in using computer vision to augment a computer's user-interface (Roy et al., 1997), including using lip-tracking to aid speech recognition (Bregler and Omohundro, 1995; Hennecke et al., 1995; Petajan and Graf, 1995). The digital desk paradigm (Wellner, 1991; Wellner, 1993) predicts a return of

the desktop metaphor to physical reality, whereby the user's desk is replaced by a projection screen and watched by a camera, and virtual documents can co-exist with real documents in the workspace. Reliable hand-tracking is vital for this goal, and various systems have been proposed for both tracking (Rehg and Kanade, 1994; Blake and Isard, 1994; Sullivan and Blake, 1997) and gesture recognition (Freeman and Roth, 1995; Kjeldsen and Kender, 1996; Cohen et al., 1996; Starner and Pentland, 1995; Yacoob and Black, 1998). Hand gestures are a special case of the developing field of "perception of action" which attempts to use tracking information to infer knowledge about a scene. This has roots in the tracking of people (Hogg, 1983; Baumberg and Hogg, 1994; Baumberg and Hogg, 1995b; Fernyhough et al., 1996; Yacoob and Black, 1998; Bregler, 1997; Haritaoglu et al., 1998; Bregler and Malik, 1998) for surveillance applications, as well as creating artificial environments (Maes, 1993; Intille et al., 1997; Wren et al., 1997) which respond to human actions, for example creating an interactive playroom for children (Intille et al., 1997). There is much current interest in learning to classify the output of such trackers into behaviours, for example (Starner and Pentland, 1995; Bobick and Wilson, 1995; Freeman and Roth, 1995; Kjeldsen and Kender, 1996). General techniques for tracking, not tied to any particular application, include the use of optic-flow information, for example (Koenderink and van Doorn, 1975; Horn and Schunk, 1981; Ju et al., 1996), rigid three-dimensional models (Harris, 1992; Lowe, 1992) and contour outlines (Kass et al., 1987; Cootes et al., 1993; Blake and Isard, 1998). More details of some of these techniques are given in the sections which follow.

1.2.1 Low-level or image-based tracking

We use the term "low-level" to informally group tracking methods which use only very weak assumptions about the object of interest in the image-processing stage. Generic features are extracted from the image, and only then grouped or interpreted according to higher-level knowledge about the scene. This contrasts with methods described in later sections which make use of strong modelling constraints to guide even the lowest-level image-processing stages. Recently a number of systems (Intille et al., 1997; Wren et al., 1997; Bregler, 1997; Maes, 1993; Haritaoglu et al., 1998; Roy et al., 1997) have been developed which conform to this notion of low-level tracking. Intille et al. (1997) construct a "blob-tracker" for real-time tracking of people viewed from directly above in a room. They use background subtraction to identify foreground regions, and then segment these regions into blobs based on colour. The blobs are then clustered using a simple algorithm based on proximity and velocity, to

identify a number of objects each of which is assumed to correspond to a single person. This technique is fast but merges blobs when people come close to each other, and relies on a top-down view and a fairly static background. Wren et al. (1997) build a system which can track a single person viewed from a camera angle in front of and above the person. They again isolate blobs using colour information, and then use prior information about e.g. the colour of hands and faces and the topology of a person's body to interpret the set of blobs as a figure. The correspondence model associating blobs with arms, torso, head, etc. is dynamically updated as blobs pass in and out of view to deal with occlusions. Since the view is frontal, gestures such as pointing and bending over can be recognised, but the blob representation precludes accurate fine-scale location of, for example, the hand's position. Bregler and Malik (1997) segment pixels into a set of blobs each represented by a Gaussian probability distribution on the basis of optic flow-determined velocities and colour. They use an expectation-maximisation (EM) algorithm in a hierarchical filtering framework, using a Kalman filter to track each blob and initialise the EM iteration for the next time-step. Finally they segment the sequence of blob parameters using a mixed discrete-continuous model, combining auto-regressive processes and Hidden Markov Models, similar to that developed in chapter 5 (although their mixed-state model is applied only once tracking is complete). Other successful tracking methodologies which do not use an explicit object model include the Hausdorff-distance tracker of Huttenlocher et al. (1993) and systems which track point features in an image-stream and use geometric rigid-body constraints to group sets of features into clusters belonging to the same object (Torr and Murray, 1994; Costeira and Kanade, 1995; Torr, 1997).

1.2.2 Optic-flow based tracking

Optic-flow has long been used (Koenderink and van Doorn, 1975; Horn and Schunk, 1981; Black and Anandan, 1993; Ju et al., 1996) as a way both to estimate dense motion fields over the entire visible region of an image sequence, e.g. (Black and Anandan, 1993; Ju et al., 1996), and to segment areas of consistent flow into discrete objects, e.g. (Black and Jepson, 1996; Weber and Malik, 1995). In order to solve the optic-flow constraint equation it is necessary to either apply regularisation, assuming change in motion is smooth over an image region, or parameterise the motion in an entire region using a low-dimensional model, for example an affine model. Black et al. have developed a series of robust methods for determining optic flow (Black and Anandan, 1993; Jepson and Black, 1993; Black and

Jepson, 1996; Black et al., 1997; Ju et al., 1996). The "skin and bones" model (Ju et al., 1996) combines many of the techniques in their earlier papers to determine a dense motion field as a tiling of the image. Each tile may contain multiple affine motions, and these motions are robustly regularised across adjoining tiles to provide smooth motion information even in regions with little texture. Bregler (1998) returns to the problem of tracking people, with the aim of determining pose very accurately over image sequences. As in (Bregler, 1997), described in the last section, body parts are represented by connected blobs segmented using parameterised affine optic-flow estimates, although now instead of using Gaussian distributions for the blobs there is an explicit pixel-based probability mask allowing blobs of arbitrary shape. The parameterised model is further constrained during the estimation process in this work by a 3D model of the human skeleton, which is broken down using twists and exponential maps to make computation locally simpler and permit tracking with reasonable computational expense. Modern developments of correlation tracking employ similar techniques to parameterised optic-flow estimation. For example the framework adopted by Hager and Toyoma (1996) for correlation tracking of a rectangular image patch undergoing affine deformations is closely related to parameterised optic-flow based methods; where optic-flow methods estimate affine parameters of deformation between consecutive images, the correlation tracker estimates parameters relative to an initial template image. A very efficient algorithm is presented in (Hager and Toyama, 1996) which transfers most of the computation to an off-line processing stage and allows affine correlation tracking to proceed in real time.

1.2.3 Curve tracking

"Snakes" were introduced by Kass et al. (1987) to perform robust segmentation and region tracking by modelling an object using outline contour information which is relatively insensitive to lighting variations, and imposing smoothness constraints on the curvature of the contour and the motion of the object. This is more general than modelling entire objects but more clutter-resistant than applying signal-processing to low-level corners or edges. This active contour idea has since been used and extended by many researchers, e.g. (Menet et al., 1990; Cipolla and Blake, 1990; Cootes et al., 1993; Blake et al., 1993b; Blake and Isard, 1994; Baumberg and Hogg, 1995b; Lanitis et al., 1995; Blake and Isard, 1998). Curves can be represented in a state-space of B-spline coefficients (Bartels et al., 1987; Menet et al., 1990; Cipolla and Blake, 1990) and shape-space models (Cootes et al., 1993) can be used to

define prior probability densities over curves and their motions (Terzopoulos and Metaxas, 1991; Blake et al., 1993b). Reasonable defaults can be chosen for those densities, however it is obviously more satisfactory to measure or estimate them from data-sequences. Algorithms to do this, assuming Gaussian densities, are known in the control-theory literature (Goodwin and Sin, 1984) and have been applied in computer vision (Blake and Isard, 1994; Baumberg and Hogg, 1995b). Blake and Isard have developed an active contour framework (Blake and Isard, 1998) which is used in this thesis and is more fully described in chapter 2. Contour models have also been extended to include deformable textured regions within the contours (Ivins and Porrill, 1995; Bascle and Deriche, 1995; Lanitis et al., 1995; Sclaroff and Isidoro, 1998; Cootes et al., 1998). A recent development is the use of level-set snakes (Paragios and Deriche, 1998) to replace traditional B-spline based snakes. An energy function is defined over the image, and fast algorithms are used to track level sets of this function. An advantage of the approach is that the topology of the level sets may change, although there is no parametric representation of the object, so the problem addressed is more akin to motion segmentation than tracking. Also, existing methods have only been applied where background subtraction can be used, and have not been demonstrated in image clutter.

1.3 Kalman filters and data-association

Spatio-temporal estimation, the tracking of shape and position over time, has been dealt with thoroughly by Kalman filtering, in the case in which the state's probability density function (p.d.f.) $p(\mathbf{X}_t|\mathcal{Z}_t)$ (where $\mathbf{X}_t$ is the object state at time t and $\mathcal{Z}_t = (\mathbf{Z}_1, \ldots, \mathbf{Z}_t)$ is the measurement history to time t) can satisfactorily be modelled as Gaussian (Dickmanns and Graefe, 1988; Harris, 1992; Gennery, 1992; Rehg and Kanade, 1994; Matthies et al., 1989) and the Kalman filter can be applied to image-curves (Terzopoulos and Szeliski, 1992; Blake et al., 1993b). For the state density to remain Gaussian it is necessary that the prior, process and measurement densities all be Gaussian also (in the usual case the measurement and process noise are Gaussian and the update equations are linear, which results in a Gaussian state density as required). Bar-Shalom and Fortmann (1988) describe a number of standard extensions to the Kalman filter for dealing with situations where non-Gaussian densities may be encountered. The Extended Kalman Filter (EKF) is appropriate in the case of a non-linear but unimodal process which can be well approximated over the length of a single timestep by its local linearisation. The EKF has been used in visual tracking,

e.g. (Harris, 1992; Jebara et al., 1998). In some cases an exact filter may be derived even in the case that the dynamics are non-linear, for example Maybank et al. (1996) construct a filter for tracking a car based on position and steering angle parameters. The introduction of clutter can cause the observation density to be highly non-Gaussian, by introducing multiple modes corresponding to the clutter features. A multi-modal measurement density necessarily induces multi-modality into the state density. The "Probabilistic Data Association Filter" (PDAF) (Bar-Shalom and Fortmann, 1988) is designed for the case of image clutter where the process is linear and Gaussian, and the observation density is a mixture of Gaussians. The PDAF continues to use the standard Kalman filter framework by approximating all the visible measurements, weighted by their predicted likelihoods, into a single Gaussian-distributed feature, and so it continues to represent the state density as a single Gaussian. When a multi-modal state density is required, one solution is to use a mixture of Gaussians to represent $p(\mathbf{X}_t|\mathcal{Z}_t)$. The "Joint PDAF" (JPDAF) (Bar-Shalom and Fortmann, 1988) is an extension of the PDAF where in principle the state density is evaluated exactly and represented as a mixture of Gaussians (Sorenson and Alspach, 1971). The number of terms in the mixture increases exponentially, however, so pruning and merging of hypotheses is required to run within a fixed computational bound. The particular form of observation density for the active contour framework used in this thesis makes the JPDAF impractical due to the very large number of terms in the observation density mixture and this is discussed in chapter 3. A multi-modal process density, as used in chapter 5, also results in the state density becoming multi-modal. The Interacting Multiple Model (IMM) filter (Blom and Bar-Shalom, 1988) is analagous to the JPDAF when it is the process rather than (or as well as) the observation density which is multi-modal. Again, the number of terms in the mixture for the state density increases exponentially and pruning must be applied (Sorenson and Alspach, 1971).

An alternative solution to the problem of clutter is to try to label features as inliers or outliers and thus avoid the problem of a multi-modal observation density by explicitly discarding clutter features. In simple cases a validation gate (Bar-Shalom and Fortmann, 1988) and heuristics may be sufficient within a Kalman filter to identify the correct features. With discrete features such as points or corners combinatorial data-association methods can be effective with clutter. The RANSAC algorithm (Fischler and Bolles, 1981) has been used to classify point features as inliers or outliers based on geometric rigid-body constraints (Torr and Murray, 1994; Costeira and Kanade, 1995; Torr, 1997; Reid and Murray, 1996).

This approach has only been presented in the literature in the case that measurements are represented as a set of point features, and it is not clear how the algorithm could be practically applied when tracking curves (ignoring the special case where it is possible to reliably find "distinguished points" (Zisserman et al., 1993) on the curve). The RANSAC approach is also unappealing as a general solution to tracking in clutter since it lacks a mechanism for temporal propagation of information. Lowe (1992) describes a system which attempts to deal with cluttered measurements by searching the image for a set of features which is consistent with an object model. A feature set is rejected if its deviation from the model exceeds a threshold set according to the expected noise of the measurement process. He uses tree searching with backtracking to find the set, but improves the efficiency by first considering those features which are closest to the predicted model position.

Finally, one very general approach to nonlinear filtering must be mentioned. This is simply to integrate the state evolution equations directly, using a suitable numerical representation of the state density such as finite elements. This in essence is what Bucy (1969) proposed and more recently Hager (1990) investigated with respect to robotics applications. It is usable in one or two dimensions but, complexity being exponential in the dimension, is altogether infeasible for problems of dimension around 6–20, typical of the tracking problems dealt with here.

1.4 Bayesian analysis of static images

A standard problem in statistical pattern recognition is to find a model parameterised by $\mathbf{X}$ in a single image $\mathbf{Z}$. The problem is usually framed in terms of finding the maximum likelihood value $\hat{\mathbf{X}}$, or possibly several modes of the likelihood. It is the generalisation of this problem to finding the trajectory of an object in an image *sequence* which is the subject of this thesis. The information of interest for the localisation of the object is expressed in the posterior distribution $p(\mathbf{X}|\mathbf{Z})$, but in general it is difficult to calculate $p(\mathbf{X}|\mathbf{Z})$ directly, so Bayes' rule is applied;

$$p(\mathbf{X}|\mathbf{Z}) = \frac{p(\mathbf{Z}|\mathbf{X})p(\mathbf{X})}{p(\mathbf{Z})} \tag{1.1}$$

where $p(\mathbf{Z})$ is a constant not dependent on $\mathbf{X}$ for a given image, and so can be neglected in the case where only relative likelihoods need be considered. Now the problem is to represent the distributions $p(\mathbf{X})$ and $p(\mathbf{Z}|\mathbf{X})$, corresponding to the prior information about the object configuration, and the observation model respectively. Here the likelihood $p(\mathbf{Z}|\mathbf{X})$

is considered a function of $\mathbf{X}$ for fixed $\mathbf{Z}$. These distributions can be estimated from real data, or by assuming reasonable defaults and relationships. In the case that $p(\mathbf{X})$ and $p(\mathbf{Z}|\mathbf{X})$ are Gaussian, it is straightforward to estimate $\hat{\mathbf{X}}$. Least-squares estimation gives $\hat{\mathbf{X}}$ as the output of a Wiener filter (Gonzales and Wintz, 1987); $p(\mathbf{X}|\mathbf{Z})$ is also Gaussian with mean $\hat{\mathbf{X}}$, and there is in principle a closed form method of obtaining the variance $V(\mathbf{X}|\mathbf{Z})$ which, with the mean, completely specifies the distribution. In cases of interest, however, $\mathbf{Z}$ may be the whole image or a complex set of features in the image, so while $p(\mathbf{X})$ may sometimes be approximated by a Gaussian, $p(\mathbf{Z}|\mathbf{X})$ is highly non-Gaussian, since each partial coincidence of a region of $\mathbf{Z}$ with a hypothesised model $\mathbf{X}$ leads to a local maximum of $p(\mathbf{Z}|\mathbf{X})$. These coincidences can be the result of image clutter, or even simply of symmetries in the model. In general therefore $p(\mathbf{X}|\mathbf{Z})$ is not available in analytic form.

Geman and Geman (1984) consider the use of pixel based models for $\mathbf{X}$, and set out a theoretical framework which is often cited in later papers. Their model is based on the assumption that different areas of the image, to be classified as distinct regions, have pixel intensities chosen from different Markov random fields (MRFs). An instance of the model, $\mathbf{X}$, assigns each pixel to a given region, and $p(\mathbf{X})$ and $p(\mathbf{Z}|\mathbf{X})$ therefore include a term for each pixel in the image, encoding the probability that it is a sample from the requisite MRF. They show that $p(\mathbf{X})$ for an MRF is equivalent to a Gibbs distribution, and thus can apply existing methods from mathematical physics to treat such distributions. They apply two algorithms to the problem — the Metropolis algorithm (Metropolis et al., 1953) to find the mean of the distribution, and a form of simulated annealing to compute the mode. Both algorithms rely on iterative simulation, where small updates in the model are made at each step, and over many iterations the set of realisations swept out approximates the prior $p(\mathbf{X})$. One disadvantage to the pixel based representation is that it does not take into account any idea of an underlying true image which is being quantised by the imaging hardware. In particular, the same image, viewed at a different pixel resolution, would have an entirely different model.

Similar iterative simulation techniques, under the name of "factored sampling" have been applied by Grenander et al. (1991) to find hands in noisy images. The factored sampling approach is to approximate the posterior density $p(\mathbf{X}|\mathbf{Z})$ by a discrete set of realisations from the prior $p(\mathbf{X})$ weighted by the observation density $p(\mathbf{Z}|\mathbf{X})$ and this is discussed more fully in section 3.3 on page 39. Rather than using a model prior based on a Markov Random Field, however, they choose to encode in $p(\mathbf{X})$ an idea of *hand shape*. The

model space $\mathcal{M}$ which they use consists of a number of nodes connected by line segments. The prior $p(\mathbf{X})$ is a Markov chain on the angles between line segments and the lengths of the segments, together with a Gaussian distribution on global translation, scale and rotation. The Markov transition matrix is estimated from data of real hands chosen to represent a spread of likely hand shapes. This model has a number of advantages over pixel-based schemes; it is independent of the imaging process, and it allows a more elegant representation of certain prior information, for example that an object boundary is simply connected, than is possible using MRFs.

The iterative simulation techniques described are part of the Markov Chain Monte Carlo (MCMC) family of algorithms which have been very widely used in statistical image analysis, e.g. (Ripley and Sutherland, 1990; Storvik, 1994; Grenander and Miller, 1994; Miller et al., 1995). They have proved very successful in exploring distributions over single images, but have no natural extension to the recursive filtering of time sequences.

1.5 Random sampling approaches to Bayesian filtering

Bayesian approaches have also been applied to temporal sequences where information from a series of measurements (in our case images) must be combined, along with a prior model of object motion. A form of sequential Bayesian filtering was first described by Handschin and Mayne (1969; 1970) and Akashi and Kumamoto (1977). Handschin and Mayne address the non-linear filtering problem: given a known process model

$$\mathbf{X}_t = f(\mathbf{X}_{t-1}, \mathbf{w}_t, t) \tag{1.2}$$

and observation model

$$\mathbf{Z}_t = g(\mathbf{X}_t, t) + \mathbf{v}_t \tag{1.3}$$

where $\mathbf{w}_t$ and $\mathbf{v}_t$ are vectors of random noise, and a series of observations $\mathcal{Z}_T$, find the best filtered state estimate

$$\hat{\mathbf{X}}_T = \mathcal{E}[p(\mathbf{X}_T|\mathcal{Z}_T)].$$

They approach the problem by generating a set of N randomly sampled sequences $\mathcal{S}_T^{(n)} = \{\mathbf{s}_1^{(n)}, \ldots, \mathbf{s}_T^{(n)}\}$. A sequence $\mathcal{S}_T^{(n)}$ is generated as follows: first sample from the prior distribution $p(\mathbf{X}_1)$ to find $\mathbf{s}_1^{(n)}$, then for $t = 2 \ldots T$ compute $\mathbf{s}_t^{(n)}$ by sampling the noise model

to generate $\mathbf{w}_t^{(n)}$ and applying (1.2). The probability of a sequence $p(\mathcal{S}_T^{(n)}|\mathcal{Z}_T)$ can then be evaluated from

$$p(\mathcal{S}_T^{(n)}|\mathcal{Z}_T) = \prod_{t=1}^{T} p(\mathbf{Z}_t|\mathbf{s}_t^{(n)})$$

where $p(\mathbf{Z}_t|\mathbf{s}_t^{(n)})$ can be derived from (1.3). It is then possible to estimate the desired expectation $\mathcal{E}[\cdot]$ from

$$\mathcal{E}[p(\mathbf{X}_T|\mathcal{Z}_T)] \approx \frac{\sum_{n=1}^{N} \mathbf{s}_T^{(n)} p(\mathcal{S}_T^{(n)}|\mathcal{Z}_T)}{\sum_{n=1}^{N} p(\mathcal{S}_T^{(n)}|\mathcal{Z}_T)}.$$

This approximation may however be expected to deteriorate rapidly as T becomes large, for fixed N, indeed for processes with Gaussian noise one might expect to need to set $N \propto \sqrt{T}$ in order to maintain a given level of accuracy in estimates over long sequences. The "control variate method" is proposed in (Handschin and Mayne, 1969; Handschin, 1970) to reduce the variance of estimates in the case that the system in (1.2) and (1.3) can be well-approximated using an Extended Kalman Filter. In that case a modified filter is derived where random sampling is used to compute only the deviations of the model from the EKF estimates found analytically. Akashi and Kumamoto (1977) propose essentially the same sequential Bayesian algorithm, but apply it to the specific problem of estimation in switching environments, where the system noise characteristics are assumed to be governed by a Markov process.

The problem of unbounded growth of the sample-set size N can be addressed by *re-sampling* the random sequences at each timestep according to the observation density. Resampling in the context of static probability distributions is described by Rubin (1988) where he calls it the "Sampling Importance Resampling" (SIR) algorithm, and performed in a Bayesian context as a weighted bootstrap by Smith and Gelfand (1992). The extension of the resampling idea to the recursive filtering of time series data was recently independently discovered by several researchers (Gordon et al., 1993; Kitagawa, 1996; Isard and Blake, 1996). The algorithm to do this in the context of computer vision, which we denote CONDENSATION (Isard and Blake, 1996; Isard and Blake, 1998a) for CONditional DENSity propagATION, is the focus of this thesis and is fully set out in chapter 3. There has been significant interest generated by these filtering methods both in the statistics and signal processing (Gordon et al., 1993; Gordon and Salmond, 1995; Kitagawa, 1996; Carpenter et al., 1997; Pitt and Shepherd, 1997; Doucet, 1998) and computer vision (Isard and Blake,

1996; Isard and Blake, 1998a; Isard and Blake, 1998c; Heap and Hogg, 1998; Isard and Blake, 1998b; Isard and Blake, 1998d; Black and Jepson, 1998) communities (chapters 3–7 contain material originally published as (Isard and Blake, 1998a; Isard and Blake, 1998c; Isard and Blake, 1998d; Isard and Blake, 1998b)). Gordon (1993; 1995) describes the algorithm, which he calls the "bootstrap filter" after the weighted bootstrap of (Smith and Gelfand, 1992), and applies it to the traditional filtering problem of bearings-only tracking. He describes some *ad-hoc* techniques to improve the efficiency of the sampling scheme, a discussion of which is deferred until section 8.3 on page 130 by which time the relevance to the work in this thesis will be clearer. The work described in (Carpenter et al., 1997; Pitt and Shepherd, 1997) will also be considered in section 8.3, for the same reasons. Kitagawa (1996) refers to the algorithm as a "Monte-Carlo filter" and his formulation is valid only for the special case that observations are one-dimensional. He also describes two smoothing algorithms which are reimplemented in the CONDENSATION framework in chapter 7. Doucet (1998) surveys a number of sequential sampling approaches, and interprets them as special cases of a general sequential importance sampling framework, which again will be considered in chapter 8. The CONDENSATION algorithm described in this thesis differs from (Gordon et al., 1993; Kitagawa, 1996; Carpenter et al., 1997; Pitt and Shepherd, 1997) largely in the form of the observation density. The algorithm is applied here to cluttered images, which we model using a mixture of a very large number of component distributions. The applications described in (Gordon et al., 1993; Kitagawa, 1996; Carpenter et al., 1997; Pitt and Shepherd, 1997) all use low-dimensional observation processes (e.g. bearings-only tracking, financial modelling using the Black-Scholes equation and biological population data) which can be modelled using comparatively simple mixture distributions. In some cases, given a tractable observation model, it is possible to take advantage of the form of the observations to improve the efficiency of the algorithm (Carpenter et al., 1997; Pitt and Shepherd, 1997; Doucet, 1998) and this is discussed in chapter 8, however it is not directly relevant to the research in this thesis, due to the complexity of the observation model used here.

Heap and Hogg (1998) describe an extension to CONDENSATION which is a special case of the mixed-model framework described in chapter 5. They represent hand outlines using a high-dimensional linear state-space, where realistic hand-shapes are a highly non-linear subset of the possible linear deformations, and are represented as a set of clusters in the high-dimensional space. A Hidden Markov Model is used to "jump" between clusters and

so track across discontinuities in the linear representation when, for example, the splayed fingers of the hand come together and the contours between individual fingers disappear. Black and Jepson (1998) also describe work related to chapter 5. They address the problem of classifying gestures tracked using an optic-flow based system. Distinct gestures are represented using trajectory models, and the problem is to find the best match of a test sequence to the available trajectories, suitably scaled and translated. A mixed-state CONDENSATION tracker is run on the test data where each distinct trajectory corresponds to a discrete state of the motion model and the continuous state-space vector $\mathbf{X}_t$ describes instantaneous scale and phase parameters of the model; the MAP trajectory is then estimated from the discrete label of the mixed-state tracker.

2

A B-spline framework for curve tracking

This chapter outlines a probabilistic "active contour" framework for visual tracking where objects are represented by B-spline curves in an image-stream. This framework was developed by Blake and a number of collaborators (Curwen, 1993; Blake et al., 1993a; Blake et al., 1995; Reynard et al., 1996; Rowe, 1996; Wildenberg, 1997; Kaucic, 1997; North and Blake, 1998) and is fully set out in (Blake and Isard, 1998). The shape and motion models used are very similar to those adopted by Cootes et al. (1994) and Baumberg and Hogg (1995b; 1995a). While the CONDENSATION algorithm and its extensions, presented in chapters 3, 5 and 6, apply generally to a large class of shape and motion models, the experiments presented in this thesis make use of the B-spline curve and auto-regressive process models which follow, with some small alterations described in later chapters.

2.1 Linear parameterisations of splines for tracking

Objects are modelled as a curve (or set of curves), typically though not necessarily the occluding contour, and represented at time t by a parameterised image curve $\mathbf{r}(s,t)$. The parameterisation is in terms of B-splines, so

$$\mathbf{r}(s,t) = (\mathbf{B}(s)\cdot\mathbf{Q}^x(t),\ \mathbf{B}(s)\cdot\mathbf{Q}^y(t)) \quad \text{for} \quad 0 \le s \le L \tag{2.1}$$

where $\mathbf{B}(s)$ is a vector $(B_0(s),\ldots,B_{N_B-1}(s))^T$ of B-spline basis functions, $\mathbf{Q}^x$ and $\mathbf{Q}^y$ are vectors of B-spline control point coordinates and L is the number of spans. For notational

simplicity the B-spline control points can be combined in a spline-vector

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

The general rule for computing B-spline basis functions in the case of multiple knots, which allow for sharp corners and breaks in the curve, is too complex to express here but is fully set out in (Blake and Isard, 1998). In the case of a closed quadratic B-spline with no multiple knots, $B_0(s)$ is given by

$$B_0(s) = \begin{cases} s^2/2 & \text{if } 0 \le s < 1 \\ \frac{3}{4} - (s - \frac{3}{2})^2 & \text{if } 1 \le s < 2 \\ (s-3)^2/2 & \text{if } 2 \le s < 3 \\ 0 & \text{otherwise} \end{cases}$$

and the others are simply translated copies:

$$B_n(s) = B_0(s - n)$$

where s is treated as periodic over the interval $0 \le s \le L$.

In practice, it is desirable to distinguish between the *spline*-vector $\mathbf{Q}$ that describes the basic shape of an object and the *shape*-vector which we denote $\mathbf{x} \in \mathcal{M}$, where $\mathcal{M}$ is a *shape-space*. Whereas $\mathcal{M}_Q$ is a vector space of B-splines and has dimension $N_Q = 2N_B$, the shape-space $\mathcal{M}_X$ is constructed from an underlying vector space of dimension N_X which is typically considerably smaller than N_Q. The shape-space is a linear parameterisation of the set of allowed deformations of a base curve. The necessity for the distinction is made clear in figure 2.1. To obtain a spline that does justice to the geometric complexity of the face shape, thirteen control points have been used. However, if all of the resulting 26 degrees of freedom of the spline-vector $\mathbf{Q}$ are manipulated arbitrarily, many uninteresting shapes are generated that are not at all reminiscent of faces. Restricting the displacements of control points to a lower-dimensional shape-space is more meaningful if it preserves the face-like quality of the shape. Conversely, using the unconstrained control-vector $\mathbf{Q}$ leads to unstable active contours and this is illustrated in figure 2.2.

The requirement that a shape-space be a *linear* parameterisation is made for the sake of computational simplicity in a Kalman filtering framework. Linearly parameterised, image-based models work well for rigid objects, and for simpler non-rigid ones. The experiments described in chapter 3 inherit this linear shape-space framework, but the CONDENSATION algorithm supports non-linear shape-spaces as naturally as linear ones, and in chapter 6

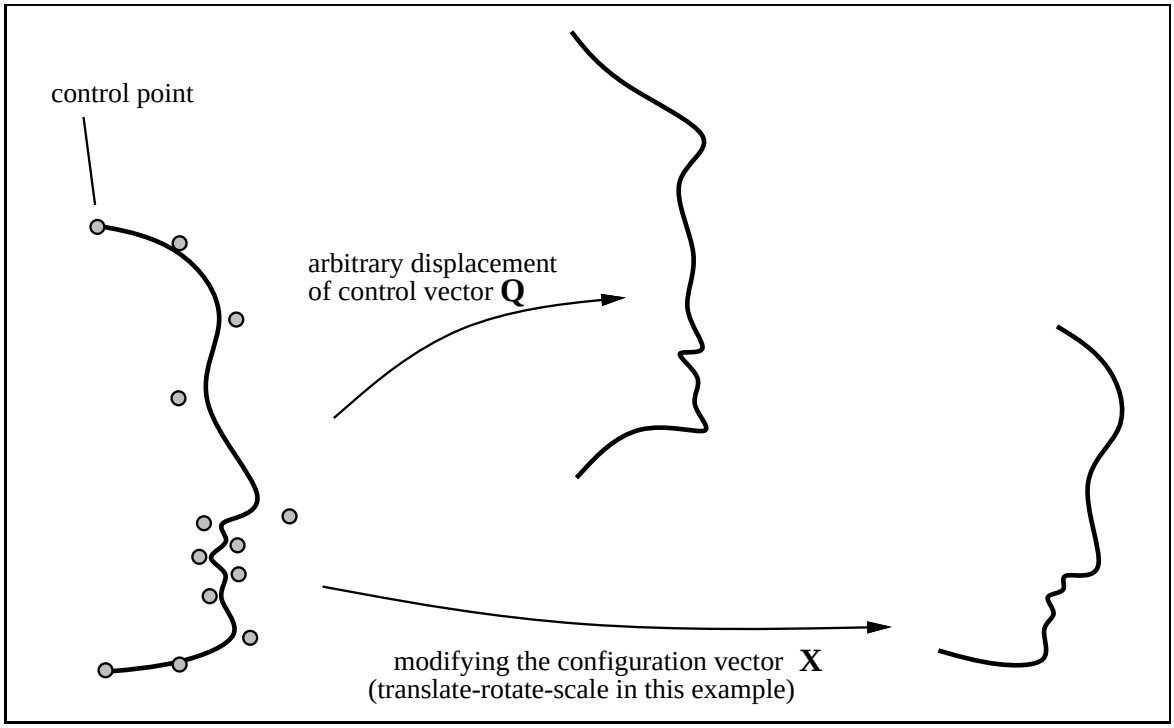


Figure 2.1: **Configuration vector.** *Arbitrary manipulation of the spline-vector* $\mathbf{Q}$ *of a spline curve is too general to be practically interesting. In this example a face curve ceases to look face-like. What is far more interesting is a restricted class* $\mathcal{M}$ *of transformations, parameterised by a relatively low-dimensional configuration vector* $\mathbf{x}$*. In this case* $\mathbf{x}$ *is a Euclidean similarity transformation which does retain the face-like character.*

a hand-tracker is implemented using a simple non-linear parameterisation of Euclidean similarity transformations. It is expected that complex articulated bodies will best be modelled using a non-linear parameterisation, perhaps similar to that used by Rehg (1994) for tracking an articulated hand using an Extended Kalman Filter or Bregler's full-body tracking using twists and exponential maps (Bregler and Malik, 1998).

A shape-space $\mathcal{M} = \mathcal{L}(W, \mathbf{Q}_0)$ is a linear mapping of a shape-space vector $\mathbf{x} \in \mathbb{R}^{N_X}$ to a spline-vector $\mathbf{Q} \in \mathbb{R}^{N_Q}$:

$$\mathbf{Q} = W\mathbf{x} + \mathbf{Q}_0, \tag{2.2}$$

where W is an $N_Q \times N_X$ "shape-matrix." The constant offset $\mathbf{Q}_0$ is a template curve against which shape variations are measured; for instance, a class of shapes consisting of $\mathbf{Q}_0$ and curves close to $\mathbf{Q}_0$ could be expressed by restricting the shape-space $\mathcal{M}$ to "small" $\mathbf{x}$. For a planar shape just six affine degrees of freedom are required to describe, to a good approximation, the possible shapes of its bounding curve. The planar affine group can be

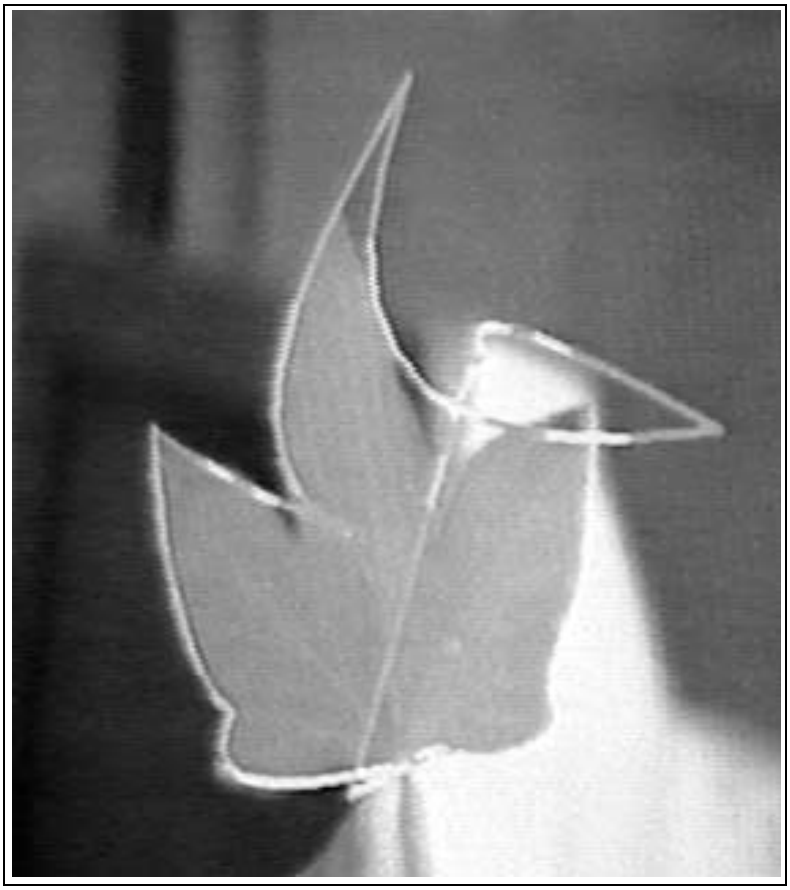


Figure 2.2: **The need for shape-spaces.** *The white curve is a B-spline with sufficient control points to do justice to the complexity of the leaf's shape. Control point positions vary over time in order to track the leaf outline. However, if the curve momentarily loses lock on the outline it rapidly becomes too tangled to be able to recover. (Figure by courtesy of R. Curwen.)*

viewed as the class of all linear transformations that can be applied to a template curve $\mathbf{r}_0(s)$:

$$\mathbf{r}(s) = \mathbf{u} + M\mathbf{r}_0(s), \tag{2.3}$$

where $\mathbf{u} = (u_1, u_2)^T$ is a two-dimensional translation vector and M is a 2×2 matrix, so that $M, \mathbf{u}$ between them represent the 6 degrees of freedom of the space. This class can be represented as a shape-space with template $\mathbf{Q}_0$ and shape-matrix:

$$W = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{Q}_0^x & \mathbf{0} & \mathbf{0} & \mathbf{Q}_0^y \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{Q}_0^y & \mathbf{Q}_0^x & \mathbf{0} \end{pmatrix} \tag{2.4}$$

where $\mathbf{1} = (1\ 1\ \ldots\ 1)^T$ and $\mathbf{0} = (0\ 0\ \ldots\ 0)^T$ are vectors each with N_B components. The first two columns of W represent horizontal and vertical translation. By convention, the template

$\mathbf{r}_0(s)$ represented by $\mathbf{Q}_0$ is chosen with its centroid at the origin. Then the remaining four affine motions (figure 2.3), which do not correspond one-for-one to the last four columns of W, can be expressed as simple linear combinations of those columns. Recall that the shape-space transformation is $\mathbf{Q} = W\mathbf{x} + \mathbf{Q}_0$ so that the elements of $\mathbf{x}$ act as weights on the columns of W. The interpretation of those weights in terms of planar transformations (2.3) of the template is:

$$\mathbf{x} = (u_1, u_2, M_{11} - 1, M_{22} - 1, M_{21}, M_{12})^T. \tag{2.5}$$

Some examples of transformations are:

1. $\mathbf{x} = (0, 0, 0, 0, 0, 0)^T$ represents the original template shape $\mathbf{Q}_0$
2. $\mathbf{x} = (1, 0, 0, 0, 0, 0)^T$ represents the template translated 1 unit to the right,
3. $\mathbf{x} = (0, 0, 1, 1, 0, 0)^T$ represents the template doubled in size
4. $\mathbf{x} = (0, 0, \cos\theta - 1, \cos\theta - 1, -\sin\theta, \sin\theta)^T$ represents the template rotated through angle θ
5. $\mathbf{x} = (0, 0, 1, 0, 0, 0)^T$ represents the template doubled in width

In practice it is convenient to arrange for the elements of the affine basis to have similar magnitudes to improve numerical stability. If the control-vector $\mathbf{Q}_0$ is expressed in pixels, for computational simplicity, the magnitudes of the last four columns of the shape-matrix may be several hundred times larger than those of the first two, and it is then necessary to scale the translation columns to match.

2.2 Key-frames

Affine spaces are appropriate for modelling the appearance of three-dimensional rigid body motion. In many applications, motion is decidedly non-rigid. A simple methodology to deal with this situation is to use "key-frames" or representative image outlines of the moving shape. Often, an effective shape-space can be built by linear combination of such key-frames. The next section describes a statistical modelling approach to learn a shape-space from a training set of sample motion.

Figure 2.4 shows a sequence of three frames which can be used to build a simple shape-space in which the first frame $\mathbf{Q}_0$ acts as the template and the shape-matrix W is constructed

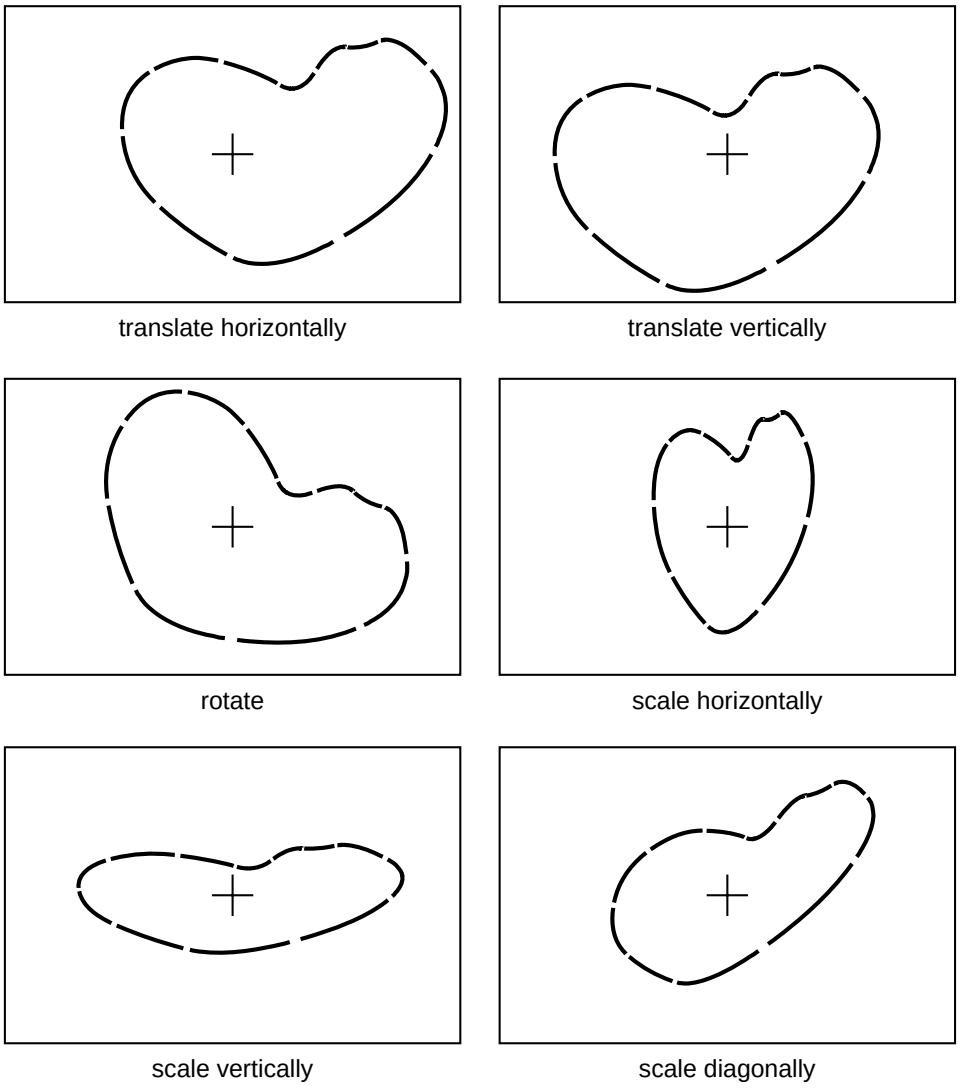
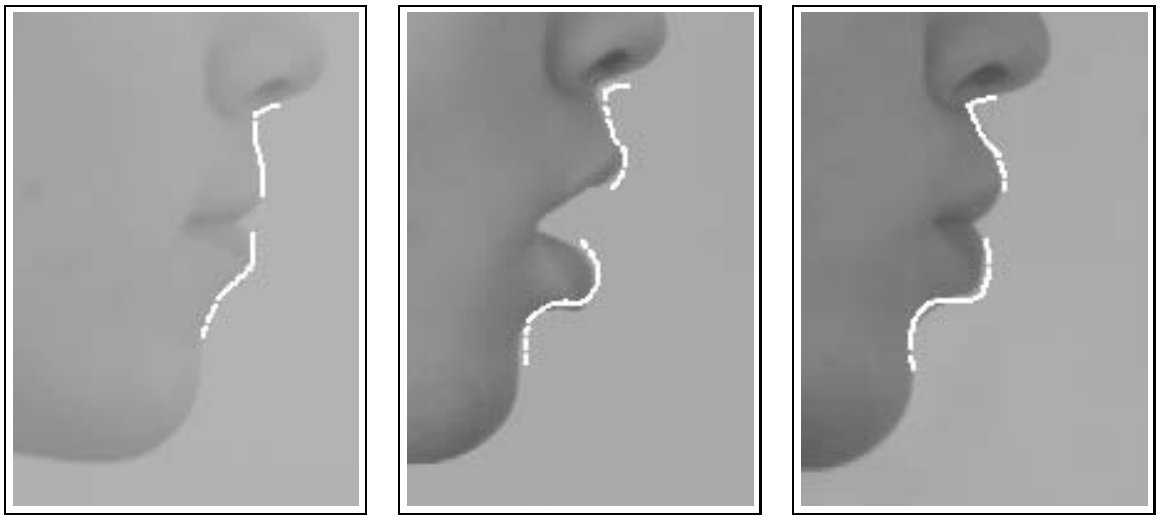


Figure 2.3: **Planar affine basis.** *The planar affine transformation group has 6 degrees of freedom and a basis for them is depicted here. The first three elements of the basis correspond to translation and rotation of a rigid template. The last three elements span a subspace that includes magnification of the template and two further degrees of freedom for directional scaling. Directional scaling occurs when a planar object, initially co-planar with the image, is allowed to rotate about an axis that lies parallel to the image plane.*

from the two key-frames $\mathbf{Q}'_1, \mathbf{Q}'_2$:

$$W = \begin{pmatrix} \mathbf{Q}_1^x & \mathbf{Q}_2^x \\ \mathbf{Q}_1^y & \mathbf{Q}_2^y \end{pmatrix}. \tag{2.6}$$

where $\mathbf{Q}_i = \mathbf{Q}'_i - \mathbf{Q}_0$. This two-dimensional shape-space is sufficient to span all linear



Template $\mathbf{Q}_0$ *Key-frame: opening* $\mathbf{Q}_1$ *Key-frame: protrusion* $\mathbf{Q}_2$

Figure 2.4: **Key-frames.** *Lips template followed by two key-frames, representing interactively tracked lips in characteristic positions. The key-frames are combined linearly with appropriate rigid degrees of freedom, to give a shape-space suitable for use in a tracker for non-rigid motion.*

combinations of the three frames. What is more, the shape-space coordinates have clear interpretations, for example:

- $\mathbf{x} = (0, 0)^T$ represents the closed mouth;
- $\mathbf{x} = (1/2, 0)^T$ represents the half-open mouth;
- $\mathbf{x} = (1/4, 1/2)^T$ represents the mouth, half-protruding and slightly open.

The same three frames can be used to build a larger shape-space that allows for translation, zooming and rotation of any of the expressions from the simple two-dimensional shape-space; this and more general methods for analytically constructing shape-spaces, for example permitting articulated objects, are described in (Blake and Isard, 1998).

2.3 Principal components analysis

When the number of available training outlines $\mathbf{Q}_1, \ldots, \mathbf{Q}_M$ of an object is much larger than the effective number of degrees of freedom of the object it is inappropriate to treat them as independent key-frames as above spanning a shape space $\mathcal{M}$. Instead they can be used to determine a smaller shape-space $\mathcal{M}' = \mathcal{L}(W', \mathbf{Q}'_0) \subset \mathcal{M}$, a subspace of $\mathcal{M}$ with dimension N'_X, that spans, at least approximately, all of the shapes in the training sequence. This idea was first introduced by Cootes and Taylor (1993), in the special case of polygonal contour models, which they dubbed the "Point Distribution Model" or PDM. Their approach, using classical principal components analysis (PCA), is not suitable for spline-based curve representations, and so the following modification (Blake et al., 1995) is used to find principal components based on the L_2 norm in spline-space. Given a long ($M > N_X$) training sequence, solve:

$$\min_{W', \mathbf{Q}'_0, \mathbf{X}'_1, \ldots, \mathbf{X}'_{N'_X}} \left(\sum_{k=1}^{M} \|\mathbf{Q}_k - \mathbf{Q}'_k\|^2 \right), \tag{2.7}$$

where

$$\mathbf{Q}'_k = W'\mathbf{X}'_k + \mathbf{Q}'_0 \quad \text{and} \quad \mathbf{Q}_k = W\mathbf{X}_k + \mathbf{Q}_0.$$

The distance measure $\|\cdot\|$ is the L_2-norm in spline-space (Blake et al., 1995), and the solution to this problem gives $\mathbf{Q}'_0 = \overline{\mathbf{Q}}$, the mean of the training sequence, and W' is a matrix whose columns are the first N'_X of the orthonormal eigenvectors of the matrix $\Sigma\mathcal{U}$ where

$$\Sigma = \frac{1}{M} \sum_{k=1}^{M} (\mathbf{Q}_k - \overline{\mathbf{Q}})(\mathbf{Q}_k - \overline{\mathbf{Q}})^T,$$

$$\mathcal{U} = \frac{1}{L} \int_0^L U(s)^T U(s)\, ds \quad \text{and}$$

$$U(s) = \begin{pmatrix} \mathbf{B}(s)^T & \mathbf{0} \\ \mathbf{0} & \mathbf{B}(s)^T \end{pmatrix}.$$

$\mathcal{U}$ is the metric matrix for B-spline curves (Blake et al., 1993a) and efficient methods for computing $\mathcal{U}$ are given in (Blake and Isard, 1998). The eigenvectors are taken in descending order of their (necessarily positive) eigenvalues. The intuitive interpretation is that the eigenvalues represent variance in the training set in the mutually orthogonal directions of the eigenvectors; the first N'_X eigenvectors form a basis for the subspace of dimension N'_X that "explains" as much as possible of the variance in the training set.

Results from the application of PCA to a lip-motion sequence are shown in figure 2.5. In this case the individual PCA components are not recognisable as particular expressions;

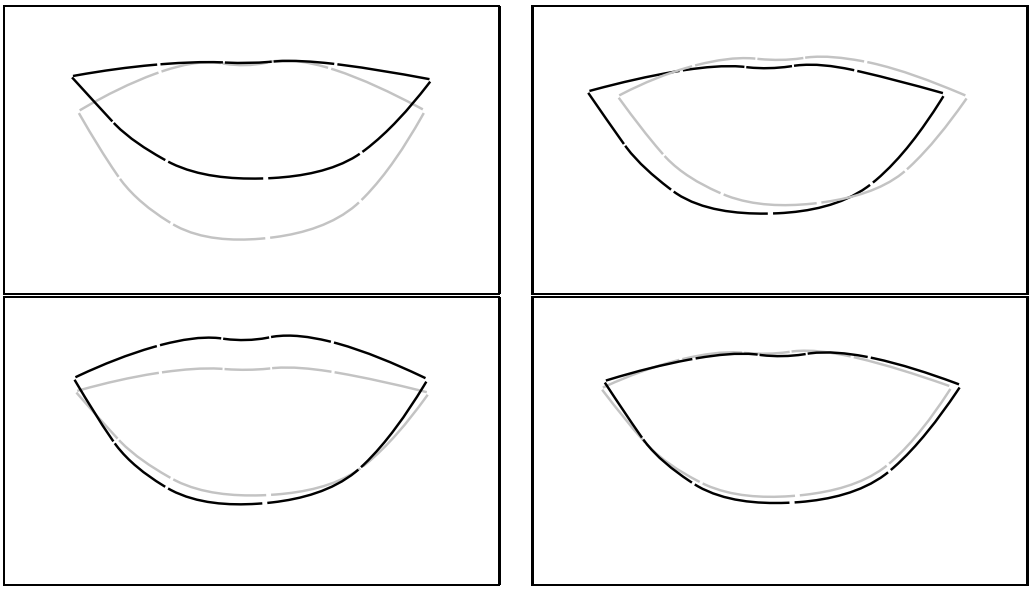


Figure 2.5: *A sequence of tracked lip-motion outlines is used here in PCA. The first four eigenvectors, illustrated here, capture over 95% of the variance in the data set. (Eigenvectors are displayed here as displacements either side of the mean, with magnitude equal to their standard deviation over the sequence.) (Figures by courtesy of Robert Kaucic.)*

rather they are mixtures of expressions. It is when they are taken as a set that they are meaningful, as a basis for the repertoire of commonly occurring deformations.

2.4 Image processing

The original implementations of active contours, or "snakes" (Kass et al., 1987), performed image processing on the entire image, and used the resulting edge-map as an energy surface across which the contour moved. For efficiency, the deformable templates described in this chapter are driven towards a distinguished feature curve $\mathbf{r}_f(s)$ rather than over the entire image landscape F that is used in the snake model. One way of doing this is to mark high strength values on the feature maps and group them to form point sets to which spline curves could be fitted. However, the wholesale application of filters across entire images is excessively computationally costly. At any given instant, an estimate is available of the position of a tracked image-contour and this can be used to define a "search-region," in which the corresponding image feature is likely to lie. Image processing can then effectively be restricted to this search region, as in figure 2.6. The search region displayed in the figure

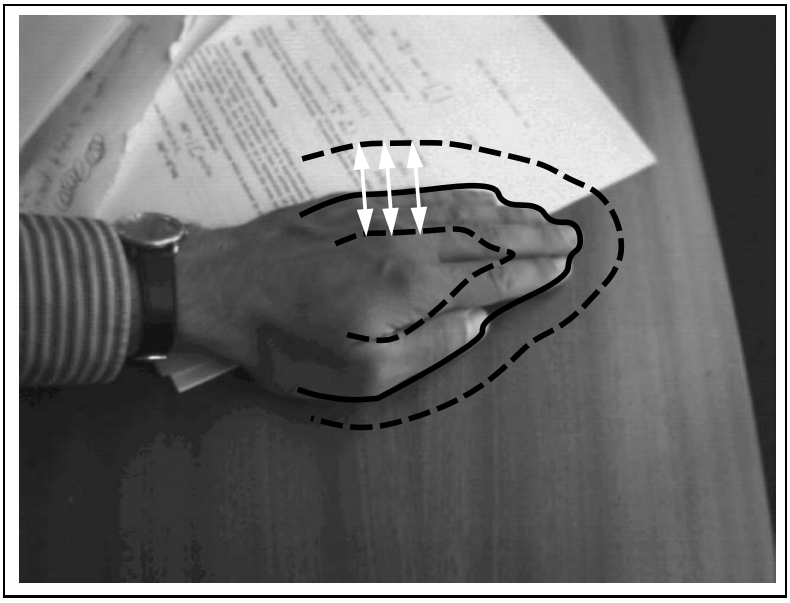


Figure 2.6: **Search region.** *It is computationally efficient to restrict image processing operations to lie within a "region of interest" (dashed lines) either side of the currently estimated contour position (solid line). Image processing operations are then performed along certain lines passing through the estimated contour. In this example, the lines are normals to the estimated curve, three of which are shown as arrowed white lines.*

is formed there by sweeping normal vectors of a chosen length along the entire contour. Features can then be detected by performing image filtering along each of the sampled normals, and this is very efficient. If normals are constructed at points $s = s_i,\ i = 1, \ldots, N$, along the curve $\mathbf{r}(s)$, this will give a sequence of sampled points $\mathbf{r}_f(s_i), i = 1, \ldots, N$ along the feature curve $\mathbf{r}_f(s)$. The s_i can either be spaced evenly around the curve or concentrated in regions where measurements are expected to be particularly informative. In general, more than one feature will be found on each normal, particularly when tracking in clutter. In the Kalman filtering framework described in this chapter, heuristics are used to choose one "favourite" feature, for example using the strongest feature response, possibly weighted towards the predicted feature position. In later chapters all detected features are used as measurements by the CONDENSATION algorithm and this is described in section 3.6 on page 46.

2.4.1 Linear scanning

In order to perform one-dimensional image processing, image intensity is sampled at regularly spaced intervals along each image normal. An arbitrarily placed normal line generally

intersects image pixels in an irregular fashion, as in figure 2.7. This generates undesirable

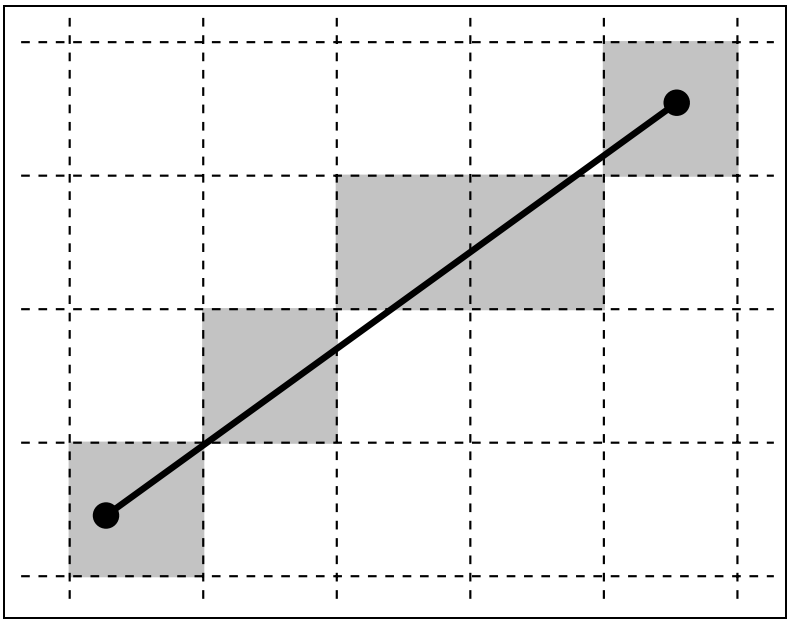


Figure 2.7: **Irregular image sampling**. *Listing the intensities of pixels crossed by a normal line would result in a non-uniform sampling of intensity that would suffer abrupt variations as the line moved over the image.*

artifacts, and an effective sampling scheme, spatially regular and temporally smooth (when the line moves) involves interpolation as follows. A sequence of regularly spaced sample points are chosen along the line. The intensity I at a particular sample point (x, y) is computed as a weighted sum of the intensities at 4 neighbouring pixels, as in figure 2.8. A pixel with centre sited at integer coordinates (i, j) has intensity $I_{i,j}$. The intensity I at (x, y) is then computed by bilinear interpolation:

$$I = \sum_{i,j} w_{i,j} I_{i,j} \tag{2.8}$$

with weights

$$w_{i,j} = \begin{cases} (1 - |x - i|)(1 - |y - j|) & \text{if } |x - i| < 1 \text{ and } |y - j| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

so that at most four pixels, the ones whose centres are closest to (x, y), have non-zero weights, as the figure depicts.

2.4.2 Image filtering

Analysis of image intensities now concentrates on the one-dimensional signals along normals. The intensity $I(x)$ along a particular normal is sampled regularly at $x = x_i$ and intensities

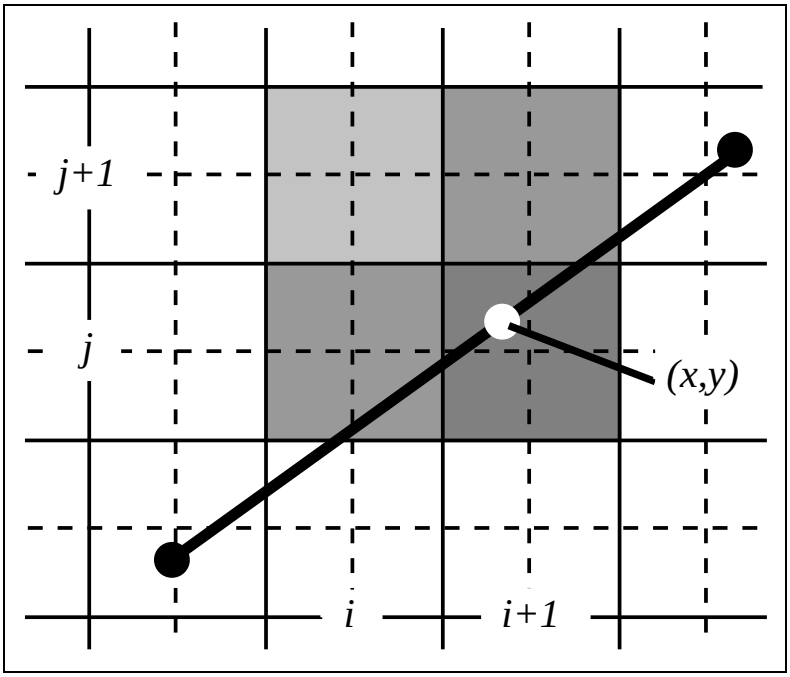


Figure 2.8: **Interpolated image sampling**. *The intensity at a chosen sampling point* (x, y) *is computed as a weighted sum of the intensities at the four immediately adjacent pixels.*

are stored in an array $I_i = I(x_i),\ i = 1, \ldots, N$. A variety of feature detection operators can be applied to the line, typically edges, valleys or ridges. Features are located by applying an appropriate operator or mask $C_n,\ -N_C \leq n \leq N_C$, by discrete convolution, to the sampled intensity signal $I_n,\ 1 \leq n \leq N_I$, to give a feature-strength signal

$$E_n = \sum_{m=-N_C}^{N_C} C_m I_{n+m}.$$

Maxima of that signal are then located, and marked wherever the value at that maximum exceeds a preset threshold (chosen to exclude spurious, noise-generated maxima). This is illustrated for edges in figure 2.9.

More sophisticated feature detection schemes can also be applied to normal lines, and these include colour-based edge-detection and template matching, whereby typical greyscale profiles for foreground and background are learned. At each position on the normal line a statistical likelihood can then be calculated that the feature point lies at that location. In addition, when the camera is stationary it may be possible to employ background subtraction to reject spurious features which do not correspond to the foreground object. These techniques are discussed more fully in (Blake and Isard, 1998).

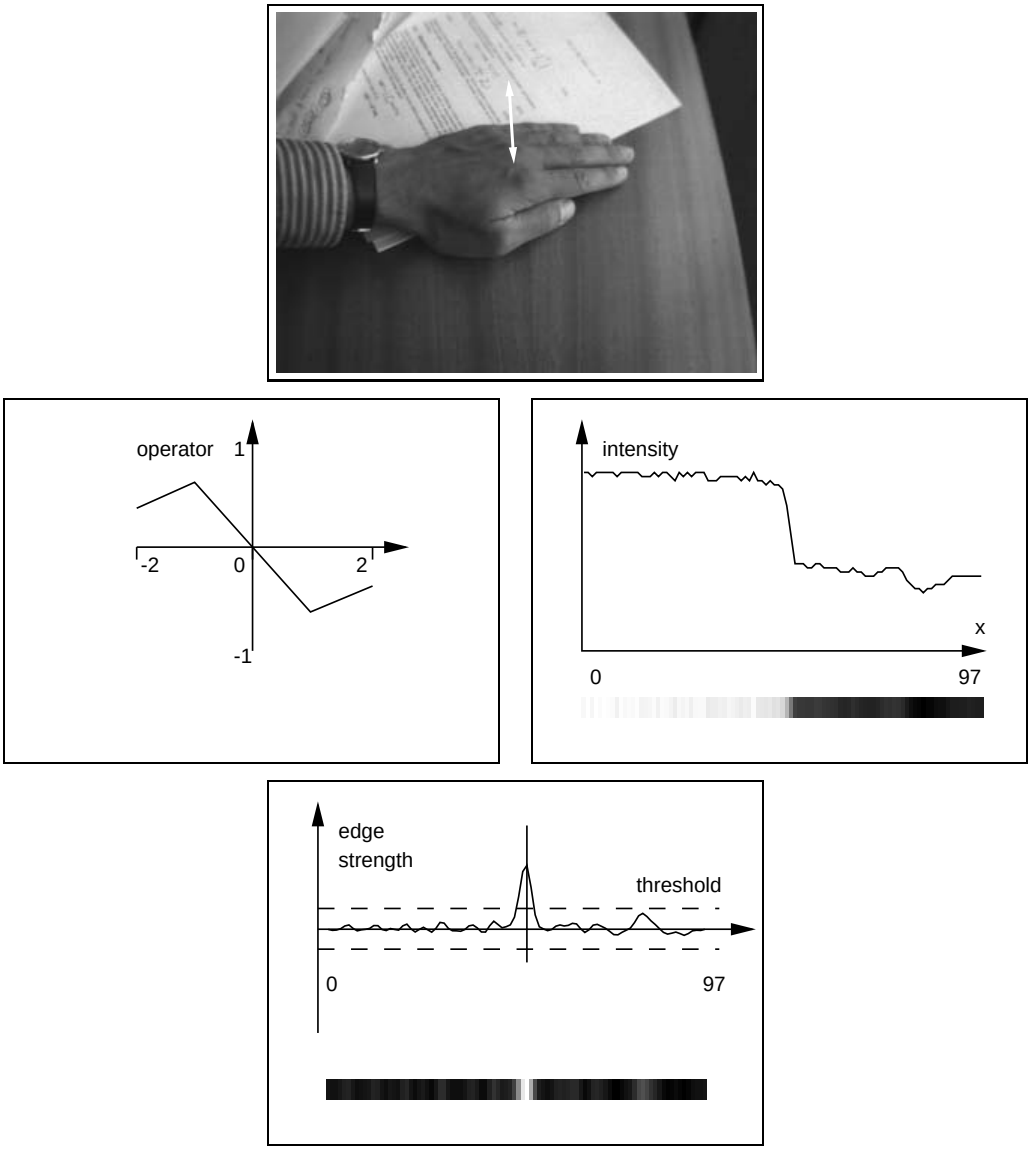


Figure 2.9: **Operator for edge detection** *The problem is to search along a line in an image (top) to find edges — locations where contrast is high. An operator (left, shown on an expanded length scale) is convolved with the image intensity function along the line (right). One edge is found, corresponding to a maximum of the feature-strength function (bottom).*

2.5 Dynamical model

The active contour framework specifies not only B-spline based shape models, but also dynamical models characterising an object's behaviour over time. To simplify tracking using a Kalman filter, the dynamical models described in (Blake et al., 1993a; Blake et al., 1995; Blake and Isard, 1998) are auto-regressive processes (ARPs), and the experiments described in chapter 4 use these models directly. As is true for shape models, the CONDENSATION algorithm permits the use of a general class of non-linear, non-Gaussian motion models, and chapter 5 introduces an extension to allow multiple ARP models with discrete switching between them. The basic ARP framework is described in this section.

Object dynamics are modelled as a 2nd order process, represented in discrete time t as a second-order ARP:

$$\mathbf{x}_t = A_2\mathbf{x}_{t-2} + A_1\mathbf{x}_{t-1} + \mathbf{D}_0 + B_0\mathbf{w}_t \tag{2.10}$$

where $\mathbf{w}_t$ are independent vectors of independent standard normal variables, $\mathbf{D}_0$ is a fixed offset, and A_2, A_1 and B_0 are matrices representing the deterministic and stochastic components of the dynamical model. For notational simplicity an augmented state-vector $\mathbf{X}_t$ can be used:

$$\mathbf{X}_t = \begin{pmatrix} \mathbf{x}_{t-1} \\ \mathbf{x}_t \end{pmatrix}, \tag{2.11}$$

and then

$$\mathbf{X}_t = A\mathbf{X}_{t-1} + \mathbf{D} + B\mathbf{w}_t \tag{2.12}$$

where

$$A = \begin{pmatrix} 0 & I \\ A_2 & A_1 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} 0 \\ \mathbf{D}_0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0 \\ B_0 \end{pmatrix}.$$

This notation also emphasises the Markov nature of the dynamical model, that $\mathbf{X}_t$ depends only on the state at the previous timestep $\mathbf{X}_{t-1}$.

In the case that the inverse of $I - A$ is defined, $\mathbf{D}$ can be interpreted in terms of a mean value $\overline{\mathbf{X}}$ of the motion

$$\overline{\mathbf{X}} = (I - A)^{-1}\mathbf{D} \tag{2.13}$$

so that (2.12) can be rewritten

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

The system is a set of damped oscillators, whose modes, natural frequencies and damping constants are determined by A, driven by random accelerations coupled into the dynamics via B from the noise term $B\mathbf{w}$. Default values for A, and B can be set by hand by specifying oscillator parameters consisting of a damping constant β, a natural frequency f and a root-mean-square average displacement ρ (Blake and Isard, 1998). These parameters can be used to determine the (zero-mean) ARP model in one dimension

$$x_t = a_2 x_{t-2} + a_1 x_{t-1} + b\omega_t$$

where ω_t is Gaussian noise drawn from $N(0,1)$, a_1, a_2 and b are given by

$$a_2 = -\exp(-2\beta\tau), \quad a_1 = 2\exp(-\beta\tau)\cos(2\pi f\tau)$$
$$b = \rho\sqrt{1 - a_2^2 - a_1^2 - 2\frac{a_2 a_1^2}{1 - a_2}}$$

and τ is the time-step length in seconds. It is straightforward to generalise this to multi-dimensional oscillators, and the shape-space can be partitioned to provide different behaviour for e.g. translation and deformation (Blake and Isard, 1998). Often it is more satisfactory and effective to estimate model parameters from input data taken while the object performs typical motions and this learning is described in the next section.

2.5.1 Learning a dynamical model

Initially, a hand-built model is used in a tracker to follow a training sequence which must be not too hard to track. This can be achieved by allowing only motions which are not too fast, and limiting background clutter. Once a new dynamical model has been learned, it can be used to build a more competent tracker, one that is specifically tuned to the sort of motions it is expected to encounter. That can be used either to track the original training sequence more accurately, or to track a new and more demanding training sequence, involving greater agility of motion. The cycle of learning and tracking is described in figure 2.10. Typically two or three cycles suffice to learn an effective dynamical model, and in practice a learned shape model can also be refined during the process by gathering new training outlines if there are some object poses in the training sequence which are poorly represented in the original shape model.

The problem is to estimate the coefficients A_1, A_2, $\mathbf{D}_0$ and B_0 which best model the motion in a training sequence of shapes $\mathbf{x}_1, \ldots, \mathbf{x}_M$, gathered at the image sampling frequency. A general algorithm to do this is described below.

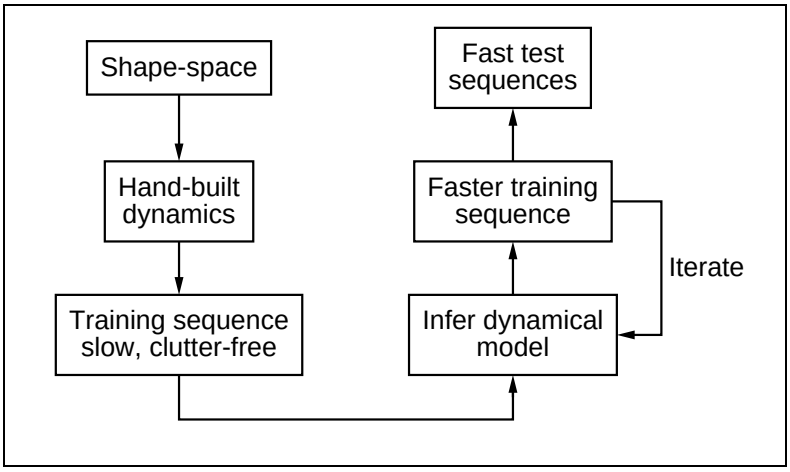


Figure 2.10: **Iterative learning of dynamics.** *The model acquired in one cycle of learning is installed in a tracker to interpret the training sequence for the next cycle. The process is initialised with hand-specified dynamics.*

The problem is expressed in terms of a "log-likelihood" function, defined up to an additive constant by

$$L(\mathbf{x}_1,\ldots,\mathbf{x}_M|A_1,A_2,B_0,\mathbf{D}_0) \equiv \log p(\mathbf{x}_1,\ldots,\mathbf{x}_M|A_1,A_2,B_0,\mathbf{D}_0) \quad + \quad \text{const}$$

where, since the w_k are independent,

$$p(\mathbf{x}_1,\ldots,\mathbf{x}_k|A_1,A_2,B_0,\mathbf{D}_0) \propto \prod_k p_{B_0\mathbf{w}_k}(\mathbf{x}_k - A_2\mathbf{x}_{k-2} - A_1\mathbf{x}_{k-1} - \mathbf{D}_0)$$

so, using the fact that the $p_{B_0\mathbf{w}_k}(\cdot)$ are standard multivariate normal distributions,

$$\begin{aligned} &L(\mathbf{x}_1\ldots\mathbf{x}_M|A_1,A_2,B_0,\mathbf{D}_0) = \\ &\quad -\frac{1}{2}\sum_{k=3}^{M}\left|B_0^{-1}\left(\mathbf{x}_k - A_2\mathbf{x}_{k-2} - A_1\mathbf{x} - \mathbf{D}_0\right)\right|^2 - (M-2)\log\det B_0. \end{aligned} \tag{2.14}$$

Minimising the log-likelihood L leads to the learning algorithm of figure 2.11 which estimates the dynamical parameters A_2, A_1, $\mathbf{D}_0$ and B (Blake et al., 1995; Wildenberg, 1997; Blake and Isard, 1998). Note that the learning algorithm as presented treats estimated shape-vectors $\mathbf{x}$ in a training sequence as if they were exact observations of the physical process, rather than noisy estimates obtained from a visual tracker. In practice this often works quite well but can give incorrect results with highly periodic training motions, and instead dynamics can be learned directly from the observations using expectation-maximisation (EM) (North and Blake, 1998).

Dynamical learning problem

Given a training set $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ of shapes from an image sequence, learn the parameters A_1, A_2, B_0 and $\mathbf{D}_0$ for a second-order AR process that describes the dynamics of the moving shape.

Algorithm

1. First, sums $\mathbf{R}_i$, $i = 0, 1, 2$ and auto-correlation coefficients R_{ij} and R'_{ij}, $i, j = 0, 1, 2$ are computed:

$$\mathbf{R}_i = \sum_{k=3}^{M} \mathbf{x}_{k-i}, \quad R_{ij} = \sum_{k=3}^{M} \mathbf{x}_{k-i}\mathbf{x}_{k-j}^T, \quad R'_{ij} = R_{ij} - \frac{1}{M-2} R_i R_j^T.$$

2. Estimated parameters $\hat{A}_1$, $\hat{A}_2$ and $\hat{\mathbf{D}}_0$ are given by

$$\begin{aligned} \hat{A}_2 &= \left(R'_{02} - R'_{01}R'^{-1}_{11}R'_{12}\right)\left(R'_{22} - R'_{21}R'^{-1}_{11}R'_{12}\right)^{-1} \\ \hat{A}_1 &= \left(R'_{01} - \hat{A}_2 R'_{21}\right) R'^{-1}_{11} \\ \hat{\mathbf{D}}_0 &= \frac{1}{M-2}\left(\mathbf{R}_0 - \hat{A}_2\mathbf{R}_2 - \hat{A}_1\mathbf{R}_1\right). \end{aligned}$$

3. The covariance coefficient B_0 is estimated as a matrix square root $\hat{B}_0 = \sqrt{\hat{C}}$ where

$$\hat{C} = \frac{1}{M-2}\left(R_{00} - \hat{A}_2 R_{20} - \hat{A}_1 R_{10} - \hat{\mathbf{D}}_0 \mathbf{R}_0^T\right).$$

Figure 2.11: **Algorithm for learning multi-variate dynamics.**

3

The CONDENSATION algorithm

This chapter sets out the basic framework used to represent and propagate conditional densities. It also describes particular models used in initial experiments to test the performance of the CONDENSATION algorithm. Later chapters extend these models to permit analysis of more complex motions, and consider modifications to the basic algorithm which improve its efficiency.

3.1 Temporal propagation of conditional densities

The Kalman filter as a recursive linear estimator is a special case, applying only to Gaussian densities, of a more general probability density propagation process. In continuous time this can be described in terms of diffusion, governed by a "Fokker-Planck" equation (Astrom, 1970), in which the density for $\mathbf{X}_t$ drifts and spreads under the action of a stochastic model of its dynamics. In the simple Gaussian case, the diffusion is purely linear and the density function evolves as a Gaussian pulse that translates, spreads and is reinforced, remaining Gaussian throughout, as in figure 3.1, a process that is described analytically and exactly by the Kalman filter. The random component of the dynamical model leads to spreading — increasing uncertainty — while the deterministic component causes the density function to drift bodily. The effect of an external observation $\mathbf{z}_t$ is to superimpose a reactive effect on the diffusion in which the density tends to peak in the vicinity of observations. In clutter, there are typically several competing observations and these tend to encourage a non-Gaussian state-density (figure 3.2).

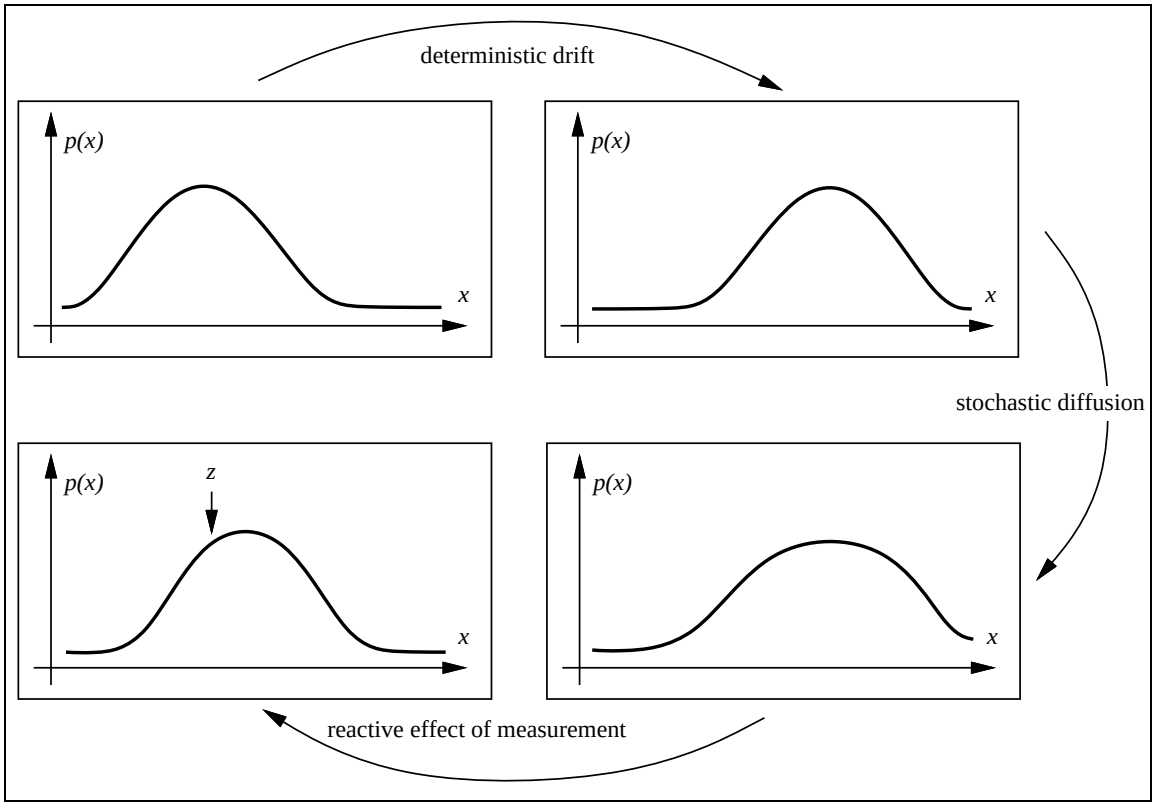


Figure 3.1: **Kalman filter as density propagation.** *In the case of Gaussian prior, process and observation densities, and assuming linear dynamics, the propagation process of figure 3.2 reduces to a diffusing Gaussian state density, represented completely by its evolving (multivariate) mean and variance — precisely what a Kalman filter computes.*

The CONDENSATION algorithm is designed to address this more general situation. It has the striking property that, generality notwithstanding, it is a considerably simpler algorithm than the Kalman filter. Moreover, despite its use of random sampling which is often thought to be computationally inefficient, the CONDENSATION algorithm runs in or near real-time. This is because tracking over time maintains relatively tight distributions for shape at successive time-steps, and particularly so given the availability of accurate, learned models of shape and motion.

3.2 Discrete-time propagation of state density

For computational purposes, the propagation process described in section 3.1 must be set out in terms of discrete time t. The state of the modelled object at time t is denoted $\mathbf{X}_t$ and its history is $\mathcal{X}_t = \{\mathbf{X}_1, \ldots, \mathbf{X}_t\}$. Similarly the set of image features at time t is $\mathbf{Z}_t$ with

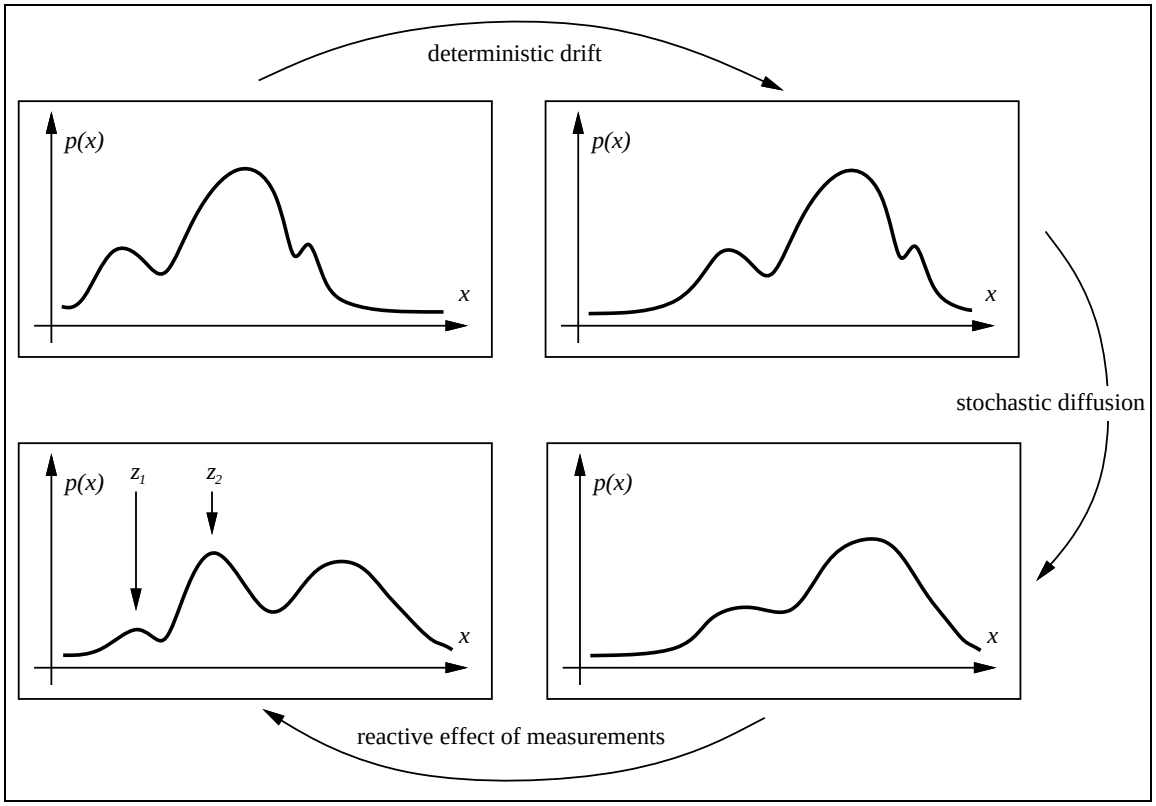


Figure 3.2: **Probability density propagation.** *Propagation is depicted here as it occurs over a discrete time-step. There are three phases: drift due to the deterministic component of object dynamics; diffusion due to the random component; reactive reinforcement due to observations.*

history $\mathcal{Z}_t = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_t\}$. Note that no functional assumptions (linearity, Gaussianity, unimodality) are made about densities in the general treatment, though particular choices will be made in due course in order to demonstrate the approach.

3.2.1 Stochastic dynamics

A somewhat general assumption is made for the probabilistic framework that the object dynamics form a temporal Markov chain so that

$$p(\mathbf{X}_t|\mathcal{X}_{t-1}) = p(\mathbf{X}_t|\mathbf{X}_{t-1}) \tag{3.1}$$

— the new state is conditioned directly only on the immediately preceding state, independent of the earlier history. This still allows quite general dynamics, including stochastic difference equations of arbitrary order; we typically use second order models as described in section 2.5 on page 28. The dynamics are entirely determined therefore by the form of

the conditional density $p(\mathbf{X}_t|\mathbf{X}_{t-1})$. For instance,

$$p(x_t|x_{t-1}) \propto \exp -\frac{1}{2}(x_t - x_{t-1} - 1)^2$$

represents a one-dimensional random walk (discrete diffusion) whose step length is a standard normal variate, superimposed on a rightward drift at unit speed. Of course, for realistic problems, the state $\mathbf{X}$ is multi-dimensional and the density is more complex (and, in the applications presented later, learned from training sequences).

3.2.2 Measurement

Observations $\mathbf{Z}_t$ are assumed to be independent, both mutually and with respect to the dynamical process. This is expressed probabilistically as follows:

$$p(\mathcal{Z}_{t-1}, \mathbf{X}_t|\mathcal{X}_{t-1}) = p(\mathbf{X}_t|\mathcal{X}_{t-1}) \prod_{i=1}^{t-1} p(\mathbf{Z}_i|\mathbf{X}_i). \tag{3.2}$$

Note that integrating over $\mathbf{X}_t$ implies the mutual independence of observations conditional on the $\mathbf{X}_i$:

$$p(\mathcal{Z}_t|\mathcal{X}_t) = \prod_{i=1}^{t} p(\mathbf{Z}_i|\mathbf{X}_i). \tag{3.3}$$

The observation process is therefore defined by specifying the conditional density $p(\mathbf{Z}_t|\mathbf{X}_t)$ at each time t, and later, in computational examples, we take this to be a time-independent function $p(\mathbf{Z}|\mathbf{X})$. Suffice it to say for now that, in clutter, the observation density is multi-modal. Details will be given in section 3.6.

3.2.3 Propagation

Given a continuous-valued Markov chain with independent observations, the conditional state-density at time t is defined by $p(\mathbf{X}_t|\mathcal{Z}_t)$. This represents all information about the state at time t that is deducible from the entire data-stream up to that time. The rule for propagation of state density over time is:

$$p(\mathbf{X}_t|\mathcal{Z}_t) = k_t \; p(\mathbf{Z}_t|\mathbf{X}_t) p(\mathbf{X}_t|\mathcal{Z}_{t-1}), \tag{3.4}$$

where

$$p(\mathbf{X}_t|\mathcal{Z}_{t-1}) = \int_{\mathbf{X}_{t-1}} p(\mathbf{X}_t|\mathbf{X}_{t-1}) p(\mathbf{X}_{t-1}|\mathcal{Z}_{t-1}) \; d\mathbf{X}_{t-1} \tag{3.5}$$

and k_t is a normalisation constant that does not depend on $\mathbf{X}_t$. The validity of the rule is proved in appendix A.1 on page 144.

The propagation rule (3.4) should be interpreted simply as the equivalent of the Bayes' rule (3.6) for inferring posterior state density from data, for the time-varying case. The effective prior $p(\mathbf{X}_t|\mathcal{Z}_{t-1})$ is actually a prediction taken from the posterior $p(\mathbf{X}_{t-1}|\mathcal{Z}_{t-1})$ from the previous time-step, onto which is superimposed one time-step from the dynamical model (Fokker-Planck drift plus diffusion as in figure 3.2), which is expressed in (3.5). Multiplication in (3.4) by the observation density $p(\mathbf{Z}_t|\mathbf{X}_t)$ in the Bayesian manner then applies the reactive effect expected from observations. When the observation density is non-Gaussian, the evolving state density $p(\mathbf{X}_t|\mathcal{Z}_t)$ is also generally non-Gaussian (although experiments in this chapter use a Gaussian process density, chapter 5 introduces a non-Gaussian form for the process as well). The problem now is how to apply a *nonlinear filter* to evaluate the state density over time, without incurring excessive computational load. Inevitably this means approximating. A survey of the non-linear filtering literature was given in section 1.3 and the following section treats Kalman-filter based solutions in more detail.

3.2.4 Non-linear extensions to Kalman filtering

There are four distinct probability distributions represented in a non-linear Bayesian filter. Three of them form part of the problem specification and the fourth constitutes the solution. The three specified distributions are:

1. the prior density $p(\mathbf{X})$ for the state $\mathbf{X}$
2. the process density $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ that describes the stochastic dynamics
3. the observation density $p(\mathbf{Z}_t|\mathbf{X}_t)$

and the filter evolves over time to generate, as the solution at each time-step, the state-density $p(\mathbf{X}_t|\mathbf{Z}_1,\ldots,\mathbf{Z}_t)$. Only when all of the three specified distributions are Gaussian is the state-density also Gaussian. Otherwise, for non-Gaussian $p(\mathbf{X}_t|\mathbf{Z}_1,\ldots,\mathbf{Z}_t)$, it is possible to use one of a number of approximate filters, depending on which of the specified densities it is that is non-Gaussian.

Non-Gaussian prior density

The case that the prior density is non-Gaussian is the simplest to deal with provided only that it can adequately be represented (or approximated) as an additive Gaussian mixture:

$$p_0(\mathbf{X}) = \sum_{m=1}^{M} w^{(m)} G(\mathbf{X}; \mu^{(m)}, P_0^{(m)}).$$

In that case, provided that other specified densities are Gaussian, the state density can also be represented as a corresponding mixture

$$p(\mathbf{X}_t|\mathcal{Z}_t) = \sum_{m=1}^{M} w^{(m)} G(\mathbf{X}_t; \mu_t^{(m)}, P_t^{(m)})$$

in which the means $\mu_t^{(m)}$ and variances $P_t^{(m)}$ vary over time but the weights $w^{(m)}$ are fixed. Each of the M mixture components evolves as an independent Gaussian so that, in fact, the state density is just a sum of densities from M independent linear Kalman filters.

Non-Gaussian process density

Non-Gaussian state densities can arise from the nature of the process either because the dynamics are driven by non-Gaussian process noise, or, more generally, because the deterministic dynamics are non-linear. One approach to filtering is then to approximate the dynamics by Taylor expansion as a linear process with time-varying coefficients and proceed as for linear Kalman filters. This generates a Gaussian representation of the evolving state-density which may be a good approximation depending on the nature of the non-linearity. This is the basis of the "Extended Kalman Filter" (EKF) (Gelb, 1974; Bar-Shalom and Fortmann, 1988). Alternatively, one can attempt a mixture representation, as earlier, but now allowing the weights $w^{(m)}$ also to vary over time. Unfortunately, even allowing dynamic re-weighting (Sorenson and Alspach, 1971) does not produce exact solutions for $p(\mathbf{X}_t|\mathcal{Z}_t)$, because the individual Gaussian components do not remain Gaussian over time. For example, consider the case in which the process density $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ is itself an additive mixture of $k > 1$ Gaussian components. According to the Bayesian propagation equation (3.5) each component of $p(\mathbf{X}_t|\mathcal{Z}_t)$ splits into k separate components in the transition from time n to time $n+1$; the total number of components in $p(\mathbf{X}_t|\mathcal{Z}_t)$ grows exponentially as k^t. Clearly $p(\mathbf{X}_t|\mathcal{Z}_t)$ must be approximated at each time-step to prune back the number of components (Anderson and Moore, 1979) within some resource-limited bound M. Effectively there are Mk full Kalman filters running at each time-step, each bringing the computational expense

of a Riccati equation step to update its covariance estimate. Clearly the success of this approach depends on how well the densities $p(\mathbf{X}_t|\mathcal{Z}_t)$ and $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ can be approximated with a modest number Mk of components.

Non-Gaussian observation density

In the case of visual tracking in clutter, non-linearity of the tracking filter arises because the observation density $p(\mathbf{Z}_t|\mathbf{X}_t)$ is non-Gaussian and, furthermore, is multi-modal so that it cannot be well approximated by a single Gaussian. Each of the methods just mentioned for handling non-Gaussian process density, the EKF and Gaussian mixtures, are relevant also to non-Gaussian observation density but continue to have the same drawbacks. Note that, in the case of Gaussian mixtures, the number of mixture components again proliferates at each time-step of (3.4), albeit via a different mechanism involving products of Gaussians rather than convolutions. Even this assumes that the observation density can be approximated as a mixture but in clutter this becomes rather inefficient, requiring at least one component per visible feature.

There is an additional class of techniques which applies to this case when the non-Gaussian state density arises from clutter of a particular sort. In the simplest case, one of a finite set of measurements $\mathbf{Z}_t = \{z_{t,1}, \ldots, z_{t,k}\}$ at time t is to be associated with the state $\mathbf{X}_t$ at time t, while the remaining $k-1$ measurements are to be regarded as clutter. Heuristic mechanisms such as the validation gate and the probabilistic data-association filter (PDAF) (Bar-Shalom and Fortmann, 1988) attempt to deal with the ambiguity of association. Alternatively it can, in principle, be dealt with exactly by "multiple hypothesis filtering" but with computational cost that grows exponentially over time and which is therefore ruled out in practice (pruning can be used to reduce computational cost, although the filter is then no longer exact. This approach is equivalent to using a Gaussian mixture model as mentioned above). The "RANSAC" algorithm (Fischler and Bolles, 1981) deals probabilistically with multiple observations but the observations have to be discrete, and there is no mechanism for temporal propagation. More complex methods including the Joint PDAF (JPDAF) (Bar-Shalom and Fortmann, 1988; Rao, 1992) address the more difficult problem of associating not simply single features but subsequences of $\mathcal{Z}_t$ with the state. However, these methods rely on the existence of discrete features. In contour tracking the features are continuous curves and so are not naturally amenable to discrete association.

3.3 Factored sampling

This section describes first the factored sampling algorithm dealing with non-Gaussian observations in single images. Then factored sampling is extended in the following section to deal with temporal image sequences.

As discussed in section 1.4 on page 9, given a single static image $\mathbf{Z}$ it is a standard problem in the statistical literature to find an object parameterised as $\mathbf{X}$ with prior $p(\mathbf{X})$. The posterior density $p(\mathbf{X}|\mathbf{Z})$ represents all the knowledge about $\mathbf{X}$ that is deducible from the data. It can be evaluated in principle by applying Bayes' rule (Papoulis, 1990) to obtain

$$p(\mathbf{X}|\mathbf{Z}) = kp(\mathbf{Z}|\mathbf{X})p(\mathbf{X}) \tag{3.6}$$

where k is a normalisation constant that is independent of $\mathbf{X}$. In cases where $p(\mathbf{Z}|\mathbf{X})$ is sufficiently complex that $p(\mathbf{X}|\mathbf{Z})$ cannot be evaluated simply in closed form, iterative sampling techniques can be used (Geman and Geman, 1984; Ripley and Sutherland, 1990; Grenander et al., 1991; Storvik, 1994). The factored sampling algorithm (Grenander et al., 1991) generates a random variate $\mathbf{X}'$ from a distribution $\tilde{p}(\mathbf{X})$ that approximates the posterior $p(\mathbf{X}|\mathbf{Z})$. First a sample-set $\{\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(N)}\}$ is generated from the prior density $p(\mathbf{X})$ and then each index $i \in \{1, \ldots, N\}$ is assigned with probability π_i, where

$$\pi_i = \frac{p_z(\mathbf{s}^{(i)})}{\sum_{j=1}^{N} p_z(\mathbf{s}^{(j)})}$$

and

$$p_z(\mathbf{X}) = p(\mathbf{Z}|\mathbf{X}),$$

the conditional observation density. The index i assigned by this procedure determines the value $\mathbf{X}' = \mathbf{X}_i$, and $\mathbf{X}'$ chosen in this fashion has a distribution which approximates the posterior $p(\mathbf{X}|\mathbf{Z})$ increasingly accurately as N increases (figure 3.3).

Note that posterior mean properties $\mathcal{E}[g(\mathbf{X})|\mathbf{Z}]$ can be generated directly from the samples $\{\mathbf{s}^{(n)}\}$ by weighting with $p_z(\mathbf{X})$ to give:

$$\mathcal{E}[g(\mathbf{X})|\mathbf{Z}] \approx \frac{\sum_{n=1}^{N} g(\mathbf{s}^{(n)}) p_z(\mathbf{s}^{(n)})}{\sum_{n=1}^{N} p_z(\mathbf{s}^{(n)})}. \tag{3.7}$$

For example, the mean $\overline{\mathbf{X}}$ can be estimated using $g(\mathbf{X}) = \mathbf{X}$ (illustrated in figure 3.4) and the variance using $g(\mathbf{X}) = \mathbf{X}\mathbf{X}^T - \overline{\mathbf{X}}^2$. In the case that $p(\mathbf{X})$ is a spatial Gauss-Markov process, Gibbs sampling from $p(\mathbf{X})$ has been used to generate the random variates

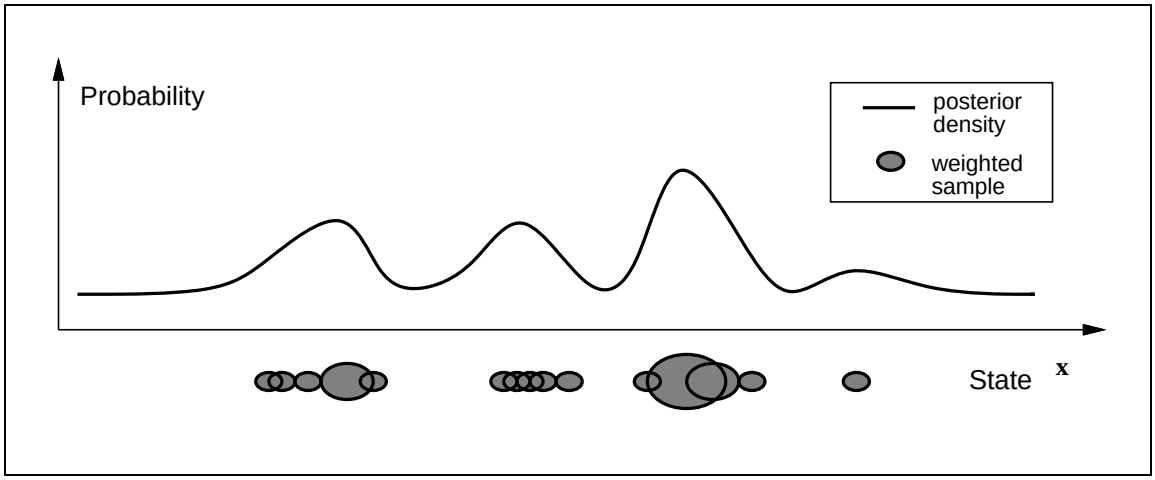


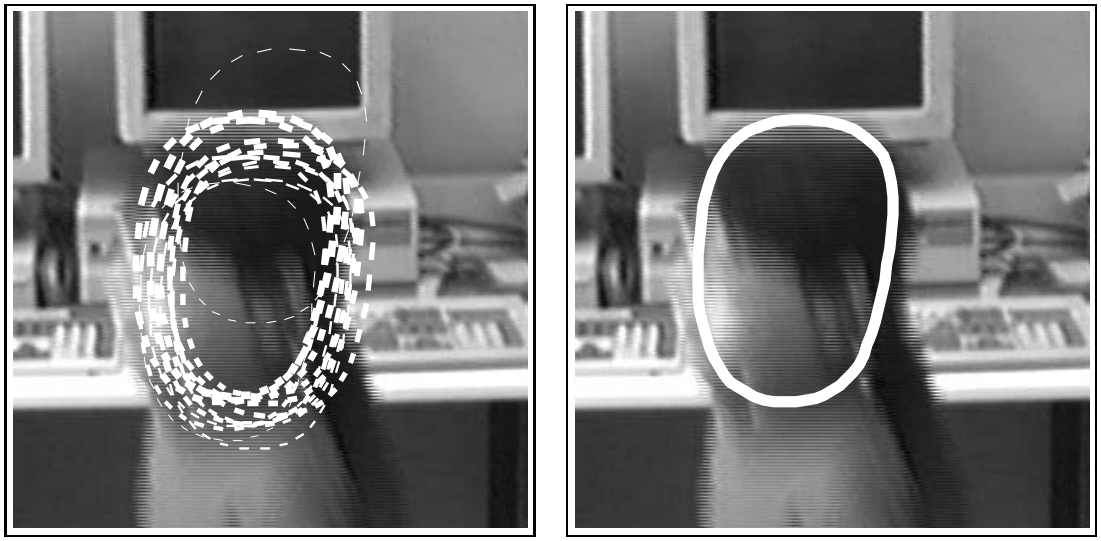
Figure 3.3: **Factored sampling.** *A set of points* $\mathbf{s}^{(i)}$*, the centres of the blobs in the figure, is sampled randomly from a prior density* $p(\mathbf{X})$*. Each sample is assigned a weight* π_i *(depicted by blob area) in proportion to the value of the observation density* $p(\mathbf{Z}|\mathbf{X} = \mathbf{s}^{(i)})$*. The weighted point-set then serves as a representation of the posterior density* $p(\mathbf{X}|\mathbf{Z})$*, suitable for sampling. The one-dimensional case illustrated here extends naturally to the practical case that the density is defined over several position and shape variables.*

$\{\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(N)}\}$. Otherwise, for low-dimensional parameterisations as in this thesis, standard, direct methods can be used for Gaussians[1] (Press et al., 1988). Note that, in the case that the density $p(\mathbf{Z}|\mathbf{X})$ is normal, the mean obtained by factored sampling is consistent with an estimate obtained more conventionally, and efficiently, from linear least squares estimation. For multi-modal distributions which cannot be approximated as normal, so that linear estimators are unusable, estimates of mean $\mathbf{X}$ by factored sampling continue to apply.

3.4 The CONDENSATION algorithm

The CONDENSATION algorithm is based on factored sampling but extended to apply iteratively to successive images in a sequence. The same sampling strategy has been developed elsewhere (Gordon et al., 1993; Kitagawa, 1996), presented as developments of Monte-Carlo methods, and various adaptations of the basic algorithm have appeared in the statistical literature (Gordon and Salmond, 1995; Carpenter et al., 1997; Pitt and Shepherd, 1997; Doucet, 1998) — these will be discussed in section 8.3.

[1] Note: the presence of clutter causes $p(\mathbf{Z}|\mathbf{X})$ to be non-Gaussian, but the prior $p(\mathbf{X})$ may still happily be Gaussian or a Gaussian mixture, and that is what will be assumed in our experiments.



(a) (b)

Figure 3.4: **Sample-set representation of shape distributions** *The sample-set representation of probability distributions, illustrated in one dimension in figure 3.3, is illustrated here (a) as it applies to the distribution of a multi-dimensional curve parameter* $\mathbf{x}$*. Each sample* $\mathbf{s}^{(n)}$ *is shown as a curve (of varying position and shape) with a thickness proportional to the weight* π_n*. The weighted mean of the sample set (b) serves as an estimator of the distribution mean.*

Given that the process at each time-step is a self-contained iteration of factored sampling, the output of an iteration will be a weighted, time-stamped sample-set, denoted $\{\mathbf{s}_t^{(n)},\ n = 1, \ldots, N\}$ with weights $\pi_t^{(n)}$, representing approximately the conditional state-density $p(\mathbf{X}_t|\mathcal{Z}_t)$ at time t. How is this sample-set obtained? Clearly the process must begin with a prior density and the effective prior for time-step t should be $p(\mathbf{X}_t|\mathcal{Z}_{t-1})$. This prior is of course multi-modal in general and no functional representation of it is available. It is derived from the sample set representation $\{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}),\ n = 1, \ldots, N\}$ of $p(\mathbf{X}_{t-1}|\mathcal{Z}_{t-1})$, the output from the previous time-step, to which prediction (3.5) must then be applied.

The iterative process as applied to sample-sets, depicted in figure 3.5, mirrors the continuous diffusion process in figure 3.2. At the top of the diagram, the output from time-step $t-1$ is the weighted sample-set $\{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}),\ n = 1, \ldots, N\}$. The aim is to maintain, at successive time-steps, sample sets of fixed size N, so that the algorithm can be guaranteed to run within a given computational resource. The first operation therefore is to sample (with replacement) N times from the set $\{\mathbf{s}_{t-1}^{(n)}\}$, choosing a given element with probability $\pi_{t-1}^{(n)}$. Some elements, especially those with high weights, may be chosen several times, leading to identical copies of elements in the new set. Others with relatively low weights

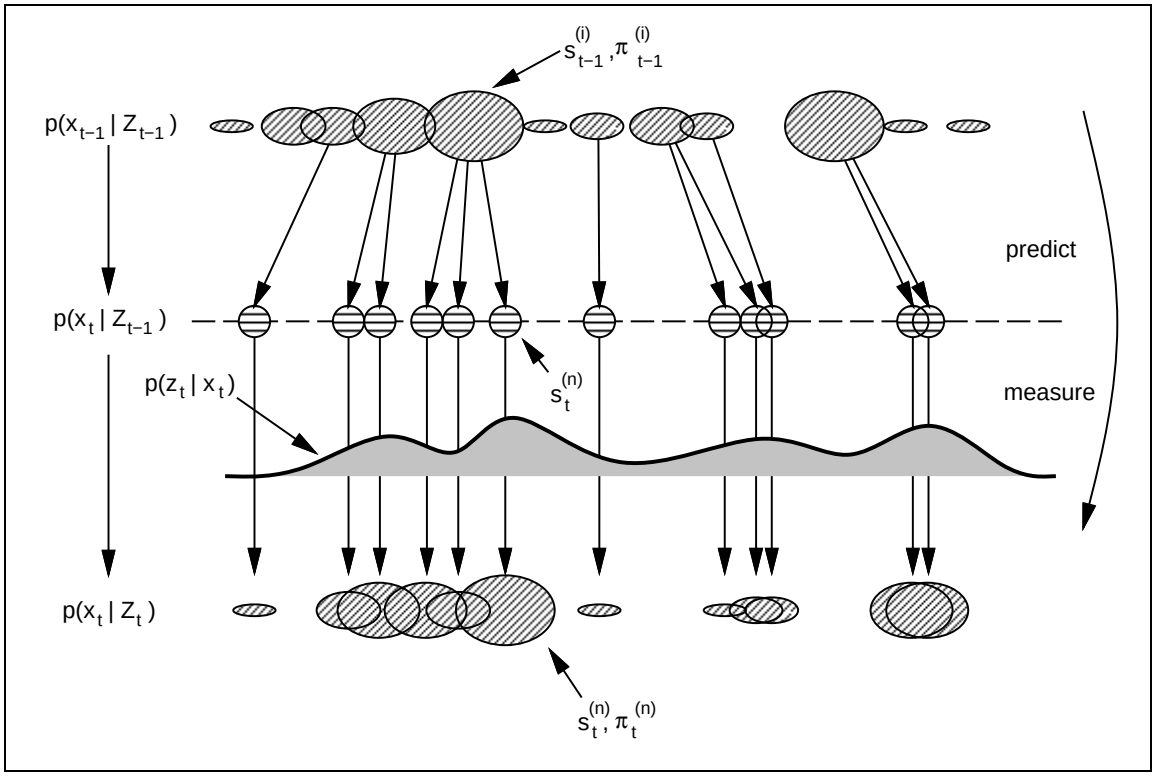


Figure 3.5: **One time-step in the** CONDENSATION **algorithm.** *The three steps — drift-diffuse-measure — of the probabilistic propagation process of figure 3.2 are represented in the* CONDENSATION *algorithm. Deterministic drift and diffusion are shown here as a single prediction stage.*

may not be chosen at all. Each element chosen from the new set is now subjected to the predictive step. The stochastic nature of the motion model causes identical elements to split and diffuse through the state-space. At this stage, the sample set $\{\mathbf{s}_t^{(n)}\}$ for the new time-step has been generated but, as yet, without its weights; it is approximately a fair random sample from the effective prior density $p(\mathbf{X}_t|\mathcal{Z}_{t-1})$ for time-step t. Finally, the observation step from factored sampling is applied, generating weights from the observation density $p(\mathbf{Z}_t|\mathbf{X}_t)$ to obtain the sample-set representation $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}$ of the state-density for time t.

Figure 3.6 gives a synopsis of the algorithm, and a proof of its asymptotic correctness is given in appendix A.2 on page 145. The algorithm makes use of cumulative probabilities $c_t^{(n)}$ calculated from the $\pi_t^{(n)}$:

$$\begin{aligned} c_t^{(0)} &= 0, \\ c_t^{(n)} &= c_t^{(n-1)} + \pi_t^{(n)} \quad (n = 1, \ldots, N). \end{aligned}$$

In step 1 a base sample $\mathbf{s}_t'^{(n)} = \mathbf{s}_{t-1}^{(j)}$ is chosen from the sample-set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}\}$ with probability $\pi_{t-1}^{(j)}$. This can be done efficiently as follows:

1. generate a random number $r \in [0, 1]$, uniformly distributed.
2. find, by binary subdivision, the smallest j for which $c_{t-1}^{(j)} \geq r$
3. set $\mathbf{s}_t'^{(n)} = \mathbf{s}_{t-1}^{(j)}$

Alternatively, the following deterministic algorithm can be used to find all the $\mathbf{s}_t'^{(n)}$ at once:

initialise: $j = 1$. **for** $n = 1 \ldots N$
  while $(c_{t-1}^{(j)} < n)$ $j{+}{+}$
  set $\mathbf{s}_t'^{(n)} = \mathbf{s}_{t-1}^{(j)}$.

The use of the random-sampling algorithm causes one iteration of the CONDENSATION algorithm to have formal complexity $O(N \log N)$ while the deterministic algorithm reduces this to $O(N)$. Carpenter et al. (1997) describe a random-sampling algorithm which again leads to $O(N)$ complexity for one iteration of CONDENSATION but they appeal to results in stratified sampling theory to argue that the deterministic method of choosing base samples is more efficient. Kitagawa (1996) also considers this problem in an Appendix and reports that the best performance is achieved using the deterministic algorithm, preceded by a sorting operation which orders the $\pi_{t-1}^{(n)}$ according to magnitude. Due to the computational cost of this sort stage, he recommends the deterministic algorithm without sorting. Only the results in chapter 6 were generated using this deterministic method; all the other examples shown use the $O(N \log N)$ random-sampling scheme.

After any time-step of the CONDENSATION algorithm, it is possible to "report" on the current state, for example by evaluating some moment of the state density as shown in figure 3.6, and in later examples typically the curve outline displayed is the estimated mean of the distribution. This can be misleading in the case of a multi-modal state distribution, and one solution to this problem is discussed in chapter 7.

One of the striking properties of the CONDENSATION algorithm is its simplicity, compared with the Kalman filter, despite its generality. Largely this is due to the absence of the Riccati equation which appears in the Kalman filter for the propagation of covariance. The Riccati equation is relatively complex computationally but is not required in the CONDENSATION algorithm which instead deals with variability by sampling, involving the repeated computation of a relatively simple propagation formula.

Iterate

From the "old" sample-set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}, n = 1, \ldots, N\}$ at time-step $t-1$, construct a "new" sample-set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}\}, n = 1, \ldots, N$ for time t.

Construct the n^{th} of N new samples as follows:

1. **Select** a sample $\mathbf{s}_t'^{(n)} = \mathbf{s}_{t-1}^{(j)}$ with probability $\pi_{t-1}^{(j)}$. This can be done efficiently with the aid of the cumulative probabilities $c_{t-1}^{(n)}$ and this is discussed in the text.

2. **Predict** by sampling from

$$p(\mathbf{X}_t | \mathbf{X}_{t-1} = \mathbf{s}_t'^{(n)})$$

   to choose each $\mathbf{s}_t^{(n)}$. For instance, in the case that the dynamics are governed by a linear stochastic differential equation, the new sample value may be generated as: $\mathbf{s}_t^{(n)} = \mathbf{A}\mathbf{s}_t'^{(n)} + B\mathbf{w}_t^{(n)}$ where $\mathbf{w}_t^{(n)}$ is a vector of standard normal random variates, and BB^T is the process noise covariance — see section 3.5.

3. **Measure** and weight the new position in terms of the measured features $\mathbf{Z}_t$:

$$\pi_t^{(n)} = p(\mathbf{Z}_t | \mathbf{X}_t = \mathbf{s}_t^{(n)})$$

   then normalise so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$ where

$$\begin{aligned} c_t^{(0)} &= 0, \\ c_t^{(n)} &= c_t^{(n-1)} + \pi_t^{(n)} \quad (n = 1, \ldots, N). \end{aligned}$$

Once the N samples have been constructed: **estimate**, if desired, moments of the tracked position at time-step t as

$$\mathcal{E}[f(\mathbf{X}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f\left(\mathbf{s}_t^{(n)}\right)$$

obtaining, for instance, a mean position using $f(\mathbf{X}) = \mathbf{X}$.

Figure 3.6: **The** CONDENSATION **algorithm.**

3.5 Stochastic dynamical models for curve motion

In order to apply the CONDENSATION algorithm, which is general, to tracking curves in image-streams, specific probability densities must be established both for the dynamics of the object and for the observation process. In the examples described here, $\mathbf{x}$ is the linear parameterisation of a B-spline curve set out in section 2.1 on page 15 and $\mathbf{X}$ is the augmented second-order state variable defined in (2.11) on page 28. The CONDENSATION algorithm itself does not necessarily demand a *linear* parameterisation and in chapter 6 a simple non-linear parameterisation of Euclidean similarities is used.

The dynamical model (2.12) on page 28 can be re-expressed in such a way as to make quite clear that it is a temporal Markov chain:

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) \propto \exp -\frac{1}{2}\|B^{-1}(\mathbf{X}_t - \mathbf{D} - A\mathbf{X}_{t-1})\|^2 \tag{3.8}$$

where $\|\ldots\|$ is the Euclidean norm. It is therefore clear that the learned dynamical models of section 2.5.1 are appropriate for use in the CONDENSATION algorithm.

3.5.1 Initial conditions

Initial conditions for tracking can be determined by specifying the prior density $p(\mathbf{X}_0)$, and if this is Gaussian, direct sampling can be used to initialise the CONDENSATION algorithm. Alternatively it is possible simply to allow the density $p(\mathbf{X}_t)$ to settle to a steady state $p(\mathbf{X}_\infty)$, in the absence of object measurements. Provided the learned dynamics are stable (free of undamped oscillations) a unique steady state exists. Furthermore, if $p(\mathbf{X}_0)$ is Gaussian, $p(\mathbf{X}_\infty)$ is Gaussian with parameters that can be computed by iterating the Riccati equation (Gelb, 1974). At this point the density function represents an envelope of possible configurations of the object, as learned during the training phase. (Background clutter, if present, will modify and bias this envelope to some extent.) Then, as soon as the foreground object arrives and is measured, the density $p(\mathbf{X}_t)$ begins to evolve appropriately. In practice, for most experiments, an initial configuration $\hat{\mathbf{X}}_0$ of the object is specified by hand and $p(\mathbf{X}_0)$ is taken to be a small Gaussian distribution about $\hat{\mathbf{X}}_0$. Chapter 6 describes a method for automatic re-initialisation which can be used when approximate information is available about the object's position.

3.6 Observation model

The observation process defined by $p(\mathbf{Z}_t|\mathbf{X}_t)$ is assumed here to be stationary in time (though the CONDENSATION algorithm does not necessarily demand this) so a static function $p(\mathbf{Z}|\mathbf{X})$ needs to be specified. The assumption will also be made throughout this thesis that the observation density depends only on the object's configuration at the current timestep, so when using an augmented state vector of the form in (2.11) on page 28 we can write

$$p(\mathbf{Z}_t|\mathbf{X}_t) = p(\mathbf{Z}_t|\mathbf{x}_t)$$

and so $p(\mathbf{Z}|\mathbf{X})$ and $p(\mathbf{Z}|\mathbf{x})$ will be used interchangeably with slight abuse of notation. As yet we have no capability to estimate this function from data, though that would be ideal, so some reasonable assumptions must be made. First a measurement model for one-dimensional data with clutter is suggested. Then an extension is proposed for two-dimensional observations that is also used later in computational experiments.

3.6.1 One-dimensional observations in clutter

In one dimension, observations reduce to a set of scalar positions $\{\mathbf{Z} = (z_1, z_2, \ldots, z_M)\}$ and the observation density has the form $p(\mathbf{Z}|x)$ where x is one-dimensional position. The multiplicity of measurements reflects the presence of clutter so either one of the events

$$\phi_m = \{\text{true measurement is } z_m\},\ m = 1, \ldots, M$$

occurs, or else the target object is not visible with probability $q = 1 - \sum_m P(\phi_m)$. Such reasoning about clutter and false alarms is commonly used in target tracking (Bar-Shalom and Fortmann, 1988). Now the observation density can be expressed as

$$p(\mathbf{Z}|x) = qp(\mathbf{Z}|\text{clutter}) + \sum_{m=1}^{M} p(\mathbf{Z}|x, \phi_m)P(\phi_m).$$

A reasonable functional form for this can be obtained by making some specific assumptions: that[2] $P(\phi_m) = p,\ \forall\ m$, that the clutter is a Poisson process along the line with spatial density λ and that any true target measurement is unbiased and normally distributed with standard deviation σ. This leads to

$$p(\mathbf{Z}|x) \propto 1 + \frac{1}{\sqrt{2\pi}\sigma\alpha} \sum_m \exp -\frac{\nu_m^2}{2\sigma^2} \tag{3.9}$$

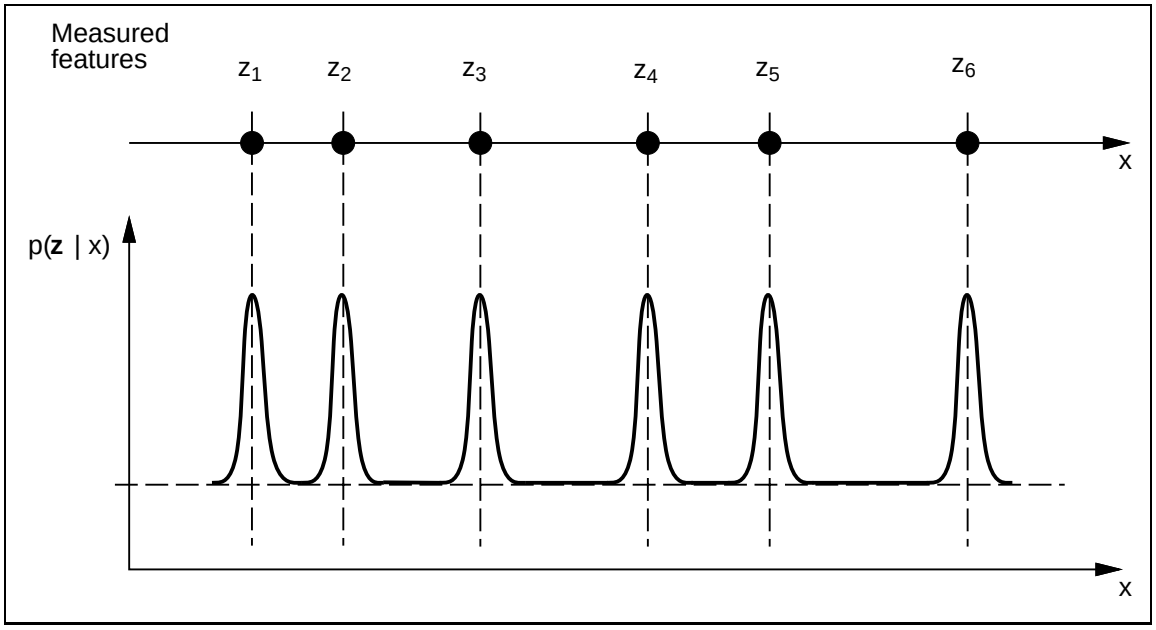


Figure 3.7: **One-dimensional observation model.** *A probabilistic observation model allowing for clutter and the possibility of missing the target altogether is specified here as a conditional density* $p(\mathbf{Z}|x)$.

where $\alpha = q\lambda$ and $\nu_m = z_m - x$, and is illustrated in figure 3.7. Peaks in the density function correspond to measured features and the state density will tend to be reinforced in the CONDENSATION algorithm at such points. The background level reflects the possibility that the true target has not been detected at all. The effect on tracking behaviour is to provide for the possibility of "tunneling": a good hypothesis should survive a transitory failure of observations due, for example, to occlusion of the tracked object. The parameters σ (units of distance) and α (units of inverse distance) must be chosen, though in principle they could be estimated from data by observing measurement error σ and both the density of clutter λ and probability of non-detection q.

Considerable economy can be applied, in practice, in the evaluation of the observation density. Given a hypothesised position x in the "observation" step (figure 3.6) it is not necessary to attend to all features $z_1, \ldots, z_M$. Any ν_m for which

$$\frac{1}{\sqrt{2\pi}\sigma\alpha} \exp -\frac{\nu_m^2}{2\sigma^2} \ll 1$$

can be neglected and this sets a search window around the position x outside which measurements can be ignored. For practical values of the constants the search window will have

[2]There could be some benefit in allowing the $P(\phi_m)$ to vary with m to reflect varying degrees of feature-affinity, based on contrast, colour or orientation.

a width of a few σ. In practice the clutter is sufficiently sparse and σ is sufficiently small that the search window rarely contains more than one feature.

Note that the density $p(\mathbf{Z}|x)$ represents the information about x given a fixed number M of measurements. Potentially, the *event* ψ_M that there are M measurements, regardless of the actual *values* of those measurements, also constitutes information about x. However, we can reasonably assume here that

$$P(\psi_M|x) = P(\psi_M),$$

for instance because x is assumed to lie always within the image window. In that case, by Bayes' theorem,

$$p(x|\psi_M) = p(x)$$

— the event ψ_M provides no additional information about the position x. (If x is allowed also to fall outside the image window then the event ψ_M *is* informative: a value of M well above the mean value for the background clutter enhances the probability that x lies within the window.)

3.6.2 Two-dimensional observations

In a two-dimensional image, the set of observations $\mathbf{Z}$ is, in principle, the entire set of features visible in the image. However, an important aspect of earlier systems in achieving real-time performance (Lowe, 1992; Harris, 1992; Blake et al., 1993b) has been the restriction of measurement to a sparse set of lines normal to the tracked curve. These two apparently conflicting ideas can be resolved as follows.

The observation density $p(\mathbf{Z}|\mathbf{X})$ in two dimensions describes the distribution of a parameterised image curve $\mathbf{z}(s')$, given a hypothetical shape in the form of a curve $\mathbf{r}(s),\ 0 \le s \le L$, represented by a shape parameter $\mathbf{x}$. The two-dimensional density can be derived as an extension of the one-dimensional case. It is assumed that a mapping $g(s')$ is known that associates each point $\mathbf{z}(s')$ on the image curve with a point $\mathbf{r}(g(s'))$ on the shape. In practice this mapping is set up by tracing normals from the curve $\mathbf{r}$. Note that $g(s')$ is not necessarily injective because $\mathbf{z}(s')$ includes clutter as well as foreground features (figure 3.8). Next the one-dimensional density (3.9) is approximated in a more amenable form that neglects the possibility of more than one feature lying inside the search interval:

$$p(\mathbf{Z}|x) \propto \exp -\frac{1}{2\sigma^2} f(\nu_1;\mu) \quad \text{where} \quad f(\nu;\mu) = \min(\nu^2, \mu^2), \tag{3.10}$$

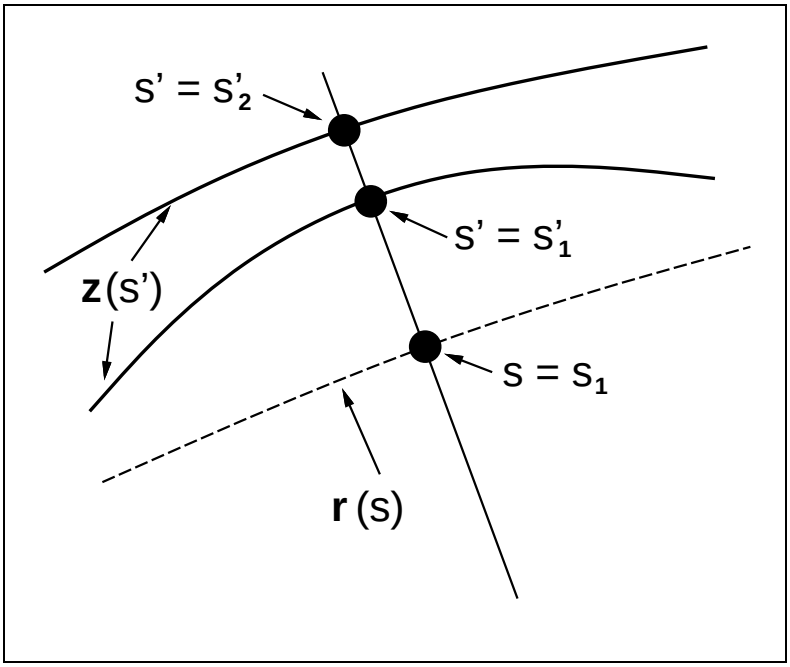


Figure 3.8: **Clutter induces multiple feature hypotheses.** *A point $s = s_1$ on a hypothetical shape curve $\mathbf{r}(s)$ is associated with two points $s' = s'_1$ and $s' = s'_2$ on the image curve $\mathbf{z}(s')$. This implies that the mapping $g(s')$, which associates each point $\mathbf{z}(s')$ on the image curve with a point $\mathbf{r}(g(s'))$, is not injective, since $g(s'_1) = g(s'_2) = s$ (see text).*

$\mu = \sqrt{2}\sigma \log(1/\sqrt{2\pi}\alpha\sigma)$ is a spatial scale constant, and ν_1 is the ν_m with smallest magnitude, representing the feature lying closest to the hypothesised position $\mathbf{x}$. A natural extension to two dimensions is then

$$p(\mathbf{Z}|\mathbf{x}) = Z \exp -\frac{1}{2r} \int_0^L f(\mathbf{z}_1(s) - \mathbf{r}(s); \mu)\, \mathrm{d}s \tag{3.11}$$

in which r is a variance constant and $\mathbf{z}_1(s)$ is the closest associated feature to $\mathbf{r}(s)$:

$$\mathbf{z}_1(s) = \mathbf{z}(s') \quad \text{where} \quad s' = \arg \min_{s' \in g^{-1}(s)} |\mathbf{r}(s) - \mathbf{z}(s')|.$$

Note that the constant of proportionality ("partition function") $Z(\mathbf{x})$ is an unknown function. We make the assumption that the variation of Z with $\mathbf{x}$ is slow compared with the other term in (3.11) so that Z can be treated as constant over the expected variation in $\mathbf{x}$ (which in principle may range over the entire image). It remains to establish whether this assumption is justified.

The observation density (3.11) can be computed via a discrete approximation, the simplest being:

$$p(\mathbf{Z}|\mathbf{x}) \propto \exp \left\{ - \sum_{m=1}^{M} \frac{1}{2rM} f(\mathbf{z}_1(s_m) - \mathbf{r}(s_m); \mu) \right\}, \tag{3.12}$$

where $s_m = m/M$. This is simply the product of one-dimensional densities (3.10) with $\sigma = \sqrt{rM}$, evaluated independently along M curve normals as in figure 3.9. The parameters which control the observation density are therefore μ, σ and M. The running time of the algorithm in current implementations is largely dependent on the number of measurements made, so a large value of M will slow the filter down considerably. Judicious positioning of the search lines s_m at informative points on the shape model, rather than spacing the s_m evenly, can allow a smaller M for equivalent tracking performance. μ very roughly controls the clutter-resistance of the tracker: if an object is expected to lie in a clutter-free environment then μ can be set quite large, and as clutter density increases its value should decrease accordingly. σ should be set according to the accuracy of the shape model. If the expected object appearance is very well modelled by the the shape-space then a small value of σ can be used since features can be expected to be found very close to the predicted curve. If however the shape model is inaccurate, a larger value of σ will permit tracking of shapes which are not exactly within the modelled space, while increasing the risk of distraction by clutter. Values of parameters used in experiments are given in the next chapter.

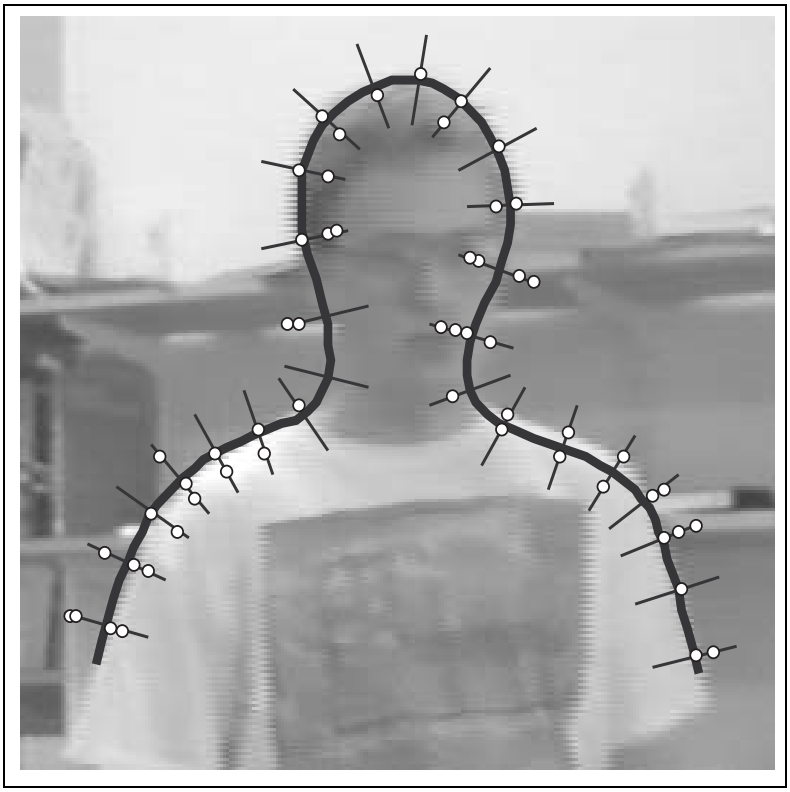


Figure 3.9: **Observation process.** *The thick line is a hypothesised shape, represented as a parametric spline curve. The spines are curve normals along which high-contrast features (white circles) are sought.*

It is now apparent why a mixture of Gaussians representation in a Kalman filtering framework would be inefficient using this observation density. The number k of mixture components in the observation density for a given image would be the product

$$k = \prod_{m=1}^{M} |\nu_m|$$

where $|\nu_m|$ is the number of features detected along search-line m. As explained in section 3.2.4 a Gaussian-mixture Kalman filter must effectively run Mk separate Kalman filter updates, where M is the desired number of mixture components after pruning. For a naive application of a Gaussian mixture filter to the image in figure 3.9, k would be in excess of 40 million. A more practical approach would be to assimilate measurements into the filter from each search-line individually, pruning after each line, but this would be expected to be very sensitive to the exact method used for pruning, as well as the order in which the search-lines were presented.

4

Applying the CONDENSATION algorithm to video-streams

This chapter describes a series of experiments which were undertaken to test the practical efficacy of the CONDENSATION algorithm.

4.1 Tracking a multi-modal distribution

The ability of the CONDENSATION algorithm to represent multi-modal distributions was tested using a 70 frame (2.8 second) sequence of a cluttered room containing three people each facing the camera (figure 4.1). One of the people moves from right to left, in front of the other two. The shape-space for tracking is built from a hand-drawn template of head and shoulders (figure 3.9 on page 51) which is then allowed to deform via planar affine transformations. A Kalman filter contour-tracker (Blake et al., 1993b) with default motion parameters is able to track a single moving person just well enough to obtain a sequence of outline curves that is usable as training data. Given the high level of clutter, adequate performance with the Kalman filter is obtained here by means of background modelling (Rowe and Blake, 1996), a statistical form of background subtraction, which effectively removes clutter from the image data before it is tracked. It transpires, for this particular training set, that the learned motions comprise primarily horizontal translation, with vertical translation and horizontal and vertical shear present to a lesser degree.

The learned shape and motion model can now be installed as $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ in the CONDEN-

Figure 4.1: **Tracking three people in a cluttered room.** *The first frame of a sequence in which one figure moves from right to left in front of two stationary figures.*

SATION algorithm which is now run on a test sequence but *without* the benefit of background modelling, so that the background clutter is now visible to the tracker. Figure 4.2 shows how the state-density evolves as tracking progresses. Initialisation is performed simply by iterating the stochastic model, in the absence of measurements, to its steady state and it can be seen that this corresponds, at time 0, to a roughly Gaussian distribution, as expected. The distribution rapidly collapses down to three peaks which are then maintained appropriately even during temporary occlusion. Although the tracker was designed to track just one person, the CONDENSATION algorithm allows the tracking of all three, for free; the ability to represent multi-modal distributions effectively provides multiple hypothesis capability. (In fact, it is not expected that multiple hypotheses will survive indefinitely as comparably-sized peaks in the posterior distribution. If, in a given image, one object is slightly more like the model than another, the first object will induce a slightly larger peak in the posterior. Over time, however, if the same object is always slightly preferred by the observation model, the cumulative effect of repeated measurement will cause the posterior density mass to focus primarily on that object and other objects' peaks will become small or disappear altogether.) Tracking is based on frame rate (40 ms) sampling in this experiment and distributions are plotted in the figure for alternate frames. The experiment was run

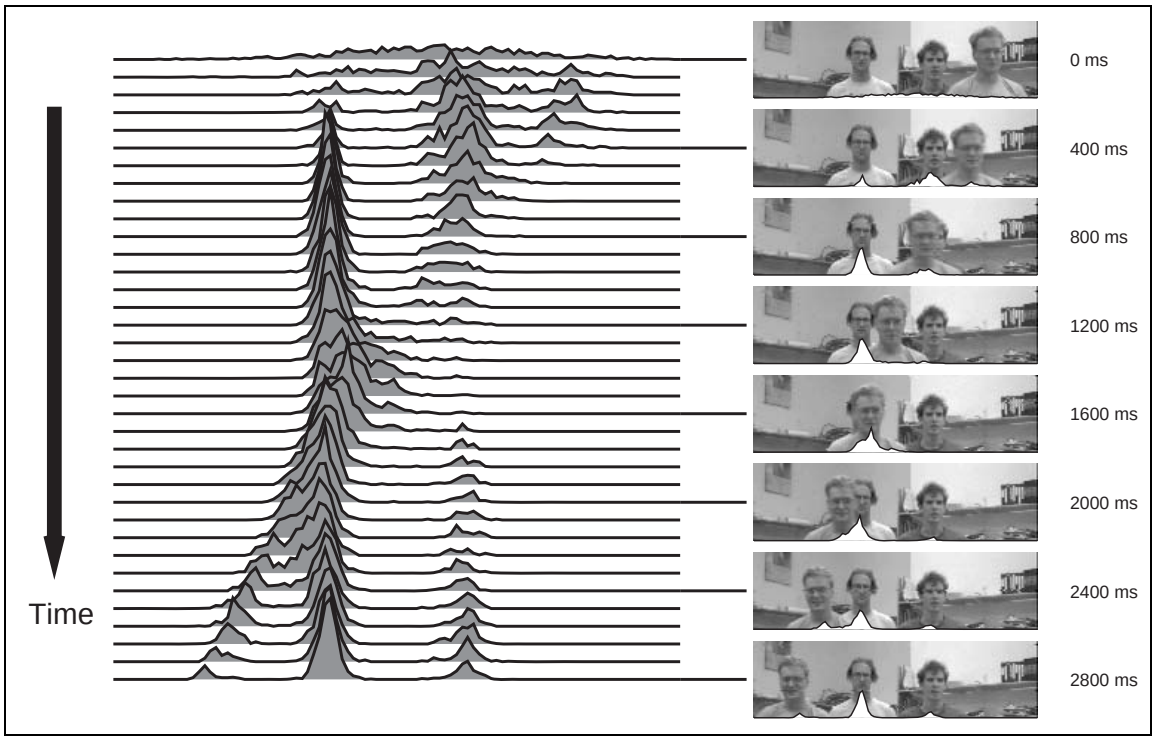


Figure 4.2: **Tracking with multi-modal state-density.** *An approximate depiction of the state-density is shown, computed by smoothing the distribution of point masses* $\mathbf{s}_t^{(1)}, \mathbf{s}_t^{(2)}, \ldots$ *in the* CONDENSATION *algorithm. The density is, of course, multi-dimensional; its projection onto the horizontal translation axis is shown here. The initial distribution is roughly Gaussian but this rapidly evolves to acquire peaks corresponding to each of the three people in the scene. The right-most peak drifts leftwards, following the moving person, coalescing with and separating from the other two peaks as it moves. Having specified a tracker for* one *person we have in a sense, for free, a multi-person tracker, owing to the innate ability of the* CONDENSATION *algorithm to maintain multiple hypotheses. There is however no constraint to ensure that the number of people being tracked remains constant, so all of the density could shift on to one person eventually.*

using a distribution of $N = 1000$ samples per time-step.

4.2 Tracking rapid motions through clutter

The ability to track more agile motion, still against clutter, was tested using a 500 field (10 second) sequence of a girl dancing vigorously to a Scottish reel. The shape-space for tracking was planar affine, based on a hand-drawn template curve for the head outline. The training sequence consisted of dancing against a largely uncluttered background, tracked by a Kalman filter contour-tracker with default dynamics to record 140 fields (2.8 seconds) of tracked head positions, the most that could be tracked before losing lock. Those 140 fields were sufficient to learn a bootstrap motion model which then allowed the Kalman filter to track the training data for 800 fields (16 seconds) before loss of lock. The motion model obtained from these 800 fields was used in experiments with the CONDENSATION tracker and applied to the test data, now including clutter. The use of a Kalman filter in the training stage was purely for experimental convenience given the experimental setup at the time. The CONDENSATION algorithm could equally well have been used with a default motion model and this was done in later experiments, for example in sections 4.5 and 5.3.

Figure 4.3 shows some stills from the test sequence, with a trail of preceding head positions to indicate motion. The motion is primarily translation, with some horizontal shear apparent as the dancer turns her head. Representing the state density with $N = 750$ samples at each time-step proves sufficient for successful tracking. As in the previous example, a prior density can be computed as the steady state of the motion model and, in this case, that yields a prior for position that spreads across most of the image area, as might be expected given the range of the dance. Such a broad distribution cannot effectively be represented by just $N = 750$ samples. One alternative is to increase N in the early stages of tracking, and this is done in a later experiment. Alternatively, the prior can be based on a narrower distribution whose centre is positioned by hand over the object at time 0, and that is what was done here. Observation parameters were $\mu = 24$, $\sigma = 7$ with $M = 18$ normals.

Figure 4.4 shows the motion of the centroid of the estimated head position as tracked both by the CONDENSATION algorithm and by a Kalman filter using the same motion model. The CONDENSATION tracker correctly estimated head position throughout the sequence, but after about 40 fields (0.80 s), the Kalman filter was distracted by clutter, never to recover.

Figure 4.3: **Tracking agile motion in clutter.** *The test sequence consists of 500 fields (10 seconds) of agile dance against a cluttered background. The dancer's head is tracked through the sequence. Several representative fields are shown here, each with a trail of successive mean tracked head positions at intervals of 40 ms. The* CONDENSATION *algorithm used* $N = 750$ *samples per time-step to obtain these results.*

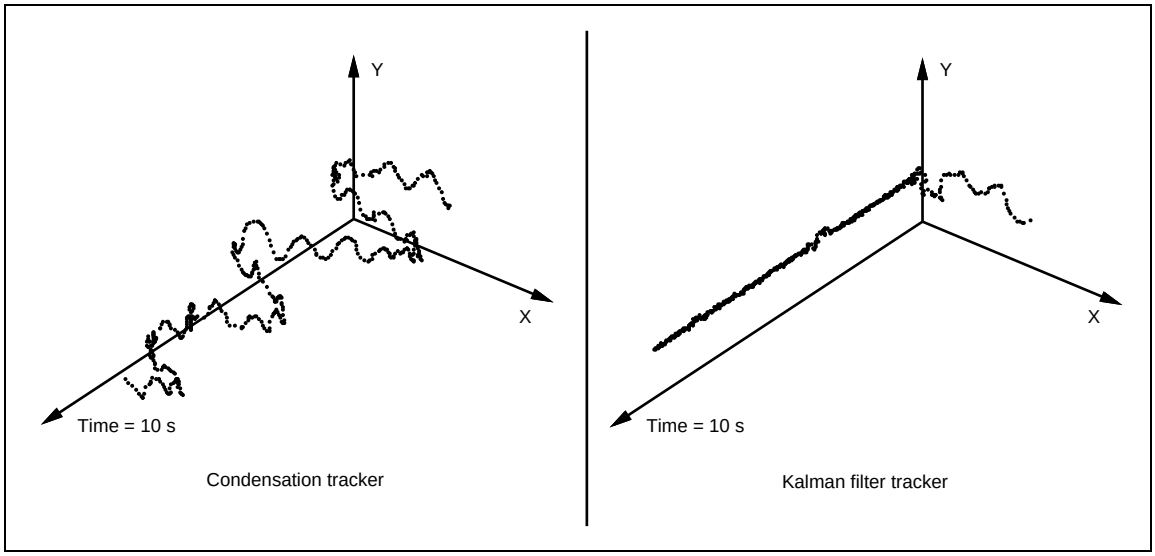
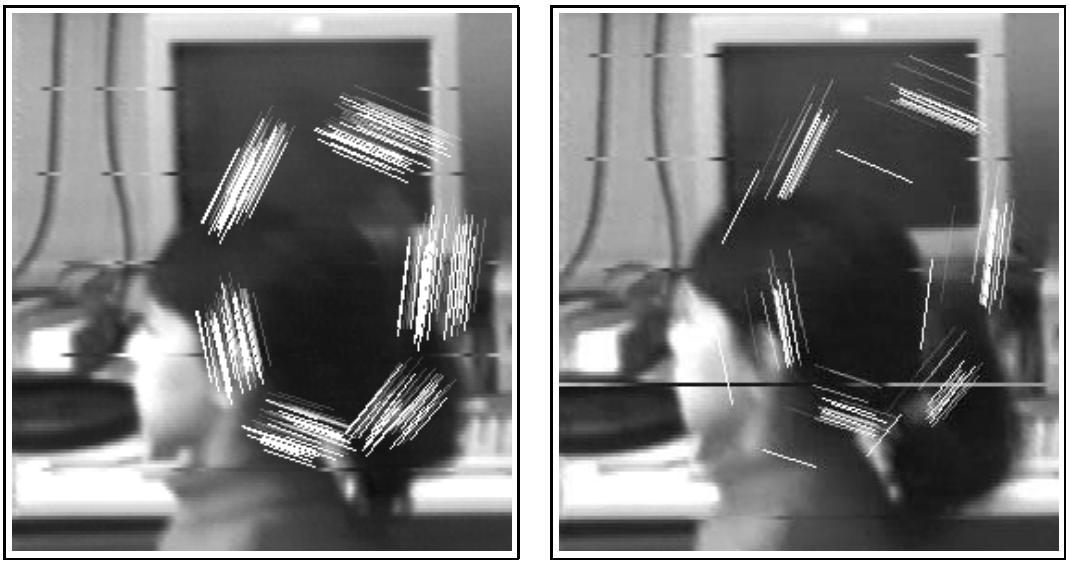
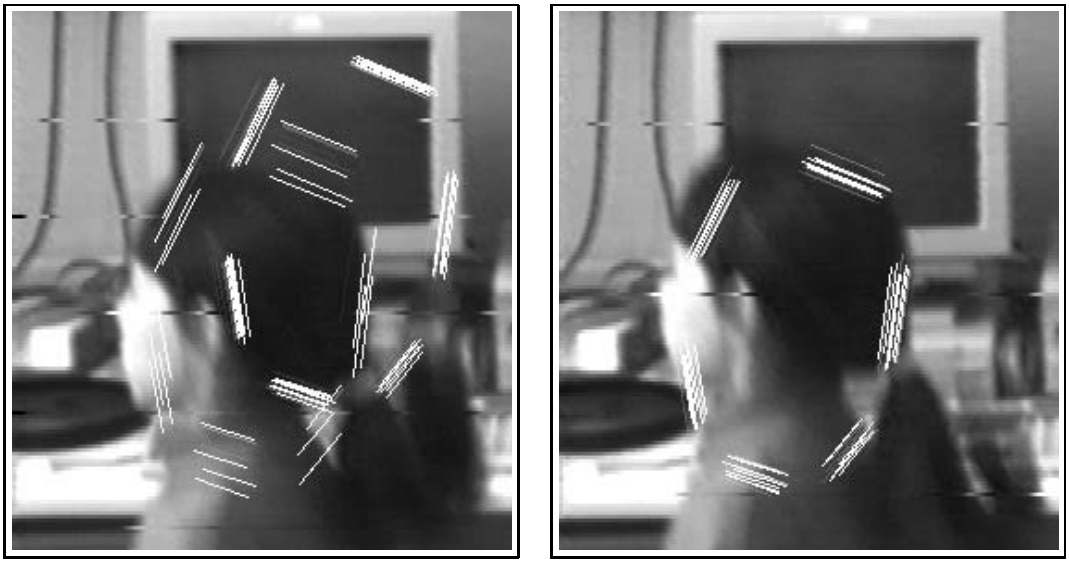


Figure 4.4: **The CONDENSATION tracker succeeds where a Kalman filter fails.** *The estimated centroid for the sequence shown in figure 4.3 is plotted against time for the entire 500 field sequence, as tracked first by the* CONDENSATION *tracker, then by a comparable Kalman filter tracker. The* CONDENSATION *algorithm correctly estimates the head position throughout the sequence. The Kalman filter tracks briefly, but is soon distracted by clutter and never recovers.*

Given that there is only one moving person in this experiment, unlike the previous one in which there were three, it might seem that a unimodal representation of the state density should suffice. This is emphatically not the case. The facility to represent multiple modes is crucial to robustness as figure 4.5 illustrates. The figure shows how the distribution becomes misaligned (at 900ms), reacting to the distracting form of the computer screen. After a further 20ms the distribution splits into two distinct peaks, one corresponding to clutter (the screen), one to the dancer's head. At this point the clutter peak actually has the higher posterior probability — a unimodal tracker, for instance a Kalman filter, would almost certainly discard the lower peak, rendering it unable to recover. The CONDENSATION algorithm however, capable as it is of carrying several hypotheses simultaneously, does recover rapidly as the clutter peak decays for lack of confirmatory observation, leaving just one peak corresponding to the dancer at 960 ms.



field 45 (900 ms) field 46 (920 ms)



field 47 (940 ms) field 48 (960 ms)

Figure 4.5: **Recovering from tracking failure.** *Detail from 4 consecutive fields of the sequence illustrated in figure 4.3. Each sample from the distribution is plotted on the image, with intensity scaled to indicate its posterior probability. (Most of the* $N = 750$ *samples have too low a probability to be visible in this display.) In field 45, the distribution is misaligned, and has begun to diverge. In fields 46 and 47 it has split into two distinct peaks, the larger attracted to background clutter, but converges back onto the dancer in field 48.*

4.3 Tracking an articulated object

The preceding sequences show motion taking place in affine shape-spaces of just 6 dimensions. High dimensionality is one of the factors, in addition to agility and clutter, that makes tracking hard (Blake et al., 1993b). In order to investigate tracking performance in higher dimensions, we used a 500 field (10 second) test sequence of a hand translating, rotating, and flexing its fingers independently, over a highly cluttered desk scene (figure 4.6). Figure 4.7 shows just how severe the clutter problem is — the hand is immersed in a dense

Figure 4.6: **A hand moving over a cluttered desk.** *Field 0 of a 500 field (10 second) sequence in which the hand translates, rotates, and the fingers and thumb flex independently.*

field of edges.

A model of shape and motion model was learned from training sequences of hand motion against a plain background, tracked by Kalman filter (using signed edges to help to disambiguate finger boundaries). The procedure comprised several stages, creative assembly of methods from the available "toolkit" for learning (Blake and Isard, 1998).

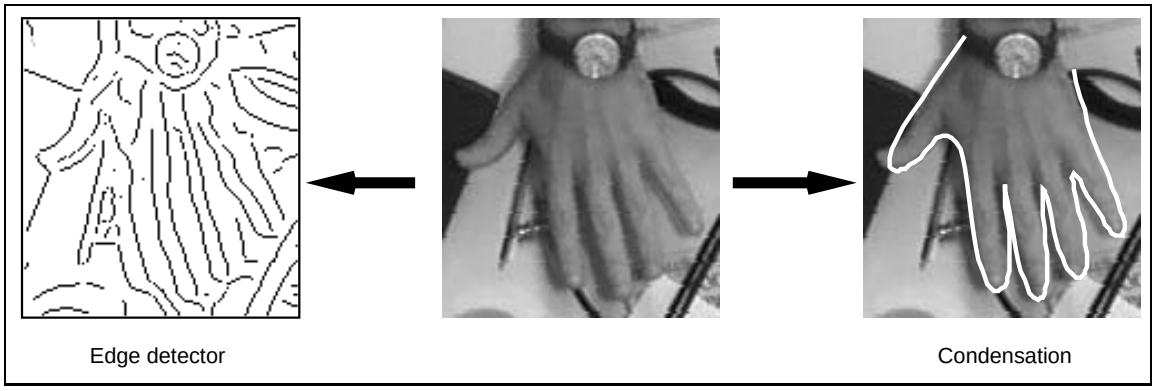


Figure 4.7: **Severe clutter.** *Detail of one field (figure 4.6) from the test-sequence shows the high level of potential ambiguity. Output from a directional Gaussian edge detector shows that there are many clutter edges present as potential distractors.*

1. **Shape-space** was constructed from 6 templates drawn around the hand with the palm in a fixed orientation and with the fingers and thumb in various configurations. The 6 templates combined linearly to form a 5-dimensional space of deformations which were then added to the space of translations to form a 7 dimensional shape-space.

2. **Default hand-specified dynamics** in the shape-space above were adequate to track a clutter-free training sequence of 600 frames in which the palm of the hand maintained an approximately fixed attitude.

3. **Principal components analysis:** the sequence of 600 hand outlines was replicated with each hand contour rotated through 90 degrees and the sequences concatenated to give a sequence of 1200 deformations. Projecting out the translational component of motion, the application of Principal Component Analysis (PCA) to the sequence of residual deformations of the 1200 contours established a 10-dimensional space that was confined almost entirely to deformation and rotation. This was then combined with the translational space to form a 12-dimensional shape-space that accounted both for the flexing of fingers and thumb and also for rotations of the palm.

4. **Bootstrapping:** a Kalman filter with default dynamics in the 12-dimensional shape-space was sufficient to track a training sequence of 800 fields of the hand translating, rotating, and flexing fingers and thumb slowly. This was used to learn a model of motion.

5. **Re-learning:** that motion model was installed in a Kalman filter used to track another, faster training-sequence of 800 fields. This allowed a model for more agile motion to be learned, which was then used in experiments with the CONDENSATION tracker.

Figure 4.8: **Tracking a flexing hand across a cluttered desk.** *Representative stills from a 500 field (10 second) sequence show a hand moving over a highly cluttered desk scene. The fingers and thumb flex independently, and the hand translates and rotates. Here the* CONDENSATION *algorithm uses* $N = 1500$ *samples per time-step initially, dropping gradually over 4 fields to* $N = 500$ *for the tracking of the remainder of the sequence. The mean configuration of the contour is displayed.*

Figure 4.8 shows detail of a series of images from a tracked, 500 field test-sequence. The initial state density was simply the steady state of the motion model, obtained by allowing the filter to iterate in the absence of observations. Tracker initialisation was facilitated by using more samples per time-step ($N = 1500$) at time $t = 0$, falling gradually to 500 over the first 4 fields. The rest of the sequence was tracked using $N = 500$. As with the previous example of the dancer, clutter can distract the tracker but the ability to represent multi-modal state density means that tracking can recover.

4.4 Tracking a camouflaged object

Next, we tested the ability of the algorithm to track rapid motion against background distraction in the extreme case that background objects actually mimic the tracked object. A 12 second (600 field) sequence showed a bush blowing in the wind, the task being to track one particular leaf. A template was drawn by hand around a still of one chosen leaf and allowed to undergo affine deformations during tracking. Given that a clutter-free training sequence was not available, the motion model was again learned by means of a

bootstrap procedure. A tracker with default dynamics proved capable of tracking the first 150 fields of a training sequence before losing the leaf, and those tracked positions allowed a first approximation to the model to be learned. Installing that in a CONDENSATION tracker, the entire sequence could be tracked, though with occasional misalignments. Finally a third learned model was sufficient to track accurately the entire 12-second training sequence. Despite occasional violent gusts of wind and temporary obscuration by another leaf, the CONDENSATION algorithm successfully followed the object, as figure 4.9 shows. In fact, tracking is accurate enough using $N = 1200$ samples to separate the foreground

Figure 4.9: **Tracking with camouflage.** *The aim is to track a single camouflaged moving leaf in this 12-second sequence of a bush blowing in the wind. Despite the heavy clutter of distractors which actually mimic the foreground object, and occasional violent gusts of wind, the chosen foreground leaf is successfully tracked throughout the sequence. Representative stills depict mean contour configurations, with preceding tracked leaf positions plotted at 40ms intervals to indicate motion.*

leaf from the background reliably, an effect which can otherwise only be achieved using "blue-screening" (figure 4.10). Having obtained the model iteratively as above, indepen-

Figure 4.10: **Automated video editing.** *Tracking the outline of a foreground object allows it to be separated automatically from the background, and manipulated as desired, a special effect which can otherwise only be achieved by "blue-screening" from specially prepared footage.*

dent test sequences could be tracked without further training. With $N = 1200$ samples per time-step the tracker runs at 6.5 Hz on a SGI Indy SC4400 200MHz workstation. Reducing this to $N = 200$ increases processing speed to video frame-rate (25 Hz), at the cost of occasional misalignments in the mean configuration of the contour. Observation parameters were $\mu = 8,\ \sigma = 3$ with $M = 21$ normals. The leaf is a good example of an object which deforms freely in a six-dimensional affine space, so the learnt motion model is shown in figure 4.11 as a representative example of second-order dynamics.

4.4.1 Investigating performance as a factor of N

Although the CONDENSATION algorithm has been shown in the preceding chapter to be asymptotically correct as $N \to \infty$, it is hard to show any analytic results concerning its behaviour for finite N. A simple experiment was conducted (data courtesy of John MacCormick) to investigate the variance of the CONDENSATION algorithm as an estimator of the x translation coordinate of the leaf in the preceding video sequence. For various values of N, the algorithm was run $M = 50$ times in succession, each time using a different seed for the random number generator. In each case, the algorithm was initialised so that all the samples were correctly positioned over the leaf in the first timestep, and then the algorithm

$$W = \begin{pmatrix}
1 & 0 & -39.428566 & 0 & 0 & -134.380931 \\
1 & 0 & -87.428566 & 0 & 0 & -86.380931 \\
1 & 0 & -75.428566 & 0 & 0 & 23.619068 \\
1 & 0 & -21.428566 & 0 & 0 & 143.619068 \\
1 & 0 & 60.571433 & 0 & 0 & 223.619068 \\
1 & 0 & 76.571433 & 0 & 0 & 115.619068 \\
1 & 0 & 88.571433 & 0 & 0 & -12.380931 \\
1 & 0 & 36.571433 & 0 & 0 & -112.380931 \\
0 & 1 & 0 & -134.380931 & -39.428566 & 0 \\
0 & 1 & 0 & -86.380931 & -87.428566 & 0 \\
0 & 1 & 0 & 23.619068 & -75.428566 & 0 \\
0 & 1 & 0 & 143.619068 & -21.428566 & 0 \\
0 & 1 & 0 & 223.619068 & 60.571433 & 0 \\
0 & 1 & 0 & 115.619068 & 76.571433 & 0 \\
0 & 1 & 0 & -12.380931 & 88.571433 & 0 \\
0 & 1 & 0 & -112.380931 & 36.571433 & 0
\end{pmatrix} \quad Q_0 = \begin{pmatrix}
332 \\ 284 \\ 296 \\ 350 \\ 432 \\ 448 \\ 460 \\ 408 \\ 40 \\ 88 \\ 198 \\ 318 \\ 398 \\ 290 \\ 162 \\ 62
\end{pmatrix}$$

$$A_2 = \begin{pmatrix}
-0.898817 & -0.027986 & -27.857566 & 22.805919 & 19.868414 & 26.293049 \\
-0.093159 & -0.828229 & 8.789815 & 46.493744 & 20.802591 & 30.512065 \\
-0.000079 & 0.000559 & -0.179707 & 0.014349 & 0.030072 & -0.110301 \\
0.000090 & -0.000116 & -0.045375 & -0.301863 & -0.013603 & -0.018119 \\
-0.000649 & 0.000688 & -0.128898 & 0.357075 & -0.014973 & 0.037543 \\
-0.000758 & -0.000146 & -0.181111 & -0.090182 & -0.000753 & -0.180315
\end{pmatrix}$$

$$A_1 = \begin{pmatrix}
1.898075 & 0.036300 & -9.538231 & 15.621520 & -10.560348 & -70.798115 \\
0.103985 & 1.826474 & 9.629748 & -60.572564 & -15.959238 & -30.105870 \\
0.000183 & -0.000680 & 1.049057 & 0.095297 & -0.050733 & 0.042289 \\
-0.000031 & 0.000207 & 0.251152 & 1.121499 & -0.004105 & 0.076337 \\
0.000663 & -0.000861 & 0.019285 & -0.320773 & 0.810857 & 0.052149 \\
0.000778 & 0.000232 & 0.127044 & 0.177370 & 0.091507 & 1.033757
\end{pmatrix}$$

$$B = \begin{pmatrix}
2.417069 & 0.964665 & 0.001856 & 0.003581 & 0.005267 & 0.004528 \\
0.964665 & 3.345752 & 0.002587 & 0.009619 & 0.002580 & 0.001781 \\
0.001856 & 0.002587 & 0.000088 & 0.000021 & 0.000084 & -0.000043 \\
0.003581 & 0.009619 & 0.000021 & 0.000127 & -0.000094 & 0.000029 \\
0.005267 & 0.002580 & 0.000084 & -0.000094 & 0.000578 & -0.000155 \\
0.004528 & 0.001781 & -0.000043 & 0.000029 & -0.000155 & 0.000109
\end{pmatrix}$$

$$\overline{\mathbf{X}} = \begin{pmatrix} 42.270703 & 72.413708 & -0.590462 & -0.584234 & -0.071800 & -0.019211 \end{pmatrix}^T$$

Figure 4.11: **Learnt motion model for the leaf tracker.** *Parameters are as described in section 2.5 on page 28. The shape-space is 2D-affine.*

was run for 5 timesteps. At that point a filtered estimate $\hat{x}^{(k,N)}$ was computed from the CONDENSATION sample set where $k = 1 \ldots M$ is the index of the trial for a given N. The variance

$$v_N = \frac{1}{M}\sum_{k=1}^{M}(\hat{x}^{(k,N)} - \bar{x}^{(k,n)})^2, \quad \text{where}$$

$$\bar{x}^{(k,n)} = \frac{1}{M}\sum_{k=1}^{M}\hat{x}^{(k,N)}$$

was then computed, and results are shown in figure 4.12. Clearly the variance of the estimated translation coordinate decreases rapidly as N increases.

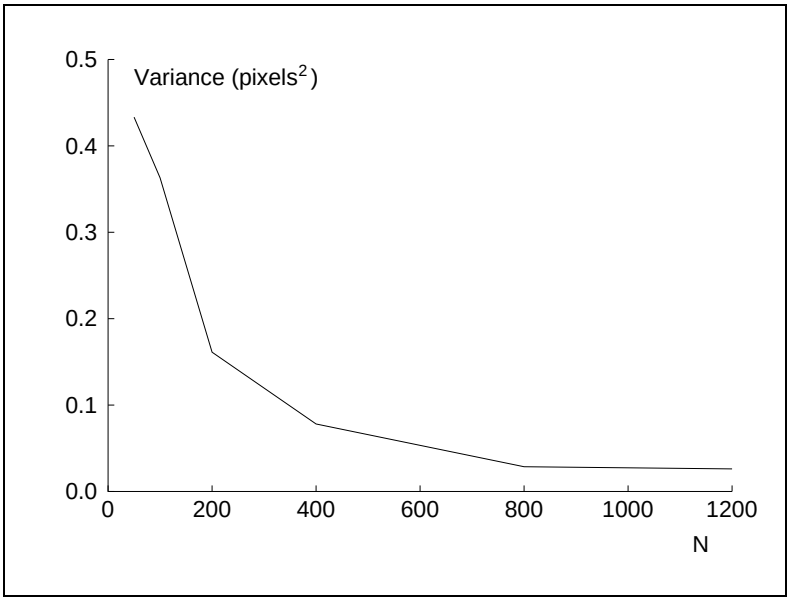


Figure 4.12: **The variance of** CONDENSATION **as an estimator decreases as N increases.** *The vertical axis shows the variance of the* CONDENSATION *estimate of the x translation coordinate of the leaf over 50 runs of the algorithm, as a function of the number of samples N. (Data courtesy of John MacCormick.)*

4.5 Pose recovery from a 3D object

Finally, a 3D-affine rigid model (Blake and Isard, 1998) was built for a sprig of leaves, and the tracked outline used to recover the pose of the object. Building the shape model was a two-stage process: first, eight representative views of the leaves were chosen by hand and a structure recovery algorithm based on factorisation (Blake and Isard, 1998; Tomasi

and Kanade, 1991) was used to determine the 3D-affine shape-space. The factorisation algorithm is modified slightly; here B-spline control points are substituted for the 2D point locations used by (Tomasi and Kanade, 1991) in the original algorithm. After initial tracking it was clear that this shape-space contained inaccuracies, with the result that from some views no vector within the shape-space would fit the observed outline. Seven more views were taken at orientations which exposed the flaws in the original shape-space and the structure algorithm was run again to construct a new shape-space from the fifteen views taken together, which proved satisfactory. When tracking in very dense clutter the shape model must be accurate, and so the second iteration of the structure-fitting process process was essential. As noted below, a value of $\sigma = 2.74$ pixels is used which does not allow high deviations from the modelled shape-space (compare with the dancer model of section 4.2, in which a simple oval is used to represent the girl's head, and $\sigma = 7$ pixels).

Default values for the dynamical parameters A, B and $\mathbf{D}$ were chosen, allowing the first 5 seconds of a clutter-free sequence to be tracked using the CONDENSATION algorithm. Initial estimates for the dynamics matrices were learned from these 250 video fields and using this learnt dynamical model, the first 10 seconds of the clutter free sequence were then tracked, more accurately than before. Another bootstrap model was learned from these 500 fields, with which the whole sequence could be tracked. The final values of the dynamical parameters were learned from these 1322 tracked fields. This final model was then used again to track all 1322 fields of the clutter free sequence, and the 250 fields of the cluttered sequence to give the results shown. It was found that the bootstrapping was necessary because at several points during the sequence a new motion was introduced which had not been present in the preceding fields. Because of the nature of the learned model, novel motions are more difficult to track, and so the filter occasionally fails. Before failure, however, it was able to track the novel motion well enough that a retrained tracker was able to follow the motion correctly. The parameters used in the CONDENSATION algorithm are as follows.

Image sequence	N	σ	μ
Clutter-free	4000	2.74	25
Cluttered	7500	2.74	25

Due to the difficulty of the tracking problem, a large value of N was necessary, and consequently the algorithm ran significantly slower than real-time — for the cluttered sequence the processing time was approximately 2 seconds per field on an SGI Indy R4400 200 MHz (100 times slower than real time).

Figure 4.13 shows that the clutter is severe enough to cause humans difficulties in identifying the leaf outlines from still images. Motion information is exploited by the CONDENSATION algorithm, as it is by human vision, to enable successful tracking. Although the background demonstrated is static, no background subtraction was performed, and a moving background could be tolerated. Indeed the hand, which moves relative to both the background and the leaves and presents significant clutter edges, does not distract the tracking.

Figure 4.13: **Tracking is robust to very heavy background clutter.** *Despite the difficulty, even for humans, of identifying the leaf outlines from a still image, the tracker is able to use information from preceding frames combined with a motion model to successfully track through very heavy clutter (see results in figure 4.16).*

Representative still frames of the tracked outlines and superimposed graphics for both sequences are shown in figures 4.14–4.17. The graphics have been rendered using simple motion blur by averaging five images to produce each field. The images are produced by interpolating the values in the linear shape-space between adjacent tracked fields, and then performing pose recovery on each interpolated shape-space vector before averaging the resulting rendered images. The addition of motion blur is very important when com-

bining computer-generated graphics with fast-moving video sequences. Without artificial blur the graphics retain sharp edges, giving a bright, unreal appearance in the final video. Figure 4.16 also includes a pose graphic which demonstrates that full 3D orientation information is extracted from the tracking. It is apparent that some images include large inter-field displacements, and tracking is successful despite these high image velocities. The tracker can follow more agile motions in the absence of clutter, since the introduction of such a cluttered background makes the tracking problem much harder. Nevertheless impressive results are achieved tracking against the woodland backdrop, and tracking is accurate except for a 10 field portion half way through the sequence, when two leaves becomes partially misaligned before tracking recovers (figure 4.19). Figure 4.18 shows the more traditional matting application of transferring an object from one image sequence to another. Since pose has been recovered, computer graphics can be rendered attached to the foreground object at the same time. Two frames are shown from a sequence in which the leaves from the sequence tracked against a blank background have been re-rendered, with their attendant flowers and flowerpot, in a sequence of a "walk" around a room.

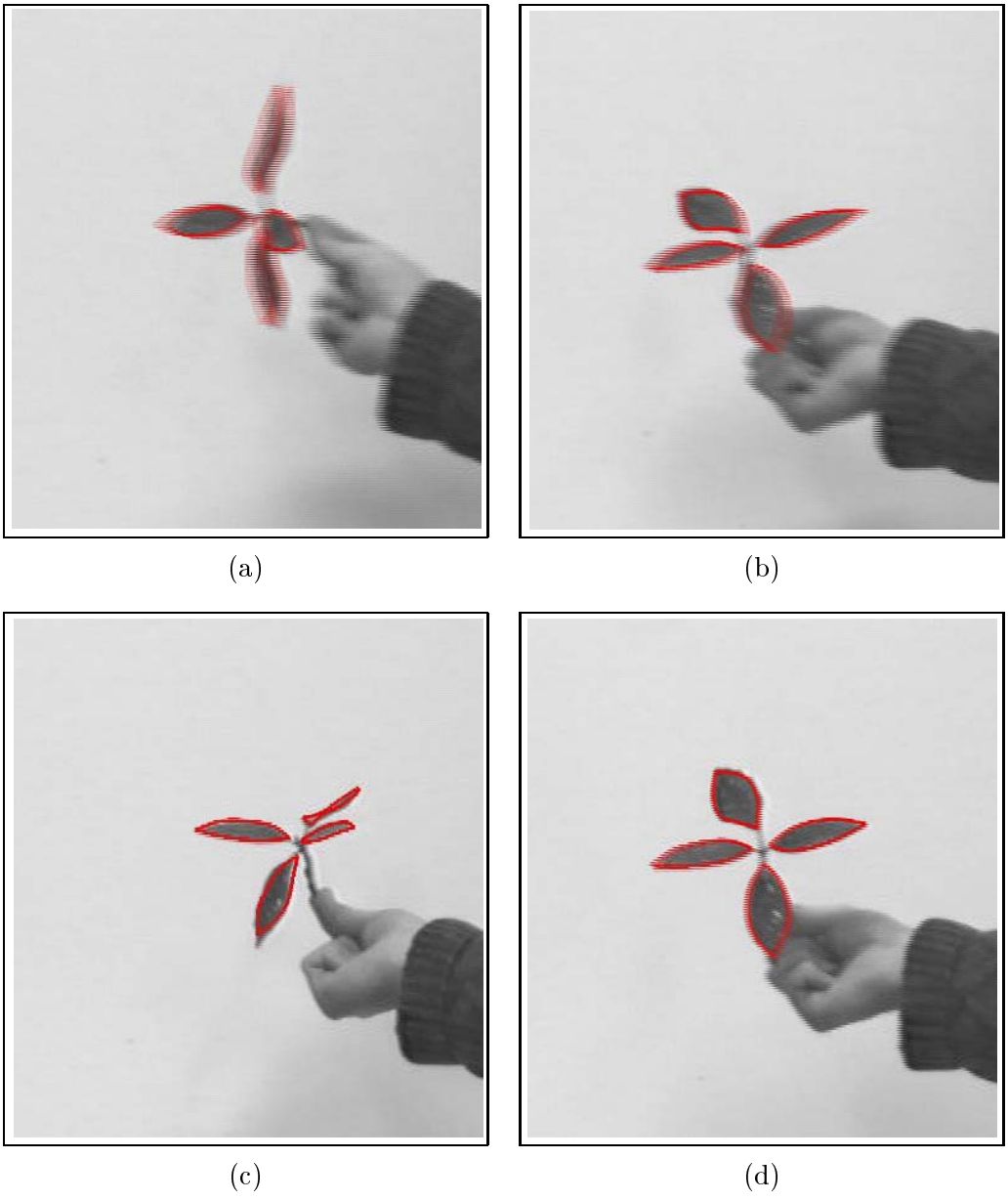
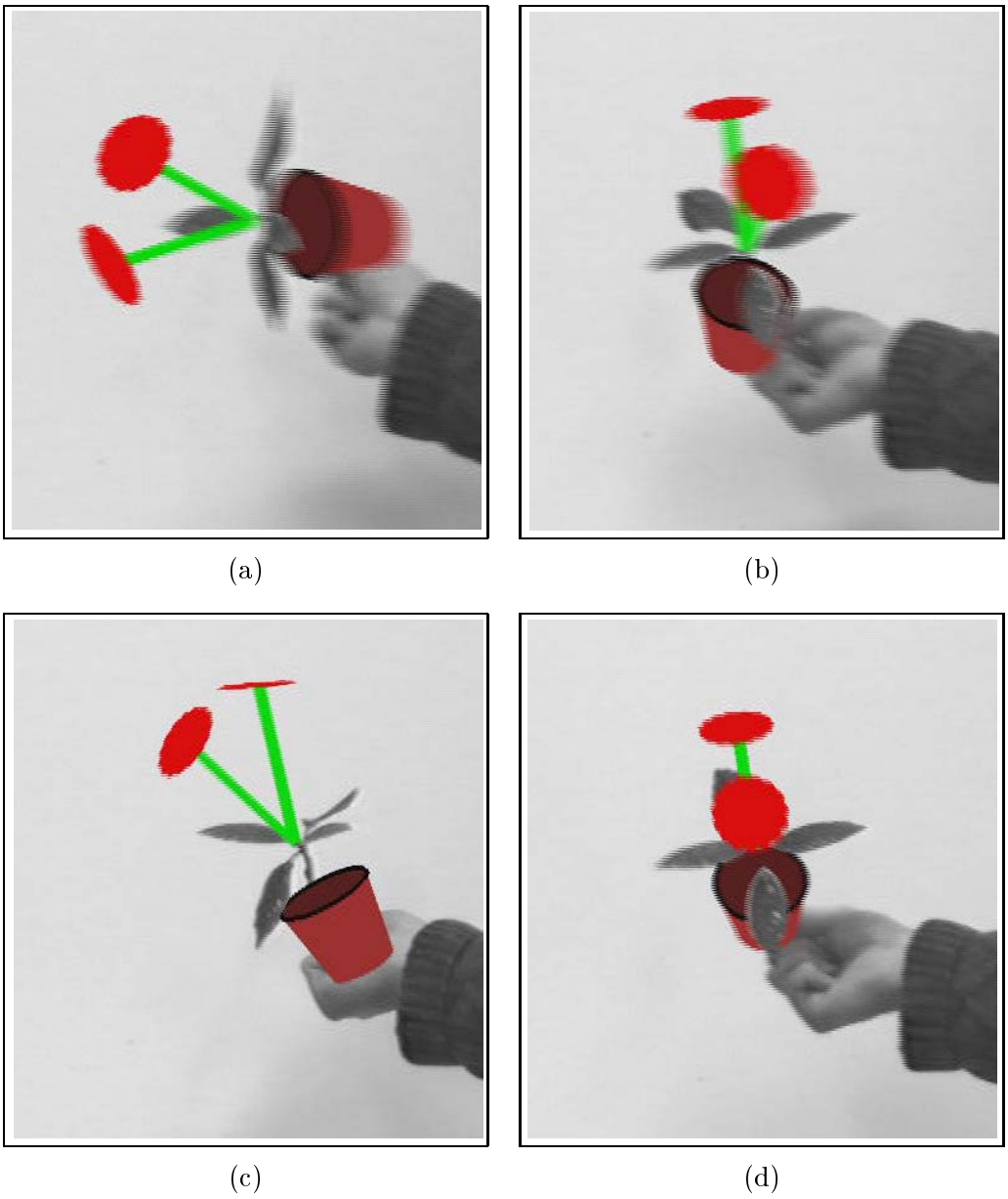


Figure 4.14: **Automatic clutter-free tracking.** *Each image shows detail from one interlaced frame. Field rate is PAL, 50 Hz. Tracking is able to cope with large inter-field motions as can be seen from image (a). Image (c) shows that even at degenerate points where one leaf is barely visible, tracking remains accurate. The tracked outlines have been rendered with simple motion blur by linearly interpolating five positions between tracked fields and averaging the resulting rendered images. Clutter-free tracking is used to learn a model for tracking in clutter.*



(a) (b)

(c) (d)

Figure 4.15: **Computer generated graphics are superimposed on real images using pose recovered from automatically tracked object outlines.** *Each image shows detail from one interlaced PAL frame. The original images with tracked outlines superimposed are shown in figure 4.14. The flowers and flowerpot have been rendered with simple motion blur by linearly interpolating five outline positions between tracked fields and performing pose recovery on each, then averaging the resulting rendered images.*

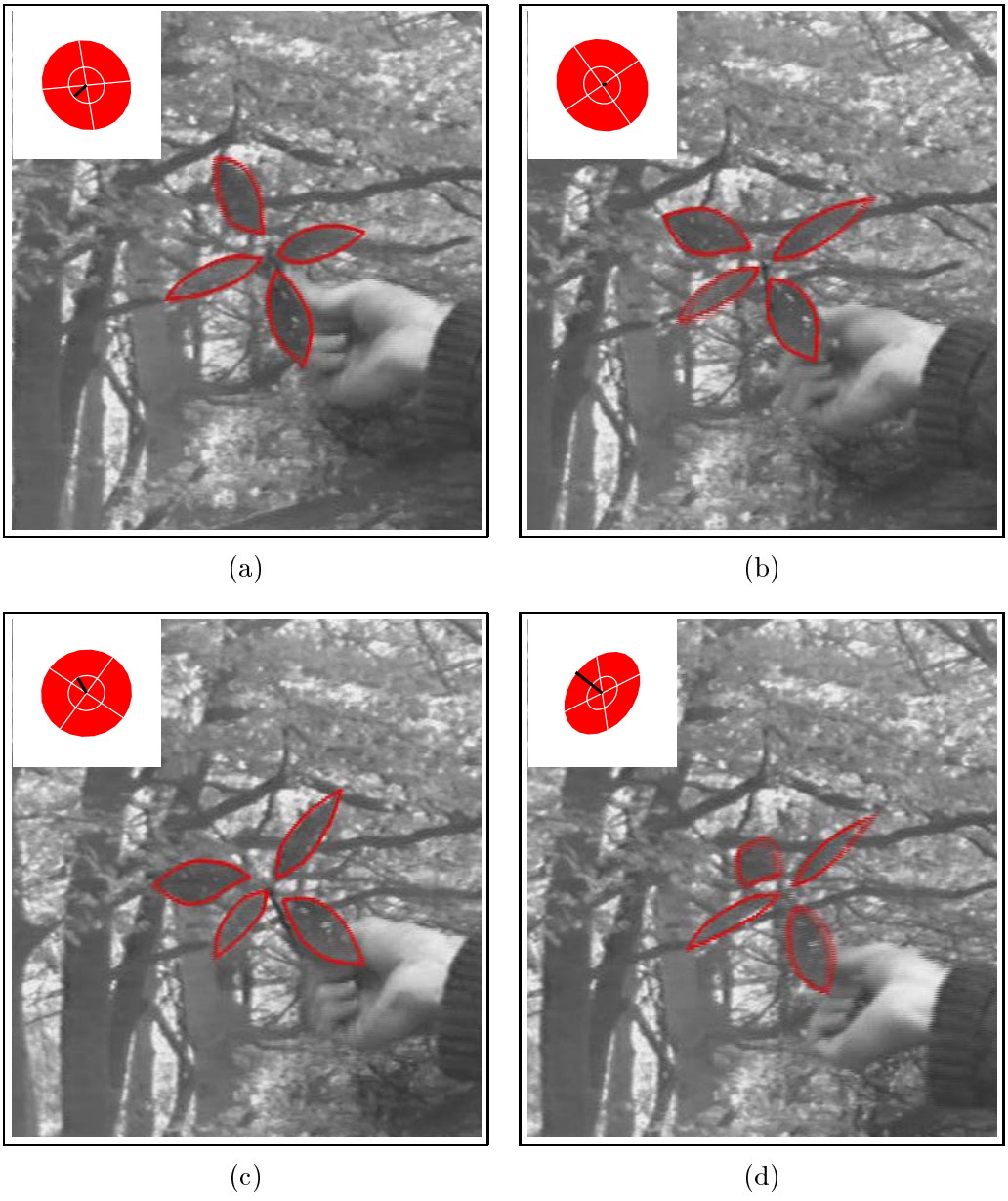


Figure 4.16: **Automatic tracking is successful even against very heavy clutter.** *The tracked position was accurate over the entire 250 field sequence, except for a modest partial misalignment in the middle of the sequence, which lasted for 10 fields. Rendering is as in figure 4.14, and the target graphic is included to demonstrate that full 3D orientation information is being extracted.*

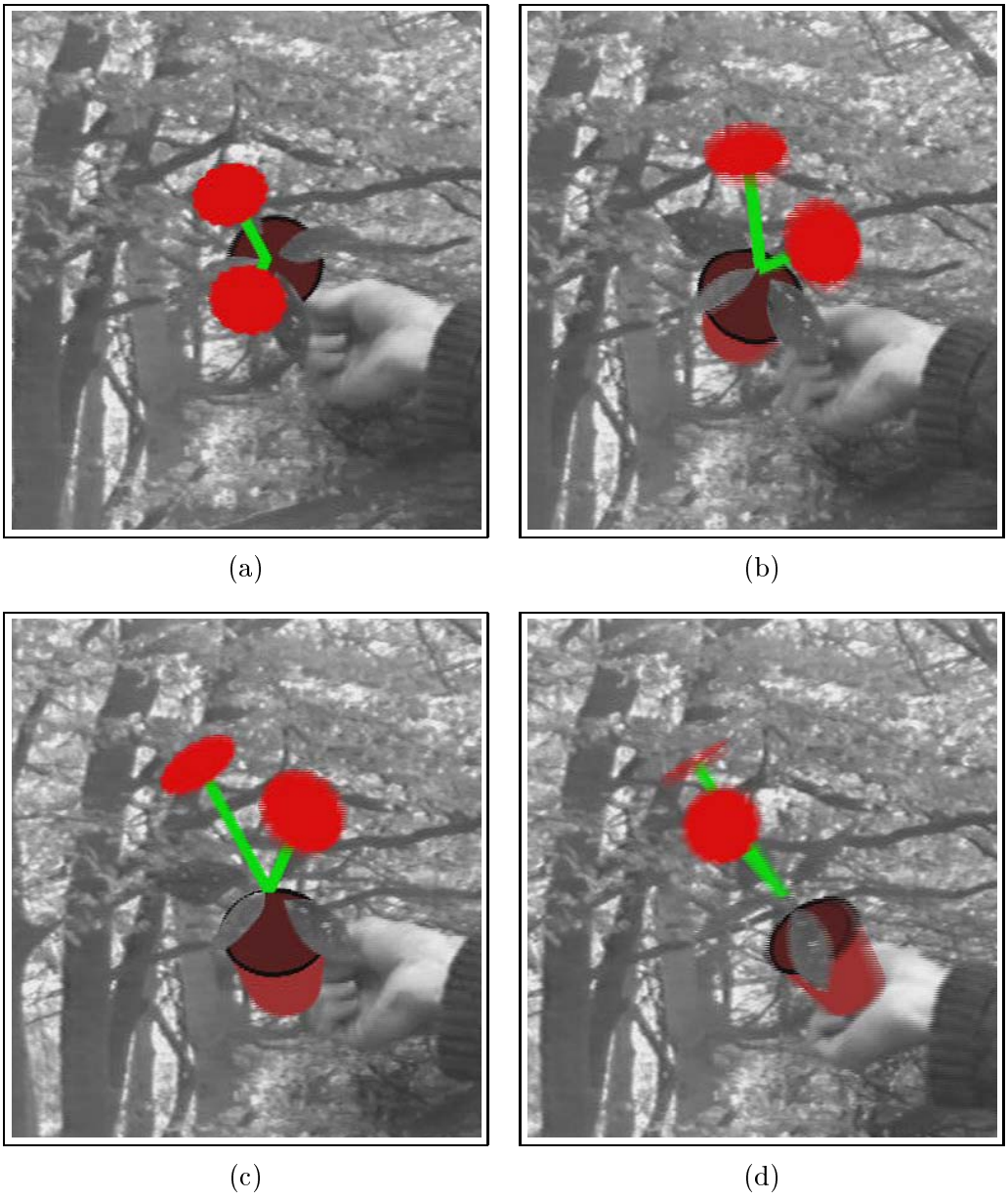


Figure 4.17: **Computer generated graphics are automatically inserted into a complex scene.** *Using the tracked outlines shown in figure 4.16 pose can be recovered allowing the virtual flowers and flowerpot to be inserted "attached" to the leaves. The computer generated objects are shown passing between the foreground and background of the real scene. Rendering is as in figure 4.15.*

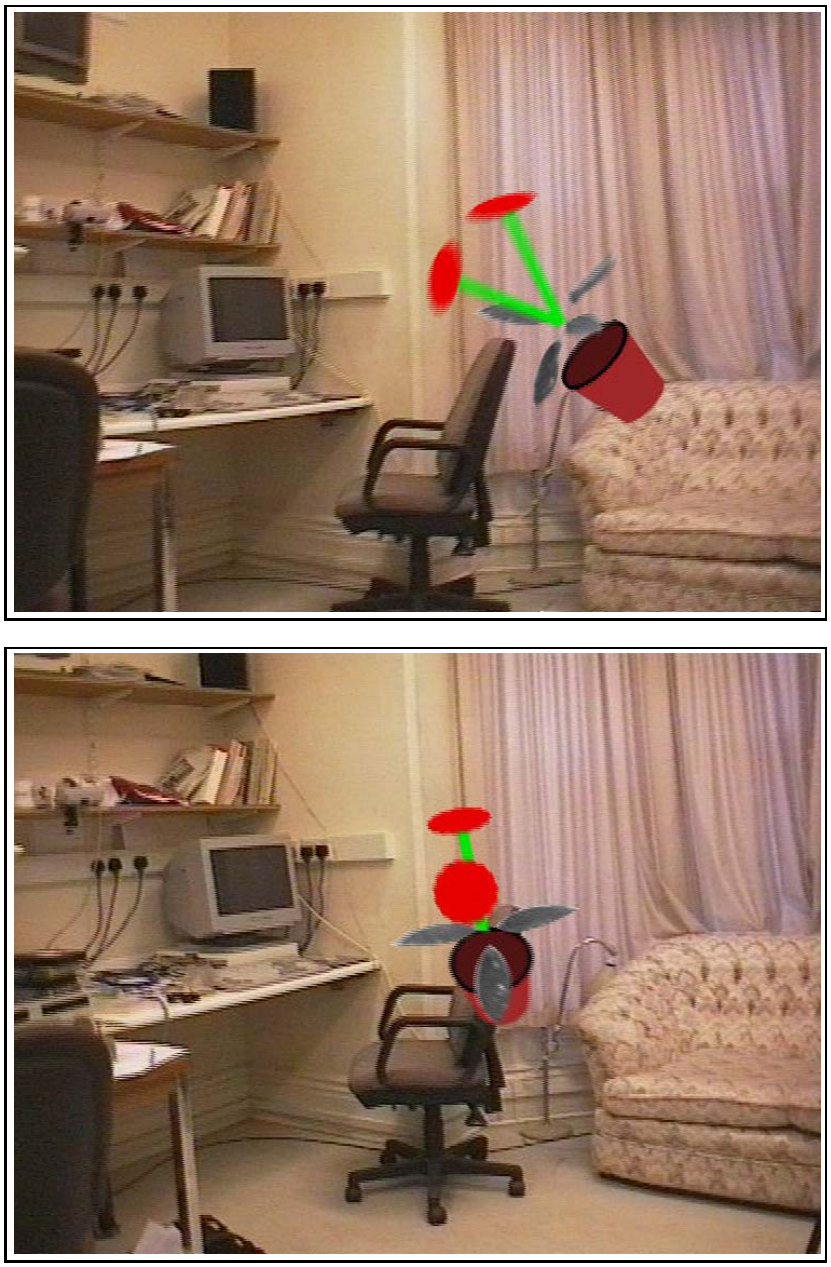


Figure 4.18: **A tracked object with its attached computer graphics can be re-rendered in another scene.** *The traditional application of matting is to transfer an object from one sequence to another. Here the leaves have been segmented from two frames of the uncluttered sequence and, along with their flowers and pot, are superimposed on another sequence — a "walk" around a room. Pose recovery is useful for attaching computer generated images to real objects even in the case that the object can be filmed against an artificial background.*

Figure 4.19: **Slight tracking failures do not affect gross pose recovery.** *Tracking the top two leaves is slightly misaligned for 10 fields in the middle of the heavily cluttered sequence. Gross pose recovery is still correct, but on the video it is apparent that a slight scaling of the rendered graphics takes place due to the tracking error.*

5

Mixed discrete-continuous motion models

This chapter describes a class of motion models which extend the 2nd-order ARPs from section 2.5, and illustrates the flexibility of the CONDENSATION algorithm in accommodating complex non-linear models.

There is significant interest in the computer vision community in the explicit representation and modelling of motion (Blake et al., 1995; Baumberg and Hogg, 1995b; Bobick and Wilson, 1995; Freeman and Roth, 1995; Bregler and Omohundro, 1995; Hennecke et al., 1995; Petajan and Graf, 1995). It is motivated by both the need for accurate motion prediction to permit robust tracking, and the desire to interpret the motion in video sequences to allow reasoning about the content of a sequence, for example for gesture or speech recognition. Considerable success has been demonstrated (Blake et al., 1995; Baumberg and Hogg, 1995b) using the learnt motion models of section 2.5.1 to improve the agility and robustness of trackers, allowing progress to be made for example in audiovisual speech recognition (Kaucic et al., 1996), by enabling lip outlines to be accurately tracked. There is a limit to the complexity of the resulting ARP based models, however, and a natural generalisation is to allow multiple models, with switching between models as appropriate. This allows a wider range of motion to be supported without losing the advantages of accurate prediction, and as a side-effect the model in use at a given timestep acts as a recogniser discriminating between the distinct motions. A mixed discrete/continuous tracker combines the flexibility of continuous-valued motion models, vital for tracking objects which have variable, complex

shape, with the powerful finite state-based descriptions used to model sequential actions, for example in Hidden Markov Models.

In order to combine continuous- and discrete-state motion models it is necessary to construct a mixed-state model representation uniting discrete and continuous random variables in a single parameter vector. A single p.d.f. then describes the joint density of the continuous and discrete variables as they evolve over time. Kalman-filter based techniques to switch between multiple models have been known for some time in the control literature, the dominant example being the Interacting Multiple Model (IMM) filter (Blom and Bar-Shalom, 1988). In order to permit multiple models it is vital to be able to represent multiple competing hypotheses, since in general each discrete state is associated with a separate peak in the joint p.d.f. at each timestep. In common with other generalisations of the Kalman filter which allow multiple hypotheses, for example the JPDAF (Bar-Shalom and Fortmann, 1988), the IMM is faced with a combinatorial explosion of hypotheses and must use pruning techniques to run within a finite computing resource.

There is a great deal of current interest in motion analysis for application to gesture recognition, usually for gestures either of a hand or the full body. Discrete-state based models have been applied successfully (Freeman and Roth, 1995; Starner and Pentland, 1995; Bobick and Wilson, 1995; Kjeldsen and Kender, 1996) when the gestures of interest comprise a predictable sequence of actions. Continuous-valued models have been used to recognise oscillatory gestures (Cohen et al., 1996) or paths in a continuous-valued pattern space (Nagaya et al., 1996). There is also active research into the problem of audiovisual speech recognition using computer vision techniques (Bregler and Omohundro, 1995; Hennecke et al., 1995; Petajan and Graf, 1995; Kaucic et al., 1996). Bregler and Malik (1997) used a hierarchical model to classify the output of a Kalman filter using a Hidden Markov Model representation of human motion. Black and Jepson (1998) used the CONDENSATION algorithm to classify gestures tracked using an optic-flow method according to one of several potential trajectory models. Existing research divides the recognition process into two stages. First, some low-dimensional feature vector is extracted from an image — this commonly takes the form of image moments, the components from an image eigendecomposition, or the output from some region- or contour-based tracker. Only when this information has been extracted is recognition performed on the low-dimensional data. A great potential advantage of the multiple-model approach is that recognition and feature extraction can be performed jointly, and so the form of the expected gesture can be used

to guide feature search, potentially making it more efficient and robust. Existing gesture recognition systems often work on rather coarse shapes; for example hands are usually represented as segmented blobs with no interior structure. For some applications, for example when fine-scale information such as the positions of individual fingers is needed, it is essential to use a framework which allows complex continuous-valued shape models rather than a finite number of states.

5.1 A mixed-state dynamical model

As described in section 3.2 on page 33, the process density $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ can have a somewhat general form, and this fact can be exploited to allow the CONDENSATION algorithm to support, and automatically switch between, multiple motion models. The extended state is defined to be

$$\mathbf{X} = (\tilde{\mathbf{X}}, y)$$

where $y \in \{1, \ldots, N_S\}$ is a discrete variable labelling the current model, and $\tilde{\mathbf{X}}$ is an augmented second-order vector in the parameter space which describes the configuration of the object, as in chapter 3;

$$\tilde{\mathbf{X}}_t = \begin{pmatrix} \mathbf{x}_{t-1} \\ \mathbf{x}_t \end{pmatrix}.$$

As in the previous chapter, the process density is first-order Markov in the augmented state, and it can then be decomposed as follows:

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) = p(\tilde{\mathbf{X}}_t|y_t, \mathbf{X}_{t-1})P(y_t|\mathbf{X}_{t-1})$$
$$T_{ij}(\tilde{\mathbf{X}}_{t-1}) \equiv P(y_t = j|\tilde{\mathbf{X}}_{t-1}, y_{t-1} = i)$$

where the T_{ij} are state transition probabilities. The continuous motion models for each transition are given by the sub-process densities

$$p_{ij}(\tilde{\mathbf{X}}_t|\tilde{\mathbf{X}}_{t-1}) \equiv p(\tilde{\mathbf{X}}_t|\tilde{\mathbf{X}}_{t-1}, y_{t-1} = i, y_t = j).$$

We assume that the discrete label is a "hidden" state, so

$$p(\mathbf{Z}_t|\mathbf{X}_t) \equiv p(\mathbf{Z}_t|\tilde{\mathbf{X}}_t)$$

and use these two distributions interchangeably by abuse of notation as in section 3.6 on page 46. In order to implement a model in the CONDENSATION framework it is sufficient

to specify a sampling algorithm for the state evolution density, to occupy step 2 in the algorithm in figure 3.6 on page 44. The procedure for mixed-state CONDENSATION is shown in figure 5.1.

Samples $\mathbf{s}_t^{(n)}$ in the algorithm in figure 3.6 on page 44 are now mixed-state samples

$$\mathbf{s}_t^{(n)} = (\tilde{\mathbf{s}}_t^{(n)}, y_t^{(n)}).$$

Step 1 in the algorithm has selected a base sample $\mathbf{s}_t'^{(n)} = (\tilde{\mathbf{s}}_t'^{(n)}, y_t'^{(n)} = i)$. The following is now used as a replacement for step 2.

Predict by sampling from $p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}_t'^{(n)})$ to choose $\mathbf{s}_t^{(n)}$.

1. Sample transition probabilities
$$P(y_t^{(n)} = j|\mathbf{X}_{t-1} = \mathbf{s}_t'^{(n)}) = T_{ij}(\tilde{\mathbf{s}}_t'^{(n)})$$
to find $y_t^{(n)}$.
2. Sample sub-process density
$$p(\tilde{\mathbf{X}}_t|\mathbf{X}_{t-1} = \mathbf{s}_t'^{(n)}, y_t^{(n)} = j) = p_{ij}(\tilde{\mathbf{X}}_t|\tilde{\mathbf{X}}_{t-1} = \tilde{\mathbf{s}}_t'^{(n)})$$
to find $\tilde{\mathbf{s}}_t^{(n)}$.
3. Store as $\mathbf{s}_t^{(n)} = (\tilde{\mathbf{s}}_t^{(n)}, y_t^{(n)})$.

Figure 5.1: **The sampling algorithm for a mixed-state CONDENSATION model.**

Merely by constructing a model of this form and implementing it within a CONDENSATION tracker, the model transitions can be expected to occur automatically when appropriate. This informal statement proceeds directly from the structure of the process density $p(\mathbf{X}_t|\mathbf{X}_{t-1})$. Each discrete state transition with non-zero probability contributes samples to the state distribution, and several such peaks are maintained while the motion is ambiguous. The weighting applied by the observation density ensures that as soon as one model predicts the object's position significantly more accurately than the others, the samples corresponding to that model will dominate, as shown in figure 5.2.

The simplifying assumption will be made here that

$$T_{ij}(\tilde{\mathbf{X}}) \equiv T_{ij}$$

and the discrete probabilities T_{ij} will be specified by hand for the results which follow.

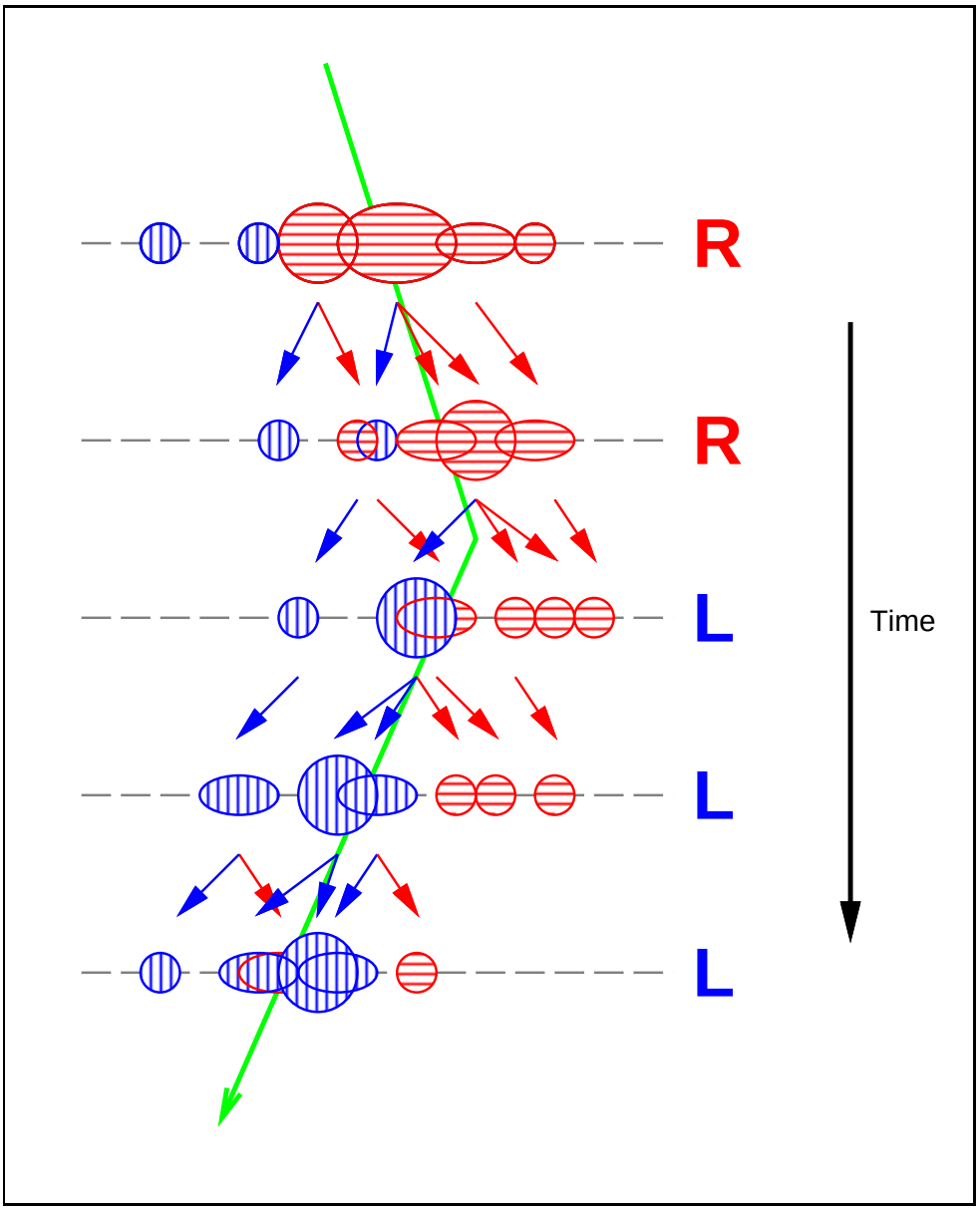


Figure 5.2: **A two-state constant velocity tracker switches state when the direction changes.** *Red ellipses belong to a model with constant velocity to the right, blue to the left. The green arrow shows the true motion of the object. The size of the ellipses indicates the weight assigned to each sample by the observation density. As the object changes direction the blue samples are more accurate predictors, so their weights increase and the model switches from red to blue. The MAP estimate of the current discrete state is shown by the label at the right at each timestep.*

It has previously been found (Blake et al., 1995) that performance of single-state Kalman filters is greatly improved when continuous motion models learned from training data are substituted for hand-coded default models, and in section 5.3 as in chapter 3 we use this methodology to determine the sub-process densities $p_{ij}(\mathbf{X}_t|\mathbf{X}_{t-1})$. It may be the case that learning is also invaluable for the joint process density of a multiple-model filter, and this estimation problem is discussed in section 8.5.

5.2 Modelling a bouncing ball

The power of the mixed-state approach to modelling motion can be demonstrated with a simple example. A mixed-state model was constructed to model a ball which falls vertically and bounces on a table-top. In this situation, there are two discrete states ($N_S = 2$). State $y = 1$ corresponds to the default behaviour of constant acceleration due to gravity, and $y = 2$ is a "bounce event" during which the vertical velocity is reversed and damped by the coefficient of restitution e. It is assumed that a bounce may not be followed immediately by another bounce, and so the model always decays immediately back to state $y = 1$ after a bounce state $y = 2$ (so $T_{21} = 1, T_{22} = 0$). Furthermore we restrict

$$p_{ij}(\ldots) \equiv p_j(\ldots)$$

so the continuous motion model depends only on the new discrete label y_t and not the label y_{t-1} from the previous timestep. The full motion model is shown in figure 5.3 — an extra random variable $\tau \in [0, 1)$ is introduced in the sampling scheme for bounce states $y = 2$ to model the exact moment of the bounce between discrete timesteps (this is an example of the flexibility in model design possible when the only constraint is that it must be possible to sample the sub-process density). The parameters which determine the model are as follows;

- a: constant acceleration due to gravity
- σ_h: standard deviation of vertical position noise added to perturb constant acceleration states $y = 1$
- b: probability of a bounce event occurring (T_{12})
- e: coefficient of restitution for vertical velocity at bounce states $y = 2$
- σ_B: standard deviation of vertical position noise added to perturb bounce states $y = 2$

- σ_v: standard deviation of vertical velocity noise added to perturb bounce states $y = 2$

Constant acceleration a **(**$y_t = 1$**).**

$$h_t = h_{t-1} + v_{t-1} + a + \omega_t$$
$$v_t = h_t - h_{t-1}$$
$$\omega_t \in N(0, \sigma_h)$$

Bounce event ($y_t = 2$**).**

$$h_\tau = h_{t-1} + \tau v_{t-1}$$
$$v_\tau = v_{t-1} + \tau a$$
$$h_t = h_\tau + (1-\tau)v_\tau + \gamma_t$$
$$v_t = -ev_\tau + (1-\tau)a + \nu_t$$
$$\tau \in U[0,1), \quad \nu_t \in N(0, \sigma_v), \quad \gamma_t \in N(0, \sigma_B)$$

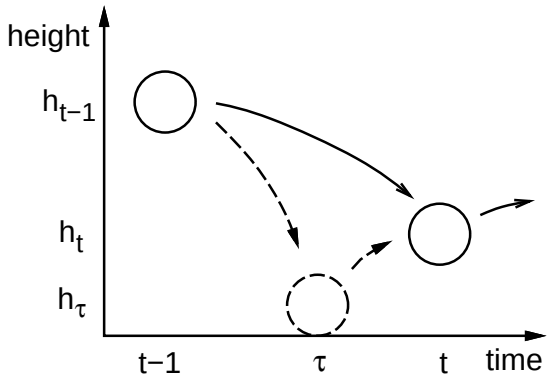


Figure 5.3: ***The two transition modes of a bouncing ball model.***

(note that T_{11} is uniquely determined by $T_{11} = 1 - b$) and the observation density $p(\mathbf{Z}|\mathbf{X})$ is the same as in chapter 3, with $\mu = 20,\ \sigma = 1,\ M = 24$. The curve estimate used for display purposes is calculated in two stages as follows. First the MAP estimate for the discrete

state y_t is found from

$$\begin{aligned}\hat{y}_t &= \arg\max_j P(y_t = j|\mathcal{Z}_t) \\ &= \arg\max_j \sum_{n\in\Upsilon_j} \pi_t^{(n)}\end{aligned}$$

where

$$\Upsilon_j = \{n|\mathbf{s}_t^{(n)} = (\tilde{\mathbf{s}}_t^{(n)}, j)\}.$$

Then the estimate for the shape-space parameter vector is found from the weighted mean of that discrete sample set:

$$\hat{\tilde{\mathbf{X}}}_t = \frac{\sum_{n\in\hat{\Upsilon}} \pi_t^{(n)} \tilde{\mathbf{s}}_t^{(n)}}{\sum_{n\in\hat{\Upsilon}} \pi_t^{(n)}}$$

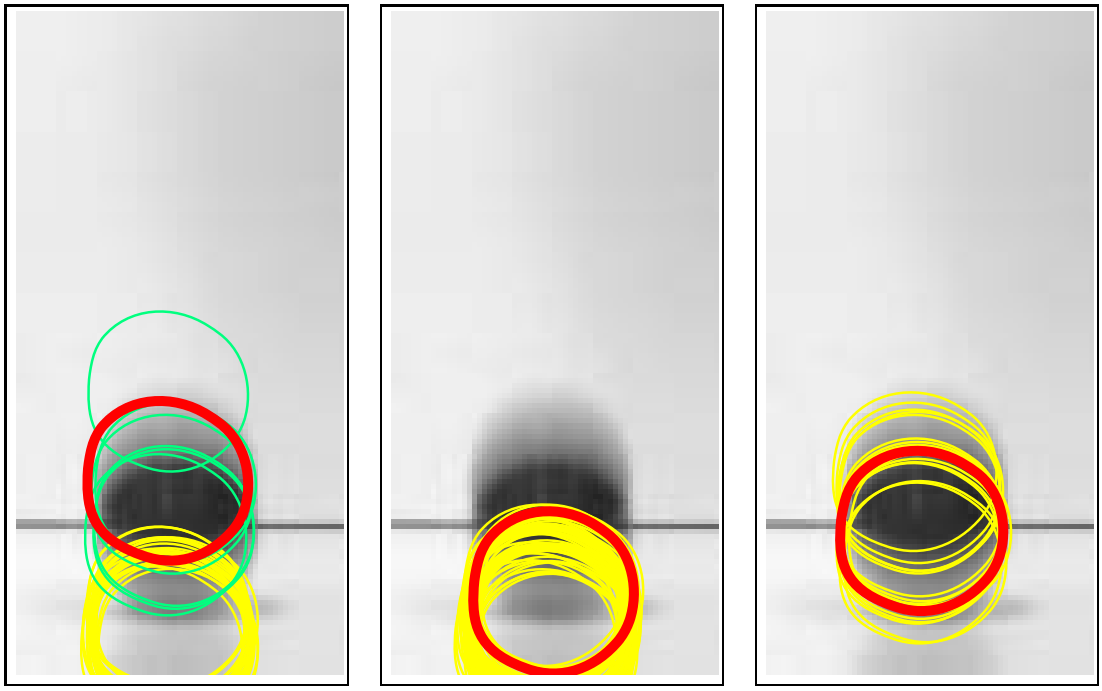
where

$$\hat{\Upsilon} = \{n|\mathbf{s}_t^{(n)} = (\tilde{\mathbf{s}}_t^{(n)}, \hat{y}_t)\}.$$

and it is this mean estimate which is displayed in the figures which follow.

To test the model, a sequence was recorded showing a ball bouncing against a blank background, and tracked twice, once with the mixed-state bounce model, and once with a single-state constant acceleration model (with parameters σ_h and a defined as for the mixed-state model). Both models use an affine shape-space for the ball, with a random walk of small amplitude on the x coordinate and the four shape parameters. The bounce transition parameters were set manually to $b = 0.1$, $e = 0.67$, $\sigma_B = 2$ pixels and $\sigma_v = 10$ pixels/second, and both models used $\sigma_h = 4$ pixels and $a = 4.17$ pixels/second^2. As figure 5.4 shows, the mixed-state model correctly follows the ball when it bounces, while the single-state constant acceleration model continues on a downward path, losing track of the ball. By increasing σ_h to 25 pixels for the single-state model, however, the filter is able to find the ball even after the bounce, as the figure shows.

The true utility of the more accurate mixed-state motion model is demonstrated when background clutter is added to the scene. Now the tracking problem becomes much harder, and a precisely tuned prediction is vital to prevent distractions by clutter features. A second sequence was recorded, showing the ball bouncing in front of a highly cluttered backdrop (figure 5.5). Setting the bounce-transition parameters as before, but reducing σ_h to 3 pixels,



(a) $\sigma_h = 4$ pixels (b) $\sigma_h = 4$ pixels (c) $\sigma_h = 25$ pixels

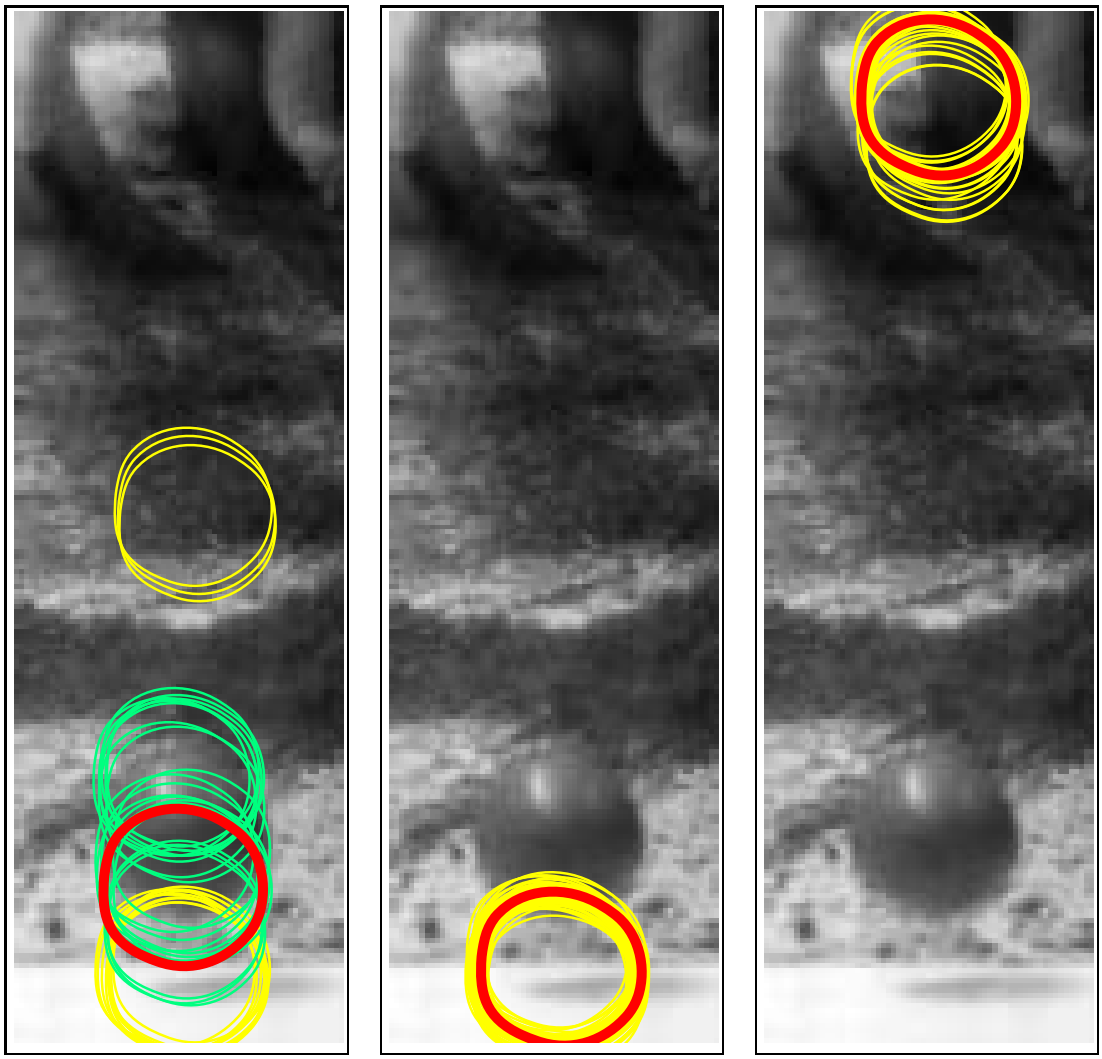
Figure 5.4: **A single-state model requires higher driving noise for successful tracking than an equivalent mixed-state model.** *The red outline shows the estimated position of the ball. Thin curves show high scoring* CONDENSATION *samples from a set of $N = 500$. Figure (a) shows the mixed-state model with $\sigma_h = 4$ pixels; yellow samples are the result of default states $y = 1$ and green are bounce states $y = 2$. Figure (b) shows that the single-state model with $\sigma_h = 4$ pixels loses track of the ball, while in figure (c) the single-state model has $\sigma_h = 25$ pixels and manages to track successfully. Detail of a single field is shown — the ball was released above the top of the visible image. The smearing of the ball is due to motion blur which is not modelled explicitly, so the tracker is equally sensitive to the leading or trailing edge, as shown in figures (a) and (c) respectively.*

the mixed-state model again tracked successfully. The single-state model with $\sigma_h = 3$ pixels tracked the ball on its initial descent as before, but lost lock once more at the moment of the first bounce. This time, increasing σ_h to 8 pixels (or any higher value) caused the single-state model to be distracted by clutter almost immediately, as shown in the figure. While the mixed-state tracker continued to follow the ball throughout the sequence, misalignments are evident in the figure, and chapter 7 discusses one method for reducing this problem.

5.3 A three-state model for freehand drawing

If a set of gestures can be discriminated according to their characteristic motions, they can be included in a mixed-state tracker as discrete states. The tracker will automatically switch into whichever state best describes the motion at a given time, thus improving tracking performance as described in the previous section, and also providing simple gesture recognition as a side-effect. To investigate this approach, a tracker was constructed which could form the basis of the back-end of an interactive drawing package. The goal is to track the outline of a hand as it draws with a pen. Three motions are included — a general line-drawing state, a stationary state, and a specialised "scribbling" state which corresponds to the rapid back-and-forth motion used when filling a region. The intention is that a sketching package driven with the tracked data could draw lines as indicated by the output of the drawing state, and then perform an accurate flood-fill to replace the crude scribbles when a solid block is required and the scribbling state is activated. The stationary state is included because otherwise pauses in motion can be misinterpreted as scribbles of zero amplitude.

A specialised observation density was constructed for the hand tracker to take advantage of the known image properties of a hand drawing with black marker pen on a white page. While thresholding is not reliable enough to separate the hand from the background, especially under variable lighting conditions, it is clear that the hand-coloured pixels are clustered around mid-grey, and the pixels on the page form two clusters, one around white and one around black. A distribution was accordingly constructed to represent this information, consisting of a single Gaussian $N(\mu_f, \sigma_f)$ for foreground pixels and a mixture $N(\mu_{b1}, \sigma_{b1}) + N(\mu_{b2}, \sigma_{b2})$ for the background. In all of the sequences used, the right edge of the hand was in slight shadow, so one set of Gaussian coefficients was used for the left hand-edge and fingers and another for the right hand-edge. The coefficients were set manually as shown in table 5.1.



(a) $\sigma_h = 3$ pixels (b) $\sigma_h = 3$ pixels (c) $\sigma_h = 8$ pixels

Figure 5.5: **Background clutter distracts a single-state tracker.** *The red outline shows the estimated position of the ball. Thin curves show high scoring* CONDENSATION *samples from a set of $N = 1500$. Figure (a) shows the mixed-state model with $\sigma_h = 3$ pixels; yellow samples are the result of default states $y = 1$ and green are bounce states $y = 2$. Although the estimate is not perfectly aligned in this field, there are sufficient samples in the correct alignment that tracking continues successfully. Figure (b) shows the result of using a single-state model with $\sigma_h = 3$ pixels; the ball is tracked in its initial descent, but lock is lost following the first bounce. Increasing the single-state model σ_h to 8 pixels, shown in (c), causes the background clutter to distract tracking almost immediately, before the ball has started to fall. The camera shutter speed is faster in this experiment than for figure 5.4, to reduce motion blur since the addition of clutter makes tracking much harder. The ball was released near the top of the visible image.*

	μ_f	σ_f	μ_{b1}	σ_{b1}	μ_{b2}	σ_{b2}
unshadowed	120	14.1	60	20	225	20
shadowed	90	14.1	60	20	180	20

Table 5.1: **Gaussian coefficients for foreground and background pixels in hand images.** *Values are in units of grey-level intensity.*

The observation density was then designed to reflect these intensity distributions. Each search line is hypothesised to lie half inside the object and half over the background, and it was assumed that along the interior half of a search line, each pixel is independently drawn from the foreground distribution, and along the exterior half each pixel is independently drawn from the background distribution. This leads to the following density:

$$p(\mathbf{Z}|\mathbf{X}) \propto \prod_{m=1}^{M} \left[y_m \left(\prod_{l=-\gamma}^{0} g_f(z_l(s_m)) \right) \left(\prod_{l=0}^{\gamma} g_b(z_l(s_m)) \right) \right]$$

where $z_l(s_m)$ is the greyscale intensity at distance l pixels along the normal to the curve at spline-parameter s_m (negative values indicate the interior of the object),

$$\begin{aligned} g_f(z) &= \mathcal{G}_{\mu_f;\sigma_f}(z) \\ g_b(z) &= \mathcal{G}_{\mu_{b1};\sigma_{b1}}(z) + \mathcal{G}_{\mu_{b2};\sigma_{b2}}(z) \end{aligned}$$

where

$$\mathcal{G}_{\mu;\sigma} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2} \right)$$

and y_m is a penalty constant which is set to 1 if an edge is detected in the direction of the normal at the position s_m and 0.3 otherwise. This observation density is effective, but it would be more satisfactory to learn the parameters from sample images. Also, it would be preferable to model the dependence between adjacent pixel values along the search line, and between the pixel values near the hypothesised curve position and the presence of an edge. It may be possible to learn a model for an entire line, perhaps in the form of a Hidden Markov Model which proceeds from state *Hand* $\rightarrow$ *Edge* $\rightarrow$ *Background*. The problem of designing observation densities is considered further in section 8.2.

A three-state ($N_S = 3$) model was built, where $y = 1$ corresponds to default motions of the hand, $y = 2$ is a stationary state, and $y = 3$ corresponds to a scribbling motion, which is effectively a high-frequency oscillator with its axis close to the horizontal direction. As

before, the restriction $p_{ij} \equiv p_j$ was made. Six training sequences were used to construct the models. The first, 2800 fields (56 seconds) long, is of slow hand-motions against an uncluttered background (a pen was held in the hand, but with the lid on to prevent lines appearing and acting as clutter). The second, 2400 field (48 second) sequence shows faster motions and this time the pen was used to draw lines, so the clutter was severe by the end of the sequence. The last four sequences, between 2000 fields (40 seconds) and 2800 fields (57 seconds) long, show pictures being drawn, and no attempt was made to use gentle hand-motions, so from time to time high image velocities (up to 25 pixels per field) are present. The training sequences were recorded at different times, and although some attempt was made at standardising the conditions, the camera angle and lighting varied slightly from sequence to sequence. Portions of each of the training sequences were hand-labelled as general motions or scribbling motions, and used to train ARP models as described in section 2.5.1 on page 29. A principal component shape-space was built for the hand as in section 2.3 on page 22; the construction of the shape-space and motion models was performed in parallel, since the output of the tracker is a useful diagnostic to find viewing angles of the hand which are not represented in the shape-space. The drawing model was bootstrapped from the training sequences, starting with the clutter-free sequence. The scribbling model was bootstrapped from the fully-trained drawing model. Ultimately the PCA space used was 12-dimensional (translation was explicitly added to a 10-dimensional space of shape variations), constructed from 80 example views. Of the example views, 15 were initially selected by eye and the B-splines constructed using an interactive drawing package. The remaining views were created by taking the slightly misaligned output of a tracker and correcting the B-spline using the drawing package, a less time-consuming job than drawing from scratch. The B-splines are quadratic, and consist of two line segments, of five and seven spans respectively. The transition probability matrix, set manually, was

$$T = \begin{pmatrix} 0.9800 & 0.0015 & 0.0185 \\ 0.0850 & 0.9000 & 0.0150 \\ 0.0050 & 0.0150 & 0.9800 \end{pmatrix}$$

which reflects the composition of a typical drawing — most of the time is spent performing default, "drawing" motions ($y = 1$), but there are occasional pauses ($y = 2$) of short duration, and less frequently somewhat longer periods of scribbling ($y = 3$). Also, scribbling motions are more likely to end with a pause than go straight into a drawing motion.

Since the scribbling motion is an oscillator with small spatial extent, a slight variant of

the standard SDE model was used which allows the means of successive oscillations to differ (Reynard et al., 1996). The concept of a "scribble unit" is introduced, which is a maximal consecutive subsequence of states $\{\mathbf{X}_i\}$ all having discrete label $y = 3$ — informally this is an entire scribbling motion, from start to finish. Each scribble unit is considered to have a fixed mean, but distinct scribbles have distinct means. This is encoded in the model by augmenting scribble-state samples with an extra vector denoting the mean:

$$\mathbf{X}_{\text{scribble}} = (\tilde{\mathbf{X}}, 3, \overline{\mathbf{x}}).$$

The x and y translation components of the mean vector $\overline{\mathbf{x}}$ are initialised to be equal to the current position $(\tilde{\mathbf{X}})_{(x,y)}$ when a scribble unit begins, and $\overline{\mathbf{x}}$ is inherited from the previous sample over the course of that scribble unit. The algorithm to implement the scribble prediction is shown in figure 5.6. When learning the SDE model for the scribble state $y = 3$

To sample from a sub-process density in the scribble state $y = 3$, given a base sample ${\mathbf{s}'}_t^{(n)}$:

1. Fix the scribble mean $\overline{\mathbf{x}}_t^{(n)}$:

   Either: ${\mathbf{s}'}_t^{(n)} = ({\tilde{\mathbf{s}}'}_t^{(n)}, 1)$ or ${\mathbf{s}'}_t^{(n)} = ({\tilde{\mathbf{s}}'}_t^{(n)}, 2)$, so initialise a scribble unit with the learned scribble model mean, translated to the current sample (x, y) position.

   $$(\overline{\mathbf{x}}_t^{(n)})_{(x,y)} = ({\tilde{\mathbf{s}}'}_t^{(n)})_{(x,y)},$$

   or: ${\mathbf{s}'}_t^{(n)} = ({\tilde{\mathbf{s}}'}_t^{(n)}, 3, {\overline{\mathbf{x}}'}_t^{(n)})$, so continue an existing scribble unit

   $$\overline{\mathbf{x}}_t^{(n)} = {\overline{\mathbf{x}}'}_t^{(n)}.$$

2. Generate an IID normal sample vector $\omega_t^{(n)}$

3. Calculate the new shape-space vector

   $$\tilde{\mathbf{s}}_t^{(n)} = A{\tilde{\mathbf{s}}'}_t^{(n)} + (I - A)\overline{\mathbf{x}}_t^{(n)} + B\omega_t^{(n)}$$

4. Store the new sample $\mathbf{s}_t^{(n)} = (\tilde{\mathbf{s}}_t^{(n)}, 3, \overline{\mathbf{x}}_t^{(n)})$

Figure 5.6: **The sampling scheme for a scribbling predictor.** *Each "scribble unit" has a unique mean — this mean is fixed when a model switch $y_{t-1} = 1 \to y_t = 3$ or $y_{t-1} = 2 \to y_t = 3$ occurs, and inherited from the previous sample for transitions $y_{t-1} = 3 \to y_t = 3$.*

from several example sequences, each sequence was first translated to have zero mean. The

stationary model is of the same, variable mean, form as the scribble model.

To test the generality of the model, a new sequence was recorded, which was not used to provide any training data. This was a 1250 field (25 second) sequence showing a drawing of a house (figure 5.7). Because of the high image velocities combined with the high dimensionality of the shape-space, 15000 samples were needed for robust tracking, which runs at approximately 0.33 Hz on an SGI O2 R5000 180MHz workstation. Tracking was accurate throughout, and sample frames are shown in the figure. The classification of motion by model-switching was also accurate, as can be seen from figure 5.8. Occasional short sequences of ambiguous motion are mis-classified, but none longer than a few fields. The onset and end points of scribble gestures are also found surprisingly accurately, although there is a slight lag in some of the switches, which is to be expected since the motion is not unambiguous until at least a quarter of an oscillatory period has elapsed. Figure 5.9 (taken from one of the test sequences) shows that significant motion of the background is discounted by the tracker. The misalignment of the thumb is due to the fact that the test sequence shown was recorded from a camera angle which deviated from those in the other sequences. As a result, there were persistent small misalignments when tracking that sequence using the final drawing model, although the hand was robustly followed throughout. A larger training set may reduce this problem.

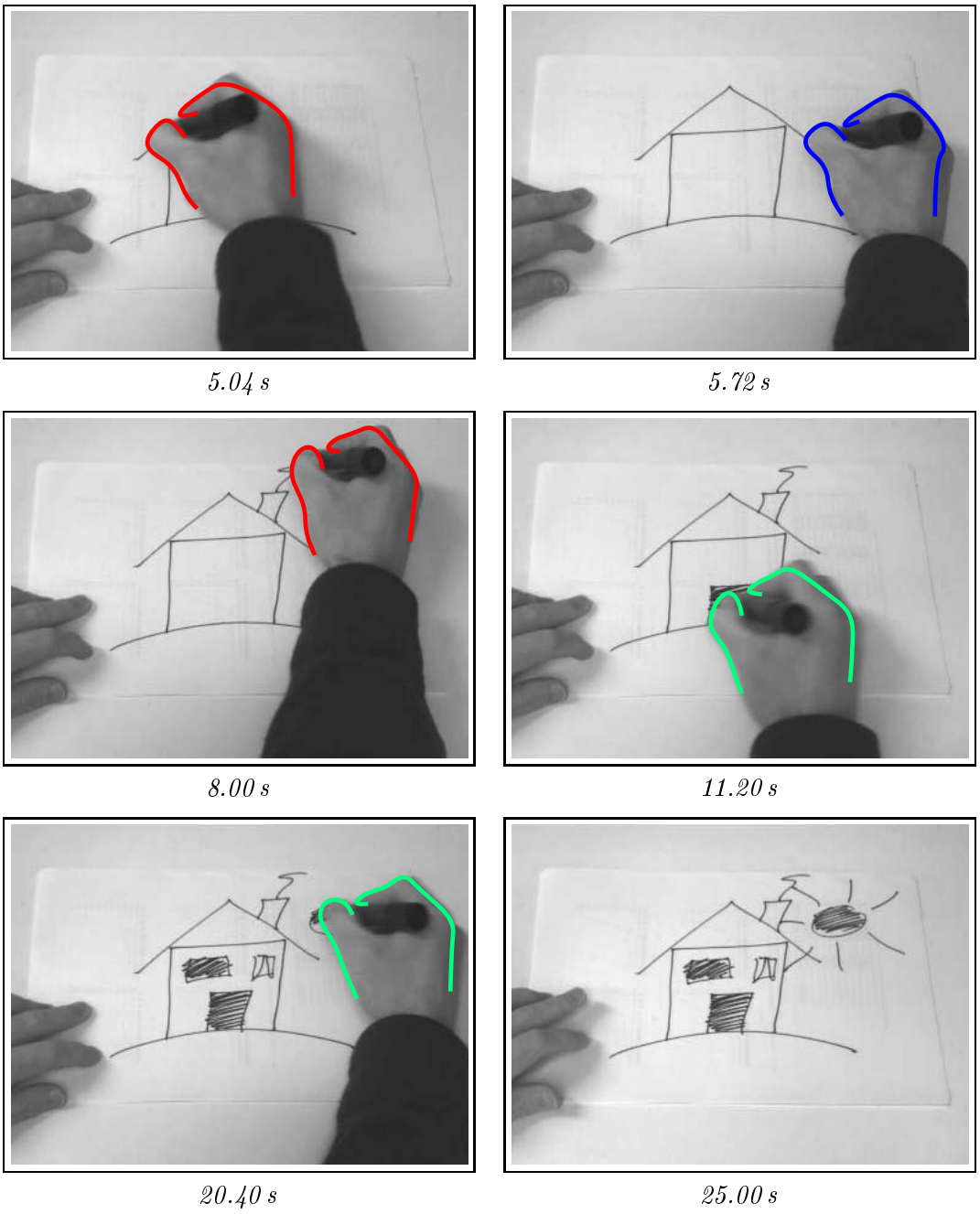


Figure 5.7: **A mixed-state tracker switches between models according to the motion of a drawing hand.** *A 25 second (1250 field) sequence was correctly tracked throughout using $N = 15000$ samples. The contour is drawn in red during default motions, green while scribbling, and blue while stationary. Tracking is accurate, although the shape-space does not quite model the knuckle of the forefinger. Note the subtle shape variation within the 12-dimensional shape-space as the thumb and forefinger move to hold the pen in different positions.*

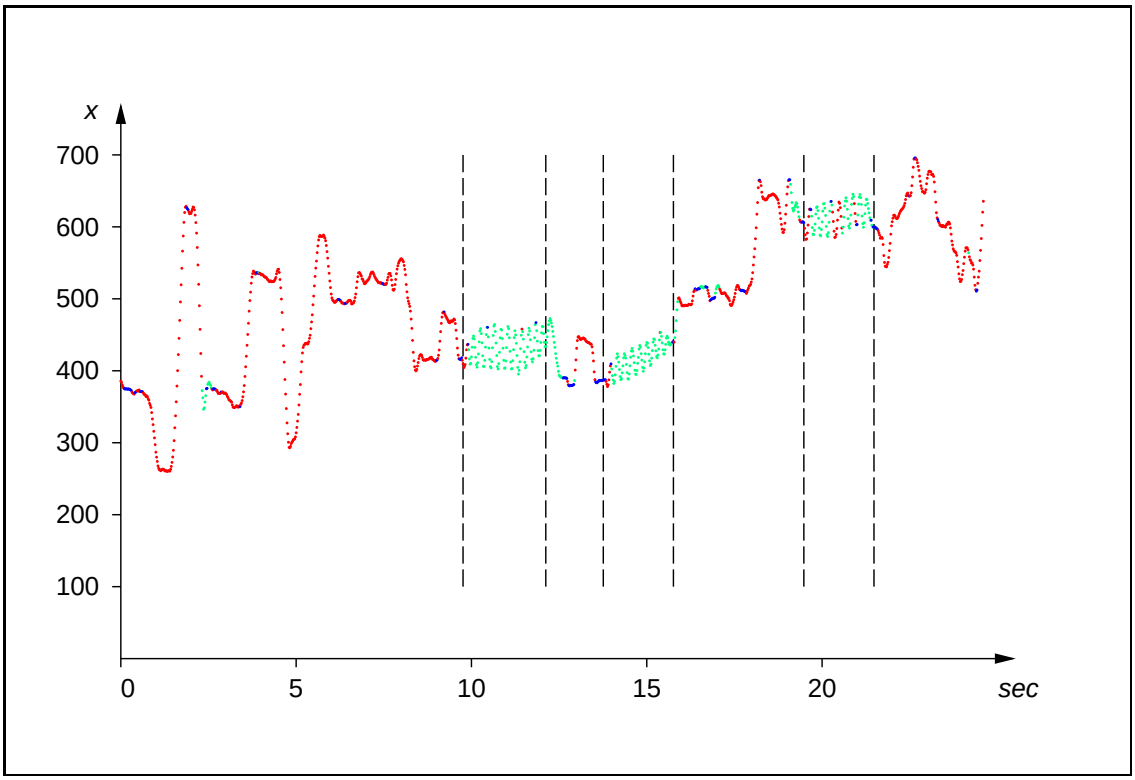


Figure 5.8: **Model switching provides an accurate classification of motion.** *Red dots show drawing motion ($y = 1$), blue dots a stationary hand ($y = 2$) and green dots denote scribbling ($y = 3$). The vertical dashed lines show the times that the scribbling gestures started and ended, found by manual segmentation. The vertical axis shows the x translation coordinate of the hand model in pixels.*

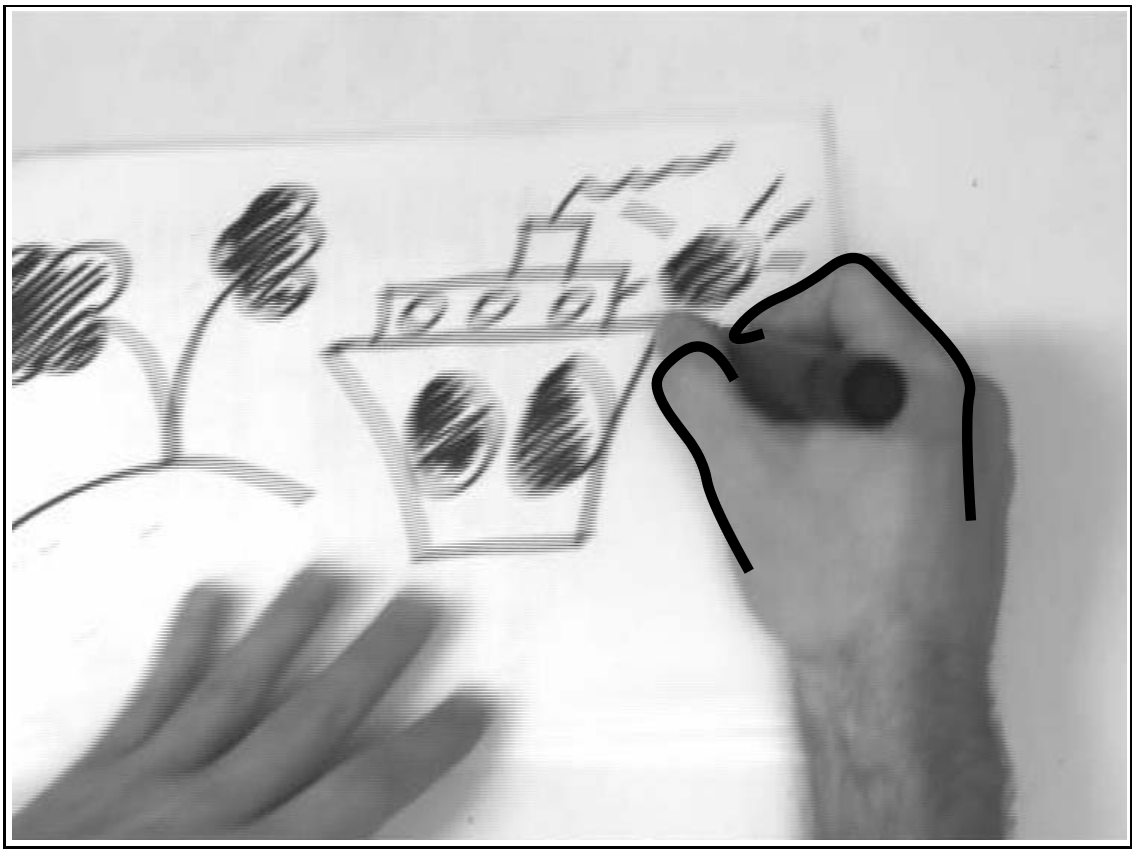


Figure 5.9: **Significant motion of the background does not distract tracking.** *An interlaced image frame shows large inter-field motion of the background as the paper is shifted to accommodate a more comfortable drawing position. Since the tracker does not rely on any explicit background subtraction or inter-field image differencing, tracking is not affected.*

6

Importance sampling and reinitialisation

The tracking framework described in chapter 2 and used in the results presented thus far can be informally characterised as high-level and thereby distinguished from low-level tracking systems. Low-level approaches include "blob trackers" (Wren et al., 1997; Kjeldsen and Kender, 1996) and systems which track sets of point features (Torr and Murray, 1994; Costeira and Kanade, 1995). Blob trackers perform low-level processing, for example colour segmentation, usually on low-resolution (subsampled or decimated) images, and are fast and robust but convey little information other than object centroid. Rigid object deformations can be tracked by matching point correspondences frame to frame (Torr, 1997), but this relies on a rich set of point features on the object of interest, and segmenting the sets of points into coherent objects is challenging. The alternative is to use higher-level information, whether by using outline curves as described in this thesis or modelling objects with specific grey-level templates (Black and Jepson, 1996) which may be allowed to deform (Hager and Toyama, 1996). By including high-level motion models (Blake et al., 1995; Baumberg and Hogg, 1995b) these trackers can follow complex deformations in high-dimensional spaces, as we have seen. The preceding two chapters have highlighted a tradeoff between speed and robustness, however. Kalman-filter based contour trackers which run in real time are very susceptible to distraction by clutter, and correlation-based systems are vulnerable to changes in object appearance and lighting, and rapidly slow down as the space of deformations increases in complexity. The CONDENSATION trackers described in

chapters 3 and 5 are highly robust to clutter but thereby sacrifice real-time performance, and this is evident also in other clutter-resistant systems (Lowe, 1992). The high-level approaches also tend to economise on processing time by searching only those regions of the image where the object is predicted to be. This diminishes robustness to unmodelled motions, and also precludes natural extensions of the trackers for the initialisation stage when the prior for object position may be broadly distributed over the whole image. The difficulty of initialisation is compounded when the dimension of the tracking space increases, since it is rapidly impractical to perform an exhaustive search for the object.

This chapter describes an extension to CONDENSATION, ICONDENSATION, to bridge the gap between low-level and high-level tracking approaches. An implementation is demonstrated which uses colour segmentation to find skin-coloured blobs in a decimated image, and feeds this information to a contour tracker specialised for hands. The techniques used, however, apply to the general sensor fusion problem of augmenting a tracker operating with one measurement modality to use information from an auxiliary measurement source. Tracking is achieved by extending the CONDENSATION filter to incorporate the statistical technique of "Importance Sampling" (Ripley, 1987). Importance sampling offers a mathematically principled way of directing search, combining prediction information based on the previous object position and motion with any additional knowledge which may be available from auxiliary sensors. This *combination* confers robustness to temporary failures in one of the measurement processes, and allows the tracker to take advantage of the distinct qualities of different information sources. In the hand-tracking system presented here, for example, colour segmentation allows rapid initialisation and robust tracking of gross motions, while the contour tracker gives fine-scale position and shape information as well as maintaining lock on the object when colour blobs merge or momentarily disappear. The hand-tracker based on ICONDENSATION, while less able to represent subtle hand gestures than that of the previous chapter, operates comfortably in real time (30 or 60 Hz) on a desktop workstation (SGI O2 R5000 180SC). The speed improvement is due partly to a reduction in the required number of samples as a result of using importance sampling, and partly to a careful implementation which is discussed in section 6.2.

6.1 Importance sampling

In the standard formulation of the CONDENSATION algorithm, positions of samples $\mathbf{s}_t^{(n)}$ are fixed in the prediction stage using only the previous approximation to the state density $\{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)})\}$ and the motion model $p(\mathbf{X}_t|\mathbf{X}_{t-1})$. The portions of state-space (and thus the image $\mathbf{Z}_t$) which are to be examined in the measurement stage are therefore determined before any measurements are made. This is appropriate when the sample-set approximation to the state density is accurate. In principle, as the state density evolves over time, the random nature of the motion model induces some non-zero probability everywhere in state-space that the object is present at that point. With a sufficiently good sample-set approximation this would tend to cause all areas of state-space to lie near some samples, so even motions which were extremely unlikely given the model would be detected, and could therefore be tracked. In practice, however, the finite nature of the sample-set approximation means that all of the samples will be concentrated near the most likely object positions as in figure 6.1. There may be several such clusters corresponding to multiple hypotheses, but

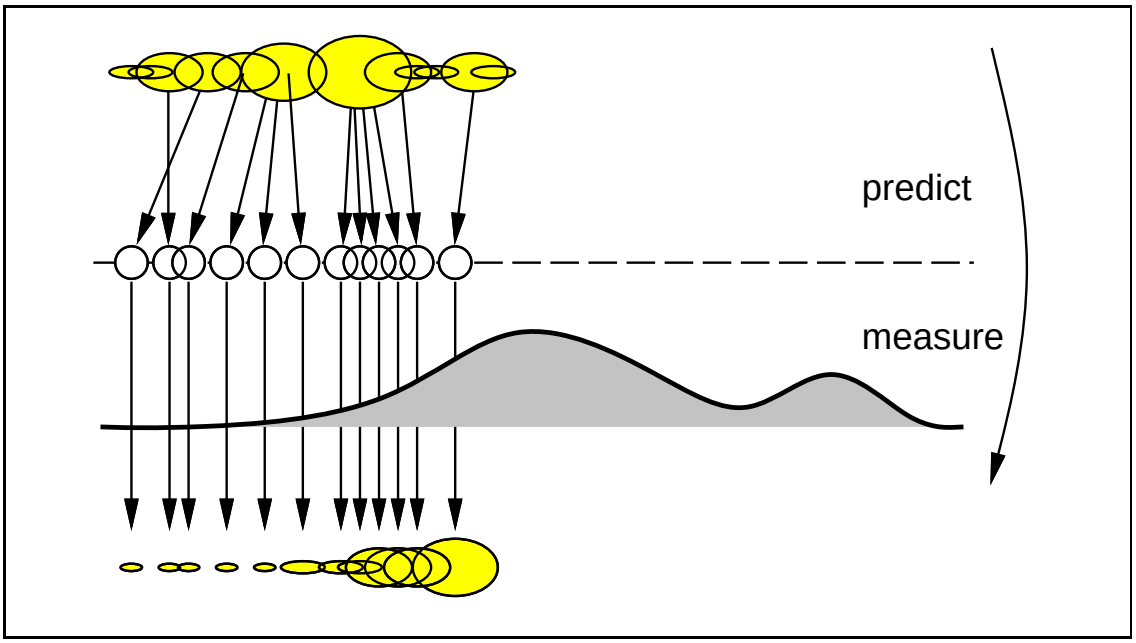


Figure 6.1: **Unexpected motion results in an inefficient sample distribution.** *The motion model has predicted that the object would remain at the left, but it has actually moved right. As a result most of the samples lie away from peaks of the observation density. Increasing the number of samples N would increase the number of samples near the peak of the observation density, making tracking more likely to continue successfully, but at the cost of increasing computational load.*

in general each cluster will be fairly localised, which in fact is precisely the behaviour which permits an efficient discrete representation of high-dimensional state spaces. The result is that large areas of state-space contain no samples at all. In order to robustly track sudden movements the process noise of the motion model must be artificially high, thus increasing the extent of each predicted cluster in state-space. To populate these larger clusters with enough samples to permit effective tracking, the sample-set size must be increased, and the algorithm therefore runs more slowly. This was evident in section 5.3 on page 85, when $N = 15000$ samples were used for the hand-tracker to permit robust tracking in the face of sudden large accelerations of the hand. A much smaller number of samples ($N = 1500$) is adequate to track the deformations of the tracker in section 5.3 as long as hand motions are slow, and the larger sample-set size is used primarily to perform search over the image in the case of sudden unexpected hand motions. Various techniques have been proposed to improve the efficiency of the representation in random sampling filters (Gordon et al., 1993; Gordon and Salmond, 1995), and these are discussed in section 8.3, but to our knowledge none have been advanced which draw on information available from alternative sensors.

Importance sampling (Ripley, 1987) is a technique developed to improve the efficiency of Monte-Carlo methods, of which factored sampling is one. It applies when auxiliary knowledge is available in the form of an importance function $g(\mathbf{X})$ describing which areas of state-space contain most information about the posterior. The idea is then to concentrate samples in those areas of state-space by generating sample positions $\mathbf{s}^{(n)}$ from $g(\mathbf{X})$ rather than sampling fairly from the prior $p(\mathbf{X})$. The desired effect is to avoid as far as possible generating any samples which have low weights, since they are "wasted" in the factored sampling representation as they provide a negligible contribution to the posterior. A correction term f/g must be added to the sample weights giving

$$\pi^{(n)} = \frac{f(\mathbf{s}^{(n)})}{g(\mathbf{s}^{(n)})} p(\mathbf{Z}|\mathbf{X} = \mathbf{s}^{(n)}) \quad \text{where} \quad f(\mathbf{s}^{(n)}) \equiv p(\mathbf{X} = \mathbf{s}^{(n)}) \tag{6.1}$$

to compensate for the uneven distribution of sample positions. This correction term ensures that, as $N \to \infty$, importance sampling has *no effect* on the consistency of the particle-set representation; the desired posterior is still correctly approximated. A high value of $g(\mathbf{X})$ implies a high probability that an importance-sampled particle will be chosen at the position $\mathbf{X}$ (a sampling bias towards $\mathbf{X}$, that is), and so $\pi^{(n)}$ for that particle is downweighted by the factor of $g(\cdot)$ in the denominator of (6.1) to correct this bias. Conversely, a high value of $f(\mathbf{X})$ implies a high probability of placing a particle at $\mathbf{X}$ in the true posterior

representation, and the $f(\cdot)$ in the numerator of (6.1) reflects this.

Any importance function could be chosen (subject to some weak constraints given in (Doucet, 1998)) and if N is sufficiently large then $\tilde{p}(\mathbf{X}|\mathbf{Z})$ will be a good approximation to $p(\mathbf{X}|\mathbf{Z})$. The purpose of importance sampling is to reduce the variance of the factored sampling procedure as an estimator for $p(\mathbf{X}|\mathbf{Z})$ given fixed N and so improve the accuracy of $\tilde{p}(\mathbf{X}|\mathbf{Z})$ when N is small. Since samples are drawn from $g(\mathbf{X})$ it plays the part of a probability density, but note that it does not necessarily correspond to the distribution of any particular random variable.

A typical Bayesian approach to sensor fusion would be to combine measurements from the various sensors in the $\mathbf{Z}_t$, weighted according to their inverse variances. This is only possible, however, when the statistical dependencies between the measurements are understood, and in practice it is often assumed that sensors produce independent measurements. The application in this chapter combines measurements made with different modalities but from the same underlying image, so it is expected that such an independence assumption would be invalid. Instead of the traditional sensor fusion approach, therefore, importance sampling allows measurements to be combined in a Bayesian framework even when no knowledge at all is available about their dependence. The tradeoff is that the symmetry between sensors is broken, since the measurements used to define the importance function are not included in the overall model, and this may result in some genuine independent information being discarded.

Importance sampling can be applied in the context of CONDENSATION sampling, and we denote this extension ICONDENSATION. Now the importance function at time t is denoted $g_t(\mathbf{X}_t)$. In standard CONDENSATION as described in section 3.4 on page 40, sample positions are drawn from the density

$$\begin{aligned} f_t(\mathbf{s}_t^{(n)}) &\equiv \tilde{p}(\mathbf{X}_t = \mathbf{s}_t^{(n)}|\mathcal{Z}_{t-1}) \\ &= \sum_{j=1}^{N} \pi_{t-1}^{(j)} p(\mathbf{X}_t = \mathbf{s}_t^{(n)}|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(j)}). \end{aligned} \tag{6.2}$$

Note that while the sample set $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}$ provides a discrete point-representation of a distribution, the prediction density $\tilde{p}(\mathbf{X}_t|\mathcal{Z}_{t-1})$ is a mixture of continuous density kernels shaped by $p(\mathbf{X}_t|\mathbf{X}_{t-1})$, representing the dynamical model. Instead of sampling from $\tilde{p}(\mathbf{X}_t|\mathcal{Z}_{t-1})$, samples $\mathbf{s}_t^{(n)}$ can instead be drawn from $g_t(\mathbf{X}_t)$ and then the weights need to

be redefined as

$$\pi_t^{(n)} = \frac{f_t(\mathbf{s}_t^{(n)})}{g_t(\mathbf{s}_t^{(n)})} p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t^{(n)}). \tag{6.3}$$

The effect of the correction ratio is to preserve the information about motion coherence which is present in the dynamical model. Although the samples are positioned according to g_t, the distribution approximated by $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}$ still generates $p(\mathbf{X}_t|\mathcal{Z}_t)$. Importance sampling is again intended to improve the efficiency of the sample-set representation, but does not change the probabilistic model (figure 6.2). It should be noted that (6.2) imposes

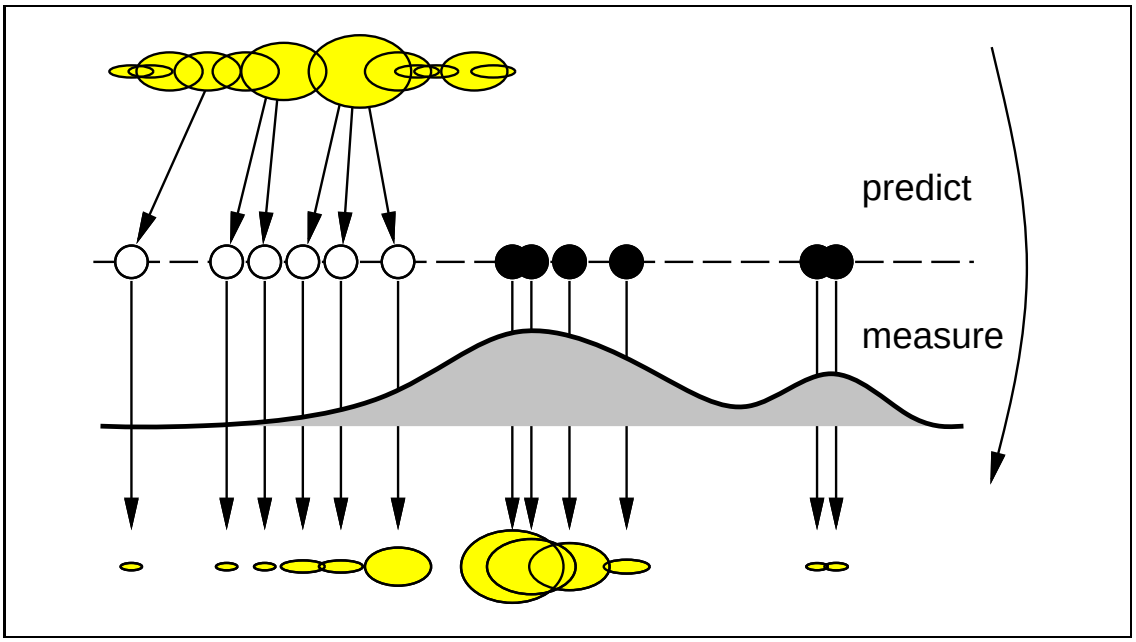


Figure 6.2: **Importance sampling improves the efficiency of the sample-set representation.** *The motion model has predicted that the object would remain at the left, and positioned white samples accordingly. In fact the object has moved right, and the black samples are positioned according to an importance density g which reflects this. The discrete set contains the same number of samples N as in figure 6.1, but now the the approximation to the density is more accurate.*

a restriction on the form of dynamical model which can be used; for CONDENSATION it is enough to be able to sample from $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ but in the ICONDENSATION algorithm this density must also be evaluated. The motion models in this thesis use Gaussians (chapter 2) or mixtures of Gaussians (chapter 5) for this process density, and so evaluation is straightforward. The sum in (6.2) must be evaluated in (6.3) for each $n = 1, \ldots, N$, which changes the complexity of the algorithm from $O(N)$ to $O(N^2)$. While this is a theoretical

disadvantage, it has little effect in practice for the application described, since the time expended on the calculation of (6.2) in an efficient implementation, for practical values of N, is dwarfed by other stages of the computation.

Variations of the importance sampling scheme

In practice the importance function g_t will derive from a measurement process, so it may be imperfect and thereby omit some likely peaks of $p(\mathbf{Z}_t|\mathbf{X}_t)$. It is therefore prudent to generate some samples using standard factored sampling and some by importance sampling using g_t. As long as $\tilde{p}(\mathbf{X}_t|\mathcal{Z}_{t-1})$ and g_t do not simultaneously fail to predict the object state, tracking will succeed.

It may be advantageous to augment the dynamical model to include some probability q of reinitialisation — repositioning the object according to a prior which is independent of past history $\mathcal{Z}_t$. This allows a tracker to lock on to an object entering the scene, or rediscover an object which has been lost due to gross failures of measurements, perhaps because the object moved while it was entirely occluded. The amended model is of the form

$$\tilde{p}'(\mathbf{X}_t|\mathcal{Z}_{t-1}) = (1-q)\tilde{p}(\mathbf{X}_t|\mathcal{Z}_{t-1}) + qp(\mathbf{X}_t)$$

where $p(\mathbf{X}_t)$ is the required initialisation prior. This is an application of mixed-state CONDENSATION with two discrete states, and in later sections a hand-tracker model will be demonstrated which is augmented to include a further two states. A mixed-state model can be included in the importance sampling scheme by choosing with probability $1-q$ to generate samples as before (using importance sampling with probability r and standard factored sampling with probability $1-q-r$) and with probability q to generate $\mathbf{s}_t^{(n)}$ by sampling directly from a prior distribution $p(\mathbf{X}_t)$. In the absence of another initialisation prior, the importance function, suitably normalised, can be used, so $p(\mathbf{X}_t) \propto g_t$. A complete sampling algorithm is shown in figure 6.3.

6.2 Experiments with a real-time hand-tracker

The framework of ICONDENSATION applies in principle to any parametric representation of objects and their motion, and any form of importance function g_t. The remainder of this chapter describes a specific implementation of a real-time hand tracker, combining blob-

Iterate

From the "old" sample set $\{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}),\ n = 1, \ldots, N\}$ at time-step $t-1$, construct a "new" sample set $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)}),\ n = 1, \ldots, N\}$ for time t. The importance function $g_t(\mathbf{X}_t)$ for time t is assumed to be known at this stage.

Construct the n^{th} of N new samples as follows:

1. **Choose** the sampling method by generating a random number $\alpha \in [0, 1)$, uniformly distributed.

2. **Sample** from the prediction density $\tilde{p}'(\mathbf{X}_t|\mathcal{Z}_{t-1})$ as follows:

   (a) **If** $\alpha < q$ use the initialisation prior. Choose $\mathbf{s}_t^{(n)}$ by sampling from $g_t(\mathbf{X}_t)$ and set the importance correction factor $\lambda_t^{(n)} = 1$.

   (b) **If** $q \leq \alpha < q + r$ use importance sampling. Choose $\mathbf{s}_t^{(n)}$ by sampling from $g_t(\mathbf{X}_t)$ and set $\lambda_t^{(n)} = f_t(\mathbf{s}_t^{(n)})/g_t(\mathbf{s}_t^{(n)})$, where

   $$f_t(\mathbf{s}_t^{(n)}) = \sum_{j=1}^{N} \pi_{t-1}^{(j)} p(\mathbf{X}_t = \mathbf{s}_t^{(n)} | \mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(j)}).$$

   (c) **If** $\alpha \geq q + r$ use standard CONDENSATION sampling. Choose a base sample $\mathbf{s}_t'^{(n)}$ as in step 1 in figure 3.6 on page 44, then choose $\mathbf{s}_t^{(n)}$ by sampling from $p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}_t'^{(n)})$ and set $\lambda_t^{(n)} = 1$.

3. **Measure** and weight the new position in terms of the image data $\mathbf{Z}_t$ and the importance sampling correction term:

   $$\pi_t^{(n)} = \lambda_t^{(n)} p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t^{(n)})$$

   then normalise multiplicatively so that $\sum_n \pi_t^{(n)} = 1$ and store as $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}$.

Figure 6.3: ICONDENSATION: CONDENSATION **with importance sampling and reinitialisation.**

tracking with a contour model. First, a crude colour-segmentation technique is described which is used to construct the importance function $g_t(\mathbf{X}_t)$.

6.2.1 Finding skin-coloured blobs

Previous researchers (Kjeldsen and Kender, 1996; Graf et al., 1996) have noted that human skin can be effectively distinguished in a typical office scene using colour segmentation, and so this method is adopted here. Training images of hands are used to construct a colour discriminant based on a Gaussian prior in (r, g, b) space which expresses the probability that a given pixel is skin-coloured. The prior is clipped with an intensity threshold to ensure that very dark pixels are not classified as skin, giving the discriminant $d_C(r, g, b)$ as:

$$d_C = \begin{cases} 10000 & \text{if } r+g+b < 192 \\ \begin{pmatrix} r' & g' & b' \end{pmatrix} \begin{pmatrix} 0.576696 & 0.761244 & -0.176944 \\ 0.761244 & 1.374758 & -0.224989 \\ -0.176944 & -0.224989 & 0.061377 \end{pmatrix} \begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} & \text{otherwise} \end{cases}$$

$$\text{where} \quad \begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} = \begin{pmatrix} r - 110.861 \\ g - 89.5033 \\ b - 70.8975 \end{pmatrix}$$

where smaller values of d_C imply a pixel is more likely to be skin-coloured. Blob-detection is performed by taking the input image and decimating to give 32×24 pixels per field, then evaluating the colour discriminant for each pixel. A 2×2 moving block average is applied to the image to reduce noise, and then pixels are grouped using a flood-fill with hysteresis (Foley et al., 1990), beginning a fill when $d_C < 27.5$ and filling all contiguous pixels where $d_C < 35$. An example image and the segmented output are shown in figure 6.4. The technique has been found to be very effective in separating skin colour from background, and works over the variation in lighting conditions from day to night in our office. Let B be the number of blobs detected, then the mean of each blob is computed as a coordinate $\mathbf{b}'_k$ in the original image, and a two-dimensional importance function $\tilde{g}_t$ is then defined to be a mixture of Gaussians over $\mathbb{R}^2$

$$\tilde{g}_t(\mathbf{x}_{trans}) = \sum_{k=1}^{B} \delta_k N(\mathbf{b}_k, \Sigma_B)$$

where $\mathbf{b}_k = \mathbf{b}'_k + \bar{\mathbf{x}}_B$, and $\bar{\mathbf{x}}_B$ and Σ_B are the mean and covariance respectively of the offset from the blob position to the centroid of the contour describing the hand. These are learned by following a user's hand against an uncluttered desk using a CONDENSATION tracker and

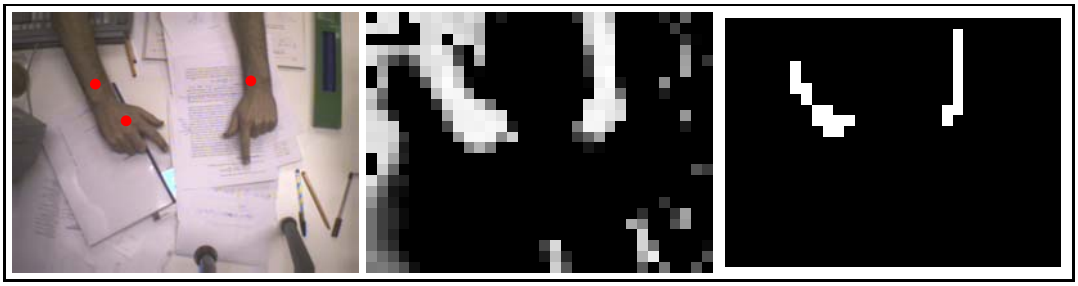


Figure 6.4: **Colour segmentation allows the detection of skin-coloured blobs.** *Circles on the input frame (left) show the centres of blobs detected using colour-segmentation and a flood-fill. The output of the colour discriminant on a* 32×24 *pixel subsampled field of the original image is shown scaled (middle) so that white corresponds to a high probability of skin-colour. The output after convolution and blob-detection (right) shows white areas belonging to blobs.*

comparing the output of the blob segmentation with the centroid of the tracked contour. The mixture weights δ_k will be discussed in the next section. A *hybrid* sampling scheme is now required since the importance function is defined only over translations, and this is outlined below.

6.2.2 A contour tracker for hands

As in section 2.5 on page 28, a second-order state-space is used to represent the hand, but now the parameterisation is non-linear, giving

$$\mathbf{X}_t = \begin{pmatrix} \mathbf{x}_{t-1} \\ \mathbf{x}_t \end{pmatrix} \quad \text{where} \quad \mathbf{x}_t = (x_t, y_t, \theta_t, r_t, \chi_t)^T.$$

Here $\mathbf{x}_t$ represents a Euclidean similarity transform applied to a spline template $\mathbf{Q}_0$ so

$$\mathbf{Q} = \begin{pmatrix} x_t \mathbf{1} \\ y_t \mathbf{1} \end{pmatrix} + r_t \begin{pmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{pmatrix} \begin{pmatrix} \chi_t \mathbf{Q}_0^x \\ \mathbf{Q}_0^y \end{pmatrix}.$$

The extra parameter $\chi_t \in \{-1, 1\}$ is a discrete label which determines whether the template is left- or right-handed — $\mathbf{Q}_0$ is reflected about the y-axis for the right hand. The left–right parameter χ_t is constant according to the motion model, and can only change as a result of a reinitialisation, and so the tracker effectively uses a four-state model (tracking or reinitialising either the left or the right hand) shown in figure 6.5.

The motion model is a second-order ARP where each of the four dimensions of the similarity is modelled by an independent one-dimensional oscillator. The oscillators are specified (Blake and Isard, 1998) by parameters defining a damping constant β, a natural

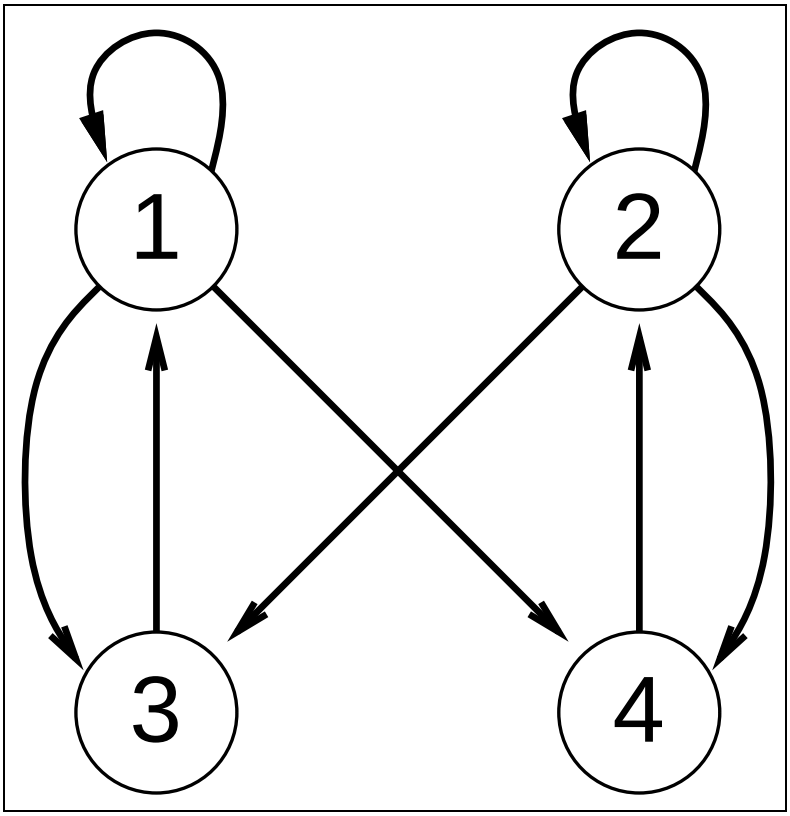


Figure 6.5: **The hand-tracker uses a four-state discrete/continuous motion model.** *States 1 and 2 denote tracking the left and right hand respectively, and states 3 and 4 correspond to initialising a sample to start tracking with either the left or the right model.*

frequency f and a root-mean-square average displacement ρ as described in section 2.5 on page 28. Sensible default parameters for the oscillators are chosen by hand and are shown in table 6.1.

	β (s^{-1})	f (Hz)	ρ
x	3	0	100 pixels
y	3	0	100 pixels
θ	5	0	0.5 rad
r	5	0	0.1

Table 6.1: **Oscillator coefficients for a hand-tracker.**

As explained in section 6.2.1, the importance function $\tilde{g}_t$ is defined here only over the space of x–y translations, being the output of a crude blob-tracker. The state-space decomposes into a translation subspace $\mathbf{x}_t^{\mathcal{T}} = (x_t, y_t)^T$ and a deformation subspace $\mathbf{x}_t^{\mathcal{D}} =$

$(\theta_t, r_t, \chi_t)^T$. Modifications to steps 2(a) and 2(b) of figure 6.3 can now be made to implement a hybrid sampling scheme. For initialisation in step 2(a), the translation component $\mathbf{s}_t^{(n)\mathcal{T}}$ is sampled from $\tilde{g}_t$ and the deformation component $\mathbf{s}_t^{(n)\mathcal{D}}$ is sampled from a fixed prior $p^{\mathcal{D}}(\mathbf{x}^{\mathcal{D}})$ which is taken to be Gaussian in θ_t and r_t and assigns equal probabilities for left and right to χ_t, then $\mathbf{s}_t^{(n)} = \mathbf{s}_t^{(n)\mathcal{T}} \oplus \mathbf{s}_t^{(n)\mathcal{D}}$. The hybrid importance sampling step 2(b) proceeds as follows. First generate a sample $\mathbf{s}_t'^{(n)} = \mathbf{s}_t'^{(n)\mathcal{T}} \oplus \mathbf{s}_t'^{(n)\mathcal{D}}$ using standard CONDENSATION sampling as in step 2(c). Then choose $\mathbf{s}_t^{(n)\mathcal{T}}$ by sampling from $\tilde{g}_t$ and set $\mathbf{s}_t^{(n)} = \mathbf{s}_t^{(n)\mathcal{T}} \oplus \mathbf{s}_t'^{(n)\mathcal{D}}$. Finally the importance correction factor $\lambda_t^{(n)}$ is replaced by $\lambda_t^{(n)\mathcal{T}} = f_t^{\mathcal{T}}(\mathbf{s}_t^{(n)})/\tilde{g}_t(\mathbf{s}_t^{(n)})$, where

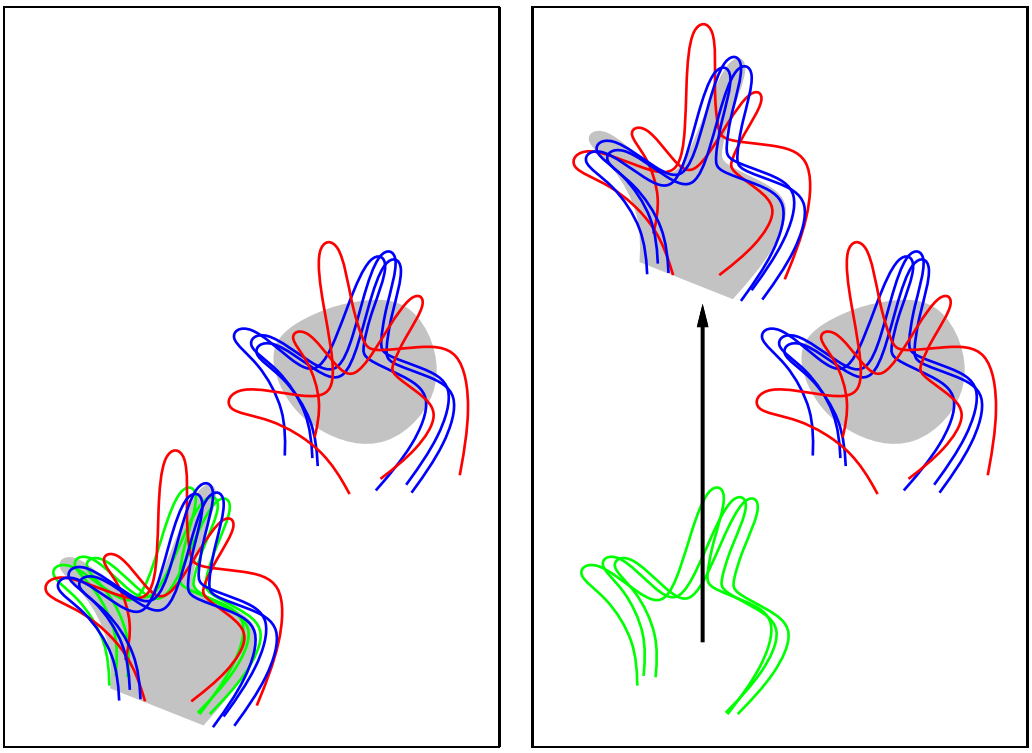
$$f_t^{\mathcal{T}}(\mathbf{s}_t^{(n)}) = \sum_{j=1}^{N} \pi_{t-1}^{(j)} p(\mathbf{X}_t^{\mathcal{T}} = \mathbf{s}_t^{(n)\mathcal{T}} | \mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(j)}).$$

It remains to specify the mixture weights δ_k for $g_t(\mathbf{X}_t)$. One reasonable choice is to set $\delta_k = 1/B$, and apportion samples equally in the vicinity of each blob. Since the motivation for importance sampling is to avoid generating samples with low weights, it may be preferable to increase δ_k for blobs which are near to many predicted sample positions. This can be done approximately by setting $\delta_k \propto f_t(\mathbf{b}_k)$, and later results are produced using these weights.

The prior distribution over translation for reinitialisation is chosen to be the distribution obtained by sampling from g_t with $\delta_k = 1/B$ for all k. The parameters θ and r are chosen from a suitable Gaussian prior density, with parameters set by hand, and χ has an equal chance of being left- or right-handed. When $k = 0$ no importance or reinitialisation samples are generated, and all of the computing time is spent on standard CONDENSATION samples. The sampling scheme is illustrated diagrammatically in figure 6.6.

6.2.3 The measurement process

Having detailed the dynamical model it remains to specify the measurement density $p(\mathbf{Z}|\mathbf{X})$. The measurement density is approximated by examining a set of points $\mathbf{z}_m$ for $m = 1 \ldots M$ which lie on the curve outline, where the normal to the curve at $\mathbf{z}_m$ is $\mathbf{n}_m$. First of all, edge-operator convolutions are taken at $\mathbf{z}_m$ in the x and y directions, and the dot product of these is taken with the normal direction $\mathbf{n}_m$ to find a directed edge strength which is scaled and interpreted directly as a log probability p_m. This is similar to the edge-detection procedure in section 3.6.2 with a normal-line length $\mu = 1$ pixel, but more efficient to compute. When the edge strength is above a certain threshold an additional colour calculation is made to examine the pixels just inside the contour. This increases p_m when the area inside the



(a) (b)

Figure 6.6: **Samples are divided between standard** CONDENSATION **samples, importance samples, and reinitialisation samples.** *Green samples are standard* CONDENSATION *samples, blue samples are importance samples and red samples are reinitialisation samples. The illustration shows two blobs in each image which are roughly skin-coloured; blue and red samples are clustered around each blob. Note that the red reinitialisation samples use a broad prior for scale and rotation, whereas the blue importance samples use motion coherence and predict a tighter distribution for scale and rotation near that of the the previous timestep. In (a) the hand is near the predicted position, so the green* CONDENSATION *samples are clustered around it. In (b), however, the hand has moved unexpectedly, so all the green samples are far from it, which would cause a standard* CONDENSATION *tracker to fail. An* ICONDENSATION *tracker will succeed due to the presence of importance samples near the hand-coloured blob.*

curve scores highly according to the skin-colour discriminant, and decreases it when it scores poorly. Independence of the measurement points is assumed, so the density is given by

$$\log p(\mathbf{Z}|\mathbf{x}) = \text{const} + \sum_m p_m.$$

Constants are set manually, and the density is somewhat ad hoc; determining a more rigorous measurement density, possibly learned from training images, is deferred to future research. The general problem of designing suitable rigorous observation densities is considered in section 8.2.

6.2.4 Speed enhancements

Importance sampling has been presented as a mechanism to use complementary sources of visual information to choose an effective set of positions in state-space for a finite set of N samples. Given the constraint of real-time operation, a certain amount of care in the detailed implementation is necessary in order to maximise N. Much of this consists simply of standard code optimisation, but some parts of the algorithm can be redesigned for greater efficiency. Firstly, the deterministic base-sampling technique described in section 3.4 on page 40 is used and this cuts down on the prediction calculations, since it allows predictions from a given base sample to be made consecutively. This offers a saving in calculating the deterministic portion of the prediction, which need only be done once per base sample. Practical experience shows that this can lead to significant economy, since typically only 10% of samples may have high enough weight to be used as a base. Experiments were also done to precompute a large number of vectors of random Gaussian noise $\mathbf{w}_k$ so that the shaped noise terms $B\mathbf{w}_k$ from (2.12) on page 28 could be computed offline, but it was found that on the experimental system used, which has a slow memory system relative to its floating point performance, this modification resulted in reduced tracking speed, and it was abandoned.

Software profiling shows that most of the tracker's computation is expended, as might be expected, on calculating the measurement density. It has been found that a significant speed improvement can be gained by presorting the measurement points in raster order before performing the convolution and colour-segmentation calculations of section 6.2.3. This has the advantage, as for the base samples, that identical measurement points are processed consecutively, which cuts down on the number of convolutions (typically the number of distinct measurement points is just over half of the total number of points). Clearly, per-

forming the sort generates a large overhead, and in fact profiling reveals that typically 30% of computation time is spent sorting compared with about 50% making the measurements. This might seem to almost cancel the performance improvement from sorting, however the speedup on an SGI O2 is significantly greater than implied by these numbers, since evaluating points in raster order allows much of the computation to take place on data which is already present in the cache, leading to a significant efficiency increase on this architecture over using unsorted points.

6.3 The tracker in operation

The hand-tracker has been implemented on an SGI O2 R5000 SC180 entry-level desktop workstation. The results shown were produced using $N = 400$ samples, which allows the tracker to run comfortably in real-time (30 Hz, using every other NTSC field). The number of samples can be increased to approximately $N = 575$ before any frames are dropped. Acceptable performance is obtained with $N = 150$ samples, and this runs comfortably at the field-rate of 60 Hz, although there is no noticeable benefit from using the additional fields. Experiments in previous chapters have benefited from using every field since this cuts in half the maximum motion of an object between sampled timesteps. Here large object motions are catered for using the importance sampling framework so a high sampling rate is less important. The main observed differences when using a smaller number of samples are a slight jitter when the hand is stationary and a longer time taken before the system reinitialises. As the number of samples is reduced below $N = 150$ the tracker begins to be distracted by clutter, although reinitialisation still functions to recover from these distractions.

The initialisation behaviour of the tracker is shown in figure 6.7. Initially the hand on the left is being tracked, and the spatial coherence inherent in the motion model means that the other blobs in the scene do not distract the tracker. When the thumb and forefinger are retracted, the shape no longer fits the template as accurately, and tracking reinitialises on the other hand within half a second. This behaviour corresponds to state transitions $2 \rightarrow 3 \rightarrow 1$ in figure 6.5. Figure 6.8 demonstrates successful tracking at high speed, with interlace shown to indicate image velocities. Even if the hand makes a sudden movement which is not predicted by the motion model, the blob tracker will detect the new position, and importance samples will be generated in the vicinity of the hand allowing the motion

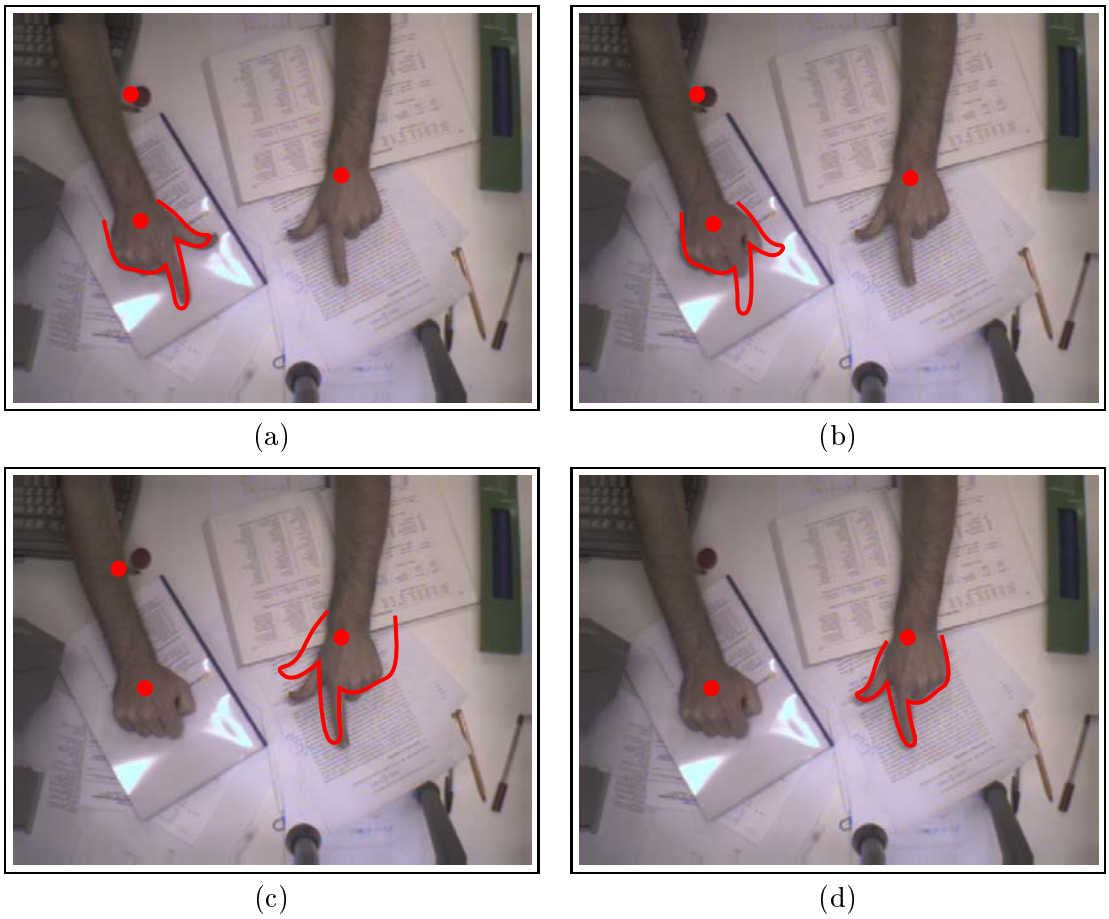


Figure 6.7: **The tracker reinitialises when one hypothesis takes precedence over another.** *While the user's right hand fits the template well (a) the motion coherence in the dynamical model ensures that the tracker remains locked on (discrete state 2 in figure 6.5). When the shape no longer fits the template (b) the probability of reinitialising to another hypothesis increases. After 5 frames (1/6 s) the tracker has switched to the neighbourhood of the left hand, selecting the left-handed template (discrete state 3 followed by 1), and another 5 frames later, a total of 1/3 s from the time that the right hand no longer fit the template, the tracker has locked on to the user's left hand (discrete state 1). Detected blobs are shown as circles, and $N=400$ samples are used.*

to be tracked. Figure 6.9 shows the advantage of using outline information as well as colour

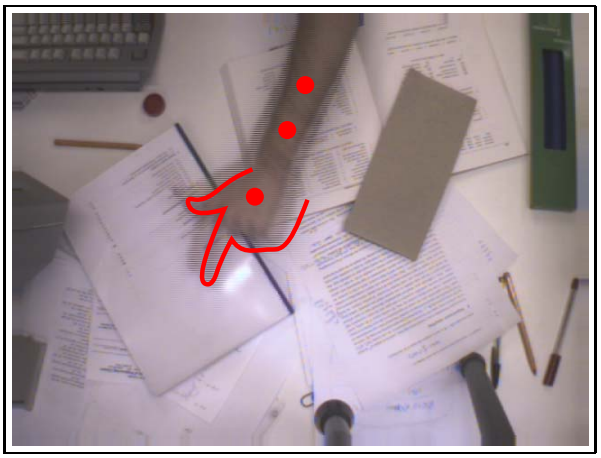


Figure 6.8: **Tracking follows high-speed motions.** *Both fields of a frame are shown, and interlacing artifacts demonstrate the rapid translation of the hand. The tracker is using N=400 samples.*

blob segmentation. The hands are adjacent, so their blobs merge, yet tracking continues to distinguish between the hands.

6.4 Extending the tracker for multiple users

The hand tracker described so far is specialised to a single hand shape, encoded in the template $\mathbf{Q}_0$. This requires the user's hand to be held fairly rigidly in the template pose, and necessarily means that some users' hands will fit the template better than others'. The main problem with an ill-fitting template is slow re-initialisation, but it also increases the chance of clutter distractions. It would be desirable, therefore, to allow some variation in $\mathbf{Q}_0$. A shape-space of hand-deformations was therefore established by using PCA on sequences of images collected from several subjects. Each subject placed his or her left hand in a reference position and orientation, with thumb and forefinger outstretched, and then made small movements to represent the variation of poses in which that user's hand will be presented to the system. These movements were recorded by a CONDENSATION tracker, the spline positions from the separate sequences were concatenated, and PCA was used to find a six-dimensional space of deformations. In the interests of real-time tracking, it is not desirable to increase the dimension of $\mathbf{x}_t$ from 4 continuous Euclidean similarity parameters to 10 for rigid transformations plus deformation. The solution adopted

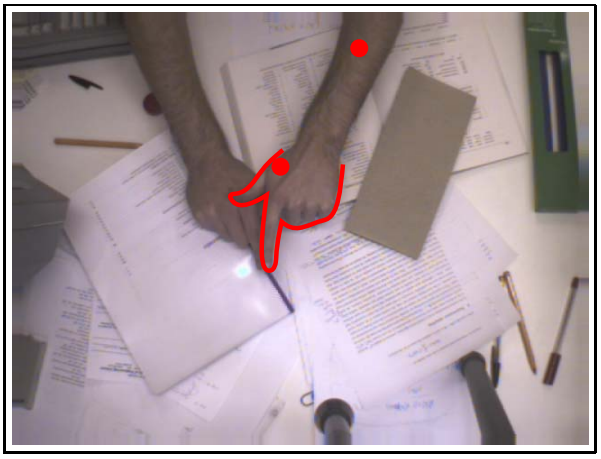


Figure 6.9: **Tracking is robust to failures of colour segmentation.** *When the hands move close to each other, their colour-segmentation blobs merge, giving a mean value (circle) between the hands. The contour tracker continues to follow the left hand using motion coherence and edge information, using N=400 samples.*

was to run two trackers, one in the Euclidean similarity space as before, and one in a separate six-dimensional deformation shape-space with N_D samples and parameter $\mathbf{d}_t$, so that $\mathbf{Q}_0 = \overline{\mathbf{Q}}_0 + W\mathbf{d}_t$ where $\overline{\mathbf{Q}}_0$ and W were estimated by PCA as in section 2.3 on page 22. Since $\mathbf{d}_t$ is expected to vary slowly (even while a single user is operating the tracker, the hand shape may alter slightly as the user adjusts to a comfortable position), only a small number of samples need be used in the deformation tracker, and good results are obtained using $N_D = 50$ samples. At each time-step, the trackers are run consecutively. First of all $\mathbf{d}_t$ is held fixed at $\mathbf{d}_t = \hat{\mathbf{d}}_{t-1}$ to establish $\mathbf{Q}_0 = \overline{\mathbf{Q}}_0 + W\hat{\mathbf{d}}_{t-1}$. The Euclidean transformation tracker is then run, exactly as before, and an estimated Euclidean vector $\hat{\mathbf{x}}_t$ is calculated as in section 5.2 on page 81. Now the Euclidean transformation is held fixed at $\mathbf{x}_t = \hat{\mathbf{x}}_t$ and the deformation tracker is run to estimate the new sample-set distribution for $\mathbf{d}_t$ from which $\hat{\mathbf{d}}_t$ can be estimated. This procedure is not entirely satisfactory, since it prevents a meaningful probabilistic interpretation of the state densities. It would be preferable to find some way of combining the estimation of deformation and rigid motion in a consistent Bayesian framework while keeping the economy of computation. It seems profligate to allow an entire 6-dimensional linear space to represent hand deformations, as most of the space consists of shapes which are very unlike a hand. It may be possible to collect a large number (perhaps 20–100) of template shapes, and include them as further discrete states in the model, thus relying on each user's hand being close to one of the example templates.

Alternatively it may be possible to specify a non-linear parameterisation of the deformations which is efficient enough to allow the deformation space to be included directly with the Euclidean similarity parameters in the main tracker. Despite this caveat, results using the two-tracker system are promising. The deformation-space tracker was run as a standard CONDENSATION tracker using dynamics learned from the hand-shape training sequences. The shape rapidly deforms to fit a hand as it enters the scene, and figure 6.10 shows that the tracker can adapt to different hands as required.

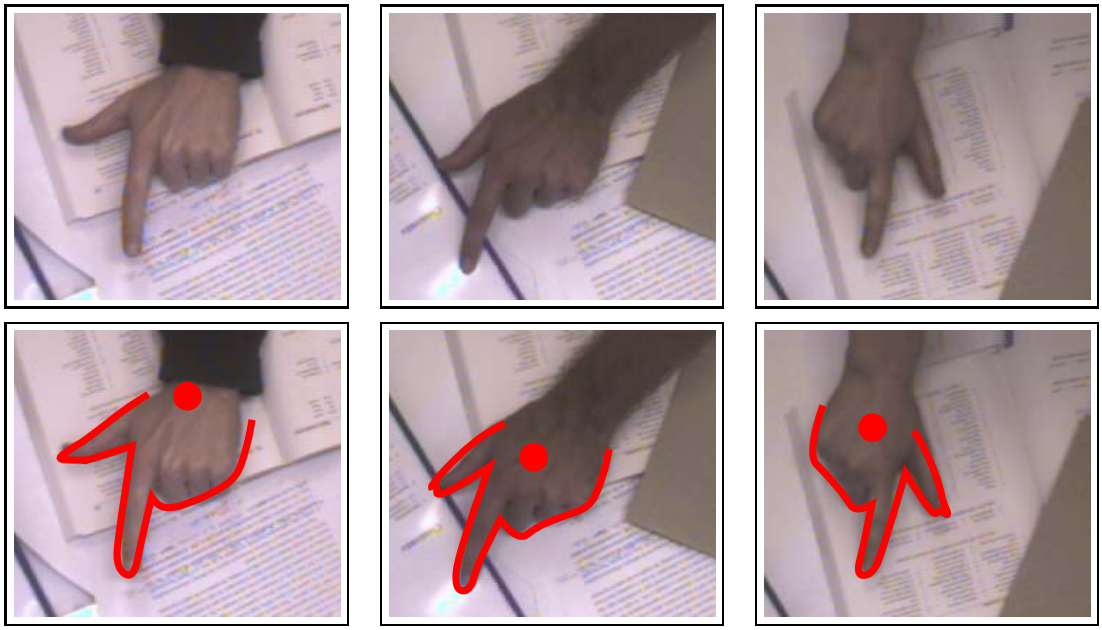


Figure 6.10: **The deformation tracker accommodates different hands.** *Columns show different users. The lower images show tracked output approximately a second after the hand was introduced into the scene. The left-hand user was included in hand-shape training data, the other two were not. In this example N=400 for the Euclidean similarity tracker and N_D=50 for the deformation tracker.*

7

Smoothing the output of a CONDENSATION filter

Kitagawa (1996), along with his formulation of the CONDENSATION algorithm, presented two smoothing algorithms which allow the state $\mathbf{X}_t$ at time t to be estimated in the light of all of the measurement data in a sequence, rather than just the data up until time t. This chapter presents an implementation of Kitagawa's smoothing algorithms in the CONDENSATION framework, and in doing so incorporates a significant simplification of one of them which extends its use to a wider class of dynamical model. Smoothing highlights a simplification which has been made in the preceding chapters. One of the distinguishing characteristics of the CONDENSATION algorithm is that it represents multiple hypotheses about object state in the form of a multi-modal state density. All of the known information about the object is contained in the state density, and this information must be processed in some way if a single estimated object position is required at each time-step. Previous chapters have described calculating simple moments of the state density, in practice the mean, for display purposes. This breaks down when the density has several peaks, and one advantage of a smoothing filter is that it tends to eliminate hypotheses which were unlikely with hindsight. The result is that the smoothed density better approximates a uni-modal density, and simple mean-estimation produces a more accurate representation of the density. Smoothing can also be used as the basis of a learning algorithm for mixed-state models, and this will be discussed in section 8.5. Note that here "smoothing" refers to the statistical technique of conditioning the state density on both past and future measurements. It has

nothing to do with the standard computer vision definition involving convolution with a smoothing kernel, either spatially or temporally.

7.1 Smoothing the output of CONDENSATION

Within the CONDENSATION algorithm the conditional state density $p(\mathbf{X}_t|\mathcal{Z}_t)$ encodes all of the known information about the object state given the current measurement history $\mathcal{Z}_t \equiv (\mathbf{Z}_1, \ldots, \mathbf{Z}_t)$. Once tracking has completed it may be desirable to return, in batch-mode, to calculate $p(\mathbf{X}_t|\mathcal{Z}_T)$, the state density for each time-step given the *entire* measurement history up until time $T \geq t$. This is particularly valuable in the case of temporary distraction, when the state density splits for a few time-steps into several distinct trajectories. During real-time[1] tracking, it is impossible to reliably determine which of these competing hypotheses corresponds to the true object trajectory, however all but one of the trajectories will "die out" eventually when it becomes apparent that they correspond to clutter, distractions or mis-estimation.

Kitagawa (1996) presents two algorithms to smooth a time-series of sample-set state estimates, which we reproduce here in the CONDENSATION framework. The first is very straightforward. Rather than storing the set $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}$ at each time t, the sample position $\mathbf{s}_t^{(n)}$ is replaced by an entire trajectory $\mathcal{S}_t^{(n)} = (\mathbf{s}_t^{(n,1)}, \ldots, \mathbf{s}_t^{(n,t)})$. The history $(\mathbf{s}_t^{(n,1)}, \ldots, \mathbf{s}_t^{(n,t-1)})$ is taken to be the trajectory of the base sample which is chosen in the first step of the CONDENSATION algorithm, and the moments of the smoothed density $p(\mathbf{X}_\tau|\mathcal{Z}_t)$ can be estimated for $1 \leq \tau \leq t$ by computing the expectation

$$\mathcal{E}[\phi(\mathbf{X}_\tau)|\mathcal{Z}_t] \approx \sum_{n=1}^{N} \pi_t^{(n)} \phi\left(\mathbf{s}_t^{(n,\tau)}\right).$$

The sequence-based smoothing algorithm is shown in figure 7.1. Note that it is reminiscent of the dynamic programming algorithm used, for example, to estimate the most likely path through a Hidden Markov Model (HMM) (Rabiner and Bing-Hwang, 1993). This algorithm has the disadvantage that in practice, the variance of the samples $\{\mathbf{s}_t^{(n,\tau)}\}$ for given t and $\tau \ll t$ is very small. In fact, for large $t - \tau$ it is typical to find that all of the $\{\mathbf{s}_t^{(n,\tau)},\ n = 1 \ldots N\}$ are identical, meaning that all of the sample-sequences share a common ancestor trajectory. (In later results this is typically true for $t - \tau > 10$.) This may

[1] Real time is used here to distinguish the standard CONDENSATION tracking algorithm from any batch-mode post-processing. It does not imply the standard computer vision meaning, that tracking is effected in the time between acquisition of consecutive images.

Iterate

From the "old" sample-sequence set $\{(\mathcal{S}_{t-1}^{(n)}, \pi_{t-1}^{(n)}),\ n = 1, \ldots, N\}$ at time-step $t-1$, construct a "new" sample-sequence set $\{(\mathcal{S}_t^{(n)}, \pi_t^{(n)}),\ n = 1, \ldots, N\}$ for time t, where $\mathcal{S}_t^{(n)} = (\mathbf{s}_t^{(n,1)}, \ldots, \mathbf{s}_t^{(n,t)})$.

Construct the n^{th} of N new samples as follows:

1. **Select** a base sequence $\mathcal{S}_{t-1}^{(i)} = (\mathbf{s}_{t-1}^{(i,1)}, \ldots, \mathbf{s}_{t-1}^{(i,t-1)})$ by sampling with probability $\pi_{t-1}^{(i)}$. This can be done efficiently, for example using cumulative probabilities as described in section 3.4.
2. **Predict** by sampling from $p(\mathbf{X}_t | \mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(i,t-1)})$ to choose $\mathbf{s}_t^{(n,t)}$.
3. **Measure** and weight the new position in terms of the image data $\mathbf{Z}_t$, setting $\pi_t^{(n)} = p(\mathbf{Z}_t | \mathbf{X}_t = \mathbf{s}_t^{(n,t)})$, then set $\mathcal{S}_t^{(n)} = \mathcal{S}_{t-1}^{(i)} \cup \mathbf{s}_t^{(n,t)}$.

Finally normalise so that $\sum_n \pi_t^{(n)} = 1$ to find the new sample-sequence set $(\mathcal{S}_t^{(n)}, \pi_t^{(n)})$. Moments ϕ of the smoothed density $p(\mathbf{X}_\tau | \mathcal{Z}_t)$ for $1 \leq \tau \leq t$ can be found from

$$\mathcal{E}[\phi(\mathbf{X}_\tau) | \mathcal{Z}_t] \approx \sum_{n=1}^{N} \pi_t^{(n)} \phi\Big(\mathbf{s}_t^{(n,\tau)}\Big)$$

Figure 7.1: **The sequence-based smoothing algorithm for** CONDENSATION**.**

be acceptable if the only required output is a single estimated position for each time-step, but in some circumstances it is preferable to maintain more detailed information as long as possible, and so a more complex algorithm follows. Note that the collapse of the trajectories $\mathcal{S}_t^{(n)}$ into common histories permits pruning, thus allowing a significant economy of storage, which is otherwise $O(Nt)$.

The second smoothing algorithm presented in (Kitagawa, 1996) is a forward–backward algorithm, analogous to the smoothing algorithm for Gaussians (Gelb, 1974) which is a two-pass extension of the Kalman filter, and also related to the forward–backward algorithm for HMMs (Rabiner and Bing-Hwang, 1993). The forward pass consists of a standard application of the CONDENSATION tracker, during which all the sets $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}$ for $t = 1 \ldots T$ are stored. Now smoothing is done purely by reweighting the $\pi_t^{(n)}$ — all of the $\mathbf{s}_t^{(n)}$ remain fixed. The algorithm presented in (Kitagawa, 1996) contains a backward filtering

step which requires access to the measurements $\mathbf{Z}_t$ during the second pass, and also means that the density $p(\mathbf{X}_{t-1}|\mathbf{X}_t)$ must be available for sampling, a condition which is not imposed by the standard CONDENSATION algorithm. We believe this backward filtering step is unnecessary and so do not include it, however the mathematical treatment and the basic structure of our algorithm are both derived from (Kitagawa, 1996). Note that our algorithm, in common with that in (Kitagawa, 1996), does require the evaluation of $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ which constitutes some restriction on the form of dynamical model used.

Defining $\mathcal{Z}_t^T = (\mathbf{Z}_t, \ldots, \mathbf{Z}_T)$ we have $\mathcal{Z}_T = \mathcal{Z}_{t-1} \cup \mathcal{Z}_t^T$. Therefore,

$$\begin{aligned} p(\mathbf{X}_t|\mathcal{Z}_T) &= p(\mathbf{X}_t|\mathcal{Z}_{t-1}, \mathcal{Z}_t^T) \\ &\propto p(\mathbf{X}_t, \mathcal{Z}_t^T|\mathcal{Z}_{t-1}) \\ &= p(\mathcal{Z}_t^T|\mathbf{X}_t)p(\mathbf{X}_t|\mathcal{Z}_{t-1}) \quad \text{by the independence of the } \mathbf{Z}_t. \end{aligned}$$

It is this rearrangement which allows the sample positions $\mathbf{s}_t^{(n)}$ to remain fixed after the smoothing step. Recall from section 3.4 on page 40 that the set $\{\mathbf{s}_t^{(n)}\}$ is approximately a fair sample from $p(\mathbf{X}_t|\mathcal{Z}_{t-1})$, so by replacing the original $\pi_t^{(n)}$ by smoothing weights

$$\psi_t^{(n)} = p(\mathcal{Z}_t^T|\mathbf{X}_t = \mathbf{s}_t^{(n)}),$$

the set $\{(\mathbf{s}_t^{(n)}, \psi_t^{(n)})\}$, when normalised, will approximate $p(\mathbf{X}_t|\mathcal{Z}_T)$ as required. It is therefore the weights $\psi_t^{(n)}$ which the backward smoothing pass will calculate.

A recursive algorithm to calculate the densities $p(\mathcal{Z}_t^T|\mathbf{X}_t)$ can be specified mathematically as follows:

$$\begin{aligned} p(\mathcal{Z}_T^T|\mathbf{X}_T) &= p(\mathbf{Z}_T|\mathbf{X}_T) \\ p(\mathcal{Z}_{t+1}^T|\mathbf{X}_t) &= \int p(\mathcal{Z}_{t+1}^T|\mathbf{X}_{t+1})p(\mathbf{X}_{t+1}|\mathbf{X}_t)\, d\mathbf{X}_{t+1} \\ p(\mathcal{Z}_t^T|\mathbf{X}_t) &= p(\mathbf{Z}_t|\mathbf{X}_t)p(\mathcal{Z}_{t+1}^T|\mathbf{X}_t) \end{aligned}$$

An implementation requires the derivation of an approximation $\delta_t^{(n)}$ to $p(\mathcal{Z}_{t+1}^T|\mathbf{X}_t = \mathbf{s}_t^{(n)})$. The integral is approximated as a sum:

$$p(\mathcal{Z}_{t+1}^T|\mathbf{X}_t = \mathbf{s}_t^{(n)}) \approx \delta_t^{(n)} = \sum_{m=1}^{N} p(\mathcal{Z}_{t+1}^T|\mathbf{X}_{t+1} = \mathbf{s}_{t+1}^{(m)}) \frac{p(\mathbf{X}_{t+1} = \mathbf{s}_{t+1}^{(m)}|\mathbf{X}_t = \mathbf{s}_t^{(n)})}{p(\mathbf{X}_{t+1} = \mathbf{s}_{t+1}^{(m)}|\mathcal{Z}_t)}$$

using the correction

$$p(\mathbf{X}_{t+1} = \mathbf{s}_{t+1}^{(m)}|\mathcal{Z}_t) = \gamma_t^{(m)} = \sum_{k=1}^{N} \pi_t^{(k)} p(\mathbf{X}_{t+1} = \mathbf{s}_{t+1}^{(m)}|\mathbf{X}_t = \mathbf{s}_t^{(k)}).$$

Perform the forward pass giving a weighted sample-set $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}$ for each $t = 1 \ldots T$, computed by the CONDENSATION algorithm.

1. **Initialise** smoothing weights $\psi_T^{(n)}$:

$$\psi_T^{(n)} = \pi_T^{(n)} \quad \text{for } n = 1 \ldots N.$$

2. **Iterate** backwards over the sequence for $t = T - 1 \ldots 1$:

   (a) **Calculate** prediction probabilities:

$$\alpha_t^{(m,n)} = p(\mathbf{X}_{t+1} = \mathbf{s}_{t+1}^{(m)} | \mathbf{X}_t = \mathbf{s}_t^{(n)}) \quad \text{for } m, n = 1 \ldots N.$$

   (b) **Calculate** correction factors:

$$\gamma_t^{(m)} = \sum_{k=1}^{N} \pi_t^{(k)} \alpha_t^{(m,k)} \quad \text{for } m = 1 \ldots N.$$

   (c) **Approximate** backward variables:

$$\delta_t^{(n)} = \sum_{m=1}^{N} \psi_{t+1}^{(m)} \frac{\alpha_t^{(m,n)}}{\gamma_t^{(m)}} \quad \text{for } n = 1 \ldots N.$$

   (d) **Evaluate** smoothing weights

$$\psi_t^{(n)} = \pi_t^{(n)} \delta_t^{(n)} \quad \text{for } n = 1 \ldots N$$

   then normalise multiplicatively so $\sum \psi_t^{(n)} = 1$, and store with sample positions as

$$\{(\mathbf{s}_t^{(n)}, \psi_t^{(n)}),\ n = 1 \ldots N\}$$

Figure 7.2: **The backward stage of the two-pass smoothing algorithm for** CONDENSATION.

It is introduced because the $\mathbf{s}_{t+1}^{(m)}$ are not distributed uniformly but are a sample from $p(\mathbf{X}_{t+1}|\mathcal{Z}_t)$, and without the correction this could bias the sum. This method of correcting the estimate of an integral over sample-sets is borrowed from the technique of importance sampling (Ripley, 1987), as used in the previous chapter for quite another purpose. The backward pass of the two-pass smoothing algorithm is shown in figure 7.2. Note that the complexity of the algorithm is $O(TN^2)$ and that $O(TN)$ storage is required for the sample-sets, and $O(N^2)$ for the $\alpha_t^{(m,n)}$. This latter storage requirement can be avoided by eliminating the $\alpha_t^{(m,n)}$ from the algorithm and instead calculating each of the $p(\mathbf{X}_{t+1} = \mathbf{s}_{t+1}^{(m)}|\mathbf{X}_t = \mathbf{s}_t^{(n)})$ twice. Since this calculation is typically the most computationally expensive step of the algorithm, this tradeoff must be carefully considered.

7.2 Applying the smoothing algorithms

First, the sequence-based smoothing algorithm was applied to a test sequence from section 5.2 which shows a ball bouncing against a backdrop of heavy clutter. The ball moves under the action of a two-state motion model, where the first state is constant acceleration due to gravity and the second state corresponds to an instantaneous bounce event during which the ball's vertical velocity is reversed. The state vector $\mathbf{X}_t$ includes a discrete variable labelling which of the two discrete states the model is in, The unsmoothed output of a mixed-state CONDENSATION tracker is depicted in figure 7.3. At each time-step, an MAP estimate is computed to determine which of the two states the tracker is in, and the mean and variance of the y translation coordinate within that state are shown, along with an indication of which time-steps were estimated to contain bounce events. The unsmoothed output is rather jittery due to the clutter, and the bounce events are not always accurately found. Figure 7.4 demonstrates the mis-estimation problem; the distribution has split into several peaks, and although one peak is present at the true ball position, the other peaks pull the distribution mean away from the desired value. After running the sequence-based smoothing algorithm (figure 7.5) most of the jitter has been eliminated and the trajectory shows smooth parabolas between bounces. One field has still been incorrectly estimated to contain a bounce. As discussed in the previous section, the variance is estimated to be zero except over the last few time-steps, since all the samples in the final distribution share the same history until $t = 60$ fields. Of course, this must be an under-estimate. Figure 7.4 shows detail from field 26 of the sequence before and after smoothing.

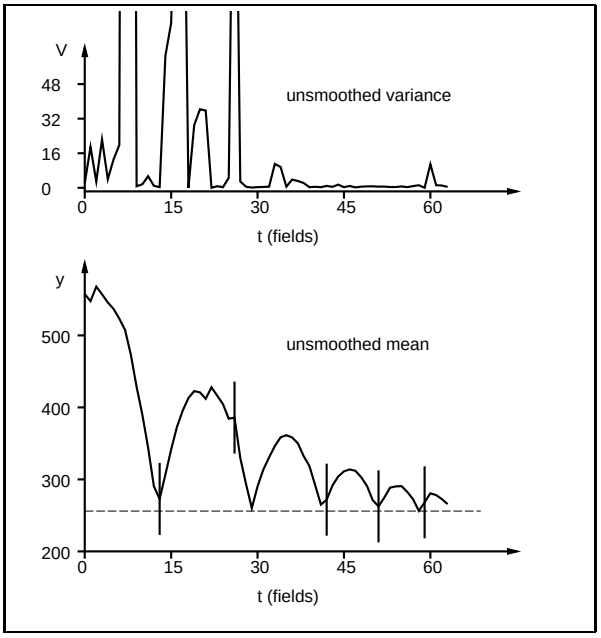


Figure 7.3: **The unsmoothed output of a mixed-state** CONDENSATION **algorithm contains estimation errors.** *The mean of the y coordinate of the distribution is shown in pixels, along with the mean-square variance around the curve in (pixels)*^{2}*. Vertical bars correspond to time-steps which are estimated to contain bounce events. The variance is high when several hypotheses have high probabilities (see figure 7.4).*

The two-pass algorithm also successfully smooths the raw tracked output (figure 7.6), and now correctly determines the bounce events. Variance information is also preserved by the two-pass filter, and a small spread of samples in the distribution can be seen in figure 7.7. Note that neither smoothing algorithm incorporates any separate machinery to estimate the mixed-state transitions. The discrete-state labels, forming part of the state-vector $\mathbf{X}_t$, are automatically estimated along with the continuous state variables. Of course, the values of the state labels of $\mathbf{s}_t^{(n)}$ and $\mathbf{s}_{t-1}^{(m)}$ play a large part in determining the density $p(\mathbf{X}_{t+1} = \mathbf{s}_{t+1}^{(n)} | \mathbf{X}_t = \mathbf{s}_t^{(m)})$ for the two-pass algorithm.

Finally, the algorithms were applied to another test sequence showing a hand moving over a cluttered desk. The hand translates and deforms in a 12-dimensional linear shape-space. After approximately 30 fields, the unsmoothed distribution splits into two peaks (figure 7.8), one of which is caused by clutter. The clutter peak dominates for 10 fields, causing a serious error in the estimated state, although the true position is maintained as a smaller peak in the distribution throughout, and the tracker recovers eventually. Figure 7.9 shows graphs of the y coordinate of the estimated mean of the distribution along with the

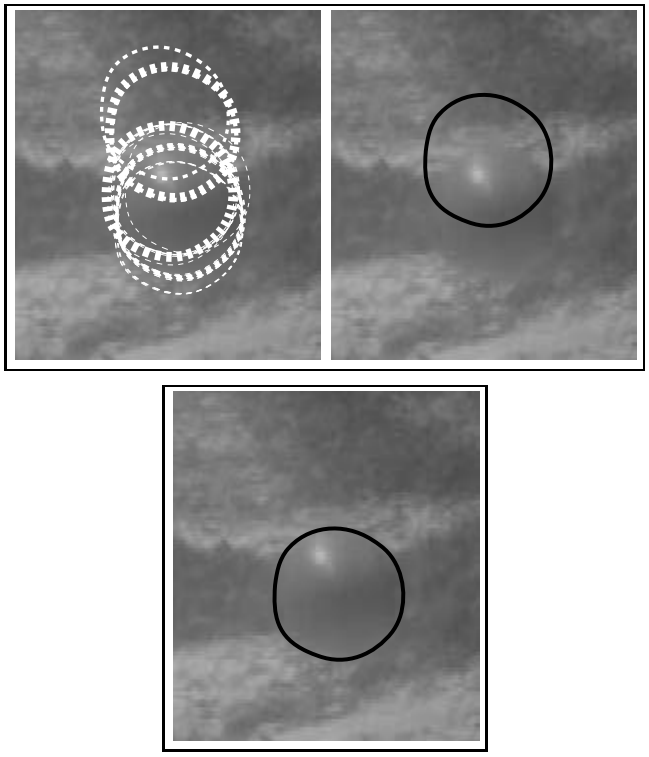


Figure 7.4: **Smoothing eliminates false hypotheses.** *Before smoothing, multiple hypotheses can increase the variance of the distribution (top left) and shift the mean away from the object position (top right). After running the sequence-based smoothing algorithm the estimated variance has dropped to zero (see text) but the mean is now correctly positioned (bottom). Detail from field 26 of the sequence is shown (see figures 7.3 and 7.5). The solid line is the distribution mean, and the dotted lines are high-scoring samples, where the width of the sample outline is proportional to its sample weight.*

variance of the sample-set. The hand moves up smoothly from field 20 to field 40, but the unsmoothed estimate is distracted between fields 30–40, before rapidly regaining the correct position at field 42. Note the very high variances, especially just before the tracker recovers.

Figure 7.10 shows the result of applying the two-pass smoothing algorithm to the hand sequence (the sequence-based algorithm provides similar state estimates and lower variance as before). When the entire sequence is taken into account, it is apparent that the lower peak in figure 7.8 corresponds to clutter, and so only the trajectory corresponding to the actual hand position survives. Figure 7.11 graphs the estimated y coordinate and the mean-square curve variances for the output of the two-pass smoother. As in the case of the ball, the jitter on the state estimates is reduced, and the hand position is significantly more

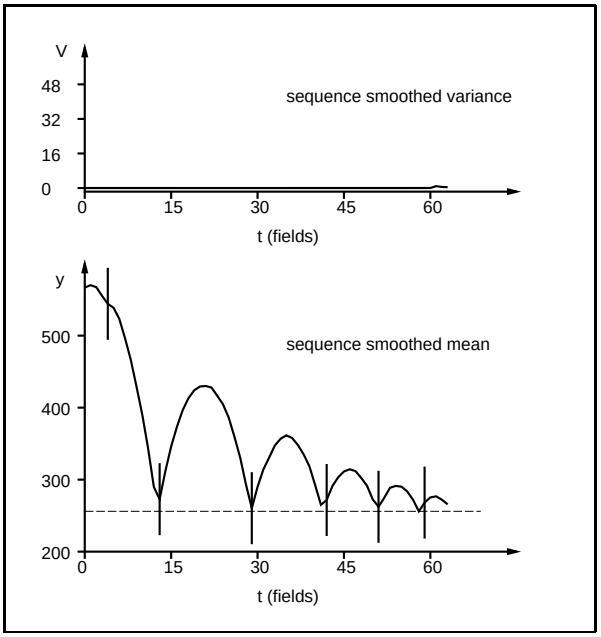


Figure 7.5: **The sequence-based algorithm smooths away jitter.** *The mean of the y coordinate of the distribution is shown in pixels, along with the mean-square variance around the curve in (pixels)*^{2}*. Vertical bars correspond to time-steps which are estimated to contain bounce events, and a bounce is incorrectly estimated at field 4. The sequence-based smoothing algorithm collapses the variance to zero for all but the last few time-steps (see text).*

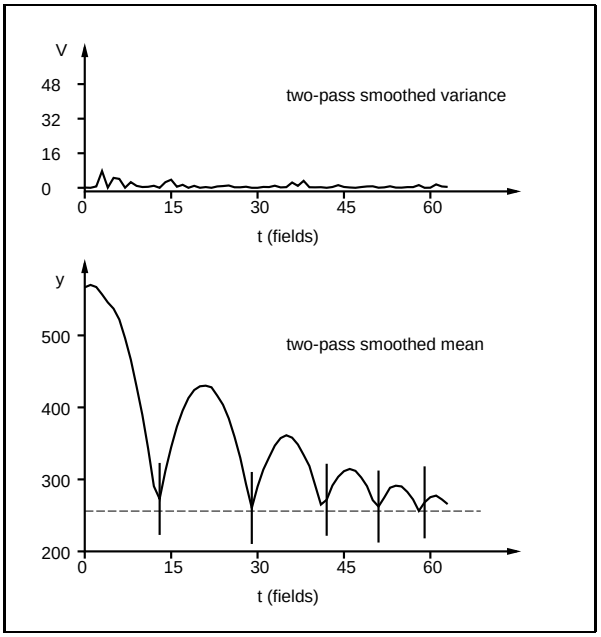


Figure 7.6: **The two-pass algorithm preserves variance information while smoothing.** *The mean of the y coordinate of the distribution is shown in pixels, along with the mean-square variance around the curve in (pixels)*^{2}*. Vertical bars correspond to frames which are estimated to contain bounce events. The bounce events are correctly identified.*

accurately determined compared with the raw CONDENSATION algorithm. The variance of the sample-sets is also much reduced, although clearly towards the end of the sequence the variance must increase to match that of the raw data.

7.3 Fixed-lag smoothing

Kitagawa (1996) also implements fixed-lag versions of both smoothing algorithms. The idea is to introduce a lag of p frames between observations and state estimates and thus to report at each timestep an estimate from the distribution $p(\mathbf{X}_{t-p}|\mathcal{Z}_t)$. This is easy to implement with either smoothing algorithm. In the sequence-based case it is enough simply to store sequences $\mathcal{S}_t^{(n)} = (\mathbf{s}_t^{(n,t-p)}, \ldots, \mathbf{s}_t^{(n,t)})$ and proceed as before to calculate estimates

$$\mathcal{E}[\phi(\mathbf{X}_{t-p})|\mathcal{Z}_t] \approx \sum_{n=1}^{N} \pi_t^{(n)} \phi\Big(\mathbf{s}_t^{(n,t-p)}\Big) .$$

For the two-pass algorithm the backward pass is restarted at every timestep, but only iterated p steps to calculate the smoothing weights $\psi_{t-p}^{(n)} = p(\mathcal{Z}_{t-p}^t|\mathbf{X}_{t-p} = \mathbf{s}_{t-p}^{(n)})$ from which

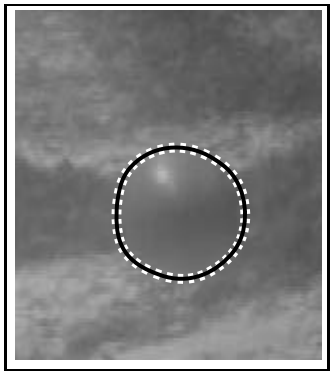


Figure 7.7: **Smoothing eliminates false hypotheses.** *The two-pass smoothing algorithm collapses the distribution down to a single peak with variance information preserved. A small spread of samples is present around the mean (figure 7.6 indicates that the estimated mean-square variance around the curve is only a few pixels). Detail from field 26 of the sequence is shown (compare with figure 7.4). The solid line is the distribution mean, and the dotted lines are high-scoring samples, where the width of the sample outline is proportional to its sample weight.*

estimates

$$\mathcal{E}[\phi(\mathbf{X}_{t-p})|\mathcal{Z}_t] \approx \sum_{n=1}^{N} \psi_{t-p}^{(n)} \phi\left(\mathbf{s}_{t-p}^{(n)}\right)$$

can be found. These fixed-lag smoothers have not been implemented in the CONDENSATION framework but it is to be expected that they would produce somewhat more accurate state estimates in cases where clutter distractions are transient (in particular where they last for fewer than p frames). The sequence-based fixed-lag smoother would probably be preferable if real-time operation were desired, since it entails a negligible computational overhead compared with an unsmoothed tracker, although there is a storage cost of order $O(Np)$.

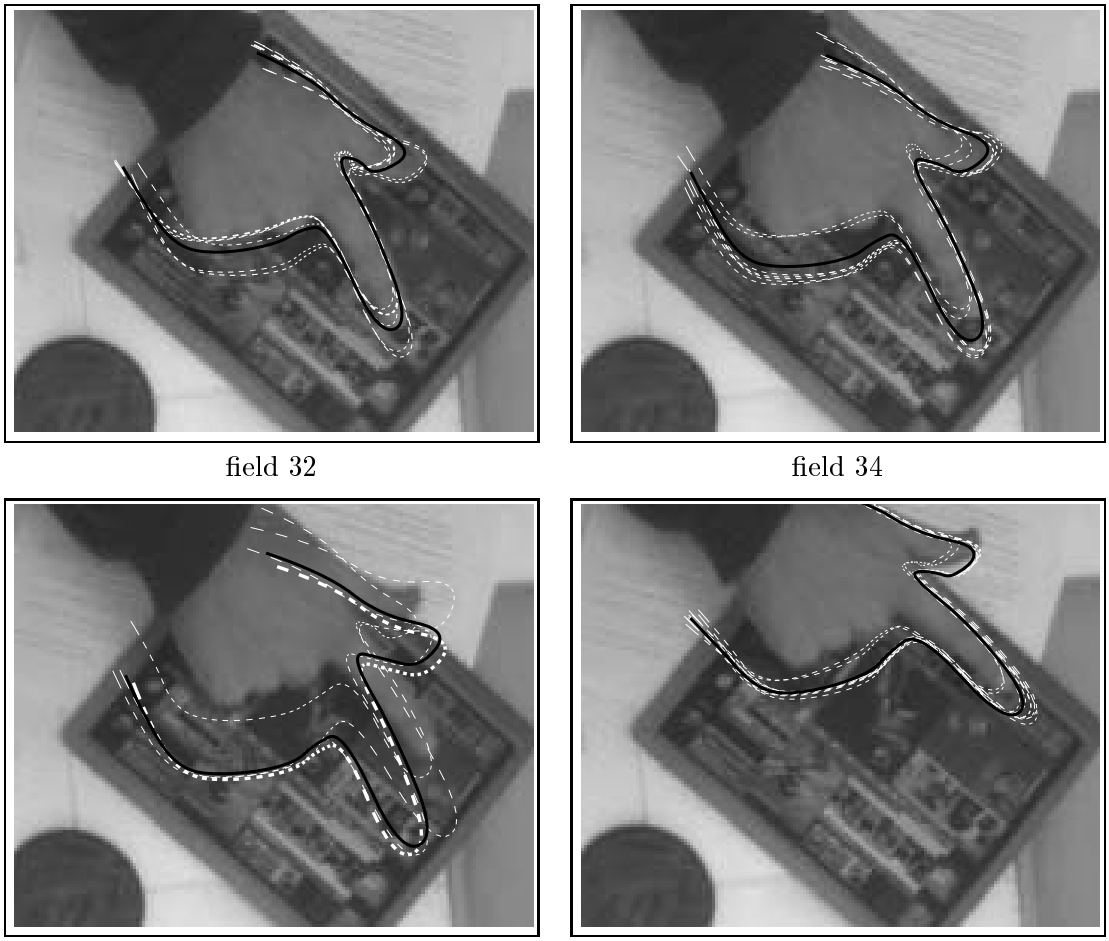


Figure 7.8: **Clutter causes temporary mis-estimation from unsmoothed data.** *The unsmoothed state distribution has begun to diverge in field 32, and by field 34 the clutter peak dominates. The multi-modality persists until field 38, after which the clutter peak rapidly dies away, leaving a single peak around the object again by field 42. The solid line is the distribution mean, and the dotted lines are high-scoring samples, where the width of the sample outline is proportional to its sample weight.*

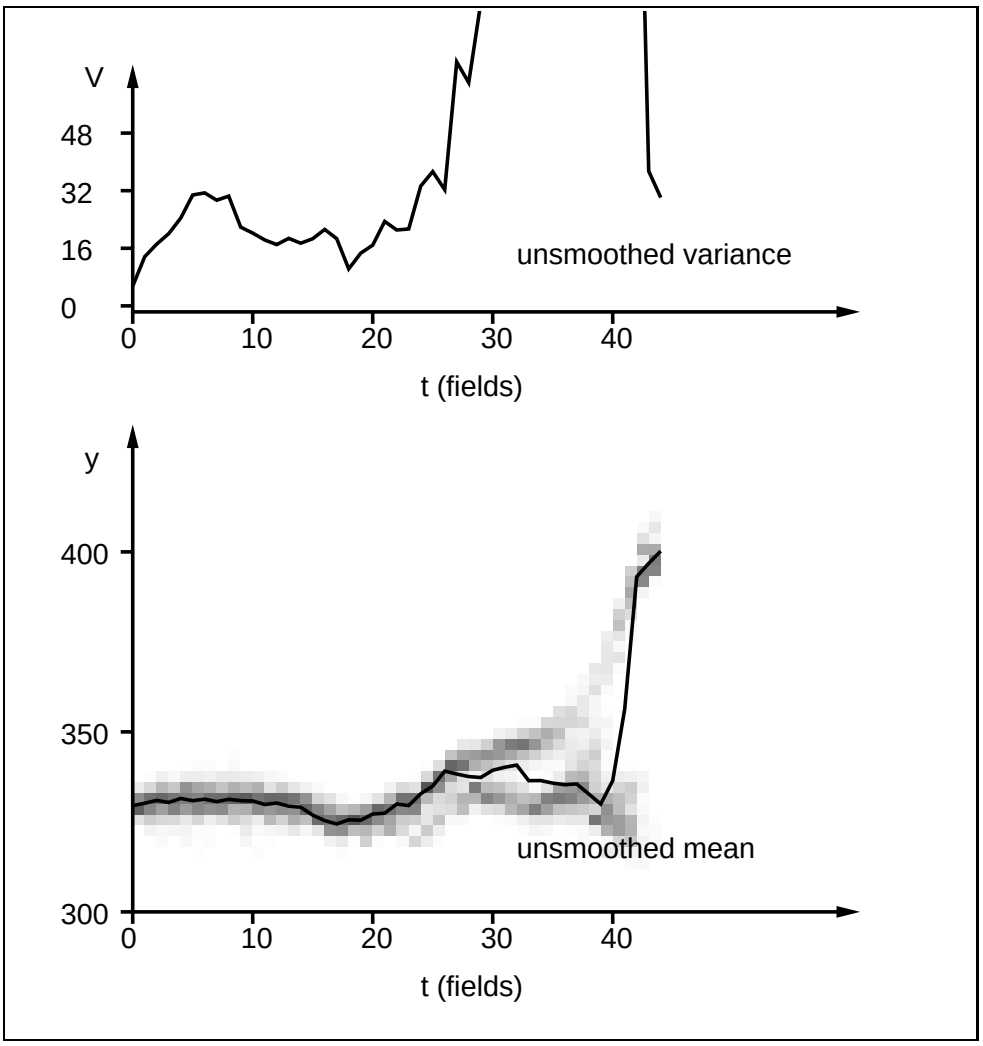


Figure 7.9: **The unsmoothed hand data leads to estimation errors.** *The mean of the y coordinate of the distribution is shown in pixels with a histogram showing the full distribution plotted behind, along with the mean-square variance around the curve in (pixels)^2. Figure 7.8 shows the distribution splitting into two peaks between fields 32–34, and this is apparent from the graphs. The clutter peak is stronger, and causes the position to be mis-estimated by shifting the distribution mean. Although the hand moves up steadily during fields 30–40, the estimated position moves down before suddenly recovering at field 42.*

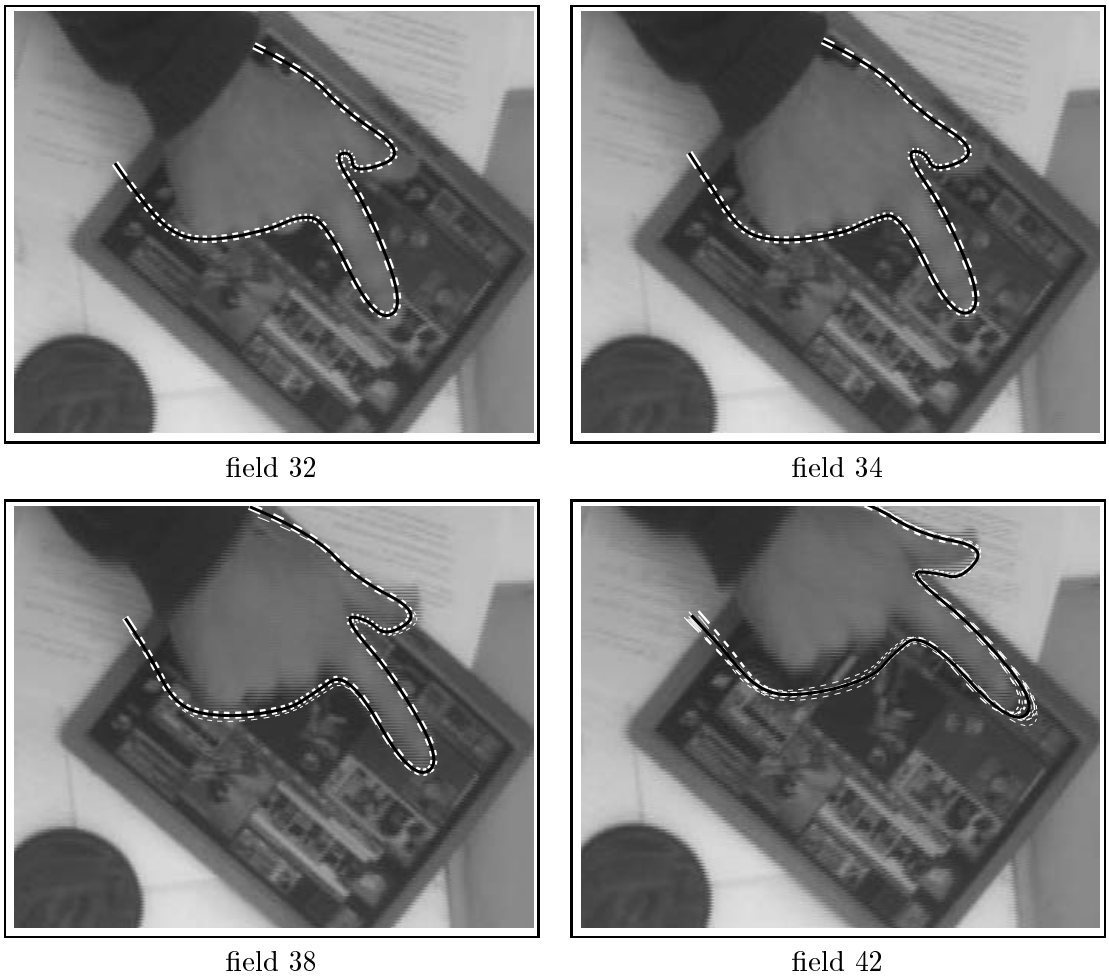


Figure 7.10: **The smoothing algorithm correctly follows the hand.** *Compare with figure 7.8; after smoothing using the two-pass algorithm, all of the visible distribution is concentrated on the correct peak. The solid line is the distribution mean after smoothing, and the dotted lines are high-scoring samples, where the width of the sample outline is proportional to its weight.*

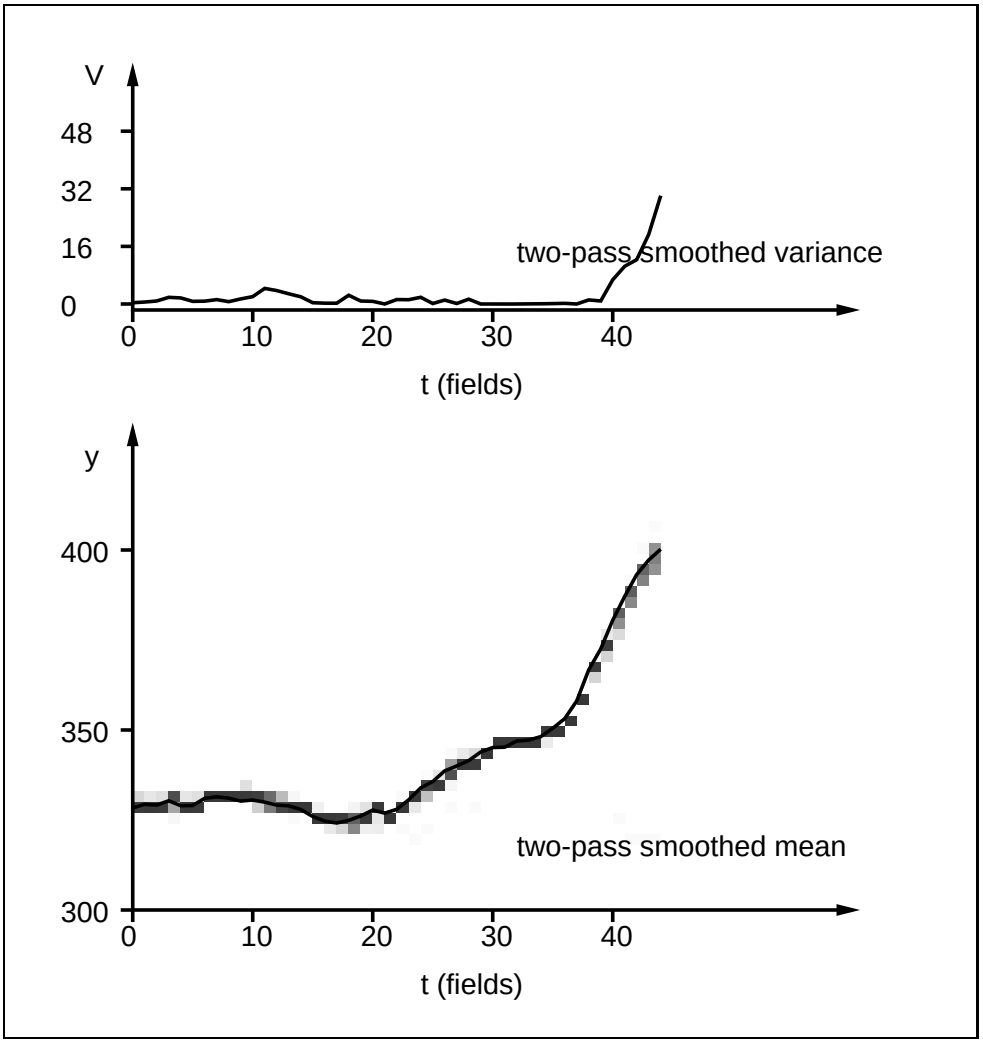


Figure 7.11: **The two-pass smoothing algorithm corrects estimation errors.** *Figure 7.10 shows that the two-pass algorithm eliminates the clutter peak which distracted the standard tracker. Now the estimated state corresponds to the true hand position as it moves steadily up the image from field 20 (compare with figure 7.9). The variance of the sample-set is also greatly reduced, although clearly at the end of the sequence the variance increases to match that of the raw output. The mean of the y coordinate of the distribution is shown in pixels with a histogram showing the full smoothed distribution plotted behind, along with the mean-square variance around the curve in (pixels)*^{2}.

8

Discussion

This chapter examines issues arising from the work presented in this thesis, and suggests some areas of future research which may prove fruitful. Section 8.3 also includes a fuller review of the literature on random-sampling methods for filtering than was given in chapter 1, since the notation and ideas presented in chapters 3–6 allow a more technical discussion of the related research.

8.1 Failure modes of the CONDENSATION algorithm

Previous chapters have not addressed the failure modes of the CONDENSATION algorithm in any detail. This is partly because unlike for example a Kalman filter, the CONDENSATION algorithm has a natural mechanism to trade off speed and robustness. When tracking fails, the sample-set size N can be increased and tracking runs slower but has a higher chance of success. For this reason, it is hard to construct a case where CONDENSATION absolutely fails to track since often selecting a larger N will allow the filter to track through any particularly difficult portion of a sequence. The main difficulties which have been found experimentally arise when the shape model is inaccurate. This is to be expected, since as $N \to \infty$ the filter increasingly well approximates $p(\mathbf{X}_t|\mathcal{Z}_t)$ and if the form of $p(\mathbf{Z}|\mathbf{X})$ chosen is a poor reflection of the desired observation density, then no amount of increase in N will improve tracking performance. It is expected that region-based models as discussed in the next section will allow much more robust modelling of shape than primarily edge-based information as used in this thesis, and this may alleviate the problem of accurate tuning of

the observation density.

The main concern of the work which will be reviewed in section 8.3 is with the efficiency of CONDENSATION-like algorithms as statistical estimators. No consideration has been given in this thesis to such notions of efficiency in the CONDENSATION algorithm — the criterion of success of the filter has in general been taken to be an ability to track an object throughout an extended sequence without suffering unrecoverable errors which cause the object's location to be lost entirely. This approach is justified partly by the existence of the aforementioned research by others. Clearly it may be advantageous to apply some of these methods in the CONDENSATION framework and this is discussed in section 8.3. One common complaint in e.g. (Gordon and Salmond, 1995; Carpenter et al., 1997; Doucet, 1998) is that N must be impractically large to adequately model the state density, especially in high-dimensional spaces (more than three or four dimensions). In the experiments described in this thesis, we have not found this to be a problem — the hand-tracker of section 4.3 on page 60 requires a 12-dimensional shape-space but uses only $N = 500$ samples for tracking. Kitagawa (1996) reports that even relatively inefficient samplers are sufficient if only the mean of the distribution need be estimated. It seems likely that we are able to use so few samples here both because only the mean of the state density is estimated rather than any higher-order moments, and because of the particular nature of our learnt motion models which allow accurate prediction even in high-dimensional spaces. Of course, just because tracking does not fail unrecoverably, this does not mean that the state density is accurately represented. It may be that when higher-order moments of a distribution are required, for example for the learning algorithms proposed in section 8.5, a more efficient form of CONDENSATION will be necessary.

8.2 Observation models for CONDENSATION

As discussed in section 3.6 on page 46, the observation model $\mathbf{Z}$ should ideally be a set of image features (whether discrete points, curves or regions) independent of $\mathbf{X}$. The observation density described in chapter 3 is based on a feature-curve approximation where the curve approximated varies according to the predicted sample curve $\mathbf{s}$. Alternative forms of $p(\mathbf{Z}|\mathbf{X})$ are used in chapters 5 and 6 for particular applications, but these share with the original form in section 3.6 the characteristic of being evaluated only over a small image region close to a set of predicted curve features. While $p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t^{(n)})$ for a given sample

position $\mathbf{s}_t^{(n)}$ involves only local computation near to the curves encoded by $\mathbf{s}_t^{(n)}$, the regions explored as this density is evaluated for all the $\{\mathbf{s}_t^{(n)}, n = 1, \ldots, N\}$ may encompass large areas of the image. As explained in section 3.6, for approximations of this form to be valid, the partition function Z in (3.12) on page 50 must vary slowly compared with $\mathbf{X}$ and it remains to be verified experimentally that this condition is true in the applications we consider.

In principle it would be possible to use an observation density such as those considered in the literature for Bayesian analysis of static images (see section 1.4 on page 9) but there are problems with this approach. The region-based observation models used for example in (Geman and Geman, 1984; Ripley and Sutherland, 1990; Storvik, 1994) are very expensive to evaluate, and so some approximate form may be necessary to deliver acceptable performance when processing long image sequences. Another problem concerns the nature of clutter in our typical applications. The models in (Geman and Geman, 1984; Ripley and Sutherland, 1990; Storvik, 1994) assume simple stationary background texture, very unlike the typical office or outdoor scenes in our experiments. For a fully Bayesian treatment of the problem it would be necessary to construct a prior model for a "typical scene" in the world. The methods proposed by Zhu and Mumford (1998) may prove useful in this context, but again they are very computationally costly. Alternative methods modelling simple aspects of the background have been proposed in (MacCormick and Blake, 1998) and may prove more appropriate for the CONDENSATION framework.

The best approach to designing an observation density which takes into account the entire image at every timestep yet supports real-time or near real-time operation may be to use a multi-scale approach. It may be possible to construct an approximation which uses information from the entire image at coarse scale but only local information at finer scale. The importance-sampling techniques of chapter 6 might be applicable to this problem if a coarse-scale observation model could be used to direct importance sampling of finer-scale images.

8.3 Sampling methods for particle filters

Much of the statistical and signal processing literature related to the CONDENSATION algorithm concerns modifications to the sampling algorithm to improve its efficiency (Gordon et al., 1993; Gordon and Salmond, 1995; Carpenter et al., 1997; Pitt and Shepherd, 1997;

Doucet, 1998). As mentioned in section 8.1 we have not devoted much study to this question, so this section considers which of these modifications may be usefully applied in the CONDENSATION framework. The typical measure of efficiency is the variance of the estimates produced by the algorithm over repeated runs using the same input data. Carpenter et al. (1997) and Doucet (1998) both advocate the use of the "effective sample size" $N_{\text{eff}} \leq N$ as a measure of the efficiency of a sample-set representation. Given a set $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}$ which approximates a density $p(\mathbf{X}_t)$, N_{eff} is related to the correlation between the samples in the set when compared to a set of fair samples from the distribution $p(\mathbf{X}_t)$. If the samples are highly correlated, then the effective sample size is low.

Gordon et al. (1993; 1995) are credited with the first publication of an algorithm equivalent to CONDENSATION. Called the "bootstrap filter," it was inspired directly by (Smith and Gelfand, 1992). Following the weighted bootstrap philosophy the output at each timestep is a *uniformly* weighted sample-set, corresponding to N fair samples from the approximating distribution $\tilde{p}(\mathbf{X}_t|\mathcal{Z}_t)$ at time t. This leads to exactly the same algorithm as set out in figure 3.6 on page 44 but with a rotation of the order of the steps corresponding to one iteration, as follows:

Algorithm 1

1. **start** with a set of uniformly weighted samples $\{(\mathbf{s}_{t-1}^{(n)}, 1/N)\}$.

2. **predict** the new positions by sampling $\mathbf{s}_t'^{(n)} \sim p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(n)})$ for each base $n = 1, \ldots, N$.

3. **measure** to calculate weights $\pi_t^{(n)} = p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t'^{(n)})$ and store as a weighted sample set $\{(\mathbf{s}_t'^{(n)}, \pi_t^{(n)}), n = 1, \ldots, N\}$.

4. **resample** to choose N evenly weighted samples $\{(\mathbf{s}_t^{(n)}, 1/N)\}$ by sampling N times from the set $\{(\mathbf{s}_t'^{(n)}, \pi_t^{(n)}), n = 1, \ldots, N\}$ choosing sample $\mathbf{s}_t'^{(n)}$ with probability $\pi_t^{(n)}$.

Carpenter et al. (1997) point out that if the output of the filter is to be used to estimate moments of the state density, it is more efficient to use the weighted sample-set $\{(\mathbf{s}_t'^{(n)}, \pi_t^{(n)})\}$ before the resampling stage, as in (3.7) on page 39, than the resampled locations $\{(\mathbf{s}_t^{(n)}, 1/N)\}$ used by Gordon et al. In fact, Gordon et al. find that resampling in step 4 to generate N unweighted samples from the set of N weighted samples leads to a poor representation of the state density, since the set $\{(\mathbf{s}_t^{(n)}, 1/N)\}$ contains many repeated

elements. To address this, they recommend "sample boosting" (Gordon and Salmond, 1995) by a factor B_f. This technique involves generating NB_f samples in steps 2 and 3 of algorithm 1, so that the resampling stage in step 4 results in many fewer repeated values in the evenly-weighted set. The sample-boosting algorithm at each timestep is as follows:

Algorithm 2

1. **start** with a set of uniformly weighted samples $\{(\mathbf{s}_{t-1}^{(n)}, 1/N)\}$.
2. **resample** by sampling NB_f indices j_n uniformly (with replacement) from $[1, N]$ to choose base samples $\{\mathbf{s}_t''^{(n)} = \mathbf{s}_{t-1}^{(j_n)}, n = 1, \ldots, NB_f\}$.
3. **predict** the new positions of NB_f samples by sampling $\mathbf{s}_t'^{(n)} \sim p(\mathbf{X}_t | \mathbf{X}_{t-1} = \mathbf{s}_t''^{(n)})$ for each base $n = 1, \ldots, NB_f$.
4. **measure** to calculate weights $\pi_t^{(n)} = p(\mathbf{Z}_t | \mathbf{X}_t = \mathbf{s}_t'^{(n)})$ and store as a weighted sample set $\{(\mathbf{s}_t'^{(n)}, \pi_t^{(n)}), n = 1, \ldots, NB_f\}$.
5. **resample** to choose N evenly weighted samples $\{(\mathbf{s}_t^{(n)}, 1/N)\}$ by sampling N times from the set $\{(\mathbf{s}_t'^{(n)}, \pi_t^{(n)}), n = 1, \ldots, NB_f\}$ choosing sample $\mathbf{s}_t'^{(n)}$ with probability $\pi_t^{(n)}$.

This technique would not seem to provide any advantage in the CONDENSATION framework where moments are computed directly from the weighted sample-set. Typically the measurement stage dominates computation time, and it would be simpler to increase N than to introduce the boosting factor.

As another method to alleviate the problem of resampling identical positions $\mathbf{s}_t^{(n)}$, Gordon et al. also propose "roughening" of the sample set during step 2 of algorithm 2 so that

$$\mathbf{s}_t''^{(n)} = \mathbf{s}_{t-1}^{(j_n)} + \beta_n$$

where $\beta_n \sim R(\cdot)$ is a random variate drawn from some distribution which is a function of the entire sample-set $\{(\mathbf{s}_{t-1}^{(n)}, 1/N)\}$. Depending on the distribution R chosen, this may be equivalent to simply altering the process model $p(\mathbf{X}_t | \mathbf{X}_{t-1})$, otherwise it destroys a meaningful interpretation of the distribution approximated by $\{(\mathbf{s}_t^{(n)}, 1/N)\}$. They implement a scheme using R which is a function of the variance of the entire sample-set and report

improved performance as a result. It would be possible to implement such a scheme in the CONDENSATION framework but it seems preferable to try to understand the reasons for poor performance and either alter the process model if the process noise is too small, or use some kind of importance sampling to redistribute samples without destroying the probabilistic interpretation of the output. Finally, the technique of "prior editing" is proposed. In this procedure samples $\mathbf{s}_t'^{(n)}$ are subjected to a rejection test after the weight $\pi_t^{(n)}$ has been calculated in step 4 of algorithm 2. If the sample is rejected, more samples $\mathbf{s}_t'^{(n)}$ are generated via steps 2 and 3 until one is accepted. Depending on the rejection criterion this may be equivalent to using an importance function as in chapter 6 (and see below) but without correcting the weights accordingly. The advantage of the procedure is that samples generated this way are unlikely to have very low weights, and therefore the variance of the sample weights is reduced. It seems inelegant to destroy the probabilistic interpretation of the sample-sets with this technique, however. Also, as in the case of sample-boosting, the computational load would be large in the CONDENSATION framework since the evaluation of the observation density is costly. Finally, a random number of evaluations of steps 2 and 3 are required for a given iteration so the algorithm no longer runs in a fixed computational bound. It seems, therefore, that simply increasing N would provide similar performance gains in CONDENSATION without disturbing the simplicity of the algorithm.

Carpenter et al. (1997) propose a number of algorithms for efficient selection of base samples which were discussed in section 3.4 on page 40. They also propose a stratified sampling scheme which is appropriate when the process and observation models have a certain tractable form. The idea is to use importance sampling so that instead of sampling new locations from $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ as in standard CONDENSATION, the importance function $\hat{p} \approx p(\mathbf{X}_t|\mathcal{Z}_t)$ is used. The intuition is similar to that of chapter 6 of this thesis, that by choosing an importance function which takes into account the observation data at time t, samples will be positioned more efficiently. In fact, as noted below, Doucet (1998) demonstrates that the optimal importance function (in the sense given below) is $p(\mathbf{X}_t|\mathbf{X}_{t-1}, \mathbf{Z}_t)$. Of course the applicability of this approach depends on the quality of the approximation $\hat{p}(\cdot)$ to $p(\mathbf{X}_t|\mathcal{Z}_t)$. An approximation is derived in (Carpenter et al., 1997) for the bearings-only tracking problem treated by Gordon et al. (1993; 1995) and significantly improved results are shown for the stratified sampling algorithm (in the sense that N_{eff} is larger) compared with the standard bootstrap filter. A stratified sampling scheme is used because the density $\hat{p}$ is constructed as a mixture model, and each mixture component is a separate stratum in

the sampling scheme. An iteration of the stratified filtering algorithm proceeds as follows:

Algorithm 3

1. **start** with a set of weighted samples $\{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)})\}$.

2. **predict** a new sampling density. The new density $p(\mathbf{X}_t|\mathcal{Z}_t)$ is estimated as

$$p(\mathbf{X}_t|\mathcal{Z}_t) \approx \sum_{n=1}^{N} \pi_{t-1}^{(n)} p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(n)}) p(\mathbf{Z}_t|\mathbf{X}_t)$$

   which is approximated using a mixture density

$$\hat{p}(\mathbf{X}_t) = \sum_{n=1}^{N} \hat{\beta}_n \hat{p}_n(\mathbf{X}_t)$$

3. **resample** a set of N indices $j_n \in [1, N]$ picking each index i N_i times where $\mathcal{E}[N_i] = N\hat{\beta}_i$. An efficient algorithm for this resampling is given in (Carpenter et al., 1997).

4. **sample** the $\mathbf{s}_t^{(n)}$ from $\hat{p}$, drawing $\mathbf{s}_t^{(n)} \sim \hat{p}_{j_n}$.

5. **weight** to determine a new weighted sample-set $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}$ where the $\pi_t^{(n)}$ are given by

$$\pi_t^{(n)} \propto \frac{\pi_{t-1}^{(j_n)} p(\mathbf{X}_t = \mathbf{s}_t^{(n)}|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(j_n)}) p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t^{(n)})}{\hat{\beta}_{j_n} \hat{p}_{j_n}(\mathbf{X}_t = \mathbf{s}_t^{(n)})}$$

There is no obvious way to construct $\hat{p}$ for the observation and process densities used in this thesis, so this approach is probably not immediately applicable in the CONDENSATION framework.

Pitt and Shephard (1997) introduce some interesting variations on sampling for particle filters. The first they call the "auxiliary particle filter," which is a particular form of importance sampling which does not require the evaluation of $p(\mathbf{X}_t|\mathbf{X}_{t-1})$, and thus avoids the $O(N^2)$ computation required for that evaluation. As in chapter 6 of this thesis, the aim is to avoid generating samples with low weights, or more generally to reduce the variance of the sample weights at a given iteration. The approach rests on the assumption that for a given base sample $\mathbf{s}_{t-1}^{(n)}$, the distribution $p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(n)})$ is much narrower than the overall prediction distribution $p(\mathbf{X}_t|\mathcal{Z}_{t-1})$. In addition, the observation density $p(\mathbf{Z}_t|\mathbf{X}_t)$ is assumed to be locally smooth, so that the variance of the weights of a set of samples generated from $p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(n)})$ is assumed in general to be much lower than the variance

of the weights of a set of samples generated from $p(\mathbf{X}_t|\mathcal{Z}_{t-1})$. The algorithm works by first computing so-called "first-stage weights" $\{\lambda_{t-1}^{(n)}\}$ where

$$\lambda_{t-1}^{(n)} = \pi_{t-1}^{(n)} p(\mathbf{Z}_t|\mathbf{X}_t = \mu_t^{(n)})$$

and $\mu_t^{(n)}$ is some estimator of $p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(n)})$ — in (Pitt and Shepherd, 1997) $\mu_t^{(n)}$ is taken to be the mean of $p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(n)})$. Following the assumptions given above, it follows that $p(\mathbf{Z}_t|\mathbf{X}_t = \mu_t^{(n)})$ is a good predictor of the weight of a sample generated from the base $\mathbf{s}_{t-1}^{(n)}$, and so the algorithm works by generating more samples from bases with high $\lambda_{t-1}^{(n)}$, in other words by concentrating sampling in areas where the predicted samples are expected to have high measurement likelihoods. One iteration is as follows:

Algorithm 4

1. **start** with a set of weighted samples $\{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)})\}$.
2. **compute** first-stage weights $\lambda_{t-1}^{(n)} = \pi_{t-1}^{(n)} p(\mathbf{Z}_t|\mathbf{X}_t = \mu_t^{(n)})$ where
$$\mu_t^{(n)} = \mathcal{E}[p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(n)})].$$
3. **resample** a set of N indices $j_n \in [1, N]$ picking each $j_n = i$ with probability proportional to $\lambda_{t-1}^{(i)}$.
4. **predict** the new sample positions $\mathbf{s}_t^{(n)}$, drawing $\mathbf{s}_t^{(n)} \sim p(\mathbf{X}_t|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(j_n)})$.
5. **measure** to determine a new weighted sample-set $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}$ where the $\pi_t^{(n)}$ are given by
$$\pi_t^{(n)} \propto \frac{p(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t^{(n)})}{p(\mathbf{Z}_t|\mathbf{X}_t = \mu_t^{(j_n)})}.$$

This algorithm restricts the process density to be of a form which allows $\mu_t^{(n)}$ to be easily computed, but unlike algorithm 3 does not put constraints on the form of the observation density, so the auxiliary particle filter may be useful in the CONDENSATION framework. In the applications shown in (Pitt and Shepherd, 1997) the calculation of $p(\mathbf{Z}_t|\mathbf{X}_t)$ is fast, so the auxiliary filtering, with its N additional evaluations of $p(\mathbf{Z}_t|\mathbf{X}_t)$ in step 2, does not greatly slow the computation. In the CONDENSATION applications considered in this thesis, the evaluation of this observation density dominates the computation, so an auxiliary particle

filter might be expected to run almost twice as slowly as a standard CONDENSATION filter; however Pitt and Shephard report significant increases in performance for some problems using the auxiliary filter. It may be that reducing N for a CONDENSATION filter at the same time as implementing an auxiliary filter will increase performance and this should be investigated.

Pitt and Shephard also consider the problem of "adaption" of a sampling algorithm, by which they mean choosing a particular importance sampling density suited to a given process and observation model to improve the efficiency of sampling. The methods they use are closely related to those of (Carpenter et al., 1997) in algorithm 3 above, and again, while (Pitt and Shepherd, 1997) demonstrates increased performance for certain applications, there is no obvious way to apply the technique to the models used in this thesis. A fixed-lag filter is also developed in (Pitt and Shepherd, 1997). The idea is to store p observations, and to design a filter which approximates the state-density $p(\mathbf{X}_t|\mathcal{Z}_{t+p-1})$. This can be done in the standard particle-filter framework by replacing the samples $\mathbf{s}_t^{(n)}$ by sample-sequences $(\mathbf{s}_t^{(n,1)}, \mathbf{s}_t^{(n,2)}, \ldots, \mathbf{s}_t^{(n,p)})$ drawn from $p(\mathbf{X}_t, \mathbf{X}_{t+1}, \ldots, \mathbf{X}_{t+p-1}|\mathbf{X}_{t-1} = \mathbf{s}_t'^{(n)})$ by repeated application of the process density $p(\mathbf{X}_t|\mathbf{X}_{t-1})$, where $\mathbf{s}_t^{(n,k)}$ is drawn from $p(\mathbf{X}_{t+k-1}|\mathbf{X}_{t-1} = \mathbf{s}_t'^{(n)})$. These samples are then weighted using

$$\pi_t^{(n)} = \prod_{k=1}^{p} p(\mathbf{Z}_{t+k-1}|\mathbf{X}_{t+k-1} = \mathbf{s}_t^{(n,k)}).$$

This is a fixed-window form of the Handschin and Mayne algorithm described in section 1.5 on page 11. Pitt and Shephard also propose using the auxiliary particle filter machinery to weight these sample-sequences where now

$$\lambda_t^{(n)} = \prod_{k=1}^{p} p(\mathbf{Z}_{t+k-1}|\mathbf{X}_{t+k-1} = \mu_t^{(n,k)})$$

and $\mu_t^{(n,k)}$ is, for example, the mean of $p(\mathbf{X}_{t+k-1}|\mathbf{X}_{t-1} = \mathbf{s}_t'^{(n)})$. It might be expected that the standard application, without the auxiliary filtering modification, would be less efficient than use of a fixed-lag sequence-smoothing filter as described in section 7.3 on page 122 since it is very similar in spirit but does not take into account the observation densities $p(\mathbf{Z}_{t+k-1}|\mathbf{X}_{t+k-1})$ when sampling. The auxiliary filter formulation may be advantageous for the applications of CONDENSATION in this thesis, and a comparison with the fixed-lag smoother would be useful.

Finally, Pitt and Shephard (1997) discuss stratified sampling, but with a different motivation to that of Carpenter et al. (1997). They propose to estimate model parameters θ

which are assumed fixed over time by using stratified sampling where each stratum corresponds to a particular value of the parameter vector θ. As evidence builds up from multiple observations, some strata which correspond to unlikely parameter choices will have very low probability, and these can then be replaced by new instantiations for θ. By careful choice of these new θ values, the filter may be able to "home in" on a good estimate of model parameters, however the method of deciding when to discard "unlikely" parameters and the choice of which new parameter vector to replace them with is somewhat *ad hoc*. Nevertheless, some form of this idea may prove useful for example in the case discussed in section 6.4 on page 110 of allowing a hand-tracker template to adapt to different users.

Doucet (1998) presents a review of random sampling methods for Bayesian filtering using an importance-sampling framework as his general case (so for example CONDENSATION is a special case where the importance function is taken to be $p(\mathbf{X}_t|\mathbf{X}_{t-1})$). He shows that the optimal choice of importance function is

$$g(\cdot) = p(\mathbf{X}_t|\mathbf{X}_{t-1}, \mathbf{Z}_t)$$

in the sense that it minimises the variance of the importance weights $\pi_t^{(n)}$ conditional on $\mathbf{s}_{t-1}'^{(n)}$ and $\mathcal{Z}_t$. It is clear that the prior editing procedure of (Gordon et al., 1993) as well as the stratified sampling scheme (Carpenter et al., 1997) in algorithm 3 and the auxiliary particle filter of (Pitt and Shepherd, 1997) are approximations to this optimal importance function. In a sense, the sampling function of chapter 6, which is a mixture of $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ and an approximation to $p(\mathbf{Z}_t|\mathbf{X}_t)$ may be seen as a very crude approximation to $p(\mathbf{X}_t|\mathbf{X}_{t-1}, \mathbf{Z}_t)$. Doucet presents various schemes to approximate the optimal $g(\cdot)$, using Monte-Carlo methods and analytic techniques where applicable. As in (Carpenter et al., 1997; Pitt and Shepherd, 1997) the analytic techniques are not suitable for the models used in this thesis, but it may be that a Monte-Carlo approximation to the optimal function could be found for the CONDENSATION framework and this should be investigated.

8.4 A comparison of the smoothing algorithms

Both of the smoothing algorithms presented in chapter 7 significantly aid the interpretation of the output of a CONDENSATION tracker. One of the major benefits of the CONDENSATION algorithm is that it allows the state density to split into several peaks to transiently represent multiple hypotheses about object configuration. This facility enables the tracker to follow the object while measurements are ambiguous, keeping track of several possible trajectories

until the true object position can once more be confidently established. During the period that the distribution contains multiple peaks, however, the unsmoothed filter may report grossly misleading state estimates even though it ultimately recovers. This is because the state estimates are based on the mean of the distribution, and thus implicitly assume a single peak. The application of a smoothing algorithm concentrates the distribution into those areas which are most likely given the entire tracking sequence, and the result is that peaks caused by temporary clutter distractions tend to be greatly reduced in size. The distribution is then more approximately uni-modal, and its mean is a good estimator for the object configuration.

The sequence-based and two-pass smoothing algorithms of chapter 7 were tested in section 7.2 on page 118 on sequences where the state distribution periodically diverges to form several hypotheses and all but one of these competing hypotheses ultimately dies out. Both algorithms successfully smoothed the test sequences, with slightly improved accuracy from the two-pass algorithm, and this suggests that for tasks of this kind the sequence-based algorithm should be used, given its greater conceptual and computational simplicity. It is anticipated that the two-pass algorithm will come into its own as more complex distributions come to be used while tracking, and more complex state estimates are required than a single configuration at each time-step. A situation could arise where the state density repeatedly splits into competing hypotheses and then merges again, for example if two similar objects move in front of one another. The sequence-based algorithm would be very unlikely to preserve the structure of the trajectories; instead it would tend to choose the most likely single path. The two-pass algorithm, on the other hand, by computing a richer representation of the past history, would be more likely to keep all the likely hypotheses and only reject genuine clutter. It may also be desirable to estimate sample-set variances, either to detect periods of uncertainty, for example due to partial occlusion of an object, or for purposes of parameter estimation as described in section 8.5 below, and the two-pass algorithm seems better suited to this task. Further experimentation is necessary to confirm this, however, since as presented, the sequence-based algorithm has complexity $O(N)$ and the two-pass smoother $O(N^2)$. It may be that using a much larger value of N for the sequence-based smoother will produce more accurate estimates at lower computational cost. Alternatively it may be possible to find some way of reducing the complexity of the two-pass smoother without greatly reducing its effectiveness.

8.5 Learning mixed-state motion models

The mixed-state models used in chapters 5 and 6 were constructed by a combination of dynamical learning using hand-segmented training sequences for individual ARP models and hand-crafted parameter-setting for the discrete transition matrices. In some situations it would be preferable to learn the entire model using a joint estimation procedure. In fact, this is possible using an EM algorithm related to that used in (North and Blake, 1998) for single-state learning from noisy measurements, and it is the subject of current research. The EM algorithm makes use of the smoothing algorithms from chapter 7 and requires estimates of the state variance as well as the mean. Preliminary research shows that the variance estimates are considerably less accurate than the mean estimates for given sample-set size N, so it may be that techniques related to those described in section 8.3 will be necessary to improve the efficiency of the sampler.

8.6 Towards a tracker-driven user-interface

Both chapters 5 and 6 demonstrate hand-trackers as applications for the CONDENSATION algorithm. Some additional development would still be necessary to produce an acceptable fully automatic hand-tracker which could be used as an input device. We take as an example the problem of using hand-tracked input to drive a drawing package as discussed in section 5.3 on page 85. The system in chapter 5 is clearly too slow to function effectively in real time with current hardware, but using the importance-sampling framework of chapter 6, real-time operation might be feasible. The challenge is to combine real-time tracking with a more comfortable hand-position than the rigidly outstretched pose used in chapter 6, and it will probably be necessary to use a model-space with more than 4 dimensions to achieve this, though the 12-dimensional space of section 5.3 may not be necessary. It will be essential to include at least one signalling gesture, if only to distinguish between drawing lines and moving the hand over the page without leaving a mark, and this could perhaps be implemented using a discrete switching mechanism. In order to justify the inclusion of a vision-driven interface it would be desirable to include some functionality which is not already available using a mouse. Perhaps the simplest feature would be to allow variation of stroke-width using some continuous hand-gesture, since the two degrees of freedom offered by a mouse do not allow this control. More research is necessary to determine how well the model-switching paradigm of chapter 5 scales when more gestures are added to the reper-

toire, and indeed what forms of motion model must be built to describe gestures which are useful in a user interface.

Before the system in chapter 6 can be used to control an interactive package it will probably be necessary to devise a robust measure to determine whether or not the hand is in the scene, which may be possible using some form of contour discriminant (MacCormick and Blake, 1998). The discrete model would then be expanded to include a fifth state (figure 8.1), denoting the event that no hand is present in the scene. A better method

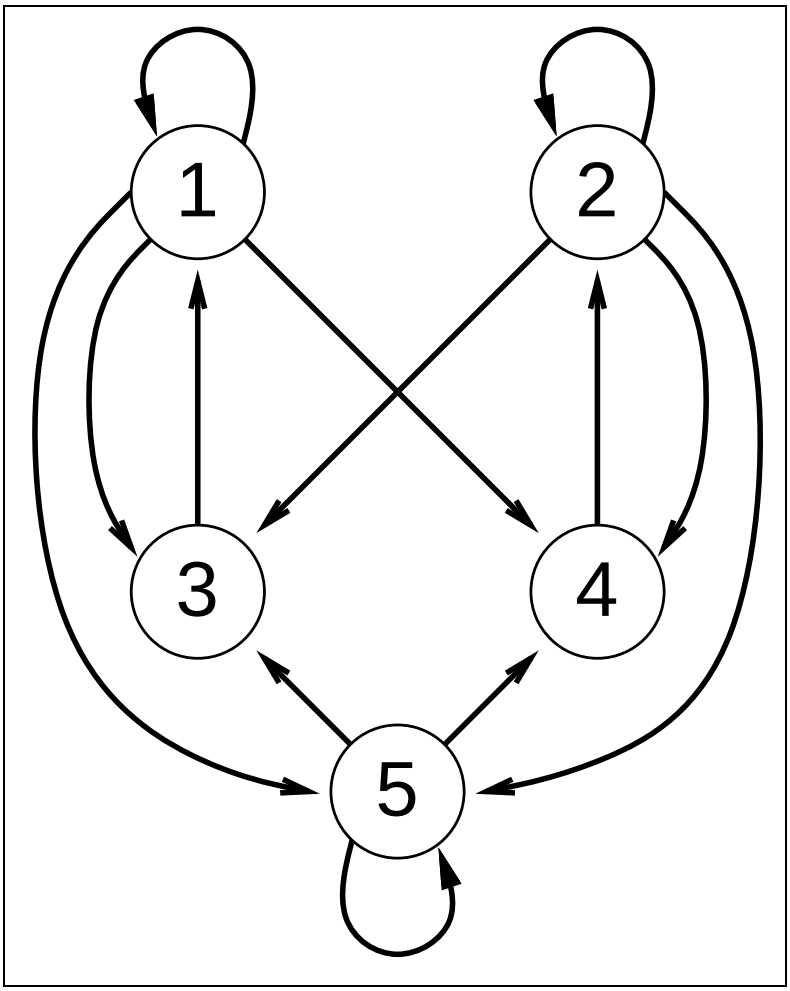


Figure 8.1: **A five-state hand-tracking model encompasses the possibility that no hand is present in the scene.** *States 1 and 2 denote tracking the left and right hand, respectively, states 3 and 4 correspond to initialising a sample to start tracking with either the left or the right model as in figure 6.5 on page 104, and state 5 is entered when no hand is present in the scene.*

of accommodating deformations of the hand-template may also be possible, and this was

discussed in section 6.4 on page 110.

8.7 Avenues of future research

A number of promising avenues of research are suggested by the work presented in this thesis, and a few of them are detailed here:

- **Measurement model.** As has been mentioned several times the measurement models used here are somewhat unsatisfactory. Perhaps the most obvious change would be to replace the purely edge-based observation densities with region-based measurements. This might result in a tracker which could combine the robustness of correlation-matching techniques with the attractive high-dimensional search properties of CONDENSATION.

- **Shape-space.** A limitation of the trackers presented here has been their reliance on a learned shape-space. This has been necessary to confer robustness against clutter, and to avoid the problems of "tangling" evident in early lightly-constrained snake models. While a learned shape-space may always provide the best performance in clutter, it would be useful to have a "default" shape model, controlled by only a few parameters, akin to the default motion models set out in chapter 2, so that CONDENSATION could be applied to general motion segmentation problems in which the object's shape-space is not known beforehand.

- **Kinematic models.** One straightforward alternative to the linear and 2D kinematic models used in preceding chapters would be full 3D kinematic models, which could for example be used for full-body person tracking. This would require explicit occlusion reasoning in the measurement density, which would be a straightforward extension. One might expect better performance across degeneracies in the observations than is provided by uni-modal trackers like the EKF. For example, a tracker using a 3D model to follow a person's arm might receive no information about the rotation angle of the arm when the elbow is straight. An EKF tracker would be forced to commit to a single estimate of the rotation angle, and if this was wrong when the elbow bent once again, the tracking might fail. A CONDENSATION tracker on the other hand might spread the hypotheses to cover all possible rotation angles and thus follow the bending elbow in any configuration.

- **Hierarchical sampling.** There are two obvious circumstances in which a hierarchical sampling scheme might be useful. The first arises when the shape model is hierarchical; for example in the full-body person tracking scenario described above, the position of the arms is strongly constrained by the position of the torso. It may be possible to search for the torso first, and only after a distribution of likely torso positions is found return to the image to search for arm positions conditional on the torso location. Similarly a multi-scale approach might allow gross localisation of the object at coarse scale which would permit searching over much smaller regions of state-space at finer scale. This hierarchical approach would be expected to greatly cut down on image measurements, and thus make a more efficient tracker, and also might bring additional robustness to clutter. It may be that hierarchical sampling schemes can be developed using a mechanism similar to importance sampling.

- **Very high accuracy tracking.** In some applications, notably for film special effects, it is desirable to compute a very accurate estimate of an object's outline, ideally to sub-pixel accuracy around the whole contour. The use of fairly low-dimensional shape models in this thesis has precluded the localisation of the bounding contour of an object to this degree of precision. It might be possible to refine the rougher estimates provided by standard CONDENSATION to produce very high quality estimates. This could also lead to more sophisticated observation models, for example including explicit modelling of motion blur using the velocity estimates produced by the tracker.

- **Graphical models.** As mentioned in section 8.5 it would be appealing to be able to learn mixed-state models without manual segmentation of the discrete labels. A recent advance in research into sequential inference and learning has been the development of "graphical models" which allow complex topologies of nodes representing observations, posterior distributions and conditional dependencies. Currently graphical models can only be constructed with discrete or Gaussian representations of probability distributions (both HMMs and Kalman filters are special cases of graphical models). It may be possible to combine the idea of graphical models with the particle-set propagation techniques of CONDENSATION and a modified learning algorithm to permit non-Gaussian densities throughout a complex topology of conditional distributions.

8.8 Conclusion

The CONDENSATION algorithm presented in chapter 3 of this thesis has been demonstrated to have performance in clutter which is significantly better than that of existing tracking methods based on Kalman filters. Perhaps more interesting, however, is the simplicity and generality of the algorithm, which allows straightforward extensions as described in chapters 5 and 6. The interest shown in the statistical and signal processing communities to improving the efficiency of CONDENSATION-like algorithms, which was summarised in section 8.3, is a testament to the wide range of applications of this approach to Bayesian filtering. Within the field of computer vision many avenues of research could be explored leading from the work in this thesis, and some of these have been discussed in this chapter. The implementation of region-based observation models for CONDENSATION as well as performing mixed-state learning are the subjects of current research, and the general problem of algorithm efficiency is also being explored. The applicability of mixed-state CONDENSATION to the emerging field of perception of action is also of great interest, and it is hoped that as we gain familiarity with the algorithm more potential applications will become apparent.

A

Derivations and proofs

A.1 Derivation of the sampling rule

The correctness of the sampling rule (3.4) on page 35 is proved by first deriving two lemmas from the independence assumption (3.2). (This is similar to the derivation found in (Bar-Shalom and Fortmann, 1988), except that our independence assumptions are explicitly specified.)

Lemma 1

$$p(\mathbf{Z}_t|\mathcal{X}_t, \mathcal{Z}_{t-1}) = p(\mathbf{Z}_t|\mathbf{X}_t).$$

Proof:

$$\begin{aligned} p(\mathcal{Z}_t|\mathcal{X}_t) &= p(\mathbf{Z}_t, \mathcal{Z}_{t-1}|\mathcal{X}_t) \\ &= p(\mathbf{Z}_t|\mathcal{Z}_{t-1}, \mathcal{X}_t)p(\mathcal{Z}_{t-1}|\mathcal{X}_t) \\ &= p(\mathbf{Z}_t|\mathcal{Z}_{t-1}, \mathcal{X}_t)\prod_{i=1}^{t-1} p(\mathbf{Z}_i|\mathbf{X}_i). \end{aligned}$$

(Taking (3.3) at time t and integrating w.r.t. $\mathbf{Z}_t$ yields the reduction of the second term in line 2.) Now, using (3.3) again gives the result.

Lemma 2

$$p(\mathbf{X}_t|\mathcal{X}_{t-1}, \mathcal{Z}_{t-1}) = p(\mathbf{X}_t|\mathbf{X}_{t-1}).$$

Proof:

$$p(\mathbf{X}_t, \mathcal{Z}_{t-1}|\mathcal{X}_{t-1}) = p(\mathbf{X}_t|\mathcal{X}_{t-1})p(\mathcal{Z}_{t-1}|\mathcal{X}_{t-1})$$

from (3.2) so

$$p(\mathbf{X}_t|\mathcal{Z}_{t-1}, \mathcal{X}_{t-1}) = p(\mathbf{X}_t|\mathcal{X}_{t-1}) = p(\mathbf{X}_t|\mathbf{X}_{t-1}),$$

using the Markov assumption (3.1).

Derivation of the propagation formula: consider

$$\begin{aligned} p(\mathcal{X}_t|\mathcal{Z}_t) &= \frac{p(\mathbf{Z}_t|\mathcal{X}_t, \mathcal{Z}_{t-1})p(\mathcal{X}_t|\mathcal{Z}_{t-1})}{p(\mathbf{Z}_t|\mathcal{Z}_{t-1})} \\ &= k_t\ p(\mathbf{Z}_t|\mathcal{X}_t, \mathcal{Z}_{t-1})p(\mathcal{X}_t|\mathcal{Z}_{t-1}) \\ &= k_t\ p(\mathbf{Z}_t|\mathbf{X}_t)p(\mathcal{X}_t|\mathcal{Z}_{t-1}) \quad \text{using lemma 1.} \end{aligned}$$

Now integrating w.r.t. $\mathcal{X}_{t-1}$ gives

$$p(\mathbf{X}_t|\mathcal{Z}_t) = k_t\ p(\mathbf{Z}_t|\mathbf{X}_t)p(\mathbf{X}_t|\mathcal{Z}_{t-1}).$$

The last term can be expanded:

$$\begin{aligned} p(\mathbf{X}_t|\mathcal{Z}_{t-1}) &= \int_{\mathcal{X}_{t-1}} p(\mathbf{X}_t|\mathcal{X}_{t-1}, \mathcal{Z}_{t-1})p(\mathcal{X}_{t-1}|\mathcal{Z}_{t-1}) \\ &= \int_{\mathbf{X}_{t-1}} \int_{\mathcal{X}_{t-2}} p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathcal{X}_{t-1}|\mathcal{Z}_{t-1}) \quad \text{using lemma 2} \\ &= \int_{\mathbf{X}_{t-1}} p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1}|\mathcal{Z}_{t-1}) \end{aligned}$$

which is the required result.

A.2 Asymptotic correctness of the CONDENSATION Algorithm

The CONDENSATION algorithm is validated here by a probabilistic argument showing that the sample-set representation of conditional density is correct, asymptotically, as the size N of the sample set at each time-step gets large. The argument is based on the one by Grenander et al. to justify their factored sampling algorithm for interpretation of static images. They use the standard probabilistic tool of "weak convergence" (Rao, 1973) and the "weak law of large numbers" to show that a posterior distribution inferred by factored

sampling can be made arbitrarily accurate by choosing N sufficiently large. No formal indication is given as to how large N should be for a given level of accuracy, something which is determined in practice by experimentation. Kitagawa (1996) performs simple experiments on simulated data from which he concludes that the mean value of a distribution can in general be estimated using a sample-set size N much lower (he cites an order of magnitude) than that necessary to get an accurate characterisation of the entire distribution.

In the proof that follows, the correctness proof for factored sampling of a static image is made inductive so that it can be applied to successive images in a sequence. This would be sufficient to apply several independent images to the estimation of a static underlying object. A further generalisation takes account of the predictive step (step 2 of the CONDENSATION algorithm) that deals with the dynamics of an object in motion.

A.2.1 Factored sampling

The asymptotic correctness of the factored sampling algorithm (section 3.3 on page 39) is expressed in a theorem of Grenander et al. (1991):

Theorem 3 (Factored sampling) *If $\alpha p_0 p_z$ is an (absolutely continuous) density function (with α a suitable normalisation constant) then for any given value* $\mathbf{X}$

$$\tilde{p}(\mathbf{X}) \to \alpha p_0(\mathbf{X}) p_z(\mathbf{X}), \text{ weakly, as } N \to \infty$$

— pointwise, weak convergence of the density function to the required posterior.

(Recall $\tilde{p}$ is the density function of the random variate $\mathbf{X}$ generated by factored sampling, as defined in section 3.3.) The proof of the theorem was given by Grenander et al.

A.2.2 Dynamic extension of factored sampling

The first step in the extension for dynamic problems is to state a corollary of the theorem above that generalises it slightly to the case where the prior is not known exactly but has itself been simulated approximately.

Cor. 4 (Weak factored sampling) *The sequence $\mathbf{s}_1, \ldots, \mathbf{s}_N$ is now generated by sampling from a density p_s chosen such that*

$$p_s(\mathbf{X}) \to p_0(\mathbf{X}), \text{ weakly, as } N \to \infty,$$

where convergence is uniform with respect to $\mathbf{X}$. *Provided* p_z *is bounded, the random variate* $\mathbf{X}'$ *generated from the* $\{\mathbf{s}_n\}$ *as before has a density function* $\tilde{p}$ *for which*

$$\tilde{p}(\mathbf{X}) \to \alpha p_0(\mathbf{X}) p_z(\mathbf{X}) \text{ weakly, as } N \to \infty$$

and convergence is uniform with respect to $\mathbf{X}$.

The proof of this corollary is straightforward.

A.2.3 Propagation of approximated state density

First note that the samples $\mathbf{s}_t^{(n)}$ generated by the algorithm can themselves be regarded as random variables. Using the corollary it is possible to establish that asymptotically the probability density of any given $\mathbf{s}_t^{(n)}$ converges to the desired probability density $p(\mathbf{X}_t|\mathcal{Z}_{t-1})$. From now on the limit symbol '$\to$' is used to denote weak, uniform convergence of density functions as $N \to \infty$. The correctness result is expressed in the theorem below. We first require a normalisation assumption[1] for the process density, that

$$\int p(\mathbf{X}_t|\mathbf{X}_{t-1})\, d\mathbf{X}_{t-1} \text{ is bounded.} \tag{A.1}$$

Theorem 5 (Weak propagation) *Each sample* $\mathbf{s}_t^{(n)}$, $n = 1, \ldots, N$ *at time* t *is drawn from a distribution with density* $\tilde{p}_t$ *such that*

$$\tilde{p}_t(\mathbf{X}_t) \to p(\mathbf{X}_t|\mathcal{Z}_{t-1}).$$

Proof

The proof is inductive. Suppose the result holds for $\tilde{p}_{t-1}$; then after step 1 of the algorithm in figure 3.6 on page 44, by the corollary, and observing that the sampling probabilities are

$$\pi_{t-1}^{(n)} \propto\ p(\mathbf{Z}_{t-1}|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(n)}),$$

each $\mathbf{s}_{t-1}'^{(n)}$ has a density p'_{t-1} such that

$$p'_{t-1} \to \alpha_{t-1} p(\mathbf{X}_{t-1}|\mathcal{Z}_{t-2}) p(\mathbf{Z}_{t-1}|\mathbf{X}_{t-1})$$

where α_{t-1} is a normalisation constant so that

$$p'_{t-1} \to p(\mathbf{X}_{t-1}|\mathcal{Z}_{t-1}).$$

[1] This assumption is not restrictive in practice but is a little inelegant and perhaps there is a way to do without it.

In step 2 of the algorithm the random dynamical step is applied to $\mathbf{s}_t'^{(n)}$ to give $\mathbf{s}_t^{(n)}$ with density p'' such that

$$\begin{aligned} p''(\mathbf{X}_t) &= \int p(\mathbf{X}_t|\mathbf{X}_{t-1}=\mathbf{s}_t'^{(n)})\ p(\mathbf{s}_t'^{(n)})\, d\mathbf{s}_t'^{(n)} \\ &= \int p(\mathbf{X}_t|\mathbf{X}_{t-1})p'(\mathbf{X}_{t-1})\, d\mathbf{X}_{t-1} \\ &\to \int p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1}|\mathcal{Z}_{t-1})\, d\mathbf{X}_{t-1} \quad \text{(making use of (A.1))} \\ &= p(\mathbf{X}_t|\mathcal{Z}_{t-1}) \end{aligned}$$

and this is the required density function for $\mathbf{s}_t^{(n)}$, establishing the inductive step as required.

Finally the ground instance is straightforward. Initial samples $\mathbf{s}_1'^{(n)}$ are drawn in step 1 from the prior p_0 so that, after step 2 of the algorithm, the $\mathbf{s}_1^{(n)}$ are sampled predictions for time $t = 1$ from a density $\tilde{p}_1$ such that

$$\tilde{p}_1(\mathbf{X}_1) = p(\mathbf{X}_1) \equiv p(\mathbf{X}_1|\mathcal{Z}_0).$$

($\mathcal{Z}_0$ is an empty set) so certainly

$$\tilde{p}_1(\mathbf{X}_1) \to p(\mathbf{X}_1|\mathcal{Z}_0)$$

as required.

Note that convergence has not been proved to be uniform in t. For a given fixed t, there is convergence as $N \to \infty$ but nothing is said about the limit $t \to \infty$. In practice this could mean that at later times t larger values of N may be required, though that could depend also on other factors such as the nature of the dynamical model.

Bibliography

Akashi, H. and Kumamoto, H. (1977). Random sampling approach to state estimation in switching environments. *Automatica*, 13, 429–434.

Anderson, B. and Moore, J. (1979). *Optimal filtering*. Prentice Hall.

Astrom, K. (1970). *Introduction to stochastic control theory*. Academic Press.

Bar-Shalom, Y. and Fortmann, T. (1988). *Tracking and Data Association*. Academic Press.

Bartels, R., Beatty, J., and Barsky, B. (1987). *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann.

Bascle, B. and Deriche, R. (1995). Region tracking through image sequences. In *Proc. 5th Int. Conf. on Computer Vision*, 302–307, Boston.

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.

Baumberg, A. and Hogg, D. (1995a). An adaptive eigenshape model. *Proc. British Machine Vision Conf.*, 87–96.

Baumberg, A. and Hogg, D. (1995b). Generating spatiotemporal models from examples. In *Proc. British Machine Vision Conf.*, 2, 413–422.

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. and Jepson, A. (1996). Eigentracking: robust matching and tracking of articulated objects using a view-based representation. In *Proc. 4th European Conf. Computer Vision*, 329–342.

Black, M. and Jepson, A. (1998). A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *Proc. 5th European Conf. Computer Vision*, 1, 909–924, Freiburg, Germany. Springer Verlag.

Black, M., Yacoob, Y., Jepson, A., and Fleet, D. (1997). Learning parameterized models of image motion. In *Proc. Conf. Computer Vision and Pattern Recognition*, 561–567, San Juan, Puerto Rico.

Blake, A., Curwen, R., and Zisserman, A. (1993a). Affine-invariant contour tracking with automatic control of spatiotemporal scale. In *Proc. 4th Int. Conf. on Computer Vision*, 66–75.

Blake, A., Curwen, R., and Zisserman, A. (1993b). A framework for spatio-temporal control in the tracking of visual contours. *Int. J. Computer Vision*, 11, 2, 127–145.

Blake, A. and Isard, M. (1994). 3D position, attitude and shape input using video tracking of hands and lips. In *Proc. Siggraph*, 185–192. ACM.

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.

Blom, H. and Bar-Shalom, Y. (1988). The interacting multiple model algorithm for systems with Markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33, 8, 780–783.

Bobick, A. and Wilson, A. (1995). A state-based technique for the summarisation and recognition of gesture. In *Proc. 5th Int. Conf. on Computer Vision*, 382–388.

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. (1995). Nonlinear manifold learning for visual speech recognition. In *Proc. 5th Int. Conf. on Computer Vision*, 494–499, Boston.

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`.

Cipolla, R. and Blake, A. (1990). The dynamic analysis of apparent contours. In *Proc. 3rd Int. Conf. on Computer Vision*, 616–625.

Cohen, C. J., Conway, L., and Koditschek, D. (1996). Dynamical system representation, generation, and recognition of basic oscillatory motion gestures. In *Proc. 2nd Int. Conf. on Automatic Face and Gesture Recognition*, 151–156.

Cootes, T., Edwards, G., and Taylor, C. (1998). Active appearance models. In *Proc. 5th European Conf. Computer Vision*, 2, 484–498, Freiburg, Germany. Springer Verlag.

Cootes, T., Hill, A., Taylor, C., and Haslam, J. (1994). The use of active shape models for locating structures in medical images. *J. Image and Vision Computing*, 12, 6, 355–366.

Cootes, T., Taylor, C., Lanitis, A., Cooper, D., and Graham, J. (1993). Building and using flexible models incorporating grey-level information. In *Proc. 4th Int. Conf. on Computer Vision*, 242–246.

Costeira, J. and Kanade, T. (1995). A multi-body factorization method for motion analysis. In *Proc. 5th Int. Conf. on Computer Vision*, 1071–1076.

Crisman, J. D. (1992). Color region tracking for vehicle guidance. In Blake, A. and Yuille, A., editors, *Active Vision*, chapter 7. The MIT Press.

Curwen, R. (1993). *Dynamic and Adaptive Contours.* PhD thesis, University of Oxford.

Davison, A. J. and Murray, D. W. (1998). Mobile robot localisation using active vision. In *Submitted to the 5th European Conference on Computer Vision, Freiburg.*

Dickmanns, E. (1992). Expectation-based dynamic scene understanding. In Blake, A. and Yuille, A., editors, *Active Vision*, 303–336. MIT.

Dickmanns, E. and Graefe, V. (1988). Applications of dynamic monocular machine vision. *Machine Vision and Applications*, 1, 241–261.

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`.

Espiau, B., Chaumette, F., and Rives, P. (1992). A new approach to visual servoing in robotics. *IEEE Trans. Robotics and Automation*, 8, 3, 313–326.

Fernyhough, J., Cohn, A., and Hogg, D. (1996). Generation of semantic regions from image sequences. In *Proc. 4th European Conf. Computer Vision*, 475–484.

Ferrier, N., Rowe, S., and Blake, A. (1994). Real-time traffic monitoring. In *Proc. 2nd IEEE Workshop on Applications of Computer Vision*, 81–88. IEEE.

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.

Foley, J., van Dam, A., Feiner, S., and Hughes, J. (1990). *Computer Graphics: Principles and Practice*, chapter 13, 563–604. Addison-Wesley.

Freeman, W. T. and Roth, M. (1995). Orientation histograms for hand gesture recognition. In Bichsel, M., editor, *Intl. Workshop on automatic face- and gesture-recognition*, Zurich.

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.

Gennery, D. (1992). Visual tracking of known three-dimensional objects. *Int. J. Computer Vision*, 7, 3, 243–270.

Gonzales, R. and Wintz, P. (1987). *Digital Image Processing.* Addison-Wesley.

Goodwin, C. and Sin, K. (1984). *Adaptive filtering prediction and control.* Prentice-Hall.

Gordon, N. and Salmond, D. (1995). Bayesian state estimation for tracking and guidance using the bootstrap filter. *Journal of guidance, control and dynamics*, 18, 6, 1434–1443.

Gordon, N., Salmond, D., and Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. F*, 140, 2, 107–113.

Graf, H., Cosatto, E., Gibbon, D., Kocheisen, M., and Petajan, E. (1996). Multi-modal system for locating heads and faces. In *Proc. 2nd Int. Conf. on Automatic Face and Gesture Recognition*, 88–93.

Grenander, U., Chow, Y., and Keenan, D. (1991). *HANDS. A Pattern Theoretical Study of Biological Shapes.* Springer-Verlag. New York.

Grenander, U. and Miller, M. (1994). Representations of knowledge in complex systems (with discussion). *J. Roy. Stat. Soc. B.*, 56, 549–603.

Hager, G. (1990). *Sensor fusion and planning: a computational approach.* Kluwer Academic Publishers.

Hager, G. and Toyama, K. (1996). X vision: Combining image warping and geometric constraints for fast visual tracking. In *Proc. 4th European Conf. Computer Vision*, 2, 507–517.

Handschin, J. (1970). Monte Carlo techniques for prediction and filtering of non-linear stochastic processes. *Automatica*, 6, 555–563.

Handschin, J. and Mayne, D. (1969). Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International Journal of Control*, 9, 5, 547–559.

Haritaoglu, I., Harwood, D., and Davis, L. (1998). w^4s: A real-time system for detecting and tracking people in 2.5D. In *Proc. 5th European Conf. Computer Vision*, 1, 877–892, Freiburg, Germany. Springer Verlag.

Harris, C. (1992). Tracking with rigid models. In Blake, A. and Yuille, A., editors, *Active Vision*, 59–74. MIT.

Heap, T. and Hogg, D. (1998). Wormholes in shape space: Tracking through discontinuous changes in shape. In *Proc. 6th Int. Conf. on Computer Vision.*

Hennecke, M., Stork, D., and Prasad, K. (1995). Visionary speech: Looking ahead to practical speechreading systems. In *Proceedings NATO ASI Conference on Speechreading by Man and Machine: Models, Systems and Applications*, 331–350. NATO Scientific Affairs Division.

Hogg, D. (1983). Model-based vision: a program to see a walking person. *J. Image and Vision Computing*, 1, 1, 5–20.

Horn, B. and Schunk, B. (1981). Determining optical flow. *J. Artificial Intelligence*, 17, 185–203.

Huttenlocher, D., Noh, J., and Rucklidge, W. (1993). Tracking non-rigid objects in complex scenes. In *Proc. 4th Int. Conf. on Computer Vision*, 93–101.

Intille, S., Davis, J., and Bobick, A. (1997). Real-time closed-world tracking. In *Proc. Conf. Computer Vision and Pattern Recognition.*

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.

Ivins, J. and Porrill, J. (1995). Active region models for segmenting textures and colours. *J. Image and Vision Computing*, 13, 5, 431–438.

Jebara, T., Russell, K., and Pentland, A. (1998). Mixtures of eigenfeatures for real-time structure from texture. In *Proc. 6th Int. Conf. on Computer Vision*, 128–135. Mumbai, India.

Jepson, A. and Black, M. (1993). Mixture models for optical flow computation. *Proc. Conf. Computer Vision and Pattern Recognition*, 760–761.

Ju, S., Black, M., and Jepson, A. (1996). Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *Proc. Conf. Computer Vision and Pattern Recognition*, 307–314.

Kass, M., Witkin, A., and Terzopoulos, D. (1987). Snakes: Active contour models. In *Proc. 1st Int. Conf. on Computer Vision*, 259–268.

Kaucic, R. (1997). *Lip Tracking for Audio-Visual Speech Recognition*. PhD thesis, University of Oxford.

Kaucic, R., Dalton, B., and Blake, A. (1996). Real-time liptracking for audio-visual speech recognition applications. In *Proc. 4th European Conf. Computer Vision*, 376–387, Cambridge, England.

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.

Kjeldsen, R. and Kender, J. (1996). Toward the use of gesture in traditional user interfaces. In *Proc. 2nd Int. Conf. on Automatic Face and Gesture Recognition*, 151–156.

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.

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.

Maes, P. (1993). Alive: An artificial life interactive video environment. In *Visual Proceedings of SIGGRAPH.*

Matthies, L., Kanade, T., and Szeliski, R. (1989). Kalman filter-based algorithms for estimating depth from image sequences. *Int. J. Computer Vision*, 3, 209–236.

Maybank, S., Worrall, A., and Sullivan, G. (1996). A filter for visual tracking based on a stochastic model for driver behaviour. In *Proc. 4th European Conf. Computer Vision*, 540–549, Cambridge, UK.

Menet, S., Saint-Marc, P., and Medioni, G. (1990). B-snakes: Implementation and application to stereo. In *Proceedings DARPA*, 720–726.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 6.

Miller, M., Srivastava, A., and Grenander, U. (1995). Conditional-mean estimation via jump-diffusion processes in multiple target tracking/recognition. *IEEE Transactions on Signal Processing*, 43, 11, 2678–2690.

Murray, D., McLauchlan, P., Reid, I., and Sharkey, P. (1993). Reactions to peripheral image motion using a head/eye platform. In *Proc. 4th Int. Conf. on Computer Vision*, 403–411.

Nagaya, S., Susumu, S., and Oka, R. (1996). A theoretical consideration of pattern space trajectory for gesture spotting recognition. In *Proc. 2nd Int. Conf. on Automatic Face and Gesture Recognition*, 151–156.

North, B. and Blake, A. (1998). Learning dynamical models using expectation-maximisation. In *Proc. 6th Int. Conf. on Computer Vision*, 384–389.

Pahlavan, K., Uhlin, T., and Eklundh, J.-O. (1993). Dynamic fixation. In *Proc. 4th Int. Conf. on Computer Vision*, 412–419, Los Alamitos, CA. IEEE Computer Society Press.

Papoulis, A. (1990). *Probability and Statistics.* Prentice-Hall.

Paragios, N. and Deriche, R. (1998). A PDE-based level-set approach for detection and tracking of moving objects. In *Proc. 6th Int. Conf. on Computer Vision*, 1139–1145.

Petajan, E. and Graf, H. (1995). Robust face feature analysis for automatic speechreading and character animation. In *Proceedings NATO ASI Conference on Speechreading by Man and Machine: Models, Systems and Applications*, 425–436. NATO Scientific Affairs Division.

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 Bing-Hwang, J. (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.

Rao, C. (1973). *Linear Statistical Inference and Its Applications*. John Wiley and Sons, New York.

Rehg, J. and Kanade, T. (1994). Visual tracking of high dof articulated structures: an application to human hand tracking. In Eklundh, J.-O., editor, *Proc. 3rd European Conf. Computer Vision*, 35–46. Springer-Verlag.

Reid, I. and Murray, D. (1996). Active tracking of foveated feature clusters using affine structure. *Int. J. Computer Vision*, 18, 1, 41–60.

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. (1987). *Stochastic simulation*. New York: Wiley.

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.

Rowe, S. and Blake, A. (1996). Statistical feature modelling for active contours. In *Proc. 4th European Conf. Computer Vision*, 560–569, Cambridge, England.

Roy, D., Hlavac, M., Umaschi, M., Jebara, T., Cassell, J., and Pentland, A. (1997). Toco the toucan: A synthetic character guided by perception, emotion, and story. In *Visual Proceedings of SIGGRAPH*, 66.

Rubin, D. (1988). Using the SIR algorithm to simulate posterior distributions. In *Bayesian Statistics*, 3, 395–402. Oxford University Press.

Sclaroff, S. and Isidoro, J. (1998). Active blobs. In *Proc. 6th Int. Conf. on Computer Vision*, 1146–1153, Mumbai, India.

Smith, A. and Gelfand, A. (1992). Bayesian statistics without tears: a sampling-resampling perspective. *The American Statistician*, 46, 84–88.

Smith, S. (1995). Asset-2: Real-time motion segmentation and shape tracking. In *Proc. 5th Int. Conf. on Computer Vision*, 237–244.

Sorenson, H. and Alspach, D. (1971). Recursive Bayesian estimation using Gaussian sums. *Automatica*, 7, 465–479.

Starner, T. and Pentland, A. (1995). Visual recognition of american sign language using hidden markov models. In *Proc. Int. Workshop on Automated Face and Gesture Recognition*.

Storvik, G. (1994). A Bayesian approach to dynamic contours through stochastic sampling and simulated annealing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16, 10, 976–986.

Sullivan, G. (1992). Visual interpretation of known objects in constrained scenes. *Phil. Trans. R. Soc. Lond. B.*, 337, 361–370.

Sullivan, J. and Blake, A. (1997). Exploiting shadows in a visual, hand-driven user interface. In *Proc. British Machine Vision Conf.*, 1, 270–279.

Taylor, M. (1995). *Visually Guided Grasping*. PhD thesis, Dept of Engineering Science, Oxford University.

Terzopoulos, D. and Metaxas, D. (1991). Dynamic 3D models with local and global deformations: deformable superquadrics. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13, 7.

Terzopoulos, D. and Szeliski, R. (1992). Tracking with Kalman snakes. In Blake, A. and Yuille, A., editors, *Active Vision*, 3–20. MIT.

Tomasi, C. and Kanade, T. (1991). Shape and motion from image streams: a factorization method. *Int. J. Computer Vision*, 9, 2, 137–154.

Torr, P. (1997). An assessment of information criteria for motion model selection. In *Proc. Conf. Computer Vision and Pattern Recognition.*

Torr, P. and Murray, D. (1994). Stochastic motion clustering. In *Proc. 3rd European Conf. Computer Vision*, 328–337. Springer-Verlag.

Weber, J. and Malik, J. (1995). Rigid body segmentation and shape description from dense optical flow under weak perspective. In *Proc. 5th Int. Conf. on Computer Vision*, 251–256, Cambridge, MA.

Wellner, P. (1991). The Digital Desk calculator — tangible manipulation on a desk-top display. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Hilton Head.

Wellner, P. (1993). Interacting with paper on the Digital Desk. *Communications of the ACM*, 36, 7, 87–96.

Wildenberg, A. (1997). *Learning and Initialisation for Visual Tracking.* PhD thesis, University of Oxford.

Wren, C., Azarbayejani, A., Darrell, T., and Pentland, A. (1997). Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 7, 780–785.

Yacoob, Y. and Black, M. (1998). Parameterized modeling and recognition of activities. In *Proc. 6th Int. Conf. on Computer Vision*, 6, 120–127.

Zhu, S. and Mumford, D. (1998). Grade: Gibbs reaction and diffusion equations. In *Proc. 6th Int. Conf. on Computer Vision*, 847–854.

Zisserman, A., Blake, A., Rothwell, C., Van Gool, L., and Van Diest, M. (1993). Eliciting qualitative structure from image curve deformations. In *Proc. 4th Int. Conf. on Computer Vision*, 340–345. IEEE.