Learning and Classification of Complex Dynamics

Ben North, Andrew Blake, *Member*, *IEEE*, Michael Isard, and Jens Rittscher

Abstract—Standard, exact techniques based on likelihood maximization are available for learning Auto-Regressive Process models of dynamical processes. The uncertainty of observations obtained from real sensors means that dynamics can be observed only approximately. Learning can still be achieved via "EM-K"—Expectation-Maximization (EM) based on Kalman Filtering. This cannot handle more complex dynamics, however, involving multiple classes of motion. A problem arises also in the case of dynamical processes observed visually: background clutter arising for example, in camouflage, produces non-Gaussian observation noise. Even with a single dynamical class, non-Gaussian observations put the learning problem beyond the scope of EM-K. For those cases, we show here how "EM-C"—based on the CONDENSATION algorithm which propagates random "particle-sets," can solve the learning problem. Here, learning in clutter is studied experimentally using visual observations of a hand moving over a desktop. The resulting learned dynamical model is shown to have considerable predictive value: When used as a prior for estimation of motion, the burden of computation in visual observation is significantly reduced. Multiclass dynamics are studied via visually observed juggling; plausible dynamical models have been found to emerge from the learning process, and accurate classification of motion has resulted. In practice, EM-C learning is computationally burdensome and the paper concludes with some discussion of computational complexity.

Index Terms—Computer vision, learning dynamics, Auto-Regressive Process, Expectation Maximization.

1 Introduction

The paper amplifies a probabilistic framework, first proposed in [8], for estimation (perception) and classification of complex time-varying signals, represented as temporal streams of states. The complexity of signals arising in practical interpretation problems may be too great to allow parameters for an estimation algorithm to be set by hand. Automated learning of dynamics is of crucial importance, therefore, as dynamical model parameters are needed in order to determine the settings of estimation parameters. The framework is particularly general, in several respects, as follows:

1. **Mixed states:** Each state comprises a continuous and a discrete component. The continuous component can be thought of as representing the instantaneous position of some object in a continuum. The discrete state represents the current class of the motion and acts as a label, selecting the current member from a set of dynamical models.
2. **Multidimensionality:** The continuous component of a state is generally multidimensional to represent motion in a higher dimensional continuum, for example, two-dimensional translation as in Fig. 1. Other examples include multispectral acoustic or image signals, or multichannel sensors such as an electro-encephalograph.
3. **Auto-Regressive Process:** Each dynamical system is modeled as an Auto-Regressive Process (ARP) and allowed to have arbitrary order K (the number of time-steps of "memory" that it carries).
4. **Stochastic observations:** The sequence of mixed states is "hidden"—not observable directly but only via observations, which may be multidimensional and are stochastically related to the continuous component of the state. This aspect is essential to represent the inherent variability of response of any real signal sensing system.

Estimation for processes with Properties 2, 3, and 4 has been widely discussed both in the control-theory literature as "estimation" and "Kalman filtering" e.g., [13], [3] and in statistics as "forecasting" e.g., [11]. Learning of models with Properties 2 and 3 is well-understood [13] and once learned can be used to drive pattern classification procedures, as in Linear Predictive Coding (LPC) in speech analysis [35], or in classification of EEG signals [32]. When Property 4 is added, the learning problem becomes harder because the training sets are no longer observed directly, but the problem can be solved [37], [29], [31], [16] by what we term "EM-K"—a combination of Kalman filtering and Expectation Maximization (EM) [12].

Discrete states (Property 1) introduce further complexities. Observing discrete states via continuous, stochastic observations leads to a "Hidden Markov Model" (HMM). The problems of classification, estimation, and learning with HMMs are precisely the three canonical problems of Rabiner for HMMs [35], whose solutions are well-known. In particular, the "Baum-Welch" learning algorithm for HMMs is an instance of EM (with discrete variables whereas EM-K used continuous ones) which has been generalized to "graphical-models" of quite general topology [28]. Investigations of visual classification with HMMs have been reported elsewhere [9]. A little less obvious, HMM

- *B. North, M. Isard, and J. Rittscher are with the Department of Engineering Science, University of Oxford, Oxford OX1 3PJ, UK.*
- *A Blake is with Microsoft Research, 1 Guildhall St., Cambridge CB2, 3NH, UK. E-mail: ablake@microsoft.com.*

Manuscript received 15 June 1999; revised 12 June 2000; accepted 19 June 2000.
Recommended for acceptance by R. Sharma.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 110054.

Fig. 1. **Learning the dynamics of juggling.** The motion of one of three juggled balls is tracked visually (circular outlines) to supply data for dynamical learning, in Section 9.

models arise also under another set of assumptions: Mixed states (Property 1) with ARP dynamics (Property 2 and 3), but with direct (noise-free) observation of the continuous state. The approach has proved to be remarkably successful in vision experiments [10]. However, it is desirable to generalize to noisy observations (Property 4) and that is what we set out to do here.

In the general case (all of Properties 1-4), an exact algorithm exists but has exponential complexity [1] in T, the duration of the time-series for estimation, so approximate algorithms are needed, as in the closely related problem of data-association tracking [3]. However, random sampling algorithms for estimation are highly effective in static, non-Gaussian problems [15], [14], [19], and can be extended to dynamical estimation. In the dynamic context, they are known variously as bootstrap filters [18], Monte-Carlo filters [27], and CONDENSATION [22], [5], [23], and are used in learning theory and experiments, in the form of the "EM-C" algorithm which is developed here. Since this idea was first developed [4], it has been proposed that the learning problem might alternatively be made tractable by a suitable variational approximation of the likelihood for the dynamical parameters [33].

2 MULTICLASS DYNAMICS

Continuous dynamical systems can be specified in terms of a continuous state vector $\mathbf{x}_t \in \mathcal{R}^{N_x}$. In machine vision, for example, $\mathbf{x}_t$ could represent the parameters of a time-varying shape at time t. Multimodal dynamics are represented by appending to the continuous state vector $\mathbf{x}_t$, a discrete state component y_t to make a "mixed" state

$$\mathbf{X}_t = \begin{pmatrix} \mathbf{x}_t \\ y_t \end{pmatrix}, \tag{1}$$

where $y_t \in \mathcal{Y} = \{1, \ldots, N_y\}$ is the discrete component of the state, drawn from a finite set of integer labels. Each discrete state represents a mode of motion, for example "stroke," "rest," and "shade" for a hand engaged in drawing. Experiments [4] have already established the resounding advantages for tracking of using mixed state dynamics as opposed to single state.

Corresponding to each state $y_t = i$ there is a dynamical model, taken to be a Markov model of order K_i that specifies $p_i(\mathbf{x}_t|\mathbf{x}_{t-1}, \ldots \mathbf{x}_{t-K_i})$. It is a Gaussian Auto-Regressive Process (ARP) defined by

$$\mathbf{x}_t = \sum_{k=1}^{K} A_k \mathbf{x}_{t-k} + \mathbf{d} + B\mathbf{w}_t \tag{2}$$

in which each $\mathbf{w}_t$ is a vector of N_x independent random $\mathcal{N}(0,1)$ variables and $\mathbf{w}_t$, $\mathbf{w}_{t'}$ are independent for $t \neq t'$. The dynamical parameters of the model are

- "deterministic" parameters $A_1, A_2, \ldots, A_K$ and $\mathbf{d}$
- "stochastic" parameters B, which are constant multiplicative weights of the stochastic process $\mathbf{w}_t$, and determine the "coupling" of $\mathbf{w}_t$ into the vector valued process $\mathbf{x}_t$.

For convenience of notation, let

$$A = (\, A_1 \quad A_2 \quad \cdots \quad A_K \,).$$

Each state $y \in \mathcal{Y}$ has a set $\lambda^y = \{A^y, B^y, \mathbf{d}^y\}$ of dynamical parameters and we denote the set of models by $\Lambda = \{\lambda^y, y \in \mathcal{Y}\}$. The goal is to learn Λ from example trajectories. Note that the stochastic component $B^y\mathbf{w}_t$ is a first-class part of a dynamical model, representing the degree and the shape of uncertainty in motion, allowing the representation of an entire distribution of possible motions for each state y.

In addition, and independently, state transitions are governed by

$$P(y_t = y' | y_{t-1} = y) = M_{y,y'},$$

the transition matrix for a first-order Markov chain. More generally, transition probabilities can be made sensitive to the context $\mathbf{x}_{t-1}$ in state space, so that

$$P(y_t = y' | y_{t-1} = y, \mathbf{x}_{t-1}) = M_{y,y'}(\mathbf{x}_{t-1}).$$

For example, this could be used to express an enhanced probability of transition into the "resting" state when the hand is moving slowly. The learning algorithm presented below assumes context insensitivity and thereby shirks the problem of trying to find a suitable (learnable) parametric form for the dependence of $M_{y,y'}(\mathbf{x})$ on $\mathbf{x}$. The joint model can be summarized, invoking both independence of the discrete transitions and the Markov properties for continuous and discrete components, as follows:

$$p(\mathbf{X}_t | \mathcal{X}_{0:t-1}, \Lambda) = p_{y_t}(\mathbf{x}_t | \mathbf{x}_{t-1}, \ldots \mathbf{x}_{t-K}) M_{y_{t-1},y_t}, \tag{3}$$

where

$$\mathcal{X}_{t_0:t_1} \equiv (\mathbf{X}_{t_0}, \ldots, \mathbf{X}_{t_1})$$

denotes a sequence of states, and p_y is the density for an ARP (2) with $\{A, B, \mathbf{d}\} = \lambda^y$. Note that initial conditions for $\mathbf{x}$ and y must also be specified, either as fixed values or as prior distributions, and this is discussed later.

3 Maximum Likelihood Learning

Maximum Likelihood Estimation (MLE) for a directly observable dynamical system is related to the well-known Yule-Walker formula [13], [17], [29], [7] for parameter estimation in ARPs, but the formula has to be generalized to include: learning of the offset $\mathbf{d}$ [36], nonasymptotic learning, i.e., from short training sets and dealing with multiple classes of motion.

3.1 Basic MLE

Consider the case of a single motion class, of order K, with dynamical parameters $\lambda = \{A, B, \mathbf{d}\}$. Given a training sequence $\mathbf{x}_1^* \ldots \mathbf{x}_T^*$, with $T > K$, learned deterministic dynamical parameters λ can be obtained from the MLE:

$$A\bar{R} = \bar{\mathbf{R}}_0; \quad \mathbf{d} = \frac{1}{T'}(R_0 - A\mathbf{R}); \quad C = \frac{1}{T'}\left(\bar{R}_{0,0} - A\bar{\mathbf{R}}_0^\top\right), \tag{4}$$

where $C = BB^\top$ and

$$\bar{R} = \begin{pmatrix} \bar{R}_{1,1} & \bar{R}_{1,2} & \cdots & \bar{R}_{1,K} \\ \bar{R}_{2,1} & \bar{R}_{2,2} & \cdots & \bar{R}_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{R}_{K,1} & \bar{R}_{K,2} & \cdots & \bar{R}_{K,K} \end{pmatrix};$$

$$\bar{\mathbf{R}}_0 = \left(\, \bar{R}_{0,1} \quad \bar{R}_{0,2} \quad \cdots \quad \bar{R}_{0,K} \,\right); \qquad \mathbf{R} = \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_K \end{pmatrix}.$$

and the first-order moments R_i and autocorrelations $\bar{R}_{i,j}$ are given by

$$R_i = \sum_{t=K+1}^{T} \mathbf{x}_{t-i}^*;$$

$$R_{i,j} = \sum_{t=K+1}^{T} \mathbf{x}_{t-i}^* \left(\mathbf{x}_{t-j}^*\right)^\top; \qquad \bar{R}_{i,j} = R_{i,j} - \frac{1}{T'} R_i R_j^\top;$$

and $T' = T - K$.

3.1.1 Notes on the MLE

1. The MLE formula (4) is asymptotically (as $T \to \infty$) consistent with the well-known Yule-Walker formula [13], [29] for estimating deterministic parameters A. The Yule-Walker formula approximates true MLE by approximating the second-order moments as

   $$R_{i,j} \approx R_{0,j-i},$$

   which corresponds to assumption of temporal stationarity that may be valid for the parent distribution, but is unlikely to be valid for a finite sample. For example, true MLE can correctly learn an oscillatory process from, say, $1\frac{1}{2}$ cycles of a sinewave, whereas Yule-Walker fails to recover the correct dynamical parameters. We have found in practice that errors introduced by the Yule-Walker approximation can be quite significant.
2. Some standard texts [11] recommend learning the process mean simply by setting it to the sample mean of the training set. Strictly, this is incorrect, in that it is not the MLE for $\overline{\mathbf{x}}$. Again, this is particularly apparent when the training set is oscillatory and of a duration that is not an integer multiple of the period of oscillation. It is approximately correct for a sufficiently long training-set, but there is no reason in practice why, for sufficiently "coherent" oscillations, a dynamical model should not be learned from

relatively few cycles of oscillation. After all, statistical reliability of learned parameters depends not on the total number of cycles in the training set, but on the number of "coherence lengths" [6].

3.2 Learning Several Classes at Once

Now, the training sequence is $\mathbf{X}_1^* \ldots \mathbf{X}_T^*$, each state having the mixed form $\mathbf{X}_t^* = (\mathbf{x}_t^*, y_t^*)$, and the problem is to learn simultaneously the dynamical models $A^y, B^y, \mathbf{d}^y$ corresponding to each discrete state $y \in \mathcal{Y}$. The MLE of each model for a given class y is computed as in (4) but from autocorrelation matrices $\bar{R}^y$, $\bar{\mathbf{R}}_0^y$, $\bar{R}_{0,0}^y$, $\mathbf{R}^y$, and R_0^y that are restricted to times t for which $y_t = y$. They are composed from blocks:

$$\bar{R}_{i,j}^y = \sum_{t=K+1}^{T} \chi_y(y_t^*)\mathbf{x}_{t-i}^* {\mathbf{x}_{t-j}^*}^\top - \frac{1}{T_y} R_i^y R_j^{y\top},$$

where

$$R_i^y = \sum_{t=K+1}^{T} \chi_y(y_t^*)\mathbf{x}_{t-i}^*; \qquad T_y = \sum_{t=K+1}^{T} \chi_y(y_t^*)$$

and χ is an indicator function:

$$\chi_y(v) = \begin{cases} 1 & \text{if } v = y \\ 0 & \text{otherwise.} \end{cases}$$

The $\bar{R}_{i,j}^y$ are then autocorrelation measures restricted to the class y. The normalizing constant T', in (4), is replaced by T_y for each class y in turn. If required, discrete states can be given models of differing order K^y and the autocorrelation matrices $\bar{R}^y$ etc., constructed of the appropriate size for each y.

3.3 Learning the Transition Matrix

Finally, on the assumption that discrete state transitions are context insensitive so that

$$P(y_t = y' | y_{t-1} = y, \mathbf{x}_{t-1}) = P(y_t = y' | y_{t-1} = y) = M_{y,y'}, \quad (5)$$

the MLE for the transition matrix M is constructed from relative frequencies as:

$$M_{y,y'} = \frac{T_{y,y'}}{\sum_{y' \in \mathcal{Y}} T_{y,y'}}, \quad (6)$$

where

$$T_{y,y'} = \sum_{t=2}^{T} \chi_y(y_{t-1}^*)\chi_{y'}(y_t^*).$$

4 Stochastic Observations

For applications, it may be important that observations $\mathbf{z}_t$ are modeled as stochastic with intrinsic error reflecting the limitations of real sensors; this is certainly the case for image and speech signals. Observations are assumed to be conditioned purely on the continuous part $\mathbf{x}$ of the mixed state, independent of y, and this maintains a healthy separation between the modeling of dynamics and modeling of observations. Observations are also assumed to be conditionally independent, both mutually and with respect to the dynamical process. This is expressed probabilistically as follows:

$$p(\mathcal{Z}_{1:t-1}, \mathbf{x}_t | \mathcal{X}_{0:t-1}) = p(\mathbf{x}_t | \mathcal{X}_{0:t-1}) \prod_{i=1}^{t-1} p(\mathbf{z}_i | \mathbf{x}_i), \quad (7)$$

where $\mathcal{Z}_{t_0:t_1} \equiv \{\mathbf{z}_{t_0}, \ldots, \mathbf{z}_{t_1}\}$ denotes a sequence of successive observations. Note that integrating over $\mathbf{x}_t$ implies the mutual conditional independence of observations:

$$p(\mathcal{Z}_{1:t} | \mathcal{X}_{1:t}) = \prod_{i=1}^{t} p(\mathbf{z}_i | \mathbf{x}_i). \quad (8)$$

The observation process is therefore defined by specifying the conditional density $p(\mathbf{z}_t | \mathbf{x}_t)$ at each time t, and often, in experiments, it is taken to be a time-independent function $p(\mathbf{z}|\mathbf{x})$.

For image data, in the special case that background clutter is sufficiently sparse, the observation density can be approximated by singly peaked density such as a Gaussian:

$$p(\mathbf{z}_t | \mathbf{x}_t) \propto \exp -k\|\mathbf{z}_t - h(\mathbf{x}_t)\|^2, \quad (9)$$

where $\| \ldots \|$ is a suitable norm measuring the difference between an observation $\mathbf{z}_t$ and the prediction $h(\mathbf{x}_t)$ based on the hypothesis $\mathbf{x}_t$. More generally, the observation density will have multiple peaks, reflecting the possible contamination of the data with additional elements generated by spurious events or features. In computer vision applications, for example, this occurs when background clutter is present. A one-dimensional illustration of the problem is given in Fig. 2, in which multiple features give rise to a multimodal observation density function $p(\mathbf{z}|\mathbf{x})$ and details are given in [22], [30]. This is similar, but not identical to observation models based on mixtures, as used in HMMs for speech [35]. The difference is that instead of the fixed pattern of mixtures that would be associated with a single discrete state in an HMM, here the placement of density kernels is variable, and is "read" as part of the observation $\mathbf{z}$. This reflects the idea that the observation contains a number of features, only one of which can be valid. The problem of deciding which is the valid feature has been termed a "data-association" problem [3]. Here, the association is not determined unambiguously; instead, all possible associations are held open and weighted in the density function $p(\mathbf{z}|\mathbf{x})$.

In practical vision problems (see later), the observation density function for these experiments is taken to be a product of multimodal densities [6] like the density in Fig. 2. Each density arises from a "measurement line" emanating from the outline of a hypothesised hand, as in Fig. 3.

5 Learning with Stochastic Observations

The learning problem is a problem of Maximum Likelihood estimation with missing variables—all of the state variables $\mathbf{X}_t^*$ are missing because they are only observed indirectly, via a stochastic process. The training sequence, therefore, is a sequence of observations $\mathcal{Z}_{1:T} = (\mathbf{z}_1, \ldots, \mathbf{z}_T)$. This arises in the well-known problem of learning a Hidden Markov Model (HMM) as done in speech analysis [21], [35], which is a special case of the mixed-state learning problem dealt with here. With HMMs, the problem is solved by the Baum-Welch algorithm, a form

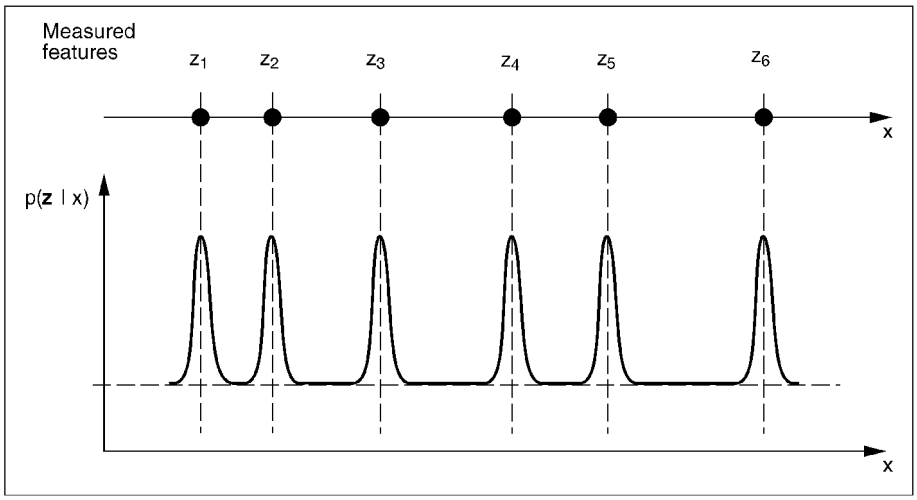


Fig. 2. One-dimensional observation model. A probabilistic observation model allowing for clutter and the possibility of missing the target altogether is specified here as a conditional density $p(\mathbf{z}|x)$. It is well-known to be a mixture of Gaussians, each of standard deviations σ and centered on an image feature [6, chapter 12]. See [31], [16] for the inference of σ from training-data.

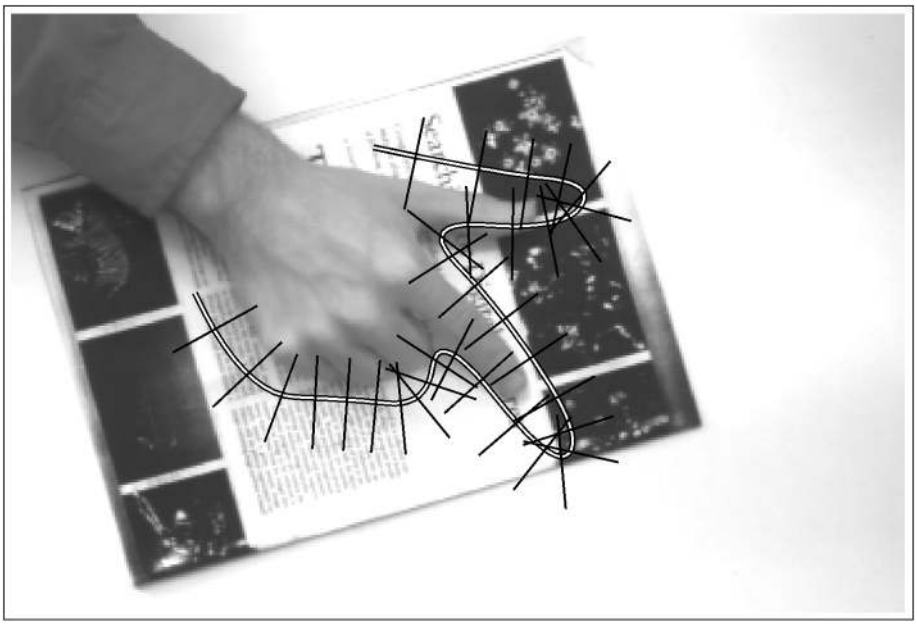


Fig. 3. **Visual Observations.** On each of several measurement lines emanating from the hand contour, the sequence of edge features generates a non-Gaussian observation density of the form seen in Fig. 2.

of Expectation-Maximization (EM) algorithm [12]. What is required here is an EM algorithm for the general problem of learning multimodal dynamics.

The EM algorithm is iterative: Given the training sequence data $\mathcal{Z}_{1:T}$, it produces a series of estimates Λ_i which converge to an MLE for Λ. Each iteration, calculating Λ_i from Λ_{i-1}, consists of an alternate application of an expectation and a maximization step which, for the dynamical learning problem, it can be shown, are as follows:

1. **Expectation.** Expected values of moments and autocorrelations

$$\begin{aligned} &\mathcal{E}[R^y{}_i|\mathcal{Z}_{1:T},\Lambda_{i-1}], \qquad \mathcal{E}[R^y{}_{i,j}|\mathcal{Z}_{1:T},\Lambda_{i-1}], \\ &\mathcal{E}[T^y{}_i|\mathcal{Z}_{1:T},\Lambda_{i-1}], \qquad \mathcal{E}[T^y{}_{i,j}|\mathcal{Z}_{1:T},\Lambda_{i-1}], \end{aligned} \tag{10}$$

conditioned on training data and the latest parameter estimate Λ_{i-1}, need to be computed.

2. **Maximization.** Setting $R^y{}_i, R^y{}_{i,j}, T^y{}_i, T^y{}_{i,j}$ to their expected values in (10), Λ_i is computed as the solution of Λ in the MLE (4) for each y, and in (6).

(Note: The correctness of this approach depends on the $R^y{}_i, R^y{}_{i,j}, T^y{}_i, T^y{}_{i,j}$ being sufficient statistics for Λ and appearing linearly in the log-likelihood $\mathcal{L}(\Lambda)$—details can be found in [37], [31], [16].)

A remaining question is, how to compute the expectations of the required moments and autocorrelations. In the special case $\mathcal{Y} = \{1\}$ of single-class dynamics and assuming a Gaussian observation density, exact methods are available for computing expected moments, using Kalman and smoothing filters [13], either by an extension of the usual forward and backward filters [37], [16], or by using an "augmented state" and the standard forward/backward filters [31]. For multiclass dynamics and/or non-Gaussian observations, exact computation is not feasible, but good

Iterate for $t = 1, \ldots, T$.

Construct the particle-set $\{(\mathbf{X}_{t|t}^{(n)}, \ldots, \mathbf{X}_{t-K|t}^{(n)}), \pi_t^{(n)}\}, n = 1, \ldots, N$ for time t.

For each n:

1. **Sample** (with replacement) $m \in \{1, \ldots, N\}$ with probability $\pi_{t-1}^{(m)}$.
2. **Predict** by sampling from

$$p\left(\mathbf{X}_t \mid \mathbf{X}_{t-K|t-1}^{(m)}, \ldots, \mathbf{X}_{t-1|t-1}^{(m)}\right)$$

to choose $\mathbf{X}_{t|t}^{(n)}$. For multi-class ARPs this is done in two steps.

Discrete: Choose $y_t^{(n)} = y' \in \mathcal{Y}$ with probability $M_{y,y'}$, where $y = y_{t-1}^{(m)}$.

Continuous: Compute

$$\mathbf{x}_{t|t}^{(n)} = \sum_{k=1}^{K} A_k^y \mathbf{x}_{t-k|t-1}^{(m)} + \mathbf{d}^y + B^y \mathbf{w}_t^{(n)},$$

where $y = y_t^{(n)}$ and $\mathbf{w}_t^{(n)}$ is a vector of standard normal random variates.

3. **Observation weights** $\pi_t^{(n)}$ are computed from the observation density, evaluated for the current observations $\mathbf{z}_t$:

$$\pi_t^{(n)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{x}_{t|t}^{(n)}),$$

then normalised multiplicatively so that $\sum_n \pi_t^{(n)} = 1$.

4. **Update history:**

$$\mathbf{X}_{t-k|t}^{(n)} = \mathbf{X}_{t-k|t-1}^{(m)}, \; k = 1, \ldots, K.$$

Fig. 4. **The CONDENSATION algorithm for forward propagation.**

approximations can be achieved based on propagation of "particle sets," as explained in Section 6.

6 Particle Smoothing Filter

A "particle-set" $\{(s_1, \pi_1), \ldots, (s_n, \pi_N)\}$ is defined as a sample $\{s_i\}$ with associated weights $\{\pi_i\}$. Such a set is said to represent approximately a particular (multivariate) distribution if choosing an i with probability proportional to π_i and then setting $\mathbf{x} = s_i$ generates a random variable $\mathbf{x}$ drawn (approximately) from the distribution.

The smoothing filter described here constructs particle sets which represent distributions[1] $p(\mathcal{X}_{t-K:t}|\mathcal{Z}_{1:T})$ for $t = K, \ldots, T$, and from which the autocorrelations $R_{i,j}$ needed for learning can be estimated.

1. In full, the distribution is $p(\mathcal{X}_{t-K:t}|\mathcal{Z}_{1:T}, \Lambda)$, but the Λ may be omitted for simplicity.

6.1 Forward Filter

The CONDENSATION algorithm [22] is a form of sample-based forward filter that can be extended to mixed states [25], to construct samples from $p(\mathcal{X}_{t-K:t}|\mathcal{Z}_{1:t})$, as a first step on the way to sampling from $p(\mathcal{X}_{t-K:t}|\mathcal{Z}_{1:T})$. The CONDENSATION forward algorithm is given in Fig. 4.

6.1.1 Notes on the Algorithm

1. The particle set is taken to be of fixed size N in each time-step t. Size N is chosen as large as possible, for the most accurate results, to fit within a given computational resource. In perception problems, the requirement may be for processing to keep pace with the cycle time of a sensor generating observations.
2. If the orders K^y of models are allowed to differ, then take $K = \max_y K^y$ in the CONDENSATION algorithm.

Initialise at $t = T$:

$$\psi_T^{(n)} = \pi_T^{(n)},\ n = 1, \ldots, N.$$

Compute smoothing-weights at time t, in terms of those at time $t+1$, as follows:

1. **Prediction probabilities**

$$a_t^{(m,n)} = p\left(\mathbf{X}_{t+1} = \mathbf{X}_{t+1|t+1}^{(m)} \mid (\mathbf{X}_{t-k|t} = \mathbf{X}_{t-k|t}^{(n)},\ k = 0, \ldots, K-1)\right),$$

calculated for a multi-modal ARP.

2. **Backward weights:**

$$\beta_t^{(n)} = \sum_{m=1}^{N} \psi_{t+1}^{(m)} a_t^{(m,n)} / \gamma_t^{(m)},\ \text{for } n = 1, \ldots, N.$$

where

$$\gamma_t^{(m)} = \sum_{n=1}^{N} \pi_t^{(n)} a_t^{(m,n)},\ \text{for } m = 1, \ldots, N.$$

3. **Smoothing weights:**

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

then normalise multiplicatively so that $\sum_n \psi_t^{(n)} = 1$.

Fig. 5. **The** CONDENSATION **backward algorithm.**

3. Particle-sets in the algorithm incorporate a history $(\mathbf{x}_{t|t}^{(n)}, \ldots, \mathbf{x}_{t-K|t}^{(n)})$, which in fact includes one element $\mathbf{x}_{t-K|t}^{(n)}$ extra to what is strictly needed for prediction. The extra element implements the "augmented state" that is needed to estimate all the autocorrelations $R_{i,j}$ needed for learning.
4. The algorithm needs to be initialized. This requires that the $(\mathbf{X}_{-k|0}^{(n)}, k = 0, \ldots, K-1)$ be drawn from a suitable (joint) prior for the multimodal process.
5. The prior for initialization may be hard to come by in practice. A straightforward alternative is available for any dynamical process which is stable and irreducible. This is to initialize the variables above in any reasonable manner, for instance,

$$y_0^{(n)} = 1, \text{and } \mathbf{x}_{-k|0}^{(n)} = \mathbf{0},\ k = 0, \ldots, K-1,$$

and run the algorithm for many iterations, without observations and setting $\pi_t^{(n)} = 1/N$. After a sufficiently long time, t_0, a statistical steady state, should be reached and can be used, thenceforth, as an initial state for subsequent runs of the algorithm.
6. The forward algorithm has computational complexity $O(NT)$, provided the sampling in Step 1 is done appropriately, for example, "deterministic sampling" [27]—see [24, Section 4.4] for details.

7 FORWARD-BACKWARD SAMPLING

The backward pass for single-class dynamics, described in [26], extends to multiple classes as in Fig 5. It produces a supplementary set of "smoothing" weights $\psi_t^{(n)}$ with the property that choosing

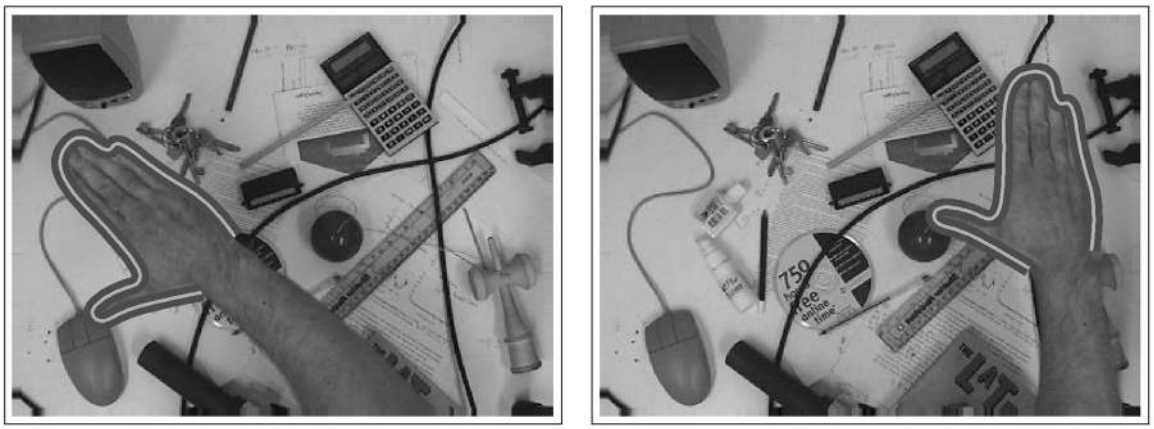
$$\left(\mathbf{X}_{t-K|t}^{(n)}, \ldots, \mathbf{X}_{t|t}^{(n)}\right)$$

for some n, with probability $\psi_t^{(n)}$, generates (in the limit $N \to \infty$) random variates from the joint distribution

$$p((\mathbf{X}_{t-K}, \ldots, \mathbf{X}_t) | \mathcal{Z}_{1:T})$$

for sections of state-sequences conditioned on the entire training set. This allows expectations of autocorrelations and frequencies to be approximated as:

$$\mathcal{E}[R_{i,j}^y] \approx \sum_{n=1}^{N} \sum_{t=K+1}^{T} \chi_y\left(y_{t|t}^{(n)}\right) \psi_t^{(n)} \mathbf{x}_{t-i|t}^{(n)} \left(\mathbf{x}_{t-j|t}^{(n)}\right)^\top \quad (11)$$



(a) (b)

Fig. 6. **Training data.** Two fields from a 5-second training sequence (sampled at 50 fields per second, so that the sequence consists of 250 fields), in which the hand oscillates to and fro over a cluttered desk. The white outlines are estimated by EM, as a by-product of training (see text).

$$\mathcal{E}[R_i^y] \approx \sum_{n=1}^{N}\sum_{t=K+1}^{T} \chi_y\left(y_{t|t}^{(n)}\right)\psi_t^{(n)}\mathbf{x}_{t-i|t}^{(n)} \tag{12}$$

$$\mathcal{E}[T_y] \approx \sum_{n=1}^{N}\sum_{t=1}^{T} \chi_y\left(y_{t|t}^{(n)}\right)\psi_t^{(n)} \tag{13}$$

$$\mathcal{E}[T_{y,y'}] \approx \sum_{n=1}^{N}\sum_{t=2}^{T} \chi_y\left(y_{(t-1)|t}^{(n)}\right)\chi_{y'}\left(y_{t|t}^{(n)}\right)\psi_t^{(n)}. \tag{14}$$

7.1 Notes on the Smoothing Algorithm

1. Backward weights. The algorithm works by applying standard Bayesian forward-backward fusion

   $$p(\mathcal{X}_{t-K:t}|\mathcal{Z}_{1:T}) \propto p(\mathcal{X}_{t-K:t}|\mathcal{Z}_{1:t-1})p(\mathcal{Z}_{t:T}|\mathcal{X}_{t-K:t})$$

   in Step 3. The $p(\mathcal{X}_{t-K:t}|\mathcal{Z}_{1:t-1})$ term is conveyed by sampling from the forward weights $\pi_t^{(n)}$, as in the forward algorithm. The backward weights $\beta_t^{(n)}$ carry the other term, as it can be shown straightforwardly that

   $$\beta_t^{(n)} \approx p(\mathcal{Z}_{t:T}|\mathcal{X}_{t-K:t}),$$

   the approximation being unbiased in the limit $N \to \infty$.
2. Computational complexity. The computational complexity of the algorithm is $O(N^2T)$ because of the sums computed in Step 2.
3. Successive averaging. One way of reducing complexity is by successive averaging, as follows: Suppose $N_{\min}$ is the smallest value of N with which a particular motion can be successfully "tracked." The smoothing algorithm can be run Q times with $N_{\min}$ particles, where $N = QN_{\min}$. Then, from the qth run, expectations are estimated as $\mathcal{E}_q[R_{i,j}^y]$ and a grand estimate

   $$\mathcal{E}[R_{i,j}^y] = \frac{1}{Q}\sum_{q=1}^{Q}\mathcal{E}_q[R_{i,j}^y]$$

   is formed. The grand estimate can probably be expected to be of similar quality to the original since it is still based on a total of N particles per time-step. Computational cost is $O(Q(N_{\min})^2T)$ which is $O(NT)$ given that $N_{\min}$ is a constant. However, the complexity reduction is only felt when $N > N_{\min}$ and, in practice, $N_{\min}$ may be large.

8 Results: Learning from Image Sequences with EM-C

The EM-C algorithm has been applied in two different visual scenes, both involving clutter. The first involves a hand moving over a desktop, illustrative of the potential application of this sort of technology for user interfaces, for example, the Xerox "Digital Desk" concept [38]. The second, a little less demanding in terms of the density of clutter, has instead the additional complexity of multiclass dynamics, the classes corresponding to the different phases of the juggling cycle.

8.1 Single-Class Dynamics: Digital Desk

In the training and test sequences used here (Fig. 6), a hand moves, without flexing, over the desk surface, and so can be regarded as a two-dimensional rigid body. This implies that the state vector $\mathbf{x}_t$ is four-dimensional, so that $\mathbf{x} \in \mathcal{S}_E$, a "shape-space" of Euclidean Similarities [6]. The space can be parameterized in terms of x, y translation, rotation and zoom. Within this four-dimensional space, and since natural motions of the hand over the desk involve roving to and fro, it is reasonable to model them as a family of damped oscillations

$$\ddot{\mathbf{x}} = F_1\dot{\mathbf{x}} + F_2\mathbf{x} + G\mathbf{w},$$

where $\mathbf{w}$ is a Wiener noise process, and F_1, F_2, G are matrix (4×4) constants. In discrete-time (with sampling interval τ), this has the form [2] of an ARP (2) with order $K = 2$. The aim of EM-C learning is to estimate the 4×4 matrices A_1, A_2, B, and the vector $\mathbf{d}$. In order to perform the EM iterative procedure, some initial values of the model parameters $\lambda = (A_1, A_2, B, \mathbf{d})$ must be fixed. Deterministic parameters are set for "constant velocity" dynamics, so that [6, chapter 9]

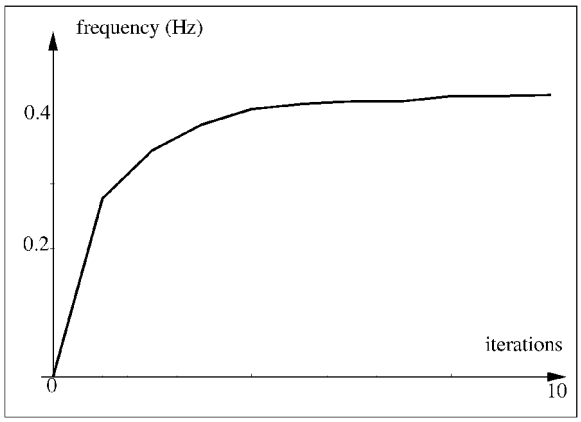


Fig. 7. **Convergence of EM-C learning.** Convergence of the frequency of the dominant oscillatory mode. Other parameters converge similarly rapidly.

$A_1 = 2I,\ A_2 = -I$, where I is the (4×4) identity matrix, and offset $\mathbf{d}$ is set to $\mathbf{0}$. Stochastic parameter-matrix B is chosen to be diagonal,

$$B = \mathrm{diag}\big(b_x, b_y, b_{\mathrm{rot}}, b_{\mathrm{zoom}}\big)$$

so that the four state-space parameters are initially decoupled, and set to give physically reasonable values for the drift that can occur, from rest, in one time-step ($\tau = 1/50$s):

$$b_x = 10 \text{ pixels},\ b_y = 10 \text{ pixels},\ b_{\mathrm{rot}} = 0.01 \text{ rad},\ b_{\mathrm{zoom}} = 1\%.$$

These values allowed the CONDENSATION algorithm to track successfully, in practice, given a sufficiently large particle-set size N. Initial conditions for $\mathbf{x}_t$ must also be set and this was done by fixing template configurations $\mathbf{x}_t$ at times $t = 1, 2$ to fit the images in those first two time-steps. The resulting EM-C learning (with particle sets of size $N = 2,048$) converges rapidly, achieving stable parameter settings after 10 iterations or so, as Fig. 7 shows.

Incorporating the learned model into the CONDENSATION estimation process should enable particles to be concentrated more efficiently. This allows the hand motion to be estimated correctly with a smaller number N of particles in each time step, as Fig. 8 shows. In EM-C training, $N = 2,048$ particles per time-step were used with the "constant velocity" dynamical model above, to give good approximations to the expected values of the moments, transition counts, and durations in (11), (12), (13), and (14). With the learned model, $N = 32$ suffices for correct tracking on the training set. Out of 16 independent test sets, motions

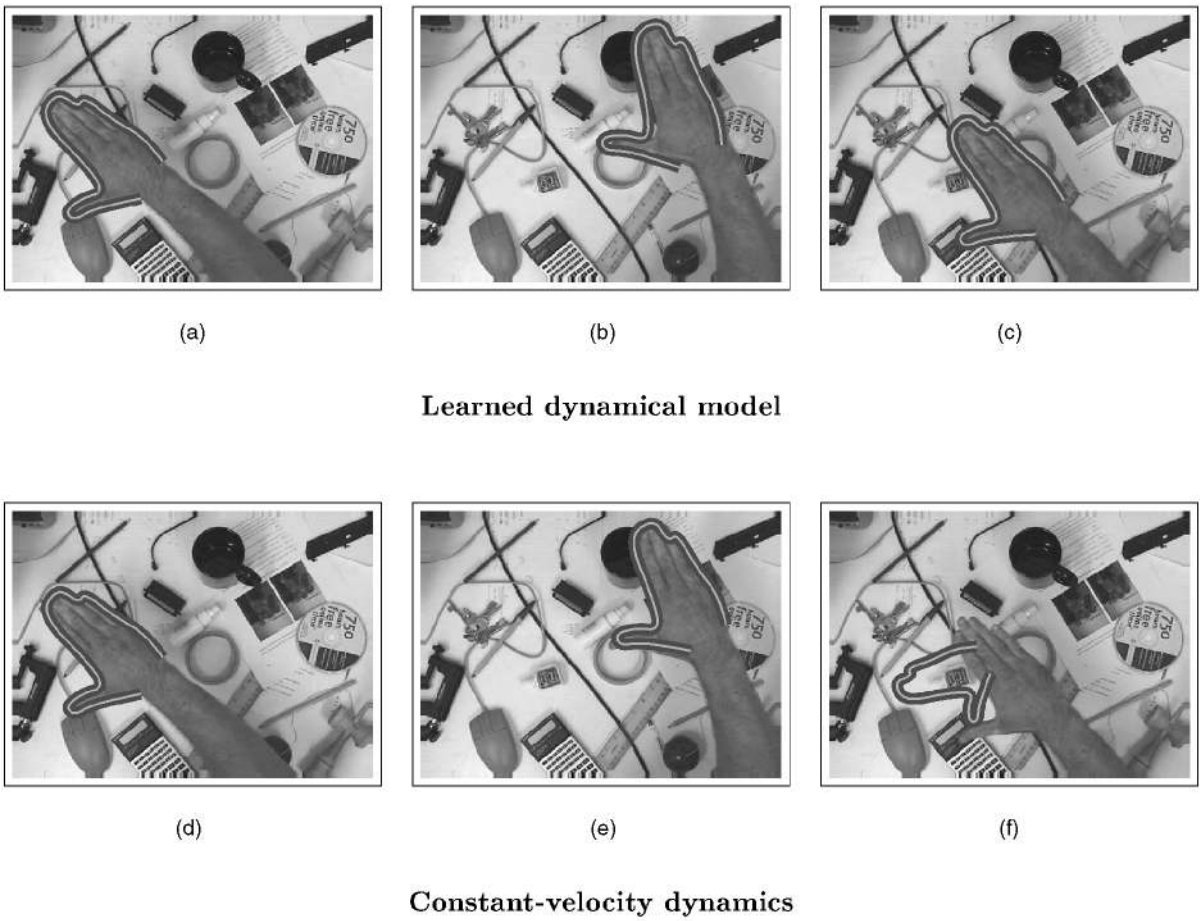


Fig. 8. **Testing learned dynamics.** Three fields from a 5-second test sequence consisting of a motion similar to that in the training sequence above. The displayed outlines show estimated motion for a rather "lean" CONDENSATION filter using just $N = 64$ particles per time-step. Note the failure after $t = 2.00$ seconds for the "constant velocity" model (as used above for training, but with $N = 2,048$). This occurs as the hand-motion reaches the end of its travel, but constant velocity prediction means that the hand is "expected" to sail on. (a) $t = 0.00$s, (b) $t = 1.00$s, (c) $t = 2.00$s, (d) $t = 0.00$s, (e) $t = 1.00$s, and (f) $t = 2.00$s.

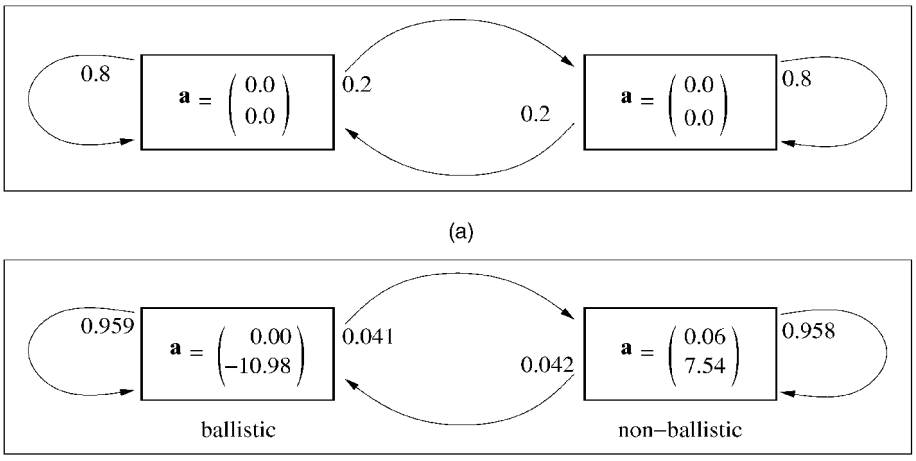


Fig. 9. **Learning two-class dynamics for juggling.** The two emergent motion classes turn out to correspond to ballistic and nonballistic motions. Note that the units of acceleration $\mathbf{a}$ are $\mathrm{m.s}^{-2}$, so that the acceleration for the ballistic class is close to the value $\mathbf{g} = -9.8\mathrm{m.s}^{-2}$ due to gravity. (a) Initial Model. (b) Learned model.

in 13 cases were estimated correctly with just $N = 64$ particles. ("Correct estimation" implies freedom from unrecoverable error judged relative to ground truth marked by eye. This is fairly well-defined given that tracking error tends to have a binary "all or nothing" behavior: Either the error is stable, recovering over time, or it diverges over time.) Using the unlearned, constant-velocity model, tracking failed for $N = 64$ on all 16 test sets. The failure to track motion occurs shortly after the motion of the hand reverses rapidly, a possibility that is anticipated by the learned model. Practically, filtering with $N = 64$ particles per time-step is sufficiently "lean" to load just 10 percent of the real time capability of a desktop workstation such as an SGI Octane. (This is potentially important for "digital desk" applications in which the hand is simply an input device and the remaining 90 percent of capacity remains available for the main graphics applications.)

9 Results: Multiclass Dynamics and Juggling

The visually tracked motion of a juggler's ball (Fig. 1) is used here to explore the learning of multiclass dynamics. Juggling takes place in a plane parallel to the image plane so that the outline of the ball is described simply by a two-dimensional state vector $\mathbf{x}$. From a juggler's point of view, the juggling cycle separates conceptually into four phases: *throw*, *ballistic*, *catch*, *carry*, then back to *throw*. It is an open question, however, whether these phases have sufficiently distinct dynamics to be classifiable from visual data. Here, experiments used both two- and three-class models. With two classes, learning and classification were entirely automatic. With three classes, learning was automatic, but some experimentation with model constraints was needed to obtain complete classifications.

9.1 Two Classes

The first experiment was to learn a two-state dynamical model, with the expectation that this might separate the process into ballistic and nonballistic phases. Each class $y \in \{1, 2\}$ is modeled by a particular form of second-order ARP:

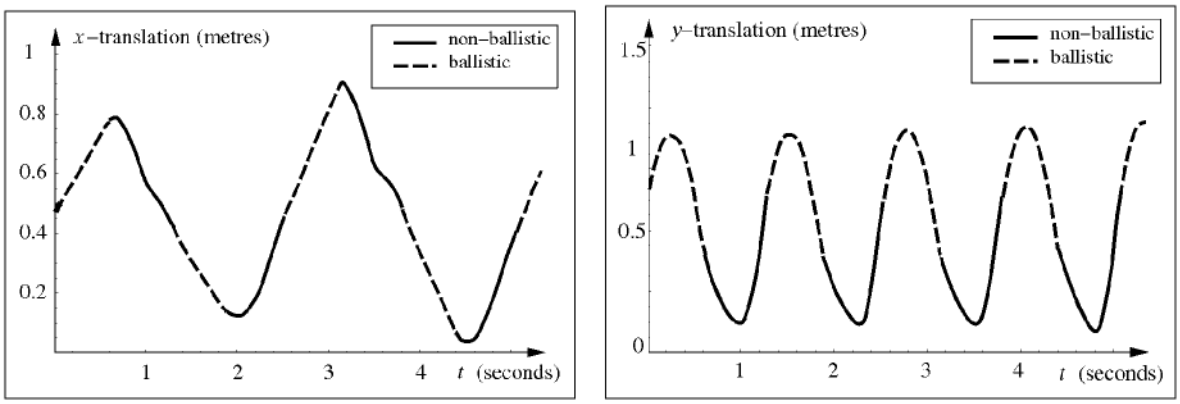
$$\mathbf{x}_t = 2\mathbf{x}_{t-1} - \mathbf{x}_{t-2} + \mathbf{d}^y + B^y\mathbf{w}_t,$$

in which the parameter $\mathbf{d}^y$ determines the (constant) acceleration. This is equivalent to a continuous-time model

$$\ddot{\mathbf{x}} = \mathbf{a} + G\mathbf{w}$$

with a fixed acceleration parameter $\mathbf{a}$ that is proportional to $\mathbf{d}$, the constant of proportionality depending simply on camera calibration and the interval τ between video frames. Visual observations are, as for the hand-tracking experiment, earlier. The duration T of the training sequence is 5.3 seconds (264 video fields), covering four juggling half-cycles. In each iteration of the EM learning algorithm, the E-step used a particle smoothing filter with $N_{\min} = 750$ particles and successive averaging over $Q = 5$ runs. The EM algorithm was initialized with bland dynamical parameters for each class—Brownian motion, unbiased ($\mathbf{a} = 0$), and driven by isotropic noise ($B = bI$, where I is the 2×2 identity matrix) with a physically reasonable magnitude $b = \sqrt{3}$ pixels. The discrete state transition matrix M is initialized symmetrically (Fig. 9) such that each state has a short initial lifetime of $1/(1 - 0.8) = 5$ time-steps, or 0.1 seconds. To avoid excessively slow convergence in EM, the value of B is constrained to be constant throughout learning—i.e., it is fixed a priori. Monitoring convergence of the EM algorithm suggested that dynamical parameters and transition probabilities had substantially converged after six iterations, and EM was continued to the 12th iteration. (When the same experiment was tried with B, also variable and being learned, the B-values failed to converge within a practical time (e.g., 20 hours).)

The resulting learned dynamics are shown in Fig. 9 and show a clear separation into a ballistic class (acceleration $\mathbf{a} \approx \mathbf{g}$, due to gravity) and a nonballistic one with strong upward acceleration. The mean duration of the ballistic



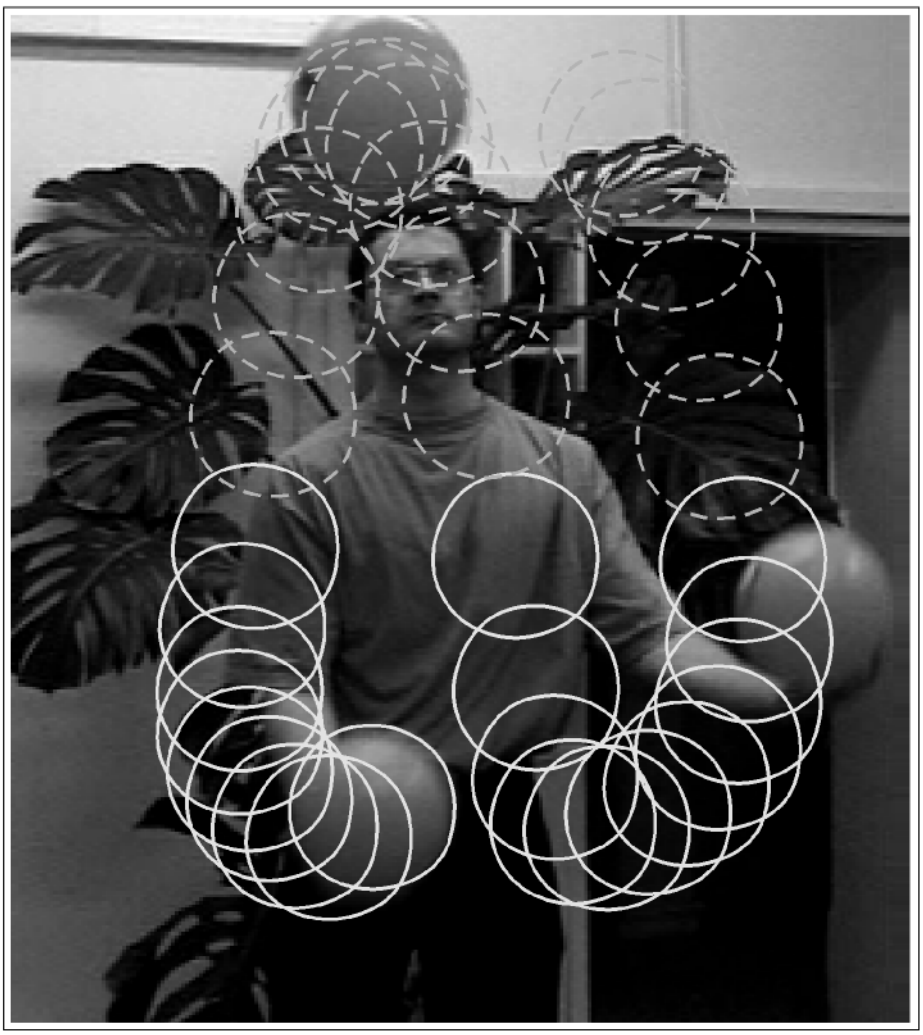
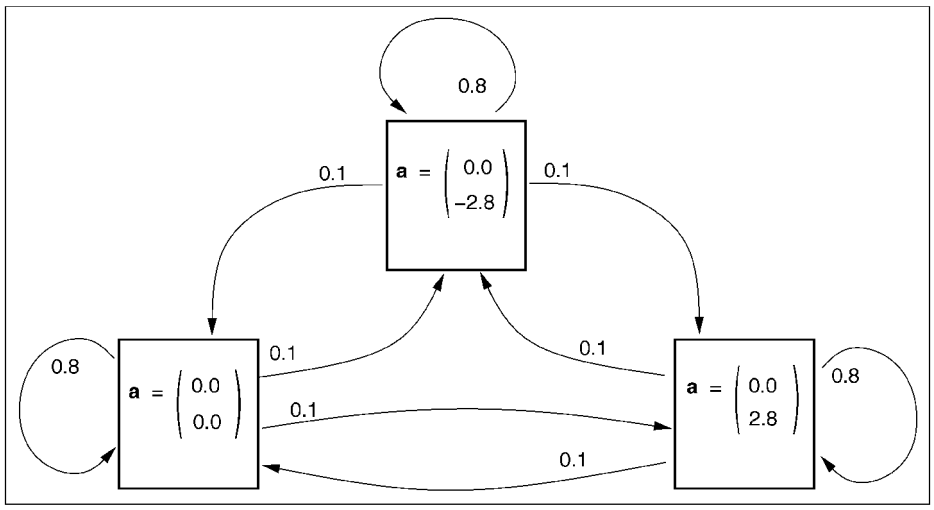


Fig. 10. **Motion classification.** Two complete juggling cycles are classified using particle smoothing, with learned class dynamics, into ballistic (dashed) and nonballistic (solid) motion. One of the classified cycles is shown overlaid on the final image frame of the cycle.

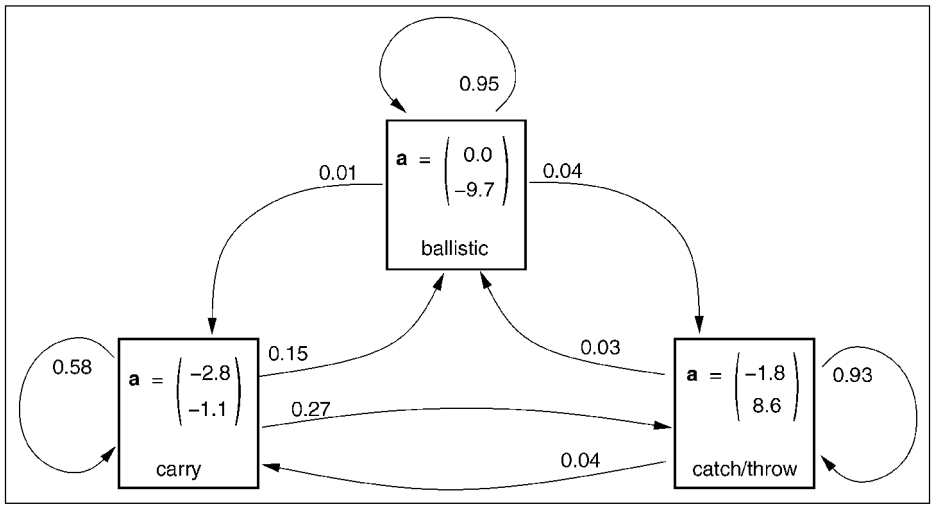
state can be calculated from its probability of reentry, as $1/(1-0.959) = 24.4$ frames, which is close to 0.5 seconds and the mean duration of the nonballistic state is very similar. This gives a total mean half-cycle time close to 1 second, reasonably consistent with the actual value for the training data of 1.25 seconds. The equal distribution of duration between the ballistic and nonballistic also appears consistent with the training data, and with the expectations of the juggler.

9.1.1 Classification

Having obtained the dynamical model, it can be used with independent test data as a motion classifier. The duration of the test sequence is also five seconds and a particle smoothing filter with $N = 1,000$ particles is used to generate the classification. Results in Fig. 10 show consistent classification into ballistic and nonballistic classes over two cycles. The figure clearly shows the ballistic phase occupying the upper part of each trajectory, as expected.



(a)



(b)

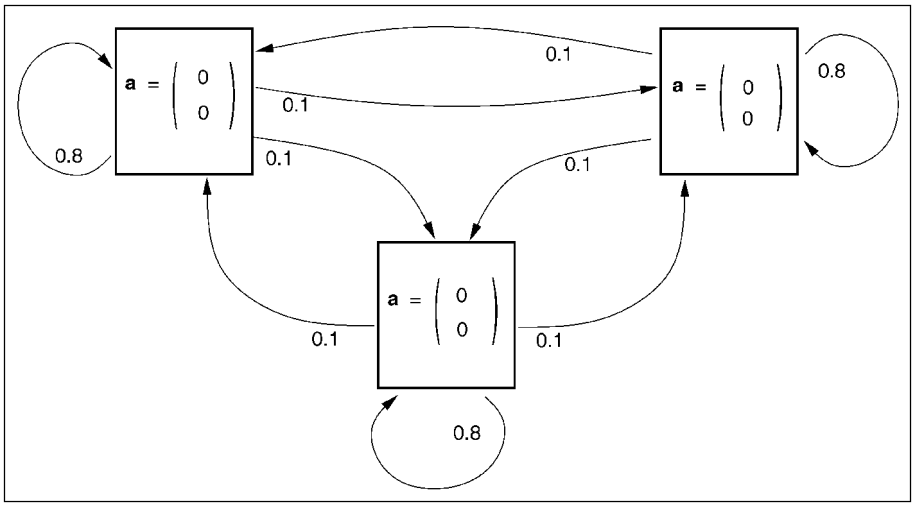
Fig. 11. **Learning three-class dynamics of juggling.** (a) Initial model and (b) learned model. Given a modest degree of "priming" in the initial settings of dynamical parameters (top), the three emergent motion classes (bottom) turn out to correspond to ballistic motion, catch/throw, and carry.

9.2 Three Classes

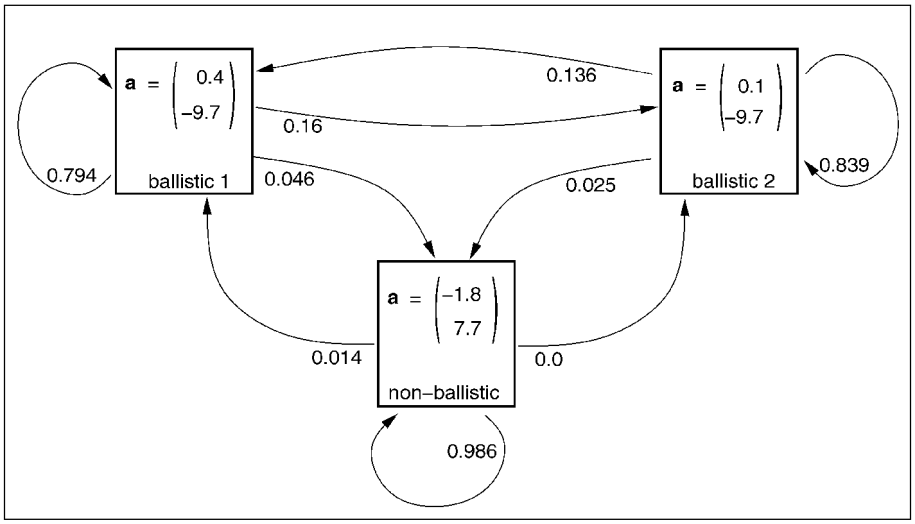
The next experiment was to learn a three-state dynamical model, in the expectation that *catch* and *throw*, being similar (each consists of strong upward acceleration) would be amalgamated into a single class. The three classes would then be: ballistic, catch/throw, and carry. The duration T of the training sequence is 1.3 seconds (65 video fields), corresponding to one juggling half-cycle. In each case, driving noise was isotropic as before, but now with amplitude $b = \sqrt{5}$ pixels.

Results with two different sets of initial dynamics are shown in Fig. 11 and Fig. 12. For Fig. 11, the initial continuous dynamical models were set with nonzero parameters to break symmetry and nudge the EM local optimization process towards a physically reasonable model. Symmetry-breaking was provided by setting the initial values for acceleration in each of the three states: one upward, one neutral, and one downward. As before, the discrete state transition matrix M was initialized symmetrically with state-lifetimes of $1/(1-0.8) = 5$ time-steps, or 0.1 seconds. The emergent dynamical model consists of classes that correspond recognizably to ballistic motion, catch/throw, and carry, consistent with the symmetry-breaking priming from the initial model. The total half-cycle-time—the sum this time of three state lifetimes—is 0.74s, which is about 40 percent too small. This bias is a property of the MLE which is unbiased only in the limit that the training sequence is long. Bias was indeed reduced by training over four half-cycles instead of one, giving 1.06s which underestimates the true lifetime by about 15 percent.

The ballistic state has the longest lifetime and this fairly reflects the characteristics of juggling. The shortest lifetime, again realistic, belongs to the carry state, and note that its acceleration is predominantly lateral, consistent with the ball being shunted sideways, between catching and relaunch. The constraint that carry motion never follows on directly from ballistic motion is captured strongly: With probability 0.01, this transition is only one quarter as likely as the alternative transition from ballistic to catch/throw. It



(a)



(b)

Fig. 12. **Local minimum in EM learning.** (a) Initial model. (b) Learned model. In this experiment, the initial dynamical model has **a**-parameters set to zero. The learned dynamics converge to a new solution, the local optimum in EM that is reached from the new initial parameters. The three classes have merged into just two distinct groups, with the two classes labeled ballistic having very similar parameters.

is also the case that ballistic motion never follows directly from carry, and this is represented more weakly, being about half as likely as the alternative transition (carry to catch/throw).

9.2.1 Initial Conditions and Local Minima

It is natural to wonder whether such clear dynamics for the three-class case would emerge from unprimed, symmetrical, initial settings, like the ones used above in the two-class experiment. In a further experiment, therefore, initial values for accelerations were set to $\mathbf{0}$, with the same symmetrical, initial M as before. The results in Fig. 12 show that the change in initial conditions has produced a marked change in learned dynamics. This is only to be expected given that EM is a gradient descent algorithm that finds local, but not necessarily global, optima of expected log-likelihood. At the new local optimum, the carry class, the most ephemeral of the classes in the previous learned model (Fig. 11), appears to have vanished, or been absorbed into catch/throw to form a single nonballistic state. Two very similar ballistic states have emerged each with $\mathbf{a} \approx \mathbf{g}$ and that two-state subsystem can be shown to have a joint lifetime of 0.58s, which is about right for the duration of the flight of a ball. However, the lifetime of the nonballistic class is about 1.4s, which is about twice as long as the actual duration of nonballistic motion in the training sequence.

9.3 Classification with Six-State Dynamics

Given a learned three-state model as above, it should be possible to classify motion. In fact, some experimentation was required before good classification was obtained. For example, the noise amplitude b in $B = bI$ is not learned, so a good value must be fixed manually, in advance of learning. A value $b = \sqrt{3}$ pixels was found to give better classification than $b = \sqrt{5}$ pixels, and is used in results shown here. Symmetry breaking was included in initial conditions as

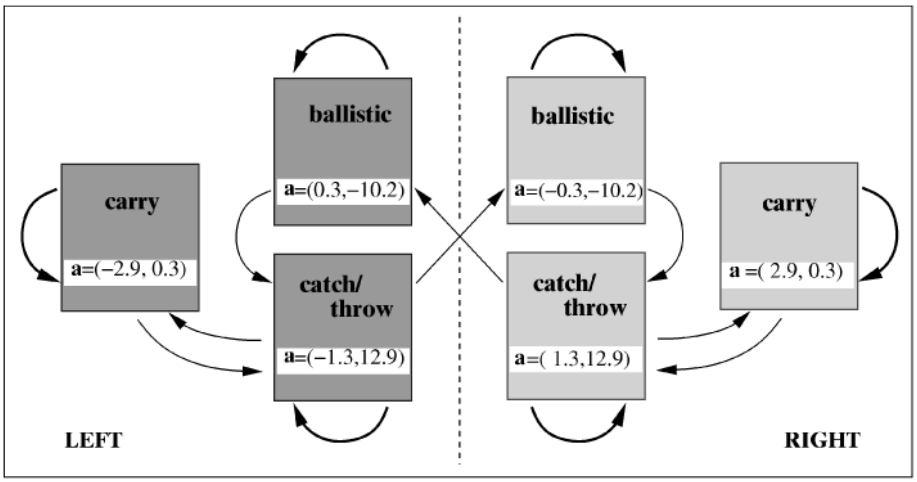


Fig. 13. **Six-class dynamics for juggling.** A three-class model, similar to the one in Fig. 11, is learned for the left hand half-cycle of juggling. Horizontal reflection of physical parameters generates a model for the right hand. Finally, the left and right hand Markov chains are broken open and connected together in a figure of eight, to form a model for the full, six-class juggling cycle.

earlier. The resulting learned three-class model is similar to the one in Fig. 11, except that now the two "illegal" transitions, ballistic $\rightarrow$ carry and carry $\rightarrow$ ballistic, acquire such low transition probabilities as to be effectively zero. (This extra information seems to have arisen as a result of improved tracking accuracy with the new b-value.) The learned model applies to a left handed half-cycle. A model for the right handed half-cycle can be obtained simply by reflecting all dynamical parameters about the image y-axis (which is parallel to gravitational force). Then, the state-chains for left and right hands can be broken open and connected in a figure of eight, as in Fig. 13.

The full cycle model of Fig. 13 was applied, via CONDENSATION smoothing, to a test sequence of 10 seconds duration, sufficient for four full juggling cycles, the first 1.2s of which formed the training sequence. The entire motion of 500 video fields is tracked accurately and the first cycle of tracked juggling is shown in Fig. 14. The figure illustrates mean positions from the smoothed distributions for successive times, together with the most probable class y_t. Note that class labels y_t are the most probable pointwise, that is maximizing over label probabilities at each time t in isolation. (An alternative would be to display the most probable *sequence* $\{y_1, \ldots, y_T\}$ of classes, for which a Viterbi algorithm has been developed recently [34], but only in the case of Gaussian observation noise.) Pointwise, most probable class labels are given for the entire test sequence in Fig. 15. The first six half-cycles are correctly classified—note the apparent periodicity both of full-cycles and (up to handedness) of half-cycles. At 7.5s, some disturbance in the data causes the handedness to flip so the final cycle is labeled right-left, in place of the true left right sequence. This is a reasonable error in that the differences in the acceleration vector $\mathbf{a}$ for corresponding left/right classes (Fig. 13) are subtle: The horizontal components of acceleration, which are reversed in exchanging hands, are small compared with the vertical components. Indeed, the flip of handedness occurs during ballistic motion which is indistinguishable, in principle, between hands.

10 Computational Complexity

A severe limitation on the scope for experimentation with learning is the very considerable computational load of the EM learning algorithm. For example, in the first (two-class) juggling experiment, 12 iterations were used with $N = 750$ and $Q = 5$, each iteration taking over an hour on a desktop workstation (SGI Octane 175MHz). Here, two possible attacks on the complexity problem are considered. The first addresses the problem posed by long-lifetimes and the associated low probabilities for transition out of a given class. Given a transition probability M_{12}, then during a class 1 phase, only NM_{12} particles on average are assigned to class 2, and if M_{12} is small, this may be insufficient to track the transition to class 2, when it occurs. One approach to this is deliberately to overstimulate low probability transitions by "partial importance sampling" in the forward filter and preliminary experiments suggest that this is useful. The other problem is the quadratic complexity of the particle smoothing algorithm, which can be mitigated by averaging (described earlier) to reduce complexity from $O(N^2T)$ to $O(NT)$, but with the limitation that the reduction in computation may only take effect for very large values of N. There is another alternative: A forward-backward algorithm with $O(NT)$ complexity which will save computational effort, again only if N is large enough.

10.1 Partial Importance Sampling

Given that off-diagonal elements of M need to be small for long-duration motion classes, only a small fraction of the N samples in a given time-step are available to change their discrete state. One general approach to such undersampling problems is "importance sampling" [20], in which areas of configuration space that are unduly sparsely populated with particles can be artificially repopulated, and the corresponding likelihood weights $\pi_t^{(n)}$ are adjusted to maintain the correct posterior distribution. This is done using an importance function $g(\mathbf{X})$ which determines the intensity of repopulation over the configuration space for X.

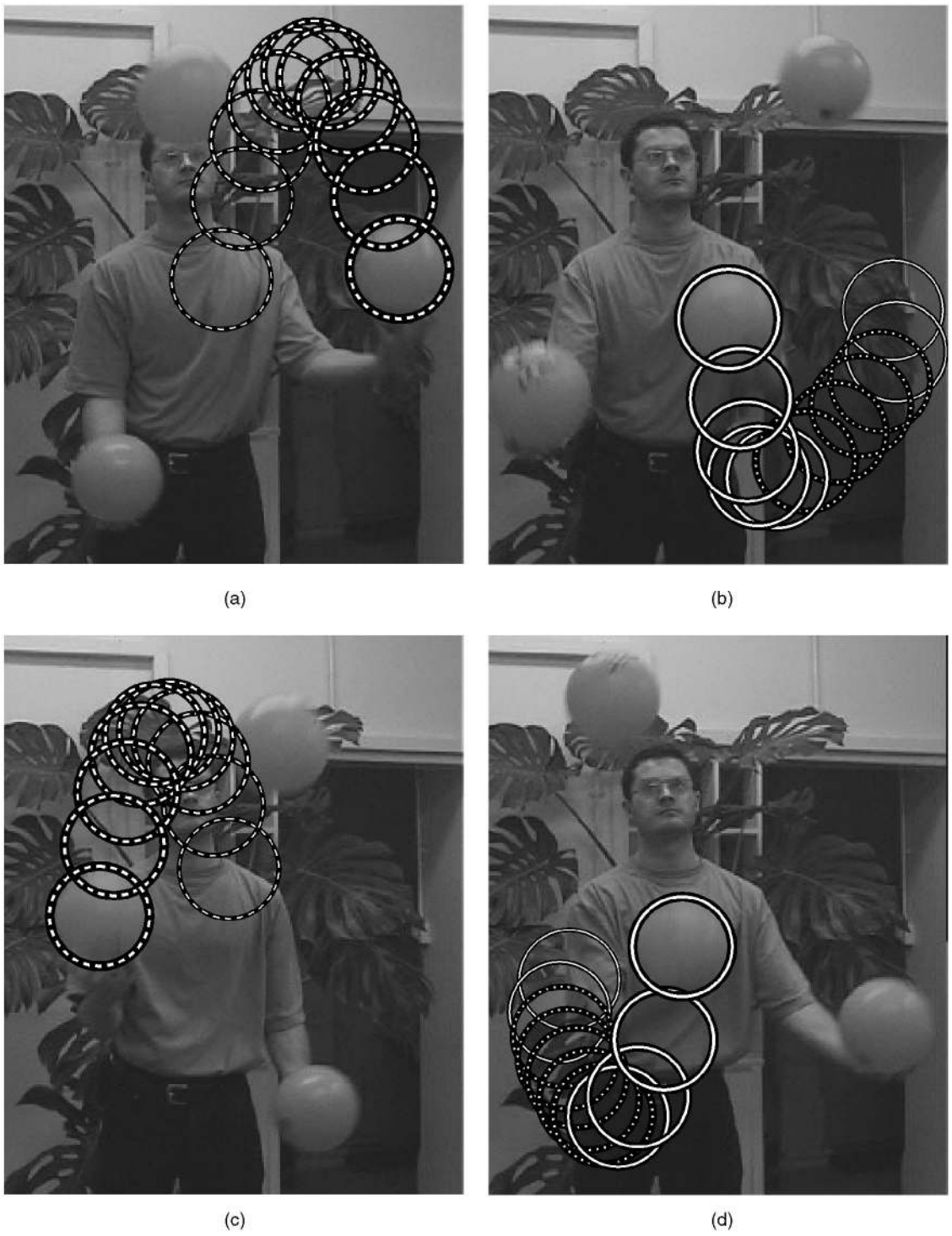


Fig. 14. **Juggling: Estimation and classification results.** One cycle of tracked motion is displayed with class labels (dashed: ballistic, solid: catch/throw, dotted: carry). (a) Ballistic (left), (b) catch, carry, throw (left), (c) ballistic (right), and (d) catch, carry, throw (right).

In the dynamical classification problem, it is just the discrete component y of the state $\mathbf{X}$ for which importance sampling is required. This can be achieved by modifying two of the steps of the forward algorithm of Fig. 4 as follows:

Step 1. Choose $y_t^{(n)} = y' \in \mathcal{Y}$ from some fixed probability distribution $\epsilon_{y'}$, for instance, uniform ($\epsilon_{y'} = 1/N_y$), regardless of the predecessor state $y_{t-1}^{(m)}$.

Step 2. Compensate for the bias introduced in Step 1 by adjusting likelihood weights which then become (before normalization)

$$\pi_t^{(n)} = \frac{M_{y,y'}}{\epsilon_{y'}} p\left(\mathbf{z}_t | \mathbf{x}_t = \mathbf{x}_{t|t}^{(n)}\right),$$

where $y = y_{t-1}^{(m)}$ and $y' = y_t^{(n)}$.

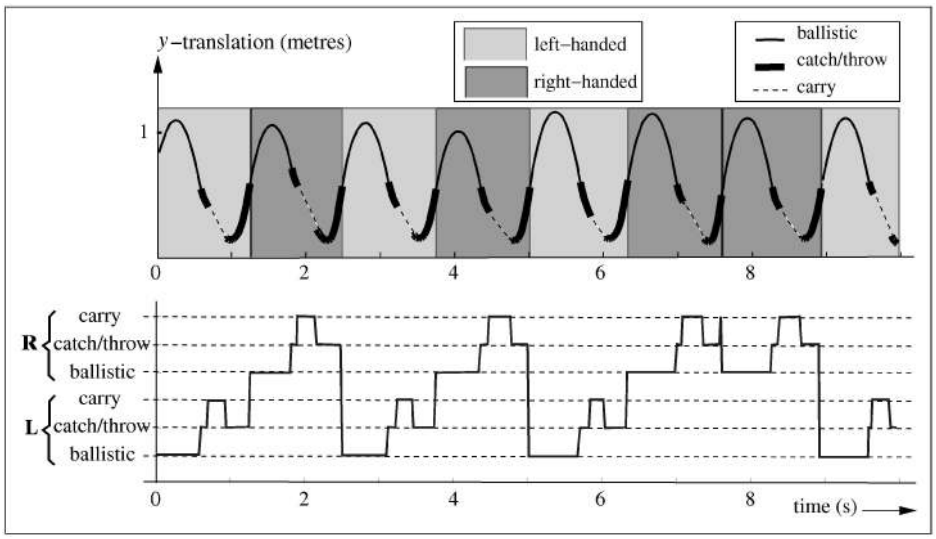


Fig. 15. **Classification of juggling motions.** The most probable classification for each time-step is shown here for the six-state model (top) and superimposed on the estimated ball-height trajectory (bottom). Some transient disturbance at 7.5s has perturbed the class-sequence—see text.

This boosts the population of particles undergoing state-transition while maintaining the asymptotic unbiasedness of the particle sets.

This strategy has been tested in preliminary learning experiments in which dynamical learning was done with just one iteration of EM and using only the forward pass as an approximation to the full forward-backward smoothing algorithm. Allowing some doubt over the extent to which this approximate learning algorithm is representative of the performance of full EM learning, the results are promising. The data was derived from visual observation of physical exercises, as in Fig. 16, the task being to classify motion into one of two classes. The quality of learned dynamical models for the two classes is measured in terms of the classification error rate on a test set of eight seconds duration, containing approximately equal durations of each class, and with two class transitions. As particle set size N increases, error rate decreases, reaching a terminal value of around 10 percent. It is clear (Fig. 17) that the value is reached considerably sooner ($N = 400$) when partial importance sampling is used than otherwise ($N = 800$). This can be taken as encouraging evidence that partial importance sampling should reduce computation times in the general EM-C setting.

10.2 Smoothing with Linear Complexity

The backward filter in Fig. 5 traverses the forward "lattice" of particles $\mathbf{X}_t^{(n)}$, which was generated by the forward filter. An alternative, based on [27] with some correction and simplification, is to generate an independent backward

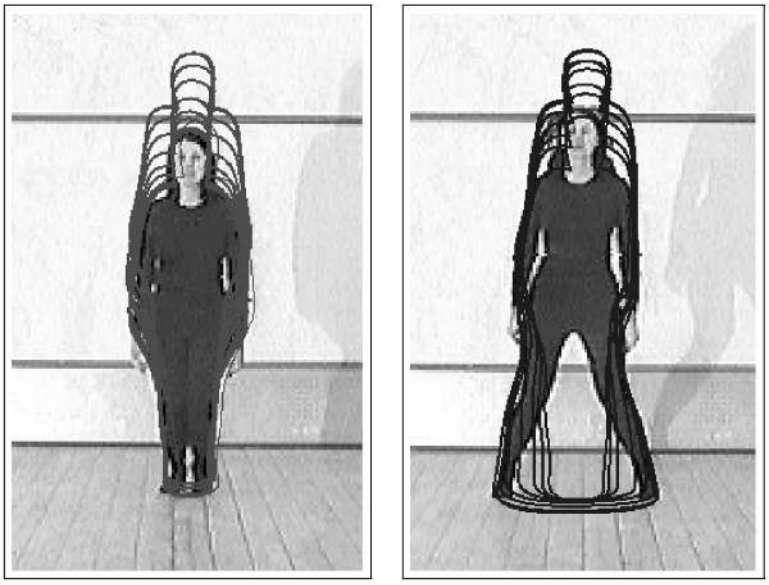


Fig. 16. **Physical exercise.** Two kinds of motion occur in alternating sequence: "jump" (left)—jumping up and down without lateral arm or leg movement and "half star"—a star jump without arm movement.

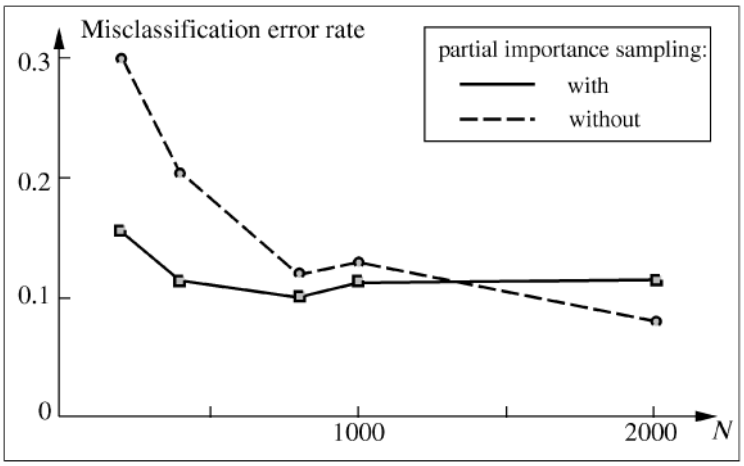


Fig. 17. **Classification error for physical exercise data.** The plots show that partial importance sampling reduces the size N of the particle set needed to learn a given quality of dynamical model.

lattice $\overleftarrow{\mathbf{X}}_t^{(n)}$ and fuse the two lattices in a final step. This alternative backward filter is given in the Appendix.

The algorithm formally has $O(NT)$ complexity compared with $O(N^2T)$ for the earlier algorithm. However, saving in computational effort may not be realized until N is quite large (see the Appendix). Furthermore, it remains to be determined experimentally whether the two smoothing algorithms are actually equally effective for equal values of N, and indeed to define a suitable measure of effectiveness.

11 Conclusion

A general tool for learning dynamics has been explored: the EM-C algorithm, a combination of CONDENSATION (particle-filtering) and Expectation Maximization. It has proved to be a versatile learning algorithm, capable of handling both inexact observations in clutter, and multiple motion classes.

Computational complexity of dynamical learning is a problem both practically and in principle. Learning dynamics from a few seconds of video has typically required several hours of processing time. Partial importance sampling is one method that promises to reduce computation times. Another source of inefficiency is that the learning algorithm used here is quadratic in N, the particle set size. An N-linear algorithm is possible for single class learning and can be modified to apply to the multiple class case. For sufficiently large values of N, this should reduce computation times for learning. Also, note that the learning algorithms are readily amenable to parallel implementation. One further possible saving of computational effort might arise if a way could be found for reusing particles in a given EM-C iteration, in the subsequent iteration.

Learning dynamics is important, both for perception and classification of motion. In perception of motion against clutter, the required size N of particle set is markedly reduced when learned dynamics are used for prediction. Experiments with learned multiclass motion show that good classification accuracy can be achieved in simpler cases. For more complex systems, scope for experimentation is somewhat limited by long computation times, but it is clear that the local nature of EM optimization becomes important. The result is that while complex dynamical models can be substantially refined by EM-C, order will not necessarily emerge from entirely bland, unprimed disorder—a clear instance of "Martin's law" [39, chapter 11] that learning generally proceeds incrementally.

Appendix

Smoothing with Linear Complexity

The alternative backward filter of Section 10.2 is given in Fig. 18 for the case of $K = 1$ order dynamics with a single motion class, and without state augmentation, so that the backward lattice is simply $\overleftarrow{\mathbf{x}}_t^{(n)}$. Then, sampling from the particle-sets $\{(\overleftarrow{\mathbf{x}}_t^{(n)}, \beta_t^{(n)})\}$ generates samples from the a density proportional to the likelihood function $p(\mathcal{Z}_{t:T}|\mathbf{x}_t)$.

A limitation on this algorithm is that it is valid only if

$$\int_{\mathbf{x}_{t-1}} p\left(\mathbf{x}_t^{(m)} \mid \mathbf{x}_{t-1}\right) \mathrm{d}\mathbf{x}_{t-1} = \mathrm{const}, \tag{15}$$

which is satisfied by the linear ARP(1) model and ARP(K) models generally, but precludes extension to augmented state filtering as used earlier. However, it is still possible to estimate all the required autocorrelations $\mathcal{E}[R_{i,j}|\mathcal{Z}_{1:T}]$ by forming particle sets as follows:

1. Draw $\mathbf{x}_{t-1}$ from $p(\mathbf{x}_{t-1}|\mathcal{Z}_{1:t-1})$ using the forward filter.
2. Draw $\mathbf{x}_t$ from $p(\mathcal{Z}_{t:T}|\mathbf{x}_t)$ using the backward filter.
3. Generate a weight $\gamma = p(\mathbf{x}_t|\mathbf{x}_{t-1})$.

Particles $((\mathbf{x}_{t-1}, \mathbf{x}_t), \gamma)$ generated in this way form sets from which samples from the distribution $p(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathcal{Z}_{1:T})$ can be drawn and used to estimate the necessary autocorrelations.

As for the forward propagation algorithm in Fig. 4, computationally complexity is $O(NT)$, determined by the sampling operation in Step 2.1. However, there is an additional cost in this alternative backward algorithm, relative to the original, namely the extra evaluations of observation likelihood required in Step 2.3. In practice, evaluation of observation likelihood often has high computational cost. Doubling this cost here means that

1. Initialise reverse lattice $\overleftarrow{\mathbf{x}}_T^{(n)}$ by drawing samples from a suitable prior $p_T(.)$.

Set initial reverse weights

$$\beta_T^{(n)} = \frac{p\left(\mathbf{z}_T | \mathbf{x}_T = \overleftarrow{\mathbf{x}}_T^{(n)}\right)}{p_T\left(\overleftarrow{\mathbf{x}}_T^{(n)}\right)}.$$

2. Iterate for $t = T-1, T-2, \ldots$:

For each n:

2.1 Sample: $m \in \{1, \ldots, N\}$ with probability $\beta_{t+1}^{(m)}$.

2.2 Predict by sampling from

$$p\left(\overleftarrow{\mathbf{x}}_t^{(m)} \mid \mathbf{x}_{t-1}\right)$$

treated as a function of $\mathbf{x}_{t-1}$, to choose $\overleftarrow{\mathbf{x}}_{t-1}^{(n)}$.

2.3 Observation weights are computed from the observation likelihood evaluated for $\mathbf{z}_t$:

$$\beta_t^{(n)} = p(\mathbf{z}_t | \overleftarrow{\mathbf{x}}_t^{(n)}),$$

and normalised.

Fig. 18. A linear complexity backward filter.

the alternative algorithm, although formally $O(NT)$ compared with $O(N^2T)$, may not actual show reduced computation time until N is quite large.

The normalization requirement (15), extended to mixed states, implies that

$$\sum_y M_{y,y'} = 1,$$

which is not in general true of a Markov chain. Strictly, therefore, the linear complexity backward filter cannot be applied to multiclass learning. Fortunately, there is a straightforward modification that deals with the problem. It involves partial importance sampling with deterministically sampled discrete variables, but details are omitted here.

ACKNOWLEDGMENTS

The authors would like to thank the Royal Society (AB), EPSRC (AB, BN), Oxford Metrics Ltd (BN), Magdalen College Oxford (MI), and the EU (JR) for their support. They also greatly appreciated the helpful comments from C. Bishop, A. Noble, B. Frey, and S. Soatto.

REFERENCES

[1] B. Anderson and J. Moore, *Optimal Filtering.* Prentice Hall, 1979.
[2] K. Astrom and B. Wittenmark, *Computer Controlled Systems.* Addison Wesley, 1984.
[3] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association.* Academic Press, 1988.
[4] A. Blake, B. Bascle, M. Isard, and J. MacCormick, "Statistical Models of Visual Shape and Motion," *Phil. Trans. Royal Soc. A,* vol. 356, pp. 1,283-1,302, 1998.
[5] A. Blake and M. Isard, "The Condensation Algorithm—Conditional Density Propagation and Applications to Visual Tracking," *Advances in Neural Information Processing Systems,* vol. 9, pp. 361-368, 1997.
[6] A. Blake and M. Isard, *Active Contours.* Springer, 1998.
[7] A. Blake, M. Isard, and D. Reynard, "Learning to Track the Visual Motion of Contours," *J. Artificial Intelligence,* vol. 78, pp. 101-134, 1995.
[8] A. Blake, B. North, and M. Isard, "Learning Multi-Class Dynamics," *Advances in Neural Information Processing Systems,* M.S. Kearns, S. Solla, and D. Cohn, eds., vol. 11, 1999.
[9] A. Bobick and A. Wilson, "A State-Based Technique for the Summarization and Recognition of Gesture," *Proc. Fifth Int'l Conf. Computer Vision,* pp. 382-388, 1995.
[10] C. Bregler, "Learning and Recognizing Human Dynamics in Video Sequences," *Proc. Conf. Computer Vision and Pattern Recognition,* 1997.
[11] P. Brockwell and R. Davis, *Introduction to Time-Series and Forecasting.* Springer-Verlag, 1996.
[12] A. Dempster, M. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc. B,* vol. 39, pp. 1-38, 1977.

[13] *Applied Optimal Estimation,* A. Gelb, ed., Cambridge, Mass.: MIT Press, 1974.
[14] A. Gelfand and A. Smith, "Sampling-Based Approaches to Computing Marginal Densities," *J. Am. Statistical Assoc.,* vol. 85, no. 410, pp. 398-409, 1990.
[15] D. Geman, "A Stochastic Model for Boundary Detection," *J. Image and Vision Computing,* vol. 5, pp. 61-65, 1987.
[16] Z. Ghahramani and S. Roweis, "Learning Nonlinear Dynamical Systems Using an EM Algorithm," *Advances in Neural Information Processing Systems,* M.S. Kearns, S. Solla, and D. Cohn, eds., vol. 11, 1999.
[17] C. Goodwin and K. Sin, *Adaptive Filtering Prediction and Control.* Prentice Hall, 1984.
[18] N. Gordon, D. Salmond, and A. Smith, "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation," *IEE Proc. F,* vol. 140, no. 2, pp. 107-113, 1993.
[19] U. Grenander, Y. Chow, and D. Keenan, *HANDS A Pattern Theoretical Study of Biological Shapes.* New York: Springer-Verlag, 1991.
[20] J. Hammersley and D. Handscomb, *Monte Carlo Methods.* Methuen, 1964.
[21] X. Huang, Y. Arika, and M. Jack, *Hidden Markov Models for Speech Recognition.* Edinburgh Univ. Press, 1990.
[22] M. Isard and A. Blake, "Visual Tracking by Stochastic Propagation of Conditional Density," *Proc. Fourth European Conf. Computer Vision,* pp. 343-356, 1996.
[23] M. Isard and A. Blake, "Condensation—Conditional Density Propagation for Visual Tracking," *Int'l J. Computer Vision,* vol. 28, no. 1, pp. 5-28, 1998.
[24] M. Isard and A. Blake, "ICondensation: Unifying Low-Level and High-Level Tracking in a Stochastic Framework," *Proc. Fifth European Conf. Computer Vision,* pp. 893-908, 1998.
[25] M. Isard and A. Blake, "A Mixed-State Condensation Tracker with Automatic Model Switching," *Proc. Sixth Int'l Conf. Computer Vision,* pp. 107-112, 1998.
[26] M. Isard and A. Blake, "A Smoothing Filter for Condensation Model Switching," *Proc. Fifth European Conf. Computer Vision,* pp. 768-781, 1998.
[27] G. Kitagawa, "Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models," *J. of Computational and Graphical Statistics,* vol. 5, no. 1, pp. 1-25, 1996.
[28] S. Lauritzen, *Graphical Models.* Oxford, 1996.
[29] L. Ljung, *System Identification: Theory for the User.* Prentice Hall, 1987.
[30] J. MacCormick and A. Blake, "A Probabilistic Contour Discriminant for Object Localization," *Proc. Sixth Int'l Conf. on Computer Vision,* pp. 390-395, 1998.
[31] B. North and A. Blake, "Learning Dynamical Models Using Expectation-Maximization," *Proc. Sixth Int'l Conf. Computer Vision,* pp. 384-389, 1998.
[32] J. Pardey, S. Roberts, and L. Tarassenko, "A Review of Parametric Modeling Techniques for EEG Analysis," *Medical Eng. Physics,* vol. 18, no. 1, pp. 2-11, 1995.
[33] V. Pavlovic, B. Frey, and T. Huang, "Time-Series Classification Using Mixed-State Dynamic Bayesian Networks," *Proc. Conf. Computer Vision and Pattern Recognition,* pp. 609-615, 1999.
[34] V. Pavlovic, J. Rehg, T.-J. Cham, and K. Murphy, "A Dynamic Bayesian Network Approach to Figure Tracking Using Learned Models," *Proc. Int'l Conf. Computer Vision,* pp. 94-101, 1999.
[35] L. Rabiner and J. Bing-Hwang, *Fundamentals of Speech Recognition.* Prentice-Hall, 1993.
[36] D. Reynard, A. Wildenberg, A. Blake, and J. Marchant, "Learning Dynamics of Complex Motions from Image Sequences," *Proc. Fourth European Conf. Computer Vision,* pp. 357-368, 1996.
[37] R. Shumway and D. Stoffer, "An Approach to Time Series Smoothing and Forecasting Using the EM Algorithm," *J. Time Series Analysis,* vol. 3, pp. 253-226, 1982.
[38] P. Wellner, "The Digital Desk Calculator—Tangible Manipulation on a Desktop Display," *Proc. ACM Symp. User Interface Software and Technology,* 1991.
[39] P. Winston, *Artificial Intelligence.* Addison Wesley, 1984.

Ben North received a degree in mathematics and the doctorate degree in computer vision from the University of Oxford in 1994 and 1998, respectively.

Andrew Blake graduated in 1977 from Trinity College, Cambridge in mathematics and electrical sciences and was a Kennedy Scholar at MIT in 1977 and 1978. He was awarded the doctorate degree in 1983 from the University of Edinburgh, Scotland. Until 1987, he was a member of the faculty in the Computer Science Department at the University of Edinburgh, as a Royal Society Research Fellow. He then joined the faculty of the Department of Engineering Science at the University of Oxford, where he became a professor in 1996. In 1999, he was appointed senior research scientist at Microsoft Research in Cambridge. His research interests are in computer vision, signal processing, and learning. He has published a number of papers in vision, and books with A. Zisserman (*Visual Reconstruction*, MIT press), and with Michael Isard (*Active Contours*, Springer-Verlag). He has won the prize of the European Conference on Computer Vision, with R. Cipolla in 1992 and with M. Isard in 1996. He served as program chairman for the International Conference on Computer Vision in 1995 and 1999 and is on the editorial board of the journals *Image and Vision Computing*, the *International Journal of Computer Vision*, and *Computer Vision and Image Understanding*. He was elected to a Royal Society Senior Research Fellowship (1998-1999) and, became a fellow of the Royal Academy of Engineering in 1998. He is member of the IEEE.

Michael Isard received a degree in mathematics from the University of Cambridge in 1993. He received the doctorate degree in computer vision from the University of Oxford in 1998 and was also a Junior Research Fellow at Magdalen College. He is currently a research scientist at the Compaq Laboratory in Palo Alto, California.

Jens Rittscher received the master's degree in mathematics from the University of Bonn, Germany in 1997. He is currently a doctoral candidate with the Department of Engineering at the University of Oxford. His research interests include computer vision, signal processing, and machine learning. In 1998, he was awarded a Marie Curie Fellowship of the European Union.