4

Fast Particle Filters and Their Applications to Adaptive Control in Change-Point ARX Models and Robotics

Yuguo Chen, Tze Leung Lai and Bin Wu
University of Illinois at Urbana-Champaign & Stanford University
USA

1. Introduction

The Kalman filter has provided an efficient and elegant solution to control problems in linear stochastic systems. For nonlinear stochastic systems, control problems become much more difficult and a large part of the literature resorts to linear approximations so that an "extended Kalman filter" or a "mixture of Kalman filters" can be used in place of the Kalman filter for linear systems. Since these linear approximations are local expansions around the estimated states, they may perform poorly when the true state differs substantially from its estimate. Substantial progress was made in the past decade for the filtering problem with the development of particle filters. This development offers promise for solving some long-standing control problems which we consider in this chapter.

As noted by Ljung & Gunnarsson (1990), a parameterized description of a dynamic system that is convenient for identification is to specify the model's prediction of the output y_t as a function of the parameter vector θ and past inputs and outputs u_s and y_s, respectively, for $s < t$. When the function is linear in θ, this yields the regression model $y_t = \theta^T \varphi_t + \epsilon_t$, which includes as a special case the ARX model (autoregressive model with exogenous inputs) that is widely used in control and signal processing. Here the regressor vector is

$$\varphi_t = (-y_{t-1}, \ldots, -y_{t-k}, u_{t-1}, \ldots, u_{t-h})^T, \tag{1}$$

consisting of lagged inputs and outputs. Whereas a comprehensive methodology has been developed for identification and control of ARX systems with time-invariant parameters (see e.g. Goodwin et al., 1981; Ljung & Soderstrom, 1983; Goodwin & Sin, 1984; Lai & Wei, 1987; Guo & Chen, 1991), the case of time-varying parameters in system identification and adaptive control still awaits definitive treatment despite a number of major advances during the past decade (Meyn & Brown, 1993; Guo & Ljung, 1993a, b). In Section 3 we show how particle filters can be used to resolve some of the long-standing difficulties due to the nonlinear interactions between the dynamics of the regressor vector (1) and of the parameter changes in the model $y_t = \theta_t^T \varphi_t + \epsilon_t$. Unlike continually fluctuating parameters modeled by a random walk in Meyn & Brown (1993) and Guo & Ljung (1993a, b), we consider here the parameter jump model similar to that in Eq. (21)-(22) of Ljung & Gunnarsson (1990). As reviewed in Ljung & Gunnarsson (1990, p. 11), an obvious way to

handle parameter jumps is to apply carefully designed on-line change detection algorithms to segment the data. Another approach, called AFMM (adaptive forgetting through multiple models), is to use Bayesian updating formulas to calculate the posterior probability of each member in a family of models locating the change-points. To keep a fixed number of such models at every stage, the model with the lowest posterior probability is deleted while that with the highest posterior probability gives birth to a new model by allowing for a possible change-point from it. The fast particle filters introduced by Chen & Lai (2007) enable them to develop a much more precise implementation of the Bayesian approach than AFMM, with little increase in computational cost, and to come up with more efficient adaptive control schemes, as shown in Section 3.

Another area where particle filters have been recognized to offer promising solutions to important and difficult control problems is probabilistic robotics. Section 4 provides a brief summary of the applications of particle filters to estimate the position and orientation of a robot in an unknown environment from sensor measurements. It also reviews previous work and ongoing work on using these particle filters to tackle the difficult stochastic control problems in robotics.

The stochastic models in Sections 3 and 4 are special cases of hidden Markov models. Section 2 gives a brief introduction to hidden Markov models and particle filters, which are sequential estimates of the hidden states by using Monte Carlo methods that involve sequential importance sampling and resampling. The basic idea underlying these sequential Monte Carlo filters is to represent the posterior distribution of the hidden state at time t given the observations up to time t by a large number of simulated samples ("particles"). Simulating a large number of samples, however, makes the Monte Carlo approach impractical for on-line identification and control applications. We show in Section 3 that by choosing appropriate resampling schemes and proposal distributions for importance sampling, we can arrive at good approximations to the optimal filter by using a manageable number (as small as 50) of simulated samples for on-line identification and adaptive control. This point is discussed further in Section 5 where we consider related issues and give some concluding remarks.

2. Particle Filters in Hidden Markov Models

A hidden Markov model (HMM) is a stochastic process (x_t, y_t) in which (i) $\{x_t\}$ is an unobservable Markov process with transition probability density function $f(\cdot|\cdot)$ with respect to some measure ν on the state space, and (ii) given $\{x_t\}$, the observable random variables y_t are conditionally independent such that y_s has density function $g(\cdot|x_s)$ with respect to some measure. The filtering problem for HMM is to find the posterior distribution of the hidden state x_t given the current and past observations $y_1,\ldots,y_t$. In particular, the optimal filter with respect to squared error loss is $E(x_t|y_1,\ldots,y_t)$. In engineering applications, there are often computational constraints for on-line updating of the filter and recursive algorithms are particularly attractive. For infinite state spaces, direct computation of the optimal filters is not feasible except in linear Gaussian state-space models, for which Kalman filtering provides explicit recursive filters. Analytic approximations or Monte Carlo methods are therefore used instead. Although Markov chain Monte Carlo has provided a versatile simulation-based tool to calculate the posterior distributions of hidden states in HMMs, it is cumbersome for updating and is too slow for

on-line filtering problems. Sequential Monte Carlo methods represent the posterior distributions by a large number M of random samples that are sequentially generated over time by using a combination of sequential importance sampling and resampling steps.

2.1 Proposal Distribution for Sequential Importance Sampling

Let $X_{1:n} = (x_1, \ldots, x_n)$, $Y_{1:n} = (y_1, \ldots, y_n)$, and $p(\cdot|\cdot)$ and $p(\cdot)$ denote the conditional and the joint density functions (under the measure P) of the random variables indicated. Given $Y_{1:n}$, the conditional distribution of $X_{1:n}$ is that of an inhomogeneous Markov chain with transition probability density

$$p(x_t|x_{t-1}, Y_{1:n}) \propto f(x_t|x_{t-1})g(y_t|x_t) \int \prod_{i=t+1}^{n} f(x_i|x_{i-1})g(y_i|x_i)d\nu(x_i), \tag{2}$$

in which the constant of proportionality is the normalizing constant that makes the left hand side of (2) integrate to 1.

It is often difficult to sample directly from this Markov chain for Monte Carlo evaluation of the posterior distribution of x_n given $Y_{1:n}$, which is used to estimate the optimal filter $E\{h(x_n)|Y_{1:n}\}$. Instead we sample from an alternative distribution Q under which $X_{1:n}$ is an inhomogeneous Markov chain with transition density

$$q_t(x_t|x_{t-1}) \propto f(x_t|x_{t-1})g(y_t|x_t), \tag{3}$$

which is tantamount to replacing $p(x_t|x_{t-1}, Y_{1:n})$ in (2) by $p(x_t|x_{t-1}, Y_{1:t})$. The optimal filter can be expressed in terms of Q via

$$E\{h(x_n)|Y_{1:n}\} = E_Q\Big\{h(x_n) \prod_{t=1}^{n} [p(x_t|x_{t-1}, Y_{1:n})/q_t(x_t|x_{t-1})]\Big\}, \tag{4}$$

where E_Q denotes expectation under the measure Q. Therefore, instead of drawing M samples $X_{1:n}^{(1)}, \ldots, X_{1:n}^{(M)}$ from (2) and using $M^{-1}\sum_{j=1}^{M} h(x_n^{(j)})$ to estimate the optimal filter (4), we can draw M samples from (3) and estimate the optimal filter by

$$\hat{h}_n = \Big\{\sum_{j=1}^{M} h(x_n^{(j)})w_n^{(j)}\Big\} / \sum_{j=1}^{M} w_n^{(j)}, \tag{5}$$

where $w_n^{(j)}$ are the *importance weights* given recursively by

$$w_1^{(j)} = 1, \quad w_t^{(j)} = w_{t-1}^{(j)} p(y_t|x_{t-1}^{(j)}), \tag{6}$$

noting that $p(y_t|x_{t-1}) = \int f(x|x_{t-1})g(y_t|x)d\nu(x)$, $p(y_1|x_0)$ and $p(Y_{1:n}|x_0)$ are proportionality constants and

$$\prod_{t=1}^{n} p(x_t|x_{t-1}, Y_{1:n}) = p(x_{1:n}|x_0, Y_{1:n}) = \Big\{\prod_{t=1}^{n} f(x_t|x_{t-1})g(y_t|x_t)\Big\}/p(Y_{1:n}|x_0),$$

$$\prod_{t=1}^{n} q_t(x_t|x_{t-1}) = \prod_{t=1}^{n}\{f(x_t|x_{t-1})g(y_t|x_t)/p(y_t|x_{t-1})\}.$$

In the case where x_0 is specified by an initial distribution π_0, we replace x_0 above by $x_0^{(j)}$ drawn from π_0 (j = 1,..., M).

In situations where the normalizing constant $\int f(x|x_{t-1})g(y_t|x)d\nu(x)$ in (3) does not have a closed-form expression, sampling from Q defined by (3) can still be carried out by rejection sampling or other methods, but the importance weights (6) do not have explicit formulas and rejection sampling slows down the procedure considerably. A better idea is to choose another Q which is easier to sample from and has explicit formulas for the importance weights, and which approximates (3) in some sense. One way to do this is to use a finite mixture of Gaussian distributions to approximate (3), with suitably chosen mixing proportions, means and covariance matrices. Using (3) or more convenient approximations thereof as the proposal distribution for sequential importance sampling provides substantial improvement over the original particle filter of Gordon et al. (1993) who simply use $q_t(x_t|x_{t-1}) = f(x_t|x_{t-1})$, not adapting to the observed data $Y_{1:n}$. Whereas the adaptive transition probability density (2) is non-recursive (because $Y_{1:n}$ and $Y_{1:n+1}$ result in different transition probabilities $p(x_t|x_{t-1}, Y_{1:n})$ and $p(x_t|x_{t-1}, Y_{1:n+1})$), the proposal distribution (3) is adaptive and recursive.

2.2 Periodic Rejuvenation via Resampling

The particle filter of Gordon et al. (1993) is often called a "bootstrap filter" because, besides sampling $\hat{x}_t^{(j)}$ from $f(\cdot|x_{t-1}^{(j)})$ to form $\hat{X}_{1:t}^{(j)} = (x_1^{(j)}, \ldots, x_{t-1}^{(j)}, \hat{x}_t^{(j)})$, it also resamples from $\{\hat{X}_{1:t}^{(1)}, \ldots, \hat{X}_{1:t}^{(M)}\}$ with probabilities proportional to the importance weights $w_t^{(j)}$ $(= w_{t-1}^{(j)} g(y_t|\hat{x}_t^{(j)}))$, thereby generating the particles (trajectories) $X_{1:t}^{(1)}, \ldots, X_{1:t}^{(M)}$. In other words, at every t there is an importance sampling step followed by a resampling step. We can think of importance sampling as generating a weighted representation $(q(x_1, \ldots, x_t),\ w_t)$ of $p(x_1, \ldots, x_t|Y_{1:t})$ and resampling as transforming the weighted representation to an unweighted approximation of $p(x_1, \ldots, x_t|Y_{1:t})$. For the bootstrap filter, since resampling introduces additional variability, resampling at every t may result in substantial loss in statistical efficiency. In addition, the computational cost of resampling at every t also accumulates over time.

If we forgo resampling altogether, then we have a weighted representation $(q(x_1, \ldots, x_n), w_n)$ of $p(x_1, \ldots, x_n|Y_{1:n})$ at stage n. In view of (4), if we use the normalized weights

$$w_t = \prod_{i=1}^{t}[p(x_i|x_{i-1}, Y_{1:n})/q_i(x_i|x_{i-1})], \tag{7}$$

then $M^{-1}\sum_{j=1}^{M} h(x_n^{(j)})w_n^{(j)}$ is an unbiased estimate of $E\{h(x_n)|Y_{1:n}\}$. However, for large n, sequential importance sampling without resampling also has difficulties because of the large variance of w_n. In particular, for the special cases (i) $q_t(x_t|x_{t-1}) = f(x_t|x_{t-1})$ and (ii) $q_t(x_t|x_{t-1}) = f(x_t|x_{t-1})g(y_t|x_t)$, $n^{-1}\log w_n$ converges almost surely under certain integrability assumptions and $E\{\mathrm{var}_Q(w_n|Y_{1:n})\} \to \infty$ as $n \to \infty$; see Chan & Lai (2008) where an asymptotic theory of particle filters in HMMs, including consistent estimation of their standard errors, is given.

A compromise between forgoing resampling altogether and resampling at every stage t is to resample periodically. The motivation for resampling is to make multiple copies of the trajectories with large weights and to prune away those with small weights. The trajectories with small weights contribute little to the final estimate and it is a waste to carry many trajectories with very small weights. In particular, Kong et al. (1994) propose to monitor the coefficient of variation (cv) of the importance weights w_t, defined by

$$\mathrm{cv}^2 = \mathrm{var}_Q(w_t)/E_Q^2(w_t), \tag{8}$$

and to perform resampling if the cv² of the current weights w_t is greater than or equal to a certain bound. Specifically the procedure can be described as follows, starting with M samples $X_{1:t-1}^{(1)}, \ldots, X_{1:t-1}^{(M)}$ having weights $w_{t-1}^{(1)}, \ldots, w_{t-1}^{(M)}$ at time t-1.

a. Draw $\hat{x}_t^{(j)}$ from $q_t(\cdot|x_{t-1}^{(j)})$ and update the weight $w_t^{(j)}$, j = 1, . . . , M.
b. If the cv² of $\{w_t^{(1)}, \ldots, w_t^{(M)}\}$ exceeds or equals a certain bound, resample from $\{\hat{x}_t^{(1)}, \ldots, \hat{x}_t^{(M)}\}$ with probabilities proportional to $\{w_t^{(1)}, \ldots, w_t^{(M)}\}$ to produce a random sample $\{x_t^{(1)}, \ldots, x_t^{(M)}\}$ with equal weights. Otherwise let $\{x_t^{(1)}, \ldots, x_t^{(M)}\} = \{\hat{x}_t^{(1)}, \ldots, \hat{x}_t^{(M)}\}$ and return to step a.

Strictly speaking, since the weight $w_t^{(j)}$ is associated with the entire path $\hat{X}_{1:t}^{(j)} = (x_1^{(j)}, \ldots, x_{t-1}^{(j)}, \hat{x}_t^{(j)})$, resampling should be performed on $\{\hat{X}_{1:t}^{(1)}, \ldots, \hat{X}_{1:t}^{(M)}\}$. However, because of the Markovian structure, the past observations x_1,..., x_{s-1} can be discarded after generating the current state x_s. This explains why $X_{1:t-1}^{(1)}, \ldots, X_{1:t-1}^{(M)}$ are discarded in Step b above. In the second paragraph of Section 3.1, since the sequential importance sampling with resampling (SISR) filter is defined via certain functions of the Markov chain (instead of the Markov chain itself), resampling has to be performed on the sample of M trajectories.

3. Fast Particle Filters in Change-Point ARX Models

3.1 Preliminaries: Normal Mean Shift Model

Before considering the more general change-point regression model $y_t = \theta_t^T\varphi_t + \epsilon_t$, we find it helpful to explain some important ideas for constructing fast particle filters in the simple case of univariate $\varphi_t \equiv 1$, dating back to Yao's (1984) simple mean shift model, in which the observations y_t are independent normal with variance 1 and means μ_t such that at time t, μ_t equals μ_{t-1} (i.e. undergoes no change) with probability $1-\rho$ and assumes a new value, which is normally distributed with mean 0 and variance σ^2, with probability ρ. Note that $\{(\mu_t, y_t),\ t \geq 1\}$ forms a HMM, with $g(\cdot|\mu)$ being the normal density function

with mean μ and variance 1 and such that the transition probability distribution of μ_t has (i) a discrete component putting mass $1-\rho$ at μ_{t-1} and (ii) an absolutely continuous component having density function $\rho\sigma^{-1}\phi(\cdot/\sigma)$, where ϕ denotes the standard normal density function. The proposal distribution (3) (with $x_t = \mu_t$) for $\mu_t|\mu_{t-1}$ is a mixture of a degenerate distribution at μ_{t-1} and a normal distribution with mean $y_t/(1+\sigma^{-2})$ and variance $1/(1+\sigma^{-2})$, with mixing probabilities proportional to $(1-\rho)\phi(y_t-\mu_{t-1})$ and $\rho(\sigma^2+1)^{-1/2}\phi(y_t/(\sigma^2+1)^{1/2})$, respectively. It is, therefore, easy to sample from this proposal distribution. Because of the discrete and absolutely continuous components of the transition probability distribution, the importance weights (see (6)) w_t are now given recursively by $w_1 = 1$ and

$$w_t = w_{t-1}\{\rho(\sigma^2+1)^{-1/2}\phi(y_t/(\sigma^2+1)^{1/2}) + (1-\rho)\phi(y_t-\mu_{t-1})\}.$$

Instead of working with the unobserved Markov chain $\{\mu_t\}$, it is more efficient to consider the latent variable $I_t = 1_{\{\mu_t \neq \mu_{t-1}\}}$, indicating whether t is a change-point. The reason is that given $Y_{1:n}$ the I_t are Bernoulli random variables that can be generated recursively and $E(\mu_n|I_{1:n}, Y_{1:n})$ can be easily computed by a closed-form formula, where $I_{1:n} = \{I_1, \ldots, I_n\}$. Following Yao (1984), we rewrite the optimal filter as

$$E(\mu_n|Y_{1:n}) = E\{E[\mu_n|I_{1:n}, Y_{1:n}]|Y_{1:n}\} = E\Big\{(n-C_n+1+\sigma^{-2})^{-1}\sum_{j=C_n}^{n} y_j|Y_{1:n}\Big\}, \quad (9)$$

where $C_t = \max\{j \le t : I_j = 1\}$ ($\max\emptyset = 1$), i.e., C_t is the most recent change-point up to time t. Consider the proposal distribution Q for which $I_t|I_{1:t-1}$ has the same distribution as $P(I_t = \cdot|I_{1:t-1}, Y_{1:t})$. It is easy to sample I_1,... ,I_n sequentially from Q, under which $I_t|I_{1:t-1}$ is Bernoulli assuming the values 1 and 0 with probabilities in the proportion

$$\frac{\rho}{(1+\sigma^2)^{1/2}}\phi\left(\frac{y_t}{(1+\sigma^2)^{1/2}}\right) \;:\; \frac{1-\rho}{(1+v_t)^{1/2}}\phi\left(\frac{y_t - v_t\sum_{j=C_{t-1}}^{t-1} y_j}{(1+v_t)^{1/2}}\right), \quad (10)$$

where $v_t = (t - C_{t-1} + \sigma^{-2})^{-1}$. Letting $a_t(\rho)$ and $b_t(\rho)$ denote the two terms in (10), note that $p(y_t|I_{1:t-1}, Y_{1:t-1}) = a_t(\rho) + b_t(\rho)$. Combining this with

$$\frac{p(I_{1:n}|Y_{1:n})}{q(I_{1:n})} = \frac{p(y_1)}{p(Y_{1:n})}\prod_{t=2}^{n}\frac{p(I_{1:t-1}, Y_{1:t})}{p(I_{1:t-1}, Y_{1:t-1})} \propto \prod_{t=2}^{n} p(y_t|I_{1:t-1}, Y_{1:t-1})$$

yields the following recursive formula for the importance weights w_t.

$$w_t = w_{t-1}\{a_t(\rho) + b_t(\rho)\},\ t \ge 2;\quad w_1 = 1. \quad (11)$$

When ρ is small, change-points occur very infrequently and many sequences $I_{1:n}^{(j)}$ sampled from Q may contain no change-points. We can modify Q by increasing ρ in (10) to $\tilde{\rho}$, thereby picking up more change-points, and adjust the importance weights accordingly.

Specifically, take $\tilde{\rho} > \rho$ and choose the proposal distribution $\tilde{Q}$ for which $I_t|I_{1:t-1}$ is a Bernoulli random variable with success probability $a_t(\tilde{\rho})/(a_t(\tilde{\rho}) + b_t(\tilde{\rho}))$. Since

$$p(I_{1:n}|Y_{1:n})/\tilde{q}(I_{1:n}) = \{p(I_{1:n}|Y_{1:n})/q(I_{1:n})\}\{q(I_{1:n})/\tilde{q}(I_{1:n})\},$$

the importance weights $\tilde{w}_t$ can be determined recursively by

$$\tilde{w}_t = \begin{cases} \tilde{w}_{t-1}\{a_t(\tilde{\rho}) + b_t(\tilde{\rho})\}a_t(\rho)/a_t(\tilde{\rho}) & \text{if } I_t = 1, \\ \tilde{w}_{t-1}\{a_t(\tilde{\rho}) + b_t(\tilde{\rho})\}b_t(\rho)/b_t(\tilde{\rho}) & \text{if } I_t = 0, \end{cases} \tag{12}$$

with $\tilde{w}_1 = (\rho/\tilde{\rho})1_{\{I_1=1\}} + 1_{\{I_1=0\}}(1-\rho)/(1-\tilde{\rho})$, assuming $\mu_0 \sim N(0, \sigma^2)$.

3.2 A Numerical Example

Table 1 studies how the cv² bound for resampling affects performance, using the sum of squared error criterion $\mathrm{SSE} = E\{\sum_{i=1}^{n}(\hat{\mu}_i - \mu_i)^2\}$ to evaluate the performance of a procedure. For ρ = 0.001, 100 sequences of observations, each of length n = 10000, were generated. We applied SISR (M = 50) with $\tilde{\rho} = \rho$ to each sequence, using different cv² bounds. As pointed out in the paragraph preceding Section 3, resampling is performed at time t with the entire vector $I_{1:t}$ (instead of I_t) so that we can keep track of the most recent change-point. Table 1 displays the average number of resamplings (Resampling #) used for each cv² bound, together with the SSE and its standard error (s.e.) based on 100 simulation runs. It shows the best value of SSE around 188 when we choose 1 as the cv² bound, involving an average of 51 resamplings.

cv² bound	0	0.5	1	2	4	10	20	30	∞
Resampling #	9999	85	51	32	22	14	10	8	0
SSE	240.38	205.24	188.06	193.24	196.19	206.01	217.51	208.75	603.94
s.e.	9.65	6.53	6.73	6.31	6.53	6.69	8.27	7.28	28.11

Table 1. Effect of cv² bound on performance of SISR for mean shift model

We have also computed the SSE of the SISR filter based on μ_n and have found over 50% reduction in SSE by working with I_n instead of μ_n. In addition, we have studied how SISR performs when different $\tilde{\rho}$'s are used in the sampling distribution by simulating data from the same setting as Table 1, but with the cv² bound fixed at 1. Our results in Table 2 show that for ρ (= 0.001) < $\tilde{\rho}$ < 100ρ, the SSE is always smaller than that of $\tilde{\rho} = \rho$, with the smallest SSE at $\tilde{\rho}$ = 5ρ, which shows the benefits of tilting.

$\tilde{\rho}$	ρ	2ρ	4ρ	5ρ	6ρ	7ρ	10ρ	15ρ	50ρ	100ρ
SSE	188.06	169.12	159.58	150.82	152.07	151.50	158.80	159.84	169.86	185.42
s.e.	6.73	5.39	5.30	4.63	4.37	4.61	4.60	5.07	5.15	5.00

Table 2. Effect of $\tilde{\rho}$ on performance of SISR for mean shift model with ρ = 0.001

3.3 Change-Point ARX Models

Letting $\theta_t = (a_{1,t}, \ldots, a_{k,t}, b_{1,t}, \ldots, b_{h,t})^T$, we can write the ARX model

$$y_t + a_{1,t}y_{t-1} + \ldots + a_{k,t}y_{t-k} = b_{1,t}u_{t-1} + \ldots + b_{h,t}u_{t-h} + \epsilon_t \tag{13}$$

in the regression form $y_t = \theta_t^T \varphi_t + \epsilon_t, t \geq t_0 \stackrel{\Delta}{=} \max(k,h) + 1$. Suppose that the change-times of θ_t form a discrete renewal process with parameter ρ, or equivalently, that $I_t \stackrel{\Delta}{=} 1_{\{\theta_t \neq \theta_{t-1}\}}$ are independent Bernoulli random variables with $P(I_t = 1) = \rho$ for $t \geq t_0 + 1$, assuming $I_{t_0} = 1$. At a change-point, θ_t takes a new value which is assumed to have the multivariate normal distribution with mean μ and covariance matrix V. Assume also that the ϵ_t are independent normal with mean 0 and variance v_ϵ, which is chosen to be 1 in the following for simplicity.

Let C_t be the most recent change-time up to time t. The conditional distribution of θ_t given C_t, y_t and $\varphi_s, s \leq t$, is normal with mean $\mu_{C_t,t}$ and covariance matrix $V_{C_t,t}$, where for $t_0 < s \leq t$,

$$V_{s,t} = \left(V^{-1} + \sum_{j=s}^{t} \varphi_j \varphi_j^T \right)^{-1}, \ \mu_{s,t} = V_{s,t} \left(V^{-1}\mu + \sum_{j=s}^{t} y_j \varphi_j \right), \tag{14}$$

which can be computed by standard recursions that follow from the matrix inversion lemma:

$$\begin{aligned}
V_{t,t} &= V - V\varphi_t \varphi_t^T V/(1 + \varphi_t^T V \varphi_t), \\
V_{s,t} &= V_{s,t-1} - V_{s,t-1}\varphi_t \varphi_t^T V_{s,t-1}/(1 + \varphi_t^T V_{s,t-1}\varphi_t) \text{ if } s < t, \\
\mu_{s,t} &= \mu_{s,t-1} + V_{s,t-1}\varphi_t (y_t - \varphi_t^T \mu_{s,t-1})/(1 + \varphi_t^T V_{s,t-1}\varphi_t).
\end{aligned}$$

Therefore, analogous to (9), the optimal filter is given by

$$\begin{aligned}
E(\theta_n | y_n, \varphi_t, t \leq n) &= E\{E(\theta_n | C_n, y_n, \varphi_t, t \leq n) | y_n, \varphi_t, t \leq n\} \\
&= E\{\mu_{C_n,n} | y_n, \varphi_t, t \leq n\}.
\end{aligned} \tag{15}$$

We can compute (15) by simulating M trajectories $I_{1:n}^{(j)}$ (j = 1,...,M) via sequential importance sampling with resampling. The proposal distribution Q is similar to that in Section 3.1. Analogous to (10), the conditional distribution of I_t given $I_{1:t-1}$ is Bernoulli assuming the values 1 and 0 with probabilities in the proportion

$$\begin{aligned}
&\frac{\rho}{(1 + \varphi_t^T V \varphi_t)^{1/2}} \ \phi \left(\frac{y_t}{(1 + \varphi_t^T V \varphi_t)^{1/2}} \right) \\
: \quad &\frac{1-\rho}{(1 + \varphi_t^T V_{C_{t-1},t-1} \varphi_t)^{1/2}} \ \phi \left(\frac{y_t - \mu_{C_{t-1},t-1}^T \varphi_t}{(1 + \varphi_t^T V_{C_{t-1},t-1} \varphi_t)^{1/2}} \right).
\end{aligned} \tag{16}$$

Letting $a_t(\rho)$ and $b_t(\rho)$ denote the two terms in (16), we can define the importance weights w_t recursively by (11). Resampling is performed when the squared coefficient of variation

of the importance weights $w_t^{(j)}$ exceeds some threshold, which we can choose as 1 that usually works quite well. When ρ is small, we can modify Q by increasing ρ in (16) to $\tilde{\rho}$ and adjusting the importance weights accordingly.

Chen and Lai (2007, Section IIIA) have applied the above particle filter, with M = 50 sequentially generated trajectories to an open-loop change-point ARX model with k = 2, h = 1, ρ = 0.001, V =identity matrix. The actual autoregressive parameters are assumed to belong to the stability region $\{(a_{1,t}, a_{2,t}) : \ |a_{1,t}| + |a_{2,t}| < 1\}$, and the inputs are assumed to be independent standard normal random variables. They carry out simulation studies of the Bayes estimate (particle filter) $\hat{\theta}_t^{(B)}$ of θ_t given by (14)-(15) that ignores, for computational and analytic tractability, the stability constraint on the prior distribution θ_t. These studies also compare $\hat{\theta}_t^{(B)}$ with the following two modifications $\hat{\theta}_t^{(W)}$ and $\hat{\theta}_t^{(F)}$ of the usual least squares estimate which have been commonly used to handle occasional jumps in θ_t (cf. Benveniste et al., 1987, pp. 140, 161):

a. Sliding window estimate $\hat{\theta}_t^{(W)}$: The least squares estimate is applied only to data in the immediate past, i.e., to the data set $\{(y_s, \varphi_s) : t - k < s \leq t\}$, where k is the window size.
b. Forgetting factor estimate $\hat{\theta}_t^{(F)}$: A weighted least squares estimate is used, with weight p^{t-s} for (y_s, φ_s), i.e., the estimate at time t minimizes $\Sigma_{s=1}^{t} p^{t-s}(y_s - \theta^T \varphi_s)^2$; where 0 < p < 1 is the "forgetting factor" to discount past observations.

They use the following two performance measures

$$\mathrm{SSE}_1 = E\{\sum_{t=n'}^{n''}[(\hat{\theta}_t - \theta_t)^T \varphi_t]^2\}, \quad \mathrm{SSE}_2 = E\{\sum_{t=n'}^{n''} ||\hat{\theta}_t - \theta_t||^2\} \tag{17}$$

to compare these estimates. The second measure considers how well $\hat{\theta}_t$ estimates θ_t, whereas the first measure evaluates how well $\hat{\theta}_t^T \varphi_t$ estimates the minimum variance predictor $\theta_t^T \varphi_t$ of y_{t+1}. The results reported in their Table 2, which chooses 1-500, 501-1000, 1001-2000 and 2001-3000 as the ranges from n' to n'', show substantial improvements of $\hat{\theta}_t^{(B)}$ over $\hat{\theta}_t^{(W)}$ and $\hat{\theta}_t^{(F)}$, especially for n' exceeding 1000.

3.4 Application of Fast Particle Filters to Adaptive Control

Section IIIB of Chen and Lai (2007) considers the control problem of choosing the inputs u_t in the ARX model (13) so that the outputs y_{t+1} are as close as possible (in L^2) to y_{t+1}^*, some reference signal such that $\{y_n^*\}$ and $\{\theta_n, \epsilon_n\}$ are independent. In the case of known θ_{t+1}, the optimal input is defined by $\theta_{t+1}^T \varphi_{t+1} = y_{t+1}^*$, assuming that $b_{1,t+1} \neq 0$. When θ_{t+1} is unknown, the *certainty equivalence rule* replaces θ_{t+1} in the optimal input by an estimate $\hat{\theta}_{t+1|t}$ based on the observations up to time t so that u_t is given by $\hat{\theta}_{t+1|t}^T \varphi_{t+1} = y_{t+1}^*$ if $\hat{b}_{1,t+1|t} \neq 0$. Letting

$$\begin{aligned}\gamma_{t+1} &= (a_{1,t+1},\ldots,a_{k,t+1},b_{2,t+1},\ldots,b_{h,t+1})^T,\\ \hat{\gamma}_{t+1|t} &= (\hat{a}_{1,t+1|t},\ldots,\hat{a}_{k,t+1|t},\hat{b}_{2,t+1|t},\ldots,\hat{b}_{h,t+1|t})^T,\\ \psi_t &= (-y_t,\ldots,-y_{t-k+1},u_{t-1},\ldots,u_{t-h+1})^T,\end{aligned}$$

they modify the certainty equivalence rule by

$$u_t = \begin{cases}(y^*_{t+1} - \hat{\gamma}^T_{t+1|t}\psi_t)/\hat{b}_{1,t+1|t} & \text{if } |\hat{b}_{1,t+1|t}| \geq \delta_t,\\ y^*_{t+1} + \omega_t & \text{if } |\hat{b}_{1,t+1|t}| < \delta_t,\end{cases} \tag{18}$$

where δ_t is some small prespecified number and ω_t is extraneous noise used to enhance the information content of the reference signal (including the case $\omega_t \equiv 0$ if the reference signal is already persistently exciting); see Caines & Lafortune (1984).

Chen and Lai (2007) also consider an alternative approximation to the optimal control in the case of unknown θ_{t+1} by using the one-step ahead error $y_{t+1} - y^*_{t+1}$ without making use of dynamic programming to determine how the current control u_t impacts on the information content of the estimates $\hat{\theta}_{s+1|s}$ for the future errors $y_{s+1} - y^*_{s+1}$ $(s > t)$. Noting that

$$\begin{aligned}&\arg\min_u E\{(b_{1,t+1}u + \gamma^T_{t+1}\psi_t - y^*_{t+1})^2 | y_t, y_{t-1}, u_{t-1}, ..., y_1, u_1\}\\ =\ &\{y^*_{t+1}E_t(b_{1,t+1}) - E_t(b_{1,t+1}\gamma^T_{t+1})\psi_t\}/E_t(b^2_{1,t+1}),\end{aligned} \tag{19}$$

where E_t denotes conditional expectation given $\{y_t, y_{t-1}, u_{t-1}, \ldots, y_1, u_1\}$, they define the following variant of (18) that incorporates uncertainty adjustments due to unknown parameters into the optimal rule $\theta^T_t\varphi_{t+1} = y^*_{t+1}$ assuming known θ_t:

$$u_t = \begin{cases}\{y^*_{t+1}E_t(b_{1,t+1}) - E_t(b_{1,t+1}\gamma^T_{t+1})\psi_t\}/E_t(b^2_{1,t+1}) & \text{if } \sqrt{E_t(b^2_{1,t+1})} \geq \delta_t,\\ y^*_{t+1} + \omega_t & \text{if } \sqrt{E_t(b^2_{1,t+1})} < \delta_t.\end{cases} \tag{20}$$

To implement this adaptive control rule, one needs to compute the one-step ahead predictors $E_t(b_{1,t+1})$, $E_t(b^2_{1,t+1})$ and $E_t(b_{1,t+1}\gamma^T_{t+1})$. Note that

$$\begin{aligned}&p(\theta_{t+1} \mid y_t, y_{t-1}, u_{t-1}, \ldots, y_1, u_1)\\ =\ &(1-\rho)p(\theta_t \mid y_t, y_{t-1}, u_{t-1}, \ldots, y_1, u_1) + \rho p(\theta_{t+1}).\end{aligned} \tag{21}$$

The first term on the right hand side of (21) can be approximated by fast particle filters, whereas the second term corresponds to a change-point at time t+1. Note that replacing $(E_t(b_{1,t+1}),\ E_t(b^2_{1,t+1}),\ E(b_{1,t+1}\gamma^T_{t+1}))$ by $(\hat{b}_{1,t+1|t},\ \hat{b}^2_{1,t+1|t}, \hat{b}_{1,t+1|t}\hat{\gamma}^T_{t+1|t})$ in (20) reduces it to the certainty equivalence rule (18), which simply uses the estimates $(\hat{b}_{1,t+1|t}, \hat{\gamma}^T_{t+1|t})$ to substitute for $(b_{1,t+1}, \gamma^T_{t+1})$ in the optimal control assuming known θ_{t+1}. The rule (20) introduces uncertainty adjustments for the unknown θ_{t+1} by considering the expected one-step ahead control error $E_t(y_{t+1} - y^*_{t+1})^2$ that leads to (19), and by introducing extraneous

white noise to enhance the information content of the control for future estimates of θ_s whenever (19) has a small denominator that may lead to a large (and numerically unstable) control action. The choice of δ_t depends on whether θ_t is known to belong to some stability region. If the θ_t are restricted to a stability region, then white noise perturbations do not destabilize the system while improving the experimental design for future estimates of θ_s. On the other hand, without such a *priori* stability assumption on the θ_t, small δ_t should be used in (20) because the perturbations can have an explosive effect.

Table III of Chen and Lai reports the results of a simulation study for an ARX model (13) in which $k = 2$, $h = 1$ and the inputs u_t are determined by the certainty equivalence rule (18) or the uncertainty-adjusted certainty equivalence rule (20), in which $\delta_t = 1/4$ and the ω_t are independent and identically distributed normal random variables. The $(b_{1,t}, b_{2,t})$ are assumed to belong to a stability region. The table shows that the certainty equivalence rules based on $\hat{\theta}_t^{(W)}$ or $\hat{\theta}_t^{(F)}$ perform much worse than those based on $\hat{\theta}_t^{(B)}$ implemented by fast particle filters, and that the modification (20) of the certainty equivalence (18) based on $\hat{\theta}_t^{(B)}$ outperforms (18).

Chen and Lai (2007, Table IV) also consider the case where the $(b_{1,t}, b_{2,t})$ do not belong to a stability region. They show that by weakening the extraneous perturbations (specifically choosing $\delta_t = \text{var}(\omega_t) = 0.04$, instead of 1/4 for which the system with inputs (20) becomes unstable), the adaptive rule (20) can stabilize the system and still performs well.

3.5 Extensions to Hammerstein and Nonlinear ARX Systems

The particle filter described in (14) - (16) can be applied to estimate the piecewise constant θ_t in the general stochastic regression model

$$y_t = \theta_t^T \varphi_t + \epsilon_t, \tag{22}$$

in which φ_t is a vector-valued function of past lagged outputs and inputs and the change-points of θ_t form a discrete renewal process with parameter ρ, with θ_t taking a new value from the $N(\mu, V)$ distribution at each change-point. The ARX model (13) is only a special case of (22) with φ_t given by (1). Another important special case is the Hammerstein system that has a static nonlinearity on the input side, replacing u_t in (13) (and therefore (1) accordingly) by some nonlinear transformation $f(u_t)$. When f is unknown, it is usually approximated by a polynomial $f(u) = c_1 u + \cdots + c_l u^l$ (Ljung, 1987). To identify the Hammerstein system, we express it in the form of (22) with

$$\varphi_t = (-y_{t-1}, \ldots, -y_{t-k}; u_{t-1}, \ldots, u_{t-h}, \ldots, u_{t-1}^l, \ldots, u_{t-h}^l)^T.$$

Instead of using a polynomial to approximate f, we can use other basis functions (e.g., splines), yielding the representation $f(u) = \sum_{i=1}^{l} c_i \psi_i(u)$. Moreover, we can allow nonlinear interactions among the lagged outputs by making use of basis function approximations, and thereby express nonlinear ARX models with occasionally changing parameters in the form of (22) with

$$\begin{aligned} \varphi_t \;=\; & (\tilde{\psi}_1(y_{t-1}, \ldots, y_{t-k}), \ldots, \tilde{\psi}_m(y_{t-1}, \ldots, y_{t-k}); \\ & \psi_1(u_{t-1}), \ldots, \psi_l(u_{t-1}), \ldots, \psi_1(u_{t-h}), \ldots, \psi_l(u_{t-h}))^T. \end{aligned}$$

4. Particle Filters in Robotic Control and Planning

The monograph by Thrun et al. (2005) gives a comprehensive treatment of the emerging field of probabilistic robotics. Here we summarize several basic concepts that are related to particle filters, referring to the monograph and other papers for details, and describe some ongoing work in this area.

4.1 Robot Motion Models

As in Thrun et al. (2005), we restrict to mobile robots in planar environments for which the *pose* x_t of a robot at time t is represented by $(x_{t1}, x_{t2}, \theta_t)^T$, where $(x_{t1}, x_{t2})^T$ represents the robot's position in the plane and θ_t its angular orientation. If we drop the restriction of planar motion, then x_t is a 6-dimensional vector in which the first three components are the Cartesian coordinates and θ_t consists of the three Euler angles relative to the coordinate frame. The *velocity motion model* of a probabilistic robot is specified by the conditional density $p(x_t|u_{t-1}, x_{t-1})$, in which u_s is a motion command that depends on all observations up to stage s and controls the robot through a translational velocity v_s and a rotational velocity w_s, i.e., $u_s = (v_s, w_s)$; see Thrun et al. (2005, pp. 127-132) for concrete examples. An alternative to the use of the robot's velocities to evaluate its motion over time is to use its odometry measurements for u_t in $p(x_t|u_{t-1}, x_{t-1})$, leading to the *odometry motion model;* see Thrun et al. (2005, pp. 133-139).

The preceding description of robot motion does not incorporate the nature of the environment. In practice, there is also a *map m,* which contains information pertaining to the places that the robot can navigate; for example, the robot's pose can only be in "free" space, which is the complement of space already occupied. A *map-based* motion model is specified by $p(x_t|u_{t-1}, x_{t-1}, m)$. A simple way to build such models is to combine $p(x_t|u_{t-1}, x_{t-1})$ and $p(x_t|m)$ by

$$p(x_t|u_{t-1}, x_{t-1}, m) \propto p(x_t|u_{t-1}, x_{t-1})p(x_t|m)/p(x_t);$$

see Thrun et al. (2005, pp. 140-143). Typical maps can be classified as feature-based or location-based. A feature-based map is a list of objects, called *landmarks,* in the environment along with the features. A prototypical location-based map is the *occupancy grid map* which assigns to any location a binary label that specifies whether the location is occupied by an object.

4.2 Environment Measurement Models

Mobile robots use their sensors to perceive their environment. Range finders, which are among the most popular sensors in robotics, measure the range to nearby objects along a beam (laser range finders) or within a cone (ultrasonic sensors). The sensor measurements y_t are typically vectors since many sensors generate more than one numerical measurement; e.g., range finders usually give entire scans of ranges. Sections 6.3 and 6.4 of Thrun et al. (2005) describe the *beam model and* an alternative model, called *likelihood field,* to model $p(y_t|x_t, m)$ for range finders. Instead of using raw sensor measurements, an alternative approach is to extract features from the measurements and it is particularly suited to feature-based maps; see Section 6.6 of Thrun et al. (2005).

4.3 Pose Estimation, Mapping and SLAM

The problem of estimating the pose of a robot relative to a given map of the environment is often called *localization.* It is a fundamental problem in robotics since nearly all tasks of a robot require knowledge of its location in the environment. In view of the hidden Markov model defined by $p(x_t|u_{t-1}, x_{t-1}, m)$ and $p(y_t|x_t, m)$ in Sections 4.1 and 4.2, estimation of the pose x_t from the measurements y_1,..., y_t for a given map representation is a filtering problem. Extended Kalman filters are often used because of their simplicity and fast updating capability; see Thrun et al. (2005, Section 7.4) for details. The most popular localization filters to date are particle filters, and Section 8.3 of Thrun et al. (2005) describes these filters and their computational issues.

The preceding paragraph assumes that the robot has a map that is given in advance. Acquiring such an *a priori* map is often a difficult problem in practice. *Mapping,* which is the task of a robot to learn the map from scratch, not only circumvents this difficulty but also enables the robot to adapt to changes in the environment. To see how the robot can learn a map, first consider occupancy grid mapping in the case where the poses are known exactly. An occupancy grid map m partitions the space into finitely many grid cells $m_1, \ldots, m_I$, where $m_i = 1$ (if occupied) or 0 (if free) for the ith cell. Putting a prior distribution on $(m_1, \ldots, m_I)$, Chapter 9 of Thrun et al. (2005) considers the posterior density $p(m|y_{1:t}, x_{1:t})$ since the poses are assumed known, and describes how the MAP (maximum a posteriori) estimate of m can be evaluated.

The ideas in the preceding two paragraphs can be combined and modified to address the actual problem of *simultaneous localization and mappling* (SLAM), which involves the posterior density $p(x_t, m|y_{1:t}, u_{1:t-1})$. A convenient way to learn the map is to use a feature-based approach involving landmarks. Typically, the robot has some uncertainty in identifying landmarks, especially those it has not observed previously. To incorporate this uncertainty in the *data association* decision, a *correspondence variable* c_t^j can be introduced to give the true identity of the jth observed feature (i.e., $c_t^j = i$ if the jth feature corresponds to the ith landmark). In this case, SLAM involves the posterior density $p(x_t, m, c_t|y_{1:t}, u_{1:t-1})$. This is sometimes called the "on-line SLAM posterior" to be distinguished from the "full SLAM posterior" $p(x_{1:t}, m, c_{1:t}|y_{1:t}, u_{1:t-1})$. Chapter 10 of Thrun et al. (2005) uses extended Kalman filters to approximate the on-line SLAM posteriors, while Chapter 11 describes an alternative linearization technique that builds a sparse graph of soft constraints to approximate the full SLAM posteriors. Chapter 12 modifies the off-line full SLAM approximation of Chapter 11 into an on-line approximation to $p(x_t, m, c_t|y_{1:t}, u_{1:t-1})$.

4.4 The FastSLAM Algorithm

FastSLAM uses particle filters to estimate the robot path and extended Kalman filters to estimate the map features. A key insight of FastSLAM is the factorization

$$p(x_{1:t}, m|y_{1:t}, u_{1:t-1}, c_{1:t}) = p(x_{1:t}|y_{1:t}, u_{1:t-1}, c_{1:t}) \prod_{i=1}^{I} p(m_i|x_{1:t}, u_{1:t-1}, c_{1:t}), \quad (23)$$

where m consists of I features $m_1, \ldots, m_I$ whose mapping errors are conditionally independent; see Section 13.2.1 of Thrun et al. (2005) for the derivation. As noted on p.

438 of Thrun et al. (2005), an important advantage of FastSLAM "stems from the fact that data association decisions can be made on a per-particle basis," and consequently, "the filter maintains posteriors over multiple data associations, not just the most likely one." Moreover, "FastSLAM is formulated to calculate the full path posterior — only the full path renders feature locations conditionally independent." While it solves the full SLAM problem, it is also an on-line algorithm "because particle filters estimate one pose at-a-time." Details of the development and implementation of FastSLAM are given in Sections 13.3-13.10 of Thrun et al. (2005). An important idea underlying FastSLAM is to use Rao-Blackwellized particle filters for certain state variables and Gaussian posteriors to represent all other state variables. Recent papers by Grisetti et al. (2005, 2007) use adaptive proposal distributions and selective resampling to improve the Rao-Blackwellized particle filters for learning grid maps, and provide a compact map model in which individual particles can share large parts of the model for the environment.

4.5 Path Planning for Robot Movement

Given an environment, the path planning problem for a robot is to choose the best path to reach a target location, starting from its initial pose. The traditional approach to robot motion planning is deterministic in nature, assuming that there is no uncertainty in the robot's pose over time and focusing on the complexities of the state space in the optimization problem. Chapter 14 of Thrun et al. (2005) incorporates uncertainty in the controls on the robot's motion by using methods from Markov decision processes (MDP) to solve the stochastic optimization problem, assuming that the robot's poses are fully observable or well approximated with negligible error. In MDP, x_{t+1} does not evolve deterministically from x_t and u_t, but is governed by a transition probability density function $p(x_{t+1}|x_t, u_t)$ with respect to some measure μ. A *Markov policy* uses control u_t that is a function of x_t at every stage t. More generally, a policy can choose u_t based on $(x_{1:t}, u_{1:t-1})$. However, because of Markovian transitions, it suffices to restrict to Markov policies in maximizing the total discounted reward

$$E\left\{\sum_{i=0}^{T} \alpha^i R(x_{t+i}, u_{t+i})|x_t\right\} \tag{24}$$

over all policies, where $0 < \alpha \leq 1$ is the discount factor, $R(\cdot)$ is the payoff function and T is the planning horizon (which may be infinite).

For the case $T = 1$, the myopic policy that chooses u_t to maximize $R(x_t, u_t)+\alpha E\{R(x_{t+1}, u_{t+1})|x_t, u_t\}$ is optimal. With longer planning horizons, one has to strike an optimal balance between the next-stage reward and the evolution of future rewards. The optimal policy can be determined by dynamic programming as follows. The value $V_T(x)$ of (24) for the optimal policy is called the *value function,* and it satisfies the Bellman equation

$$V_T(x) = \max_u \left\{R(x, u) + \alpha \int V_{T-1}(x')p(x'|x, u)d\mu(x')\right\}. \tag{25}$$

The optimal policy chooses the control $u = u_T(x)$ that maximizes the right-hand side of (25).

Unless the state space is finite or of dimension 1 or 2, direct computation of (25) via discretization of the state space is prohibitively difficult. One approach is to use a low-dimensional state approximation that assumes the value function to be relatively constant in the complementary state variables; see Thrun et al. (2005, pp. 505-507) who also note that "in higher dimensional situations it is common to introduce learning algorithms to represent the value function." Instead of working directly with the value function, it is more convenient to use the functions Q_n defined by backward induction via $Q_T(x) = \max_u R(x,u)$ and

$$Q_n(x) = \max_u \{R(x,u) + \alpha E[Q_{n+1}(x_{n+1})|x_n = x, u_n = u]\} \text{ for } T-1 \geq n \geq 1, \quad (26)$$

noting that $V_T = Q_1$. Since conditional expectation is a regression function, one can approximate Q_{n+1} in (26) by using ideas from nonparametric regression, which basically uses certain basis functions to approximate Q_{n+1} and estimates the coefficients of the basis functions by the method of least squares from simulated samples drawn from the conditional distribution of x_{n+1} given x_n and $u_n = u$; see Bertsekas and Tsitsiklis (1996) and Tsitsiklis and Van Roy (2001) for details.

4.6 Robotic Control via Approximate Dynamic Programming

Whereas path planning is usually carried out off-line before the robot is in motion, robotic control is concerned with on-line control of the motion of the robot to maximize a total discounted reward. It has to address the uncertainties in both the robot's poses and the control effects, which are incorporated in $p(x_t|u_{t-1}, x_{t-1}, m)$ and $p(y_t|x_t, m)$ in Sections 4.1 and 4.2. Accordingly Thrun et al. (2005, Chapter 15) use methods from partially observable Markov decision processes (POMDP) to address the corresponding stochastic control problem of maximizing (24) over control policies that are functions of the posterior distribution π_n of x_n given $y_{1:n}$ and $u_{1:n-1}$, instead of functions of x_n as in MDP because the x_n cannot be fully observed. Calling these posterior distributions *beliefs*, Thrun et al. (2005, p. 514) extend the Bellman equation (25) formally to

$$V_T(\pi) = \max_u \left\{ R(\pi, u) + \alpha \int V_{T-1}(\pi') p(\pi'|\pi, u)\, d\mu(\pi) \right\}, \quad (27)$$

where μ is a measure on the space $\mathcal{B}$ of beliefs and $p(\pi'|\pi, u)$ is the one-step transition probability density function of the belief π to π' when control u is taken. The optimal control chooses the maximizer u in (27) when one's current belief is π. This is tantamount to working with the Markov chain (x_t, π_t) on the state space $\mathcal{X} \times \mathcal{B}$, where $\mathcal{X}$ is the state space of the poses x_t. Since $\mathcal{B}$ is a set of probability measures, μ is a measure on the space of probability measures and the existence of the transition density function in (27) is "problematic". Moreover, "given the complex nature of the belief, it is not at all obvious that the integration can be carried out exactly, or that effective approximation can be found" (Thrun et al., 2005, p. 514).

Because of the inherent complexity of POMDP problems, the literature has focused almost exclusively on the infinite-horizon case $(T = \infty)$ so that the value function in the Bellman equation (27) does not depend on T and is a function of the posterior distribution (belief) only. Thrun et al. (2005, Sections 15.3 and 15.4) consider the case where the state space $\mathcal{X}$,

the observation space and the control set are all finite. If $\mathcal{X}$ has k elements, then any probability distribution π on $\mathcal{X}$ can be represented by (p_1,... ,p_k) such that $0 \leq p_i \leq 1$ and $\sum_{i=1}^{k} p_i = 1$. In this case, the value function (27) can be expressed as a convex, piecewise linear function of (p_1,... ,p_k).

For more general state spaces $\mathcal{X}$, Thrun (2000) has proposed a Monte Carlo procedure, called MC-POMDP, involving particle filters to perform approximate value iteration in $\mathcal{B}$. MC-POMDP uses a finite particle set to approximate a probability distribution π. Specifically, this iterative procedure updates the value function V at π by simulating for each applicable control u a sample of possible subsequent beliefs π' and then averaging over the simulated sample:

$$Q(\pi, u) = R(\pi, u) + \alpha E\{V(\pi')|\pi, u\}, \tag{28}$$

so that V is updated by $V(\pi) = \max_u Q(\pi, u)$ in the iterative procedure. The basic idea is taken from model-based reinforcement learning (Gordon, 1995; Kaebling et al., 1996; Sutton & Barto, 1998), in which function approximations such as neural networks, decision trees and spline basis functions are used to represent the value function V in MDPs. To extend the idea to POMDPs, the challenge lies in how to represent V since it is a function of a probability distribution on the state space, instead of the state itself. Thrun (2000) uses a "nearest neighbor" approximation to represent $V(\pi')$ in (28). His MC-POMDP algorithm keeps a set (database) of reference beliefs π_i, and associated values V_i. When a new belief state π' (not in the database) is generated, its V value is obtained by finding the k nearest neighbors in the database and taking a weighted average of the corresponding V_i values. To measure the distance of π_i from π', he convolves each particle with a normal $N(0,v)$ distribution having a small variance v so that π_i and π' can be represented by Gaussian mixtures, and then uses the Kullback-Leibler divergence to measure the distance (divergence) d_i of π_i from π'. Denoting the k nearest neighbors of π' by $\pi_1, \ldots, \pi_k$, $V(\pi')$ in (28) is approximated by $(\sum_{i=1}^{k} d_i^{-1} V_i)/(\sum_{i=1}^{k} d_i^{-1})$; see Section 16.4 of Thrun et al. (2005) for further details and refinements of the MC-POMDP algorithm. Because of the complexity of the actual value function, how well the above nearest neighbor method approximates V is formidable to explore. Although the value-function approximation approach has been the "dominant approach" in reinforcement learning, Sutton et al. (2000) have pointed out various limitations of this approach and have proposed an alternative approach that uses function approximations for the optimal policy to carry out approximate policy iteration in MDPs. Instead of extending this approximate policy iteration approach to POMDPs whose optimal policies are prohibitively difficult to visualize, Lai and Wu (2008) propose to begin with a good (albeit sub-optimal) policy that captures the essence of the control objective and quantifies the uncertainties concerning the current state reflected by the particle filter, and to use it as the base policy in a *rollout algorithm* (Bayard, 1991; Bertsekas, 2000; Han et al., 2006) that successively improves the base policy into a new base policy, iterating till convergence to the optimal policy.

5. Concluding Remarks

In this section we describe some issues that have not been addressed in the preceding sections and summarize the connections between the two control problems in Sections 3 and 4.

5.1 Adaptive Particle Filters

In the previous sections, we have assumed that the hidden Markov model has specified parameters. However, in practice, the HMM usually has unknown parameters that need to be estimated besides the unobservable states. We consider here a Bayesian formulation in which the unknown parameter vector θ has a prior distribution, so that θ can be incorporated into the state vector at the expense of increasing the dimension. Such augmentation of the state vector does not pose additional difficulties if it can still be conveniently simulated. Here we show that it is sometimes even possible to integrate out the unknown parameter vector θ, with respect to a posterior distribution, in the SISR filter. Whenever this is possible, integrating out θ can improve substantially the performance of the Monte Carlo method; this principle is called *marginalization* (Kong et al., 1994).

As an illustration, suppose that in the normal mean shift model, the probability ρ of change is unknown and is specified by a prior $Beta(\alpha, \beta)$ distribution with mean $\alpha/(\alpha+\beta)$, where α and β are positive integers. It turns out that when ρ in (10) is unknown but has a Beta prior distribution, we can follow the same arguments to come up with an analogous proposal distribution Q from which I_1,..., I_t are sampled sequentially. Using the closed-form expression for $P(I_t = \cdot | I_{1:t-1}, Y_{1:t})$, it can be shown that under Q, $I_t | I_{1:t-1}$ is Bernoulli assuming the values 1 and 0 with probabilities in the proportion

$$\begin{array}{c} \dfrac{n_{t-1,1}+\alpha}{(t-1+\alpha+\beta)(1+\sigma^2)^{1/2}}\phi\left(\dfrac{Y_t}{(1+\sigma^2)^{1/2}}\right) \\ : \quad \dfrac{n_{t-1,0}+\beta}{(t-1+\alpha+\beta)(1+v_t)^{1/2}}\phi\left(\dfrac{Y_t - v_t\sum_{j=C_{t-1}}^{t-1} Y_j}{(1+v_t)^{1/2}}\right), \end{array} \tag{29}$$

where $v_t = (t - C_{t-1} + \sigma^{-2})^{-1}$, $n_{t-1,1}$ is the number of 1's in $\{I_1, I_2,..., I_{t-1}\}$ and $n_{t-1,0} = t - 1 - n_{t-1,1}$ is the number of 0's in $\{I_1, I_2,..., I_{t-1}\}$. Note how closely (29) resembles (10). Accordingly, letting a_t and b_t denote the two terms in (29), the importance weights w_t are given recursively by $w_t = w_{t-1}(a_t + b_t)$, $t \geq 1$, with $w_0 = 1$.

Parallel to Table 1 that studies the performance of the SISR filter (with $M = 50$ and various cv^2 bounds) when $\rho = 0.001$ is known, Table 3 considers the performance of the adaptive filter when ρ is assumed to have the *Beta*(1,999) distribution (with mean 0.001). Performance is measured by the SSE(ρ) $= E_\rho\{\sum_{i=1}^{n}(\hat{\mu}_i - \mu_i)^2\}$ at $\rho = 0.001$, and also by the Bayes SSE, which is the expected value of SSE(ρ) when ρ is regarded as a *Beta*(1,999) random variable. Each result is based on 100 sequences of $n = 10000$ observations. As in Table 1, σ is assumed to be 1. Comparison of SSE(ρ) with the SSE values in Table 1 shows that the performance of the adaptive filter is comparable to that of the optimal filter that assumes ρ to be known. Table 3 shows a minimal value of SSE(ρ) and the best Bayes performance when the cv^2 bound is 2.

CV^2 Bound	0	0.5	1	2	4	10	20	30	∞
Resampling #	9999	100	50	35	30	15	12	9	0
SSE(ρ)	288.52	200.82	201.92	200.55	205.57	206.72	207.43	215.75	478.93
Bayes SSE	240.52	174.73	158.16	153.56	184.35	188.47	190.20	186.31	501.01
Resampling #B	9999	80	50	30	20	15	10	7	0

Table 3. Effect of cv² bound on performance of SISR for mean shift model with unknown ρ
Note: SSE(ρ) = Expected sum of squared errors at ρ = 0.001; Bayes SSE = Expected sum of squared errors when $\rho \sim Beta(1,999)$; Resampling # = Average number of resamplings when ρ = 0.001; Resampling #B= Expected number of resamplings when $\rho \sim Beta(1,999)$

5.2 Control Law Heuristics, Approximate Policy Iteration and On-Line Implementation

In Section 3.4 we have addressed the adaptive control problem by using simple heuristics like certainty equivalence and uncertainty adjustment. The treatment of robotic control reviewed in Section 4.6 uses more general and formal tools like approximate value iteration, reinforcement learning and MC-POMDP. Although these tools are powerful, they may be difficult to implement, as noted in Section 4.6. A major difficulty is that since the form of the value function is hard to guess, it is not clear what basis functions should be chosen to approximate the value function in carrying out approximate value iteration. This difficulty is compounded for POMDPs because the value function is a function of a probability distribution on the state space, as we have explained in Section 4.6. An alternative approach is to use control law heuristics to come up with a good practical policy as in Section 3.4, and to study its performance by Monte Carlo simulations and refine it by using the rollout method as in Lai and Wu (2008). Another challenge is the real-time computational requirement for on-line control, under which one can only afford to perform few iterations, especially when there is a Monte Carlo inner loop involving particle filters. This is why fast particle filters, which use efficient proposal distributions and resampling schemes like those in Section 3, or which use conditional independence of feature locations to perform data association on a per-particle basis as in FastSLAM in Section 4.4, are of particular importance for applications of particle filters to control of HMMs.

6. References

Bayard, D. S. (1991). A forward method for optimal stochastic nonlinear and adaptive control. *IEEE Trans. Automat. Contr.,* Vol 36, pp. 1046-1053.

Benveniste, A.; Metivier, M. & Priouret, P. (1987). *Adaptive Algorithms and Stochastic Approximations,* New York: Springer-Verlag.

Bertsekas, D. P. (2000). *Dynamic Programming and Optimal Control,* Athena Scientific, Belmont, MA.

Bertsekas, D. P. & Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming.* Belmont, MA: Athena Scientific.

Caines, P. E. & Lafortune, S. (1984). Adaptive control with recursive identification for stochastic linear systems. *IEEE Trans. Automat. Contr.,* Vol. 29, pp. 312-321.

Chan, H. P. & Lai, T. L. (2008). A general theory of particle filters in hidden Markov models and some applications. Tech. Report, Department of Statistics, Stanford University.

Chen, Y. & Lai, T. L. (2007). Identification and adaptive control of change-point ARX models via Rao-Blackwellized particle filters. *IEEE Trans. Automat. Contr.,* Vol 52, pp. 67-72.

Goodwin, G. C.; Ramadge, P. J. & Caines, P. E. (1981). Discrete time stochastic adaptive control. *SI AM J. Contr. Optimiz.,* Vol. 19, pp. 829-853.

Goodwin, G. C. & Sin, K. S. (1984). *Adaptive Filtering, Prediction and Control.* Englewood Cliffs, NJ: Prentice Hall.

Gordon, G. J. (1995). Stable function approximation in dynamic programming. *Proc. 12th Internal. Conf. Machine Learning,* pp. 261-268, Morgan Kaufmann.

Gordon, N. J.; Salmond, D. J. & Smith, A. F. M. (1993). A novel approach to non-linear and non-Gaussian Bayesian state estimation. *IEEE Proc. Radar & Signal Processing,* Vol. 140, pp. 107-113.

Grisetti, G.; Stachniss, C. & Burgard, W. (2005). Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation,* pp. 2432-2437, ISBN: 0-7803-8914-X.

Grisetti, G.; Tipaldi, G. D.; Stachniss, C.; Burgard, W. & Nardi, D. (2007). Fast and accurate SLAM with Rao-Blackwellized particle filters. *Robotics and Autonomous Systems archive,* Vol. 55 , No. 1, (January 2007) pp. 30-38, ISSN:0921-8890.

Guo, L. & Chen, H. F. (1991). The Astrom-Wittenmark self-tuning regulator revisited and ELS-based adaptive trackers. *IEEE Trans. Automat. Contr.,* Vol. 36, pp. 802-812.

Guo, L. & Ljung, L. (1993a). Exponential stability of general tracking algorithms. *IEEE Trans. Automat. Contr.,* Vol. 40, pp. 1376-1387.

Guo, L. & Ljung, L. (1993b). Performance analysis of general tracking algorithms. *IEEE Trans. Automat. Contr.,* Vol. 40, pp. 1388-1402.

Han, J.; Lai, T. L. & Spivakovsky, V. (2006). Approximate policy optimization and adaptive control in regression models. *Comput. Econ.,* Vol. 27, 433-452.

Kaebling, L. P.; Littman, M. L. & Moore, A., W. (1996). Reinforcement learning: A survey. *J. Articial Intelligence Research,* Vol. 4, pp. 237-285.

Kong, A.; Liu, J. S. & Wong, W. H. (1994). Sequential imputations and Bayesian missing data problems. *J. Amer. Statist. Assoc.,* Vol. 89, pp. 278-288.

Lai, T. L. & Wei, C. Z. (1987). Asymptotically efficient self-tuning regulators. *SIAM J. Contr. Optimiz,* Vol. 25, pp. 466-481.

Lai, T. L. & Wu, B. (2008). Approximate policy iteration in partially observable Markov decision processes with applications to robot control. *Tech. Report*, Department of Statistics, Stanford University.

Ljung, L. (1987). *System Identification: Theory for the User,* Englewood Cliffs, NJ: Prentice Hall.

Ljung, L. & Gunnarsson, S. (1990). Adaptation and tracking in system identification - a survey. *Automatica,* Vol. 26, pp. 7-21.

Ljung, L. & Soderstrom, T. (1983). *Theory and Practice of Recursive Identification.* MIT Press.

Meyn, S. P. & Brown, L. J. (1993). Model reference adaptive control of time varying and stochastic systems. *IEEE Trans. Automat. Contr.,* Vol. 38, pp. 1738-1753.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT Press.

Sutton, R. S.; McAllester, S. S. & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In: *Adv. Neural Information Processing Systems,* Vol. 12, pp. 1057-1063, MIT press.

Thrun, S. (2000). Monte Carlo POMDPs. In: *Advances in Neural Information Processing Systems 12,* S.A. Solla, T.K. Leen, and K.-R. Mller (Ed.), pp 1064-1070. MIT Press.

Thrun, S.; Burgard, W. & Fox, D. (2005). *Probabilistic Robotics.* MIT Press, Cambridge, MA.

Tsitsiklis, J. N. & Van Roy, B. (2001). Regression methods for pricing complex American-style options. *IEEE Trans. Neural Networks,* Vol. 12, pp. 694-703.

Yao, Y. (1984). Estimation of a noisy discrete-time step function: Bayes and empirical Bayes approaches. *Ann. Statist.,* Vol. 12, pp. 1434-1447.

Frontiers in Adaptive Control

Edited by Shuang Cong

ISBN 978-953-7619-43-5
Hard cover, 334 pages
Publisher InTech
Published online 01, January, 2009
Published in print edition January, 2009

The objective of this book is to provide an up-to-date and state-of-the-art coverage of diverse aspects related to adaptive control theory, methodologies and applications. These include various robust techniques, performance enhancement techniques, techniques with less a-priori knowledge, nonlinear adaptive control techniques and intelligent adaptive techniques. There are several themes in this book which instance both the maturity and the novelty of the general adaptive control. Each chapter is introduced by a brief preamble providing the background and objectives of subject matter. The experiment results are presented in considerable detail in order to facilitate the comprehension of the theoretical development, as well as to increase sensitivity of applications in practical problems

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen