Kalman and Particle Filters

Carl Richardson

15th April 2021

Contents

1 Introduction

The goal of online state estimation is to sequentially infer the state of the system, in real-time, using the sequence of past observations. Naturally, the state space structure provides a framework for modelling the dynamics of an unobserved state and an observed function of that state. The probabilistic Bayesian setting gives an approach to modelling how a prior belief, the predicted density of the state conditioned on the past sequence of observations, is updated given new information, the arrival of the new observation, to provide a posterior density of the state. Through marginal and conditional probability, Bayesian state estimation can be written as a recursive relationship using the densities of the state and observation equations. Unfortunately, obtaining and sampling from the posterior distribution is not always easy so algorithms which can approximate the posterior whilst accounting for these problems must be considered.

If the system is linear, the Kalman filter (KF) provides optimal state estimation [1]. However, realistic systems are non-linear so alternative approaches must be found. Section two of this report considers a non-linear state estimation problem in the form of a time-varying autoregressive (AR) process. For this study, two Monte Carlo approaches were considered, sequential importance sampling (SIS) and it's extension- the particle filter (PF). Furthermore, section three formulates a logistic regression problem as a state estimation problem and this time, the extended Kalman filter (EKF) and the PF were used. Both studies begin by briefly describing the problem, followed by a description of the algorithms and finally, analysis of their performance.

2 Synthetic Time Varying Autoregressive Problem

The second order time-varying AR problem is described by the discrete state space formulation, shown by equation 1. The two dimensional state equation is shown explicitly, where each state variable, $a_{i,n}$, represents a time varying parameter of the AR process. The time varying parameters are functions of time, n, with additive process noise, $v_n \backsim N(0, 0.1)$. The AR process is described by the observation equation, where the process output, at time n, is the dot product of the parameters at time n and the two previous process outputs, $x_n = [S_{n-1}, S_{n-2}]$, plus some observation noise $w_n \backsim N(0, 1)$. Figure 1 illustrates one realisation of this system over 100 iterations. To simulate the online estimation problem, observations of the time varying AR process arrive sequentially and the parameters are unobserved. For the purpose of filter implementation, the AR process and the state equations are assumed to be known, along with the distribution of the process and observation noise.

$$\begin{aligned} a_{0,n} &= A_{0,0} + 0.1cos(\frac{2\pi n}{N}) + v_n \\ a_{1,n} &= A_{1,0} + 0.1sin(\frac{\pi n}{N}) + v_n \\ S_n &= a_n^T x_n + w_n \end{aligned} \tag{1}$$

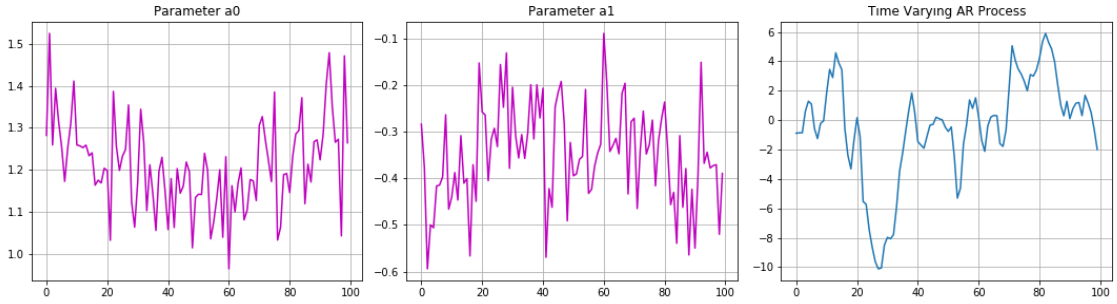

Figure 1: Time varying AR process

2.1 Sequential Importance Sampling

Sequential Importance Sampling (SIS) is a Monte Carlo method which approximates a target density, that cannot be sampled from, by random sampling from a proposal density. This is shown by equation 2, where the posterior is the target density and the state equation provides the proposal density which L samples are drawn from $\hat{a}_n^l \backsim N(a_n, 0.1I)$. To complete the approximation of the posterior, each sample has an associated weight. These are recursively computed by normalising equation 3, where the previous weights are scaled by the

likelihood of the latest observation conditioned on the sampled states. Once the posterior distribution, for each time step, has been approximated, the state can be estimated by computing the mean of the samples, for each time step. This is done by summing the product of the samples and their corresponding weights. Furthermore, in the same way, the variance of each state can be estimated to provide confidence intervals of the estimate.

$$p(a_n|S_{1:n}) \approx \sum_{l=1}^{L} \hat{w}_n^l \delta_{\hat{a}^l}(a_n) \tag{2}$$

$$w_n^l = w_{n-1}^l p(S_n|\hat{a}_n^l) \tag{3}$$

Figure 2 illustrates the statistical properties of the SIS estimate, using 100 samples, compared to the noise free, time varying parameters. The root mean squared error (rmse) between the true parameters and the SIS estimates was 0.056 and 0.059, respectively. This small error is illustrated by the accurate SIS estimate of both parameters over 100 iterations. The true trajectory of the time varying parameters falls comfortably within the 95% confidence interval (two standard deviations either side of the mean); however, the boundaries of this confidence interval vary rapidly over time. Figure 2 (right) shows a rapid decline in the effective sample size (ESS). After 30 iterations, the ESS varies between 1 and 5 illustrating significant weight degeneracy.

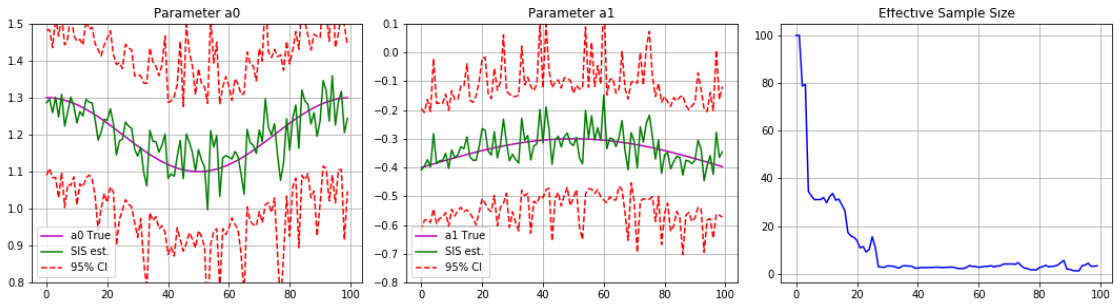

Figure 2: SIS estimation, 95% confidence intervals and effective sample size

2.2 Particle Filter

The PF, also known as the sequential importance resampling (SIR) filter, is an extension of the SIS filter. As shown by figure 2 (right), the ESS rapidly decayed over time when the SIS filter was used. This represents how the number of non-negligible weights rapidly decayed over time. The result of this undesirable effect is the number of samples with negligible weights increasing, leading to more samples having a negligible contribution to the approximation of the posterior density. To counteract the weight degeneracy, the PF adds a resampling step after the weights are evaluated in the SIS algorithm. In the case of systematic resampling, a random number is sampled from a uniform density between 0 and $1/L$, $u_1 \backsim U[0, 1/L]$. The sample corresponding to the weight which occupies the interval of the cumulative density function (CDF), that u_1 falls in, is the first sample. The remaining $(L-1)$ samples are drawn by recursively adding $1/L$ to the cumulative probability of the first sample and sampling at each of those points. This results in a new set of samples (particles) with uniform probability. The samples with high probability in the evaluation stage will still occur most frequently, but additional samples will also be considered instead of effectively being discarded. Keeping low probability samples is important as they may be important later in the estimation process.

The results of performing this resampling step, each time the ESS fell below $0.75L$, are shown by figure 3. The rmse between the true parameters and the PF estimate was 0.012 and 0.011 respectively. This improved estimate, compared to the SIS filter, is illustrated by the estimated trajectory tracing out the true trajectory of the time varying parameters more tightly. Furthermore, the 95% confidence intervals of both parameters have become smoother indicating the reduced variance in the estimate. Figure 3 (right) shows how the ESS now remains equal to or greater than 75% of the actual sample size across the full simulation.

2.3 Analysis

In the examples above, the rmse between the estimated and true parameters reduced by 79% and 82%, respectively, when the PF filter was used. A more extensive comparison of the two algorithms was performed by computing the mean and variance of the rmse, over 30 experiments, for five different sample sizes. Figure 4 shows the rmse, of each filter, varying in a similar way for both parameters. The rmse of the SIS filter has a large variance for each sample size, and displays no general trend; this may have been skewed due to the small

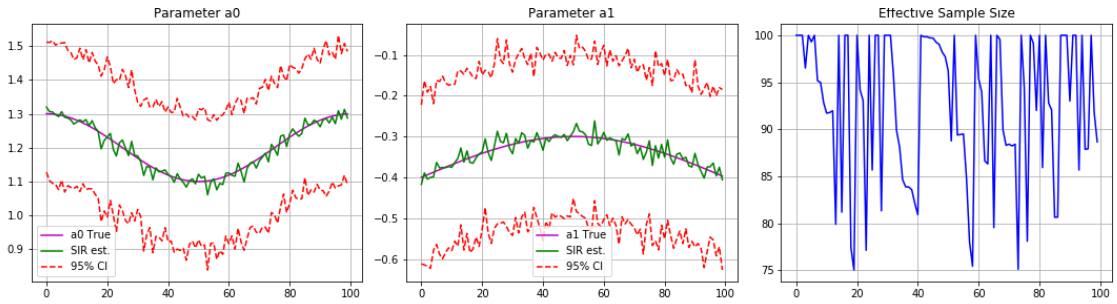

Figure 3: SIR estimation, 95% confidence intervals and effective sample size using 100 particles

error. On the other hand, the rmse of the PF decays and plateaus as the sample size increases, with minimal variance. The smaller rmse and variance validates the benefit of the additional resampling step. Furthermore, although the performance of the PF was an improvement on the SIS filter; both performed much better than the KF. Due to the non-linear process of the time varying parameters, the KF failed to predict the next state well and these errors compounded as the estimates propagated through the system.

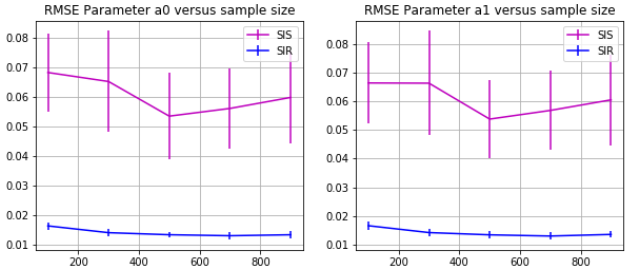

Figure 4: Average RMSE between true and estimated parameters dependent on sample size

3 Logistic Regression Problem

This case study investigated the use of state estimation tools to estimate the parameters of a logistic function, used to model the posterior probability a given sample was drawn from a binary class. The likelihood functions, which the samples were drawn from, are given by equations 4 and 5, where c_i represents class $i = \{0, 1\}$. The common covariance was given in the problem definition and the mean was of the form $\mu = [-\alpha, \alpha]$, where $\alpha = 2.5$. Figure 5 shows one realisation when 300 samples were drawn from class 0 and 700 from class 1. The posterior probability of sample n being in class 0 is given by equation 6 where the normalised decision boundary was given by the true parameters $\theta = [0.70, -0.72]$. The vector in figure 5 illustrates the direction of this decision boundary which provides perfect classification of the sampled data.

$$p(x|c_0) = N(x|\mu, \Sigma) \tag{4}$$

$$p(x|c_1) = N(x|-\mu, \Sigma) \tag{5}$$

$$y_n = p(c_0|x_n) = f(x_n; \theta) = \frac{1}{1 + exp(-\theta^T x_n)} \tag{6}$$

To formulate this as an online learning problem, data (x_n, y_n) would arrive sequentially. Equation 7 gives the state space model of this problem, where the estimated boundary parameters, $\hat{\theta}_n$, are randomly updated by sampling from $\hat{\theta}_n \backsim N(\hat{\theta}_{n-1}, Q)$, since the process noise, w_n, is zero mean Gaussian with covariance Q. The observation equation models the observed class probability and is based on equation 6. In the state space model, the sample x_n is known and becomes the parameter of the time varying logistic function whilst $\hat{\theta}_n$ is the state variable to be estimated. The observation noise is modelled by $v_n \backsim N(0, R)$. For these experiments, a process covariance, $Q = 10I$, and a measurement variance $R = 1$ were used. In this case study, the goal is for the estimated state trajectory to converge to the true parameter values as opposed to estimating an unknown time varying state, as in the previous task.

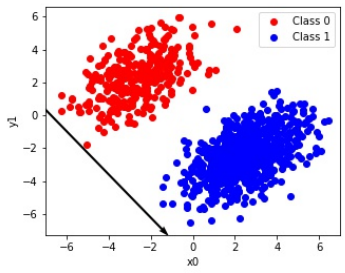

Figure 5: 2-d binary classification data

$$\begin{aligned} \hat{\theta}_n &= \hat{\theta}_{n-1} + w_n \\ y_n &= f(\hat{\theta}_n; x_n) + v_n \end{aligned} \tag{7}$$

3.1 Extended Kalman Filter

This section investigates the performance of the EKF for estimating the boundary parameters. This algorithm extends the KF by including an additional linearisation step which makes it a feasible option for non-linear state estimation. The algorithm is described below; the estimated boundary parameters are initialised by randomly sampling from the uniform distribution on the interval $[-1, 1]$ since we are only interested in the direction of the boundary. The same reason applies for normalising the parameters in step 10.

Algorithm 1 Extended Kalman Filter

1: Initialise $\hat{\theta}_0^+$ by sampling both parameters from a uniform distribution on [-1,1]
2: Initialise $P_0^+ = 10^3 I$ ▷ high uncertainty in initial estimate
3: **for** $n = 1 : N$ **do**
4: $\quad \hat{\theta}_n^- = \hat{\theta}_{n-1}^+$ ▷ Time update (State)
5: $\quad P_n^- = P_{n-1}^+ + Q$ ▷ Time update (Error Covariance)
6: $\quad F_n = f^2(\hat{\theta}_n^-; x_n) exp(-x_n^T \hat{\theta}_n^-) x_n^T$ ▷ Linearise output equation. $F_n \in R^{1\times 2}$
7: $\quad K_n = P_n^- F_n^T (F_n P_n^- F_n^T + R)^{-1}$ ▷ Kalman Gain
8: $\quad \hat{\theta}_n^+ = \hat{\theta}_n^- + K_n(y_n - f(\hat{\theta}_n^-; x_n))$ ▷ Observation Update (State)
9: $\quad P_n^+ = (I - K_n F_n) P_n^i$ ▷ Observation Update (Error Covariance)
10: $\quad$ Normalise $\hat{\theta}_n^+$ ▷ Boundary Direction
11: Return $\{(\hat{\theta}_n^+, P_n^+)\}_{n=0}^N$

Figure 6 shows the evolution, over 250 iterations, of the estimated boundary parameters and error covariance. The figure shows, in this case, the parameters converge to the true values within 82 iterations. This coincided with the time the uncertainty in the estimates reduced significantly since both estimation error variances dropped and varied between 5 and 120. The average convergence time over 20 experiments was 73.1 iterations with a standard deviation of 73.5. Furthermore, the same experiment was repeated when $\alpha = 1$, this had an average convergence time of 11.6 and a standard deviation of 6.5. The data for this experiment is shown by figure 8, along with one trial, figure 9 in the appendix. These results show convergence occurs much quicker when the classes are not linearly separable. The reason for this is that the set of boundary parameters which could have produced the observations is much smaller; hence the solution is found more quickly.

3.2 Particle Filter

This section investigates the performance of the PF for estimating the boundary parameters. This is the same algorithm as described in section 2.2, but repurposed for the parameter estimation task. The algorithm is described below; the estimated boundary parameters are initialised by randomly sampling from the uniform distribution on the interval $[-1, 1]$. The parameters falling within this region was assumed since only the direction of the boundary vector was important. For this same reason, the particles were normalised in step 8 as well.

Figure 7 shows the evolution of the estimated boundary parameters and the 95% confidence intervals, over 250 iterations and using 100 particles. After this time had elapsed, the rmse between the true and estimated parameters was 0.01 and 0.16 respectively. Although an error existed, the estimated parameters still produced

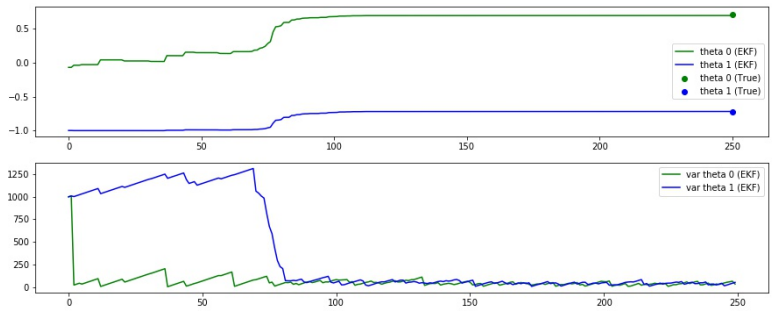

Figure 6: Evolution of boundary estimate and error covariance

Algorithm 2 Particle Filter

Initialise L particles, $\{\hat{\theta}_0^l\}_{l=1}^L$, by sampling both parameters from a uniform distribution on [-1,1]
Normalise particles ▷ Unit vectors
Initialise weights, $\{w_0^l\}_{l=1}^L$, where $w_0^l = 1/L$
for $n = 1 : N$ **do**
    Sample L particles $\hat{\theta}_n^l \backsim N(\hat{\theta}_{n-1}^l, Q)$ ▷ Predict next state
    Normalise particles
    Evaluate weights $w_n^l = w_{n-1}^l N(y_n | f(\hat{\theta}_n^l; x_n), R)$ ▷ Update based on observation
    Normalise weights
    if $ESS(w_n) < 0.75L$ **then**
        Resample ▷ As described in section 2.2
        Return $\{(\hat{\theta}_n^l, w_n^l = 1/L)\}_{l=1}^L$
Return $\{\{(\hat{\theta}_n^l, w_n^l)\}_{l=1}^L\}_{n=0}^N$

100% classification accuracy, just like the true parameters. Furthermore, the true parameters fall comfortably within the 95% confidence interval. The time until both parameters converged to within 10^{-3} of their final estimate was 231 iterations. The effect of the resampling step is also shown in figure 7 (right), as can be seen, the ESS never drops below 75% of the total number of particles. The average convergence time over 20 experiments was 195 iterations with a standard deviation of 60.2. The same experiment was repeated using 1000 particles, the results of which are shown in the appendix by figure 10. Under these conditions, the estimation trajectories are slightly smoother along with smaller confidence intervals. For 100 particles, the same experiment was repeated once more when $\alpha = 1$, this had an average convergence time of 189 and a standard deviation of 47.2. The distribution of the data and the evolution of the state, for one realisation, is shown by figure 11 and 12 included in the appendix. Like in the EKF case, this decrease in convergence time could be due to a reduced set of possible boundaries which could produce the observations.

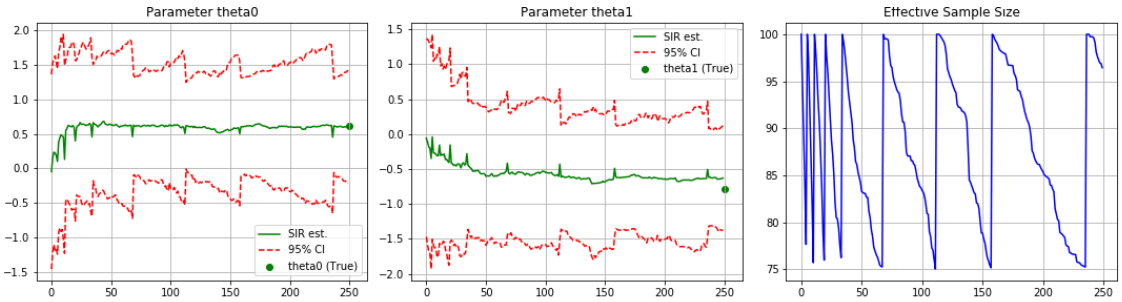

Figure 7: SIR estimation, 95% confidence intervals and effective sample size using 100 particles

3.3 Analysis

For the case when $\alpha = 2.5$, the average convergence time was 2.7 times slower for the PF compared to the EKF; however, the standard deviation of the convergence time for the EKF was 1.22 times larger. The ratio was amplified for the case when $\alpha = 1.0$, this time the PF converged 16.3 times slower than the EKF, but the

standard deviation of the EKF's convergence time was smaller by a factor of 7.26. This suggests the EKF is the optimal choice if it can capture the non-linearity of the system. In severely non-linear systems, it has been shown that the EKF will estimate poorly whereas the PF will still provide a good estimate [1].

4 Summary

This report considered two case studies. It showed that adding a resampling step to the SIS filter significantly reduced the uncertainty in the estimate and produced a much smoother estimate in the time varying AR problem. It also considered a logistic regression. Results from this experiment showed that for this particular problem, the EKF's estimate of the boundary parameters converged significantly quicker than the PF (100 particles) for both the linearly separable and non-separable case. It was also noted that this strongly depends on the level of non-linearity in the system. For highly non-linear systems, it has been shown the EKF produces a poor estimate of the state.

5 Appendix

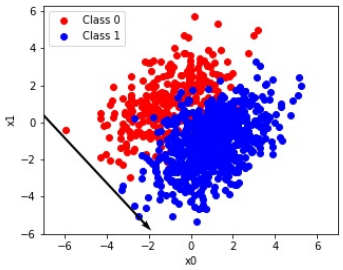

Figure 8: 2-d binary classification data with overlap

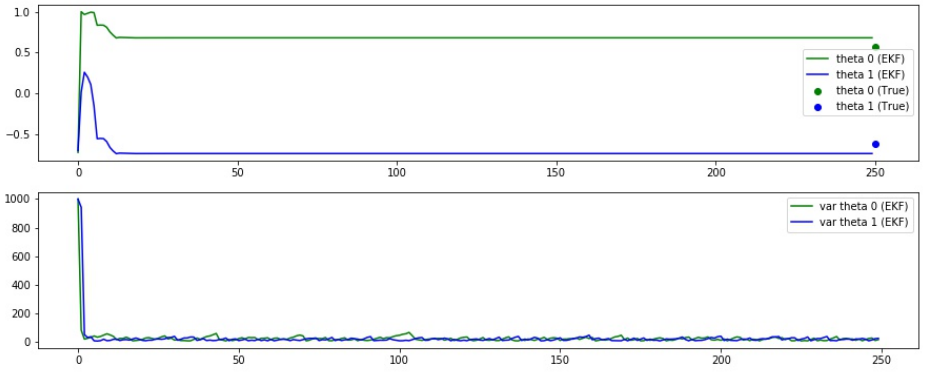

Figure 9: Evolution of boundary estimate and error covariance

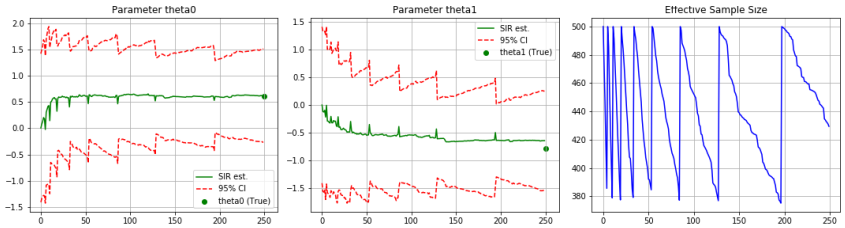

Figure 10: SIR estimation, 95% confidence intervals and effective sample size using 1000 particles

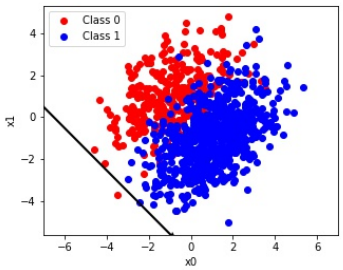

Figure 11: 2-d binary classification data with overlap

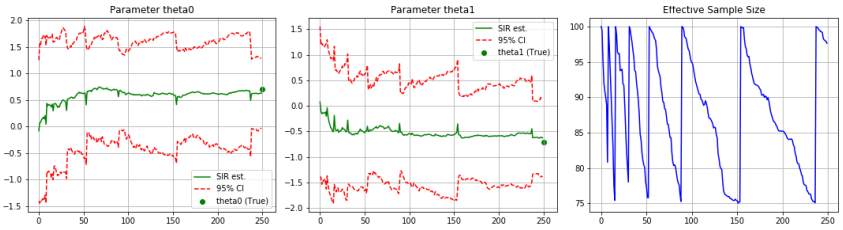

Figure 12: SIR estimation, 95% confidence intervals and effective sample size using 100 particles

References

[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 174–188, 2002.