Fast Particle Smoothing: If I Had a Million Particles

Mike Klaas KLAAS@CS.UBC.CA
Department of Computer Science, University of British Columbia, Canada

Mark Briers MB511@CAM.AC.UK
Department of Information Engineering, Cambridge University, UK

Nando de Freitas NANDO@CS.UBC.CA
Arnaud Doucet ARNAUD@CS.UBC.CA
Department of Computer Science, University of British Columbia, Canada

Simon Maskell S.MASKELL@SIGNAL.QINETIQ.COM
Advanced Signal and Information Processing Group, QinetiQ, UK

Dustin Lang DSTN@CS.TORONTO.EDU
Department of Computer Science, University of Toronto, Canada

Abstract

We propose efficient particle smoothing methods for generalized state-spaces models. Particle smoothing is an expensive $O(N^2)$ algorithm, where N is the number of particles. We overcome this problem by integrating dual tree recursions and fast multipole techniques with forward-backward smoothers, a new generalized two-filter smoother and a maximum *a posteriori* (MAP) smoother. Our experiments show that these improvements can substantially increase the practicality of particle smoothing.

1. Introduction

Belief inference, also known as smoothing in the signal processing and control literature, is at the heart of many learning paradigms. Given a sequence of observations $y_{1:T} = \{y_1, \ldots, y_T\}$, an initial distribution over the hidden states $p(x_1)$, a transition model $p(x_t|x_{t-1})$ and an observation model $p(y_t|x_t)$, Bayesian theory provides a sound framework for estimating the filtering distribution $p(x_t|y_{1:t})$, the smoothing distribution $p(x_t|y_{1:T})$ and the MAP sequence $x_{1:T}^{\mathrm{MAP}} \triangleq \arg\max_{x_{1:T}} p(x_{1:T}|y_{1:T})$. The Bayesian solution can be obtained in closed form when the probabilistic graphical model, defined by the transition and observation models, is either a Gaussian tree model or a discrete tree model. Otherwise, numerical approximation techniques must be employed. In this paper, we present algorithms for smoothing in tree-structured graphical models when the model distributions are non-Gaussian, nonlinear and non-stationary. Though many of the ideas can be extended to junction trees, we simplify the presentation by focusing on the widely-used chain-structured graphs.

Historically, particle filtering (PF) has proved to be an efficient method for approximating the filtering distribution; see for example (Doucet et al., 2001) or the many hundreds of scientific publications on the topic. In contrast, particle smoothing has hardly received any attention. There are two reasons for this:

1. Smoothing algorithms, such as the two-filter smoother (TFS), the forward-backward smoother (FBS), and the MAP smoother, incur an $O(N^2)$ cost, where N is a typically large number of particles. Note that in contrast PF is only $O(N)$.
2. The existing particle TFSs (Kitagawa, 1996; Isard & Blake, 1998) make assumptions that are hard to verify or satisfy in practice.

Practitioners often use filtering as a proxy to smoothing. For example, PF is often used when post-processing video sequences in sports (Vermaak et al., 2003; Okuma et al., 2004). This is terribly limiting given that the smoothed estimates, with access to the data in the entire video sequence, can be far more accurate. In addition, there has also been a historic focus on inference but hardly any work on learning (parameter estimation) with particle methods. This is not surprising. In order to obtain the MAP or maximum likelihood parameter estimates one needs the smoothed

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

estimates of the states; as is done often with the expectation maximization algorithm for linear-Gaussian models or discrete hidden Markov models. One could use Markov chain Monte Carlo (MCMC) techniques, but these can perform poorly as, by the model definition, the states are strongly correlated in time.

In order to make particle smoothing viable, this paper proposes the following:

- A unifying treatment and comparison of particle smoothing methods for general state spaces, including FBS, TFS and MAP smoothing.
- Fast implementation of these methods with dual metric-tree recursions (Gray & Moore, 2000; Klaas et al., 2005) and fast multipole algorithms (Greengard & Rokhlin, 1987). Anecdotally, we will show that for models where the likelihood is easy to evaluate, one can carry out smoothing on a chain of length 10 with 1,000,000 particles in about 1 minute!
- A generalized two filter particle smoother that circumvents previous unrealistic assumptions.

In our state space formulation, the models can be non-stationary, nonlinear and multi-modal, but we make the assumption that the transition prior $p(x_t|x_{t-1})$ is defined on a metric space. This assumption is not very restrictive as it applies in many domains of interest, including image tracking, beat tracking, robotics, econometric models and so on. If the interest is in fast methods for discrete state spaces, we refer the reader to (Felzenswalb et al., 2003). We should also mention that for maximum a posteriori (MAP) estimation with deterministic grids one should use the distance transform (Felzenswalb et al., 2003), however this algorithm is not applicable to the type of multivariate Monte Carlo grids that arise in particle smoothing.

2. Bayesian filtering

General Bayesian filtering is a two-step procedure. Given an estimate of $p(x_{t-1}|y_{1:t-1})$, we can obtain $p(x_t|y_{1:t})$ as follows:

- **Forward prediction:**

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}$$

- **Forward update:**

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t}.$$

The integrals in this procedure are often intractable, but can be approximated with standard PF methods (Doucet et al., 2001). In particular, through PF, one obtains the following Monte Carlo approximation of the filtering distribution:

$$\widehat{p}(dx_t|y_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{x_t^{(i)}}(dx_t).$$

where $w_t^{(i)}$ denotes the *normalized* importance weight of the i-th particle and $\delta_{x_t^{(i)}}(dx_t)$ is the Dirac distribution; see for example the introductory chapter in (Doucet et al., 2001) for a detailed derivation. This filtering step is carried out in all smoothers.

3. Bayesian Smoothing

In this section, we present the forward-backward smoother, the MAP smoother as well as a new generalized two-filter smoother.

3.1. Forward-Backward smoother

The smoothed density $p(x_t|y_{1:T})$ can be factored as follows:

$$\begin{aligned} p(x_t|y_{1:T}) &= \int p(x_t, x_{t+1}|y_{1:T})dx_{t+1} \\ &= \int p(x_{t+1}|y_{1:T})p(x_t|x_{t+1}, y_{1:t})dx_{t+1} \\ &= \underbrace{p(x_t|y_{1:t})}_{\text{filtered}} \int \underbrace{\frac{\overbrace{p(x_{t+1}|y_{1:T})}^{\text{smoothed}}\overbrace{p(x_{t+1}|x_t)}^{\text{dynamics}}}{\underbrace{\int p(x_{t+1}|x_t)p(x_t|y_{1:t})dx_t}_{\text{state prediction}}}}_{} dx_{t+1}. \end{aligned}$$

Hence, we obtain a recursive formula for the smoothed density $p(x_t|y_{1:T})$ in terms of the filtering density $p(x_t|y_{1:t})$, the state prediction density $p(x_{t+1}|y_{1:t})$, and the smoothed density at time $t+1$ $p(x_{t+1}|y_{1:T})$. The algorithm proceeds by making first a forward filtering pass to compute the filtered distribution at each time step, and then a backward smoothing pass to determine the smoothing distribution (see Figure 1).

We can approximate this distribution by defining:

$$\widehat{p}(dx_t|y_{1:T}) = \sum_{i=1}^{N} w_{t|T}^{(i)} \delta_{x_t^{(i)}}(dx_t),$$

where the importance weights $w_{t|T}^{(i)}$ are obtained through the following backward recursion:

$$w_{t|T}^{(i)} = w_t^{(i)} \left[\sum_{j=1}^{N} w_{t+1|T}^{(j)} \frac{p\left(x_{t+1}^{(j)} \middle| x_t^{(i)}\right)}{\sum_{k=1}^{N} w_t^{(k)} p\left(x_{t+1}^{(j)} \middle| x_t^{(k)}\right)} \right] \quad (1)$$

with $w_{T|T}^{(i)} = w_T^{(i)}$. Note that FBS maintains the original particle locations and reweights the particles to obtain an approximation to the smoothed density. Thus, its success depends on the filtered distribution having support where the smoothed density is significant.

Equation (1) costs $O(N^3)$ operations to evaluate directly, but can easily be performed in $O(N^2)$ by observing that the denominator of the fraction in equation (1) is independent of i, hence can be performed independently from i for each j. $O(N^2)$ is still too expensive, however, but we reduce this cost later.

0. *Filtering*. For $t = 1, \ldots, T$, perform particle filtering to obtain the weighted measure $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$.
1. *Initialization*. For $i = 1, \ldots, N$, set $w_{T|T}^{(i)} = w_T^{(i)}$.
2. *Backward recursion*. For $t = T-1, \ldots, 1$ **and** $i = 1, \ldots, N$

$$w_{t|T}^{(i)} = w_t^{(i)} \left[\sum_{j=1}^{N} w_{t+1|T}^{(j)} \frac{p\left(x_{t+1}^{(j)} \middle| x_t^{(i)}\right)}{\sum_{k=1}^{N} w_t^{(k)} p\left(x_{t+1}^{(j)} \middle| x_t^{(k)}\right)} \right]$$

Figure 1: The forward/backward particle smoother.

3.2. Two-Filter smoother

A smoothed marginal distribution can also be obtained by combining the result of two independent filter-like procedures; one running forward in time and another backward. We use the following factorization:

$$\begin{aligned} p(x_t|y_{1:T}) &= p(x_t|y_{1:t-1}, y_{t:T}) \\ &= \frac{p(x_t|y_{1:t-1}) p(y_{t:T}|y_{1:t-1}, x_t)}{p(y_{t:T}|y_{1:t-1})} \\ &\propto \underbrace{p(x_t|y_{1:t})}_{\text{filter one}} \underbrace{p(y_{t+1:T}|x_t)}_{\text{filter two}}. \end{aligned}$$

Filter one is our familiar Bayesian filter. *Filter two* can be computed sequentially using the *Backward Information Filter* (Mayne, 1966), noting that:

$$p(y_{t:T}|x_t) = \int p(y_{t+1:T}|x_{t+1}) p(x_{t+1}|x_t) p(y_t|x_t) dx_{t+1}.$$

The problem with this recursion is that $p(y_{t:T}|x_t)$ is not a probability density function in argument x_t and thus its integral over x_t might not be finite (the recursion blows up). Hence, existing Monte Carlo particle methods *cannot be used* to approximate $p(y_{t:T}|x_t)$, unless one makes unrealistic assumptions. References (Isard & Blake, 1998; Kitagawa, 1996) implicitly assume that $\int p(y_{t:T}|x_t) dx_t < \infty$ and develop sequential Monte Carlo methods in this context. However, if this assumption is violated, these methods do not apply.

We present a solution to this problem that relies on the introduction of an *artificial* distribution $\gamma_t(x_t)$. This distribution enables us to define a recursion in terms of finite quantities and thus allow Monte Carlo methods to be used. We will first present the new recursion assuming that $\gamma_t(x_t)$ is known. Subsequently, we will present several choices of $\gamma_t(x_t)$ and their impact upon the approximation at hand; see (Briers et al., 2004).

The key idea of the new algorithm is to express the backward filter in terms of two finite quantities:

$$p(y_{1:T}|x_t) \propto \frac{\widetilde{p}(x_t|y_{t:T})}{\gamma_t(x_t)}, \qquad \text{where}$$

$$\widetilde{p}(x_{t:T}|y_{t:T}) \propto \gamma_t(x_t) \prod_{i=t+1}^{T} p(x_i|x_{i-1}) \prod_{i=t}^{T} p(y_i|x_i).$$

The derivation of the above ratio is as follows:

$$\begin{aligned} p(y_{t:T}|x_t) &= \int \cdots \int p(y_{t:T}, x_{t+1:T}|x_t) dx_{t+1:T} \\ &= \int \cdots \int \frac{\gamma_t(x_t)}{\gamma_t(x_t)} \prod_{i=t+1}^{T} p(x_i|x_{i-1}) \prod_{i=t}^{T} p(y_i|x_i) dx_{t+1:T} \\ &\propto \int \cdots \int \frac{\widetilde{p}(x_{t:T}|y_{t:T})}{\gamma_t(x_t)} dx_{t+1:T} \quad = \quad \frac{\widetilde{p}(x_t|y_{t:T})}{\gamma_t(x_t)}. \end{aligned}$$

It is clear from the way in which $\gamma_t(x_t)$ is introduced in this derivation that it is an arbitrary distribution (as long as it does not cause obvious divisions by zero). To derive a recursive algorithm for $\widetilde{p}(x_t|y_{t:T})$, we use the following expansion:[1]

$$\begin{aligned} p(y_{t:T}|x_t) &= \int p(y_{t+1:T}|x_{t+1}) p(x_{t+1}|x_t) p(y_t|x_t) dx_{t+1} \\ &\propto \int \frac{\widetilde{p}(x_{t+1}|y_{t+1:T})}{\gamma_{t+1}(x_{t+1})} p(x_{t+1}|x_t) p(y_t|x_t) dx_{t+1} \\ &= \frac{p(y_t|x_t)}{\gamma_t(x_t)} \int \widetilde{p}(x_{t+1}|y_{t+1:T}) \frac{p(x_{t+1}|x_t)\gamma_t(x_t)}{\gamma_{t+1}(x_{t+1})} dx_{t+1}. \end{aligned}$$

This expansion implies the following two-step backward filtering procedure:

- **Backward prediction:**

$$\widetilde{p}(x_t|y_{t+1:T}) = \int \widetilde{p}(x_{t+1}|y_{t+1:T}) \frac{p(x_{t+1}|x_t)\gamma_t(x_t)}{\gamma_{t+1}(x_{t+1})} dx_{t+1}$$

- **Backward update:**

$$\widetilde{p}(x_t|y_{t:T}) = \frac{p(y_t|x_t)\widetilde{p}(x_t|y_{t+1:T})}{\int p(y_t|x_t')\widetilde{p}(x_t'|y_{t+1:T}) dx_t'}.$$

This backward recursion is initialized with

$$\widetilde{p}(x_T|y_T) = \frac{p(y_T|x_T)\gamma_T(x_T)}{p(y_T)}.$$

[1] Note: $\widetilde{p}(x_t|y_{t+1:T})$ may not be a proper probability distribution, but its finiteness is sufficient to guarantee convergence in a Monte Carlo setting.

Now that we have a recursion for $\widetilde{p}(x_{t:T}|y_{t:T})$, it only remains to specify $\{\gamma_t(x_t)\}$. It is clear from the derivation that any γ_t would work (as long as it is positive when $p(y_{1:T}|x_t)$ is positive). It is not possible to define optimal $\{\gamma_t(x_t)\}$ with respect to minimizing the variance of the resulting incremental importance weights, as is normally performed within standard particle filtering, since the choice of $\{\gamma_t(x_t)\}$ is dependent upon the specific choice of proposal distribution.

However some choices are intuitively better than others. If the prior is analytic, that is if one can compute:

$$p(x_t) = \int \cdots \int p(x_1) \prod_{t'=1}^{t-1} p(x_{t'+1}|x_{t'}) dx_{1:t-1}$$

then a sensible choice is $\gamma_t(x_t) = p(x_t)$, since:

$$\begin{aligned} p(x_1, \ldots, x_T) &= p(x_1) \prod_{k=2}^{T} p(x_k|x_{k-1}) \\ &= p(x_T) \prod_{k=1}^{T-1} b(x_k|x_{k+1}) \end{aligned}$$

where $b(x_k|x_{k+1}) = \frac{p(x_k)p(x_{k+1}|x_k)}{p(x_{k+1})}$.

References (Bresler, 1986; Fraser & Potter, 1969) consider this case. However, the algorithm presented here is more general and does not require that the prior be analytic. Therefore, any approximation of the prior will suffice. For example, in systems where one can produce realizations of the Markov chain prior $p(x_t|x_{t-1})$, one can obtain many samples from $p(x_t)$ and $\gamma_t(x_t)$ can be obtained by fitting these samples with an analytical distribution. Moreover, if the Markov chain is ergodic then we can run a single Markov chain to obtain (approximate) samples of the invariant distribution. Then we can use for all γ_t the analytical approximation of the invariant distribution.

For readers more familiar with the notion of message passing in discrete probabilistic graphical models, we note that the introduction of the artificial distribution allows one to avoid numerical underflow issues associated with calculating the messages in the belief propagation algorithm. This is exactly what Pearl is suggesting when he suggests normalizing messages to avoid numerical problems (Pearl, 1988). He is implicitly choosing $\gamma(x_t) = p(x_t)$ so that the message is a probability measure. Our approach is more general than this, and with thought, clever choices of γ_t can lead to messages with good numerical properties.

In terms of Monte Carlo implementation, let $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ be the weighted measure approximation produced by *filter one* (conventional forward filter). *Filter two* is given by the backward recursion of $\widetilde{p}(x_t|y_{t:T})$. The algorithm is shown in Figure 2. The particles are resampled if the discrepancy of the weights is too high. Finally, we combine the forward and backward approximations, $\widehat{p}(x_{t-1}|y_{1:t-1})$ and $\widehat{\widetilde{p}}(x_t|y_{t:T})$, to obtain through $p(x_t|y_{t:T}) \propto \frac{p(x_t|y_{1:t-1})\widetilde{p}(x_t|y_{t:T})}{\gamma_t(x_t)}$ the following approximation:

$$\widehat{p}(dx_t|y_{1:T}) \propto \sum_{j=1}^{N} \widetilde{w}_t^{(j)} \sum_{i=1}^{N} w_{t-1}^{(i)} \frac{p\left(\widetilde{x}_t^{(j)}|x_{t-1}^{(i)}\right)}{\gamma_t\left(\widetilde{x}_t^{(j)}\right)} \delta_{\widetilde{x}_t^{(j)}}(dx_t).$$

Note the same $O(N^2)$ computational bottleneck arising again in the above expression.

Forward filtering. For $t = 1, \ldots, T$, perform particle filtering to obtain the weighted measure $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$.

Backward filtering.

1. Initialization:
   (a) For $i = 1, \ldots, N$, sample a candidate from a proposal distribution $\widetilde{x}_T^{(i)} \sim q(\cdot|y_T)$.
   (b) Compute the importance weights $\widetilde{w}_T^{(i)} \propto \frac{p(y_T|\widetilde{x}_T^{(i)})\gamma_T(\widetilde{x}_T^{(i)})}{q(\widetilde{x}_T^{(i)}|y_T)}$.
2. For $t = T, \ldots, 2$ **and** $i = 1, \ldots, N$
   (a) Sample a candidate particle from a proposal distribution: $\widetilde{x}_{t-1}^{(i)} \sim q(\cdot|\widetilde{x}_t^{(i)}, y_{t-1})$.
   (b) Calculate the backward importance weights:
   $$\widetilde{w}_{t-1}^{(i)} \propto \frac{\widetilde{w}_t^{(i)} p(y_{t-1}|\widetilde{x}_{t-1}^{(i)})\gamma_{t-1}(\widetilde{x}_{t-1}^{(i)})p(\widetilde{x}_{t-1}^{(i)}|\widetilde{x}_t^{(i)})}{\gamma_t(\widetilde{x}_t^{(i)})q(\widetilde{x}_{t-1}^{(i)}|\widetilde{x}_t^{(i)}, y_t)}.$$
   (c) Resample if necessary.

Figure 2: The two-filter smoother. *Note*: The order of running the two filters may be interchanged.

3.2.1. An Important Warning

A common mistake in the smoothing literature is to use the the inverse of the dynamics to obtain the backward transition $p(x_t|x_{t+1})$. To see why this is erroneous, consider the following auto-regressive process:

$$x_{t+1} = ax_t + \sigma\nu_{t+1}, \quad x_1 \sim \mathcal{N}\left(0, \sigma^2/(1-a^2)\right)$$

which admits through the proper application of Bayes rule $p(x_t|x_{t+1}) = \frac{p(x_{t+1}|x_t)p(x_t)}{p(x_{t+1})}$ and marginalization the following backward kernel:

$$p(x_t|x_{t+1}) = \mathcal{N}\left(ax_{t+1}, \sigma^2\right).$$

On the other hand, inversion of the dynamics $x_t = a^{-1}(x_{t+1} - \sigma\nu_{t+1})$ incorrectly leads to:

$$p(x_t|x_{t+1}) = \mathcal{N}(a^{-1}x_{t+1}, a^{-2}\sigma^2).$$

3.3. Maximum *a posteriori* (MAP) smoother

In many applications, one is interested in the maximum *a posteriori* sequence

$$x_{1:T}^{\mathrm{MAP}} \triangleq \arg\max_{x_{1:T}} p(x_{1:T}|y_{1:T})\,. \tag{2}$$

For continuous state spaces, equation (2) almost never admits an analytic form. In (Godsill et al., 2001), a sequential Monte Carlo approximation to (2) is developed. First, standard particle filtering is performed. At time t, the set of particles $\left\{x_t^{(i)}\right\}_{i=1}^{N}$ can be viewed as a discretization of the state space. Since particles are more likely to survive in regions of high-probability, this discretization will be sensible. We can approximate (2) by finding the sequence of maximum probability on this grid, namely:

$$\widetilde{x}_{1:T}^{\mathrm{MAP}} \triangleq \arg\max_{x_{1:T} \in \bigotimes_{k=1}^{T} \left\{x_k^{(i)}\right\}_{i=1}^{N}} p(x_{1:T}|y_{1:T})\,. \tag{3}$$

This approximation requires consideration of a number of paths that grows exponentially with time. However, the maximization can be solved efficiently using dynamic programming.

Due to the Markov decomposition of the model, equation (2) is additive in log space, thus the Viterbi algorithm can be used to compute eq. (3) efficiently as shown in Figure 3.

0. *Filtering*. For $t = 1, \ldots, T$, perform particle filtering to obtain the weighted measure $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^{N}$.
1. *Initialization*. For $i = 1, \ldots, N$

$$\delta_1(i) = \log p(x_1^{(i)}) + \log p(y_1|x_1^{(i)})$$

2. *Recursion*. For $t = 2, \ldots, T$ **and** $j = 1, \ldots, N$

$$\delta_t(j) = \log p(y_t|x_t^{(j)}) + \max_i \left[\delta_{t-1}(i) + \log p(x_t^{(j)}|x_{t-1}^{(i)})\right]$$
$$\psi_t(j) = \arg\max_i \left[\delta_{t-1}(i) + \log p(x_t^{(j)}|x_{t-1}^{(i)})\right]$$

3. *Termination*.

$$i_T = \arg\max_i \ \delta_T(i)$$
$$\widetilde{x}_T^{\mathrm{MAP}} = x_T^{(i_T)}$$

4. *Backtracking*. For $t = T-1, \ldots, 1$

$$i_t = \psi_{t+1}(i_{t+1})$$
$$\widetilde{x}_t^{\mathrm{MAP}} = x_t^{(i_t)}$$

5. *Aggregation*. $\widetilde{x}_{1:T}^{\mathrm{MAP}} \triangleq \left\{\widetilde{x}_1^{\mathrm{MAP}}, \widetilde{x}_2^{\mathrm{MAP}}, \ldots, \widetilde{x}_T^{\mathrm{MAP}}\right\}$

Figure 3: The MAP smoothing algorithm produces an approximation to $x_{1:T}^{\mathrm{MAP}}$ by employing the Viterbi algorithm on the discretized state space induced by the particle filter approximation at time t: $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^{N}$. The computational expense of the algorithm is $O(N^2T)$ due to the recursion step (2).

4. Fast N-body Monte Carlo

The key N^2 computational bottleneck in FBS and TFS is the following evaluation:

$$f^{(j)} = \sum_{i=1}^{N} g^{(i)} p(x_{t+1}^{(j)}|x_t^{(i)})$$

given the appropriate substitutions for f and g in each case. In MAP smoothing, the bottleneck is given by:

$$f^{(j)} = \max_{i=1}^{N} g^{(i)} p(x_{t+1}^{(j)}|x_t^{(i)}).$$

We shall refer to these two bottlenecks as the sum-kernel and max-kernel problems, respectively.

Assuming that the transition model is a similarity kernel defined on a metric space $p(x_{t+1}|x_t) = K(d(x_{t+1}, x_t))$, where $d(\cdot)$ denotes distance, the sum-kernel problem can be approximated in $O(N \log N)$ steps using algorithms from N-body simulation. These algorithms guarantee an approximate solution within a pre-specified error tolerance, and so one can control the approximation error *a priori*. The most general and popular examples of these N-body algorithms for the sum-kernel problem include fast multipole expansions (Greengard & Rokhlin, 1987), box-sum approximations (Felzenswalb et al., 2003) and spatial-index methods (Moore, 2000).

Fast multipole methods tend to work only in low dimensions (up to 6D) and need to be re-engineered every time a new kernel (transition model) is adopted. The most popular multipole method is the fast Gauss transform (FGT) algorithm (Greengard & Sun, 1998), which as the name implies applies to Gaussian kernels. In the particular case of Gaussianity, it is possible to attack larger dimensions (say up to 10D) by adopting clustering-based partitions as in the improved fast Gauss transform (Yang et al., 2003). Both the computational and storage cost of fast multipole methods is claimed to be $O(N)$, though empirically the algorithms behave as $O(N \log N)$ with $O(N)$ storage.

Spatial-index methods, such as KD-trees and ball trees, are very general, easy to implement and can be applied in high-dimensional spaces (Gray & Moore, 2000; Gray & Moore, 2003). In particular, they apply to any monotonic kernels defined on a metric space (most continuous distributions and some discrete ones). Building the trees costs $O(N \log N)$ and in practice the run-time cost behaves as $O(N \log N)$.

Tree methods can also be applied to *exactly* solve the max-kernel problem (Klaas et al., 2005).

To provide some intuition on how these fast algorithms work, we will present a brief explanation of tree methods for the sum-kernel problem. The first step in these methods involves partitioning the particles recursively as shown in Figure 4.

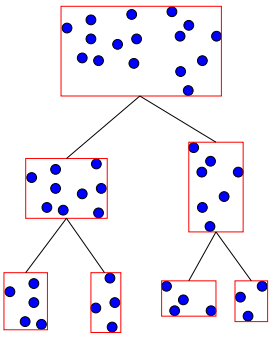

Figure 4: KD-tree partition of the state space.

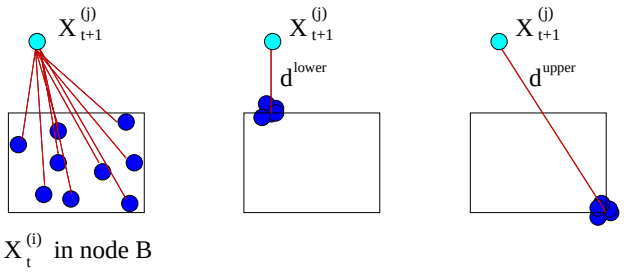

Figure 5: To bound the influence of the node particles $x_t^{(i)}$ on the query particle $x_{t+1}^{(j)}$, we move all the node particles to the closest and farthest positions in the node. To compute each bound, we only need to carry out a single kernel evaluation.

The nodes of the tree will maintain statistics, such as the sum of the weights in the node. We want to evaluate the effect of particles $x_t^{(i)}$ in a node B on the query particle $x_{t+1}^{(j)}$, that is:

$$f^{(j)} = \sum_{i \in B} g^{(i)} K\left(d(x_t^{(i)}, x_{t+1}^{(j)})\right).$$

where $d(\cdot,\cdot)$ is the metric component of the transition kernel $K(\cdot)$. As shown in Figure 5, this sum can be approximated using upper and lower bounds as follows:

$$\begin{aligned} f^{(j)} &\approx \frac{1}{2}\left(f^{(j)upper} + f^{(j)lower}\right) \\ &= \frac{1}{2}\sum_{i \in B} g^{(i)}\left[K(d^{lower}) + K(d^{upper})\right], \end{aligned}$$

where d^{lower} and d^{upper} are the closest and farthest distances from the query particle to node B. The error in this approximation is: $\frac{1}{2}\left(f^{(j)upper} - f^{(j)lower}\right)$. One only needs to recurse down the tree to the level at which the pre-specified error tolerance is guaranteed.

Since there are many query particles, it is possible to improve the efficiency of these tree methods by building trees for the source *and* query points. Then, instead of comparing nodes to individual query particles, one compares nodes to query nodes. Detailed explanations of these *dual-tree* techniques appear in (Gray & Moore, 2000; Gray & Moore, 2003; Klaas et al., 2005).

5. Experiments

5.1. Linear-Gaussian Model

In this simple experiment, we show that it is now possible to conduct particle smoothing for 10 observations in about one minute with as many as 1,000,000 particles. In particular, FBS was tested on a three-dimensional linear-Gaussian state space model populated with synthetic data. We keep the model sufficiently simple to compare the particle smoother to an analytic solution (in this case, a Kalman smoother). Since this is a sum-kernel problem, the acceleration method necessarily introduces error. Thus, we report both the cpu time for the naïve and fast method (in this case, the FGT) and the error of both algorithms after a given amount of cpu time. The results in Figure 6 verify that using more particles—smoothed approximately using the FGT—is better in terms of RMSE than using fewer particles smoothed exactly.

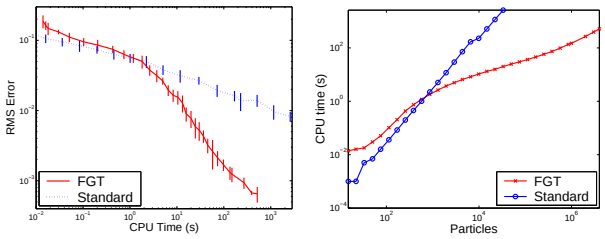

Figure 6: Particle-smoothing results on synthetic data, shown on a log-log scale for clarity. The data in both plots stem from the same experiment. For the same computation cost, FGT achieves an RMSE two orders of magnitude lower. We are able to smooth with as many as 4,000,000 particles with this method!

5.2. Non-Linear and Non-Gaussian Model

We consider the following synthetic model:

$$\begin{aligned} x_t &= \frac{x_{t-1}}{2} + \frac{10x_{t-1}}{1+x_{t-1}} + 8\cos(1.2t) + \mathcal{N}(0,\sigma_x) \\ y_t &= x_t^2/20 + \mathcal{N}(0,\sigma_y). \end{aligned}$$

The posterior is multi-modal, non-Gaussian, and non-stationary. It is as such an ideal setting to compare

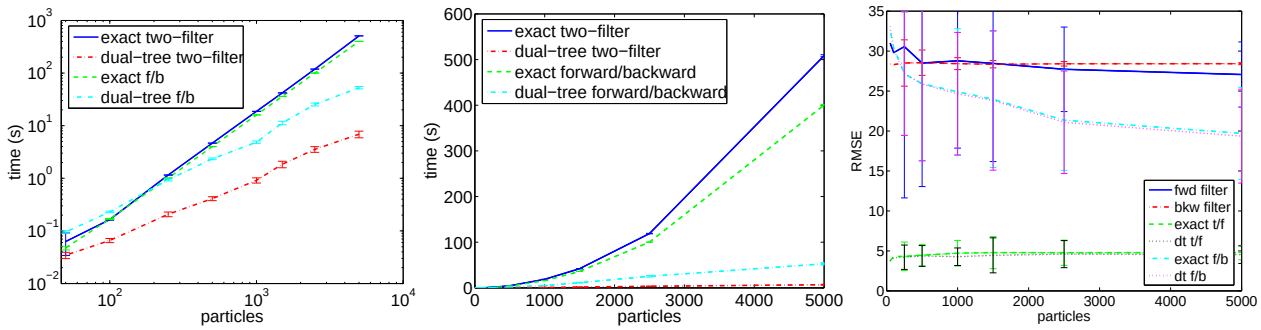

Figure 7: Non-linear, non-Gaussian model. Exact and dual-tree accelerated algorithms ($\epsilon = .005$). *Left/Middle*: Dual-tree algorithms provide up to two orders of magnitude speedup. The TFS is more amenable to acceleration than the FBS, up to a factor of ten. *Right*: FBS out-performs filtering, but TFS significantly out-performs both. The RMSE of the approximate algorithms is comparable to the exact versions.

the particle smoothers. There is no closed-form expression for the natural choice of $\gamma_t(x_t)$, so we instead simply use a flat Gaussian distribution. We tested forward filtering, backward (information) filtering, forward/backward smoothing, and two-filter smoothing over 15 Monte Carlo runs for each of 30 realizations of the model. Figure 7 contains the results. The TFS is significantly more accurate than filtering or FBS, and (surprisingly) faster as well. We conjecture that this is due to the resulting kernel matrix being sparser, hence providing more opportunities for node pruning.

5.3. Parameter Learning in SV Models

Stochastic volatility (SV) models are used extensively in the analysis of financial time series data (Kim et al., 1998). A common inference problem is to estimate the parameters in such complex models; coordinate ascent algorithms such as EM are used to perform such inference.

The SV model can be written as follows:

$$x_{t+1} = \theta_1 x_t + \theta_2 \nu_{t+1}, \; x_1 \sim \mathcal{N}\left(0, \theta_2^2/(1-\theta_1^2)\right)$$
$$y_t = \theta_3 \exp\left(x_t/2\right) \omega_t,$$

where ν_{t+1} and ω_t are two independent standard Gaussian noise processes. The initial state, x_1, which is independent of the states at all other times, is distributed according to the invariant distribution of the process evolution.

We use a standard EM algorithm: given a current estimate $\theta^{(i-1)}$ of θ then

$$\theta^{(i)} = \arg\max_{\theta \in \Theta} Q\left(\theta^{(i-1)}, \theta\right),$$

where

$$Q\left(\theta^{(i-1)}, \theta\right) = \mathbb{E}_{\theta^{(i-1)}}\left[\log\left(p_\theta\left(x_{1:T}, y_{1:T}\right)\right) \middle| y_{1:T}\right]$$
$$= \int \log\left(p_\theta\left(x_{1:T}, y_{1:T}\right)\right) p_{\theta^{(i-1)}}\left(x_{1:T} \middle| y_{1:T}\right) dx_{1:T}.$$

In our application, we use the natural invariant distribution as artificial prior γ_t. Hence, we cannot maximize Q analytically, but it is possible to maximize a modified Q function where the initial state is discarded, by setting[2]

$$\theta_1^{(i)} = \frac{\sum_{t=2}^{T} \mathbb{E}_{\theta^{(i-1)}}\left[x_{t-1} x_t \middle| y_{1:T}\right]}{\sum_{t=1}^{T-1} \mathbb{E}_{\theta^{(i-1)}}\left[x_t^2 \middle| y_{1:T}\right]},$$
$$\theta_2^{(i)} = \Bigg[(T-1)^{-1} \Bigg(\sum_{t=2}^{T} \mathbb{E}_{\theta^{(i-1)}}[x_t^2 | y_{1:T}] + \theta_1^{(i)2} \sum_{t=2}^{T} \mathbb{E}_{\theta^{(i-1)}}[x_{t-1}^2 | y_{1:T}] - 2\theta_1^{(i)} \sum_{t=2}^{T} \mathbb{E}_{\theta^{(i-1)}}[x_{t-1} x_t | y_{1:T}] \Bigg) \Bigg]^{1/2},$$
$$\theta_3^{(i)} = \left(T^{-1} \sum_{t=2}^{T} y_t^2 \mathbb{E}_{\theta^{(i-1)}}\left[\exp\left(-x_t\right) \middle| y_{1:T}\right] \right)^{1/2}. \quad (4)$$

The algorithm proceeds by iteratively computing the smoothed marginals $p(x_t|y_{1:T})$ (E step) and updating the parameters using eq. (4) (M step). Figure 8 contains results applying the TFS to the stochastic volatility model.

[2]Note that the expectations in eq. (4) involve smoothed distributions—this is an application where smoothing is not merely a higher-quality version of filtering but *integral* to the problem.

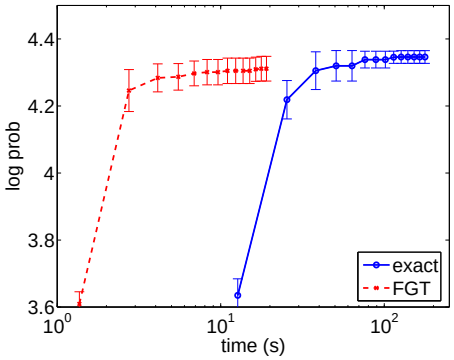

Figure 8: Parameter estimation example; log likelihood v. smoothing (E step) time. We use $N = 1000$ particles over 6 runs, and set the FGT tolerance to $\epsilon = 10^{-6}$. In the time the exact algorithm takes for one iteration, the FGT has already converged.

5.4. Beat Tracking

Beat-tracking is the process of determining the time slices in a raw song file that correspond to musical beats. This is a challenging multi-modal MAP problem (Cemgil & Kappen, 2003). We applied the model in (Lang & de Freitas, 2004) to "I Will Survive" by Cake, using the MAP particle smoothing algorithm described in section 3.3, and the exact dual-tree acceleration method. The dual-tree version is faster for $N > 100$ particles, and then dominates: it takes 2.53s to smooth $N = 50000$ particles, while naïve computation requires 1m46s (see Figure 9). This problem benefits from a high particle count: Using 1000 particles results in a MAP sequence of probability $p = 0.53$, while 50000 particles results in $p = 0.87$.

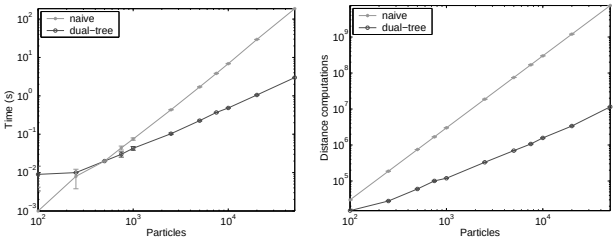

Figure 9: Beat-tracking results; log-log scale. The dual-tree method becomes more efficient at $t = 10$ms, and thereafter dominates the naïve method.

6. Conclusion

Our experiments demonstrate that practical particle smoothing can now become part of the standard statistical machine learning toolbox. To assist in this, the software has been made available on the author's website. One nice feature of all the smoothing algorithms is that if one already has a working PF, then adding the smoothing step is trivial. This should be noted by people working in image tracking, where the MAP smoother is likely to improve results.

References

Bresler, Y. (1986). Two-filter formula for discrete-time non-linear Bayesian smoothing. *IJC*, *43*, 629–641.

Briers, M., Doucet, A., & Maskell, S. R. (2004). *Smoothing algorithms for state-space models* (Technical Report CUED/F-INFENG/TR.498). Cambridge University Engineering Department.

Cemgil, A., & Kappen, H. (2003). Monte Carlo methods for tempo tracking and rhythm quantization. *JAIR*, *18*, 45–81.

Doucet, A., de Freitas, N., & Gordon, N. J. (Eds.). (2001). *Sequential Monte Carlo methods in practice*. Springer-Verlag.

Felzenswalb, P. F., Huttenlocher, D. P., & Kleinberg, J. M. (2003). Fast Algorithms for Large-State-Space HMMs with Application to Web Usage Analysis. *NIPS 16*.

Fraser, D. C., & Potter, J. E. (1969). The optimum linear smoother as a combination of two optimum linear filters. *IEEE Transactions on Automatic Control*, 387–390.

Godsill, S. J., Doucet, A., & West, M. (2001). Maximum a posteriori sequence estimation using Monte Carlo particle filters. *Ann. Inst. Stat. Math.*, *53*, 82–96.

Gray, A., & Moore, A. (2000). 'N-Body' Problems in Statistical Learning. *NIPS 4* (pp. 521–527).

Gray, A., & Moore, A. (2003). Rapid evaluation of multiple density models. *Artificial Intelligence and Statistics*.

Greengard, L., & Rokhlin, V. (1987). A fast algorithm for particle simulations. *JCP*, *73*, 325–348.

Greengard, L., & Sun, X. (1998). A new version of the Fast gauss transform. *Doc. Math., ICM*, 575–584.

Isard, M., & Blake, A. (1998). A smoothing filter for Condensation. *ECCV* (pp. 767–781). Freiburg, Germany.

Kim, S., Shephard, N., & Chib, S. (1998). Stochastic volatility: Likelihood inference and comparison with ARCH models. *Review of Economic Studies*, *65*, 361–93.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *JCGS*, *5*, 1–25.

Klaas, M., Lang, D., & de Freitas, N. (2005). Fast maximum a posteriori inference in Monte Carlo state spaces. *Artificial Intelligence and Statistics*.

Lang, D., & de Freitas, N. (2004). Beat tracking the graphical model way. *NIPS*. Cambridge, MA: MIT Press.

Mayne, D. Q. (1966). A solution of the smoothing problem for linear dynamic systems. *Automatica*, *4*, 73–92.

Moore, A. (2000). *The Anchors Hierarchy: Using the triangle inequality to survive high dimensional data* (Technical Report CMU-RI-TR-00-05). Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Okuma, K., Taleghani, A., de Freitas, N., Little, J., & Lowe, D. (2004). A boosted particle filter: Multitarget detection and tracking. *ECCV*. Prague.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan-Kaufmann.

Vermaak, J., Doucet, A., & Peŕez, P. (2003). Maintaining multi-modality through mixture tracking. *ICCV*.

Yang, C., Duraiswami, R., Gumerov, N. A., & Davis, L. S. (2003). Improved fast Gauss transform and efficient kernel density estimation. *ICCV*. Nice.