Local Importance Sampling: A Novel Technique to Enhance Particle Filtering

Péter Torma
Eötvös Loránd University, Budapest, Hungary
Email: tyus@axelero.hu

Csaba Szepesvári
Computer and Automation Research Institute
of the Hungarian Academy of Sciences, Budapest, Hungary
Email: szcsaba@sztaki.hu

***Abstract* – In the low observation noise limit particle filters become inefficient. In this paper a simple-to-implement particle filter is suggested as a solution to this well-known problem. The proposed Local Importance Sampling based particle filters draw the particles' positions in a two-step process that makes use of both the dynamics of the system and the most recent observation. Experiments with the standard bearings-only tracking problem indicate that the proposed new particle filter method is indeed very successful when observations are reliable. Experiments with a high-dimensional variant of this problem further show that the advantage of the new filter grows with the increasing dimensionality of the system.**

***Index Terms* – Sequential Monte Carlo Methods, Particle Filters, Hidden Markov Models**

I. INTRODUCTION

In this paper we consider the filtering of non-linear stochastic processes. The problem studied can be formalized as follows. We consider a sequence of values $Y_0, Y_1, Y_2, \ldots$ of some Euclidean space, governed by the equations

$$X_{t+1} = a(X_t; \xi_{t+1}), \quad X_0 \sim p_0(\cdot), \tag{1}$$
$$Y_{t+1} = b(X_{t+1}; \eta_{t+1}), \quad t = 0, 1, \ldots \tag{2}$$

Here X_t is the (hidden) state of the system at time step t and Y_t is the observed signal at the same time step, p_0 is the initial distribution of states at time step zero, and ξ_t and η_t are the process and observation noise processes, respectively. It is assumed that $\{\xi_t\}$ and $\{\eta_t\}$ are sequences of independent, identically distributed random variables and that they are also independent of each other. The filtering task is to determine the posterior, $\pi_t(x) = p(X_t = x|Y_1, \ldots, Y_t)$, over the states.

Three major factors can be identified that influence the performance of filtering algorithms: *(i)* the spread/peakiness of the process noise; *(ii)* the spread/peakiness of the observation noise; and *(iii)* the severity of 'perceptual aliasing' due to b that makes the recovery of the state from the sequence of observations hard even in the 'zero noise' limit.

Particle filters (see e.g. [7, 9, 5]) approximate the posterior by empirical measures of the form

$$\pi_t(x) = \frac{\sum_{k=1}^N w_t^{(k)} \delta(x - X_t^{(k)})}{\sum_{k=1}^N w_t^{(k)}},$$

where $X_t^{(k)}$ and $w_t^{(k)}$ represent the kth particle's position and weight, respectively, and $\delta(\cdot)$ is Dirac's delta function. Here $X_t^{(k)}$ and $w_t^{(k)}$ are (random) quantities that depend on the sequence of past observations $Y_{1:t} \stackrel{\text{def}}{=} (Y_1, \ldots, Y_t)$. Using the above empirical measure $\pi_t(x)$, the estimate of the expectation of an arbitrary function h with respect to the posterior can obtained by

$$I_{t,N}(h) = \frac{\sum_{k=1}^N w_t^{(k)} h(X_t^{(k)})}{\sum_{k=1}^N w_t^{(k)}}. \tag{3}$$

The goal is to construct π_t such that $I_{t,N}(h)$ is close to $\mathbb{E}[h(X_t)|Y_{1:t}]$, independently of h.

Based on "On Likelihood Adjusted Proposals in Particle Filtering: Local Importance Sampling", by P.Torma, and Cs.Szepesvri which appeared in the Proceedings of 4th International Symposium on Image and Signal Processing and Analysis 2005, Zagreb, Croatia, September 2005 [17].

The generic particle filter works by updating the particle's positions and weights in a recursive manner. The update is composed of two steps: computation of the particle's new positions is done by sampling from the so-called *proposal function*, followed by the update of the weights and the optional resampling step [7]. In the bootstrap filter [7] the particle's positions are updated independently of each other by sampling the kth particle's new position using $a(X_t^{(k)}; \xi_t^{(k)})$, where $\xi_t^{(k)}$ is sampled from the common underlying distribution of the process noise variables ξ_t, whilst the weight of the kth particle is computed by evaluating the observation likelihood $p(Y_t|X_{t+1}^{(k)})$.

When the level of the observation noise is low, the observation likelihood function becomes 'peaky'. In other words it becomes concentrated around its modes. In such a case if the position of a particle is not sufficiently close to one of these modes then the particle's weight will become small. This causes a serious increase of the variance of the estimate when most except a few particles have very small weights. This is easy to see if we notice that those particles whose weights are small do not effectively contribute to the estimate $I_{t,N}(h)$. Hence, the estimate's variance will be of the same order as if the number of particles would be changed to the number of particles that have significantly non-zero weights. Since the variance is inversely proportional to the number of particles, we get that if many particles have close to zero weights except a few ones then the variance of the estimate increases. The situation is not better either if all particles have small weights. In this case all particles are in the tails of the observation likelihood. Since the tail is typically very poorly modelled, the quality of the posterior's estimate will be poor, too. We call the problem that with increasingly more peaky observation functions the performance of particle filter can be expected to decrease the *"curse of reliable observations"*. Note that the curse of reliable observations is not a theoretical question: In many real world applications when the sensors are accurate (e.g. vision) the observations are very reliable.

In fact, the curse of reliable observations is a well-known peculiarity of particle filters. The general advise to remedy this problem is to use a proposal that depends on the most recent observation [5]. However, in some cases finding a proposal that is tractable, has no false peaks and still yields good performance can be notoriously hard. Another often used method is to replace the observation likelihood function by one which is less peaky. This method trades off approximation error (of the system's observation likelihood) and estimation error. Finding the right balance between these two sources of error can be quite cumbersome. A straightforward alternative to these methods is to increase the number of particles until a sufficiently high number of particles is placed close to the peaks of the observation likelihood function (in fact, this can be done adaptively). Unfortunately, for high dimensional systems, quite common in applications (e.g. [12, 8]), this approach may become infeasible because it may require an enormous number of particles: One expects that the number of particles will scale exponentially with the dimensionality of the system. Hence new methods that scale well to high-dimensional spaces even when the observations are reliable are of substantial interest.

The main idea underlying the algorithms of this paper relies on the following simple observation: Since the main source for the inefficiency of particle-filters is that the particles' positions are far from the modes of the observation likelihood, one should let the modes of the likelihood function 'attract' the particles. Hence, after the particles' positions are proposed using the prediction density as in the bootstrap filter, they are allowed to undergo a further randomized adjustment when the most recent observation is taken into account.

In this paper we consider two methods in details. The first method, the *local likelihood sampling* based particle filter (LLS filter) was first introduced by us in [16] and it is used to motivate the second one. In the LLS filter the second sampling step employs a localized version of the observation likelihood function. Weights are calculated so that the process remains unbiased. The second algorithm, that we call the *local importance sampling* based particle filter (LIS filter) is the main theme of this paper.[1] In this algorithm, the second step employs a generic proposal density function that, in general, should match closely the observation likelihood function. The weight update equations are modified so that unbiasedness is retained. A practical variant of the LIS filter that uses mixture of Gaussian proposal and hence is universal and is still easy to implement is studied in greater detail.

The paper is organized as follows: In Section II we introduce the necessary notation. This section is followed by a description of the algorithms (Section III), which is followed by a description of related research (Section IV). Experimental results on variants of the bearings-only tracking problem are presented in Section V. Here we compare the performance of the mixture of Gaussian's based LIS filter with that of the baseline bootstrap filter and the Auxiliary Variable Particle Filter (AVPF) [11]. The standard bearings-only problem is selected as it is a very good example of a problem when the observations are reliable. We also study the algorithm in a high-dimensional version of the basic problem. Finally, our conclusions drawn in Section VI.

II. NOTATION

The following notations will be used throughout the article: for an integrable function f, $I(f)$ will denote the integral of f with respect to the Lebesgue measure. Convolution is denoted by $*$:

$$(f * g)(x) = \int f(y)g(x-y)dy \ .$$

Capital letters such as X, Y will be used to denote random variables, whilst their corresponding lower counter-

[1] This algorithm has been introduced in the conference paper [17] serving as the basis of this communication.

parts (x, y, etc.) will be used to denote values that they can take on. Expectation and variance are denoted by $\mathbb{E}$ and $\mathbb{V}\mathrm{ar}$, respectively.

Let us denote the transition kernel corresponding to the dynamics a (cf. (1)) by $K = K(u|v)$, i.e., $\int_{\mathcal{U}} K(u|v)\,du = P(a(v,\xi_t) \in \mathcal{U})$, where $\mathcal{U}$ is any measurable subset of the state-space. Further, let us denote the observation likelihood density by $r = r(y|x)$, i.e., $\int_{\mathcal{Y}} r(y|x)\,dy = P(b(x,\eta_t) \in \mathcal{Y})$, where $\mathcal{Y}$ is any measurable subset of the observation space.

III. ALGORITHMS

A. LLS filters

The basic idea of LLS filters [16] is to draw a sample from the prediction density as in the bootstrap filter and then to allow the particles attracted by the modes of the observation density. A window-function (g) is used to localize the observation density's effect on the sample, hence the name of the procedure. The role of localization is to prevent particles 'jump around' in the state-space. This allows LLS filters to retain the information available in the previous time step's posterior in an effective way. The procedure is shown in Fig. 1.

Initialize $\{(Z_0^{(k)}, 1/N)\}_{k=1}^N$ from the prior $p_0(\cdot)$.
For time points $t = 1, 2, \ldots$
  For particles $k = 1, 2, \ldots, N$
    Resample $S_{t-1} = \{(Z_{t-1}^{(j)}, w_{t-1}^{(j)})\}_{j=1}^N$ to obtain a new sample $({Z_{t-1}^{(k)}}', 1/N)$
    Predict $X_t^{(k)}$ by drawing a sample from $K(\cdot|{Z_{t-1}^k}')$
    Perturb $X_t^{(k)}$ by drawing $Z_t^{(k)}$ from $\frac{1}{\alpha_t^{(k)}} r(Y_t|\cdot) g(X_t^{(k)} - \cdot)$, where

$$\alpha_t^{(k)} = \int r(Y_t|x) g(X_t^{(k)} - x)\,dx\ .$$

    Update weight $w_t^{(k)}$ using

$$\hat{w}_t^{(k)} = \alpha_t^{(k)} \frac{K(Z_t^{(k)}|{Z_{t-1}^{(k)}}')}{K(X_t^{(k)}|{Z_{t-1}^{(k)}}')}\ .$$

  EndFor
  Normalize weights using $w_t^{(k)} = \hat{w}_t^{(k)} / \sum_{j=1}^N \hat{w}_t^{(j)}$.
EndFor

Figure 1. LLS Filter

The following proposition, showing the unbiasedness of the two-stage sampling step of the LLS filter was shown to hold in [16]:

Proposition 1. *Let (X_t, Y_t) evolve according to Equations (1)-(2). Assume that g is a non-negative, integrable function satisfying $I(g) = 1$. Let $\{(\hat{w}_t^{(k)}, Z_t^{(k)})\}$ be the particle step (with un-normalized weights) obtained at time step t of the LIS filter. Then*

$$\mathbb{E}[\hat{w}_t^{(k)} h(Z_t^{(k)})|Y_{1:t}] = \mathbb{E}[h(X_t)|Y_{1:t}] p(Y_t|Y_{1:t-1}).$$

This result allows us to conclude that weighted averages of the form (3) can be used to approximate $\mathbb{E}[h(X_t)|Y_{1:t}]$. In particular, as a consequence convergence results (almost sure convergence, convergence in distribution, mean-square, finite sample performance bounds) can be derived along the lines of previous proofs (see [3] and references therein).

Proposition 1 leaves open the question whether the two-stage sampling step buys anything compared to the simpler procedure of the bootstrap filter. The following result proven in [16] can be used as a starting point for a positive answer in some cases:

Theorem 1. *Fix $X_{t-1}^{(k)}$. Assume that both the bootstrap filter and LLS draw $X_t^{(k)}$ from $p(\cdot) \stackrel{\mathrm{def}}{=} K(\cdot|X_{t-1}^{(k)})$. Let $s \in [1, \infty]$ and $f(\cdot) \stackrel{\mathrm{def}}{=} r(Y_t|\cdot)$. Then there exists a positive number ϵ that depends on s, g and p such that when*

$$\langle f,\, fp\rangle \geq \langle f * g,\, (fp) * g\rangle + \epsilon I(f) \|f\|_s. \qquad (4)$$

then LLS is more efficient as measured by the variance of its weights than the bootstrap filter.

The efficiency condition (4) depends crucially on the size of ϵ. Using the constructions of [16] it can be shown that ϵ measures how fast the tail of g decays as compared to the decay rate of the tail of p. For $s = 1$ and when p is unimodal and the tail of g decays faster than the tail of p, ϵ can be shown to be 'small'. In such a case, condition (4) becomes $\langle f,\, p\rangle^2 \gtrsim \langle f * g,\, (fp) * g\rangle$. Since for typical choices of g, convolving functions with g cuts high frequencies, the efficiency condition (4) can be expected to hold for problems when the prediction density, p, is 'broad' as compared with the observation density, f, i.e., when observation noise is low compared with the noise of the dynamics.

Note that the algorithm can be generalized easily to use window functions that change depending on $X_t^{(k)}$ by replacing $g(X_t^{(k)} - x)$ in the sample-perturbation step by $g(X_t^{(k)} - x; X_t^{(k)})$ and redefine $\alpha_t^{(k)}$ by $\alpha_t^{(k)} = \int p(Y_t|x) g(X_t^{(k)} - x; X_t^{(k)})\,dx$. The simplest application of this is to fit g to match the energy distribution of the observation noise conditioned on $X_t^{(k)}$. In some applications it may worth to make g dependent on the observation Y_t.

B. Local Importance Sampling

One problem with LLS filters is that unlike bootstrap filters, they rely on whether sampling from $r(Y_t|\cdot) g(X_t^{(k)} - \cdot)/\alpha_t^{(k)}$ is possible (in an efficient way). One possible remedy to this problem is to sample from a proposal $q_{X_t,Y_t}(\cdot)$

instead of sampling from $r(Y_t|\cdot)$. The resulting algorithm that we call Local Importance Sampling is given in Fig. 2.

Initialize a sample set $\{(Z_0^{(k)}, 1/N)\}_{k=1}^N$ according to the prior $p_0(\cdot)$.
For $t = 1, 2, \ldots$
  For $k = 1, 2, \ldots, N$
    Resample $S_{t-1} = \{(Z_{t-1}^{(j)}, w_{t-1}^{(j)})\}_{j=1}^N$
      to obtain a new sample $(Z_{t-1}^{(k)\prime}, 1/N)$
    Predict $X_t^{(k)}$ by drawing a sample from $K(\cdot|Z_{t-1}^{k\prime})$
    Perturb $X_t^{(k)}$ by drawing $Z_t^{(k)}$ from the proposal q modulated by the window function g, i.e:
      $Z_t^{(k)} \sim \frac{1}{\alpha_t^{(k)}} q_{X_t^{(k)},Y_t}(\cdot) g(X_t^{(k)} - \cdot)$, where

$$\alpha_t^{(k)} = \int q_{X_t^{(k)},Y_t}(x) g(X_t^{(k)} - x)\, dx\ .$$

    Update weight $w_t^{(k)}$ using

$$\hat{w}_t^{(k)} = \alpha_t^{(k)} \frac{r(Y_t|Z_t^{(k)})}{q_{X_t^{(k)},Y_t}(Z_t^{(k)})} \frac{K(Z_t^{(k)}|Z_{t-1}^{(k)\prime})}{K(X_t^{(k)}|Z_{t-1}^{(k)\prime})}\ .$$

  EndFor
  Normalize weights using $w_t^{(k)} = \hat{w}_t^{(k)} / \sum_{j=1}^N \hat{w}_t^{(j)}$
EndFor

Figure 2. LIS Filter

The following proposition holds:

Proposition 2. *Let (X_t, Y_t) evolve according to Equations (1)-(2) and consider the LIS filter. Assume that g is a non-negative, integrable function satisfying $I(g) = 1$ and let $q_{x,y}(\cdot) > 0$ be a bounded, integrable function for all x, y. Let $\{(\hat{w}_t^{(k)}, Z_t^{(k)})\}_{k=1}^N$ be the particle set obtained at time step t. Then*

$$\mathbb{E}[\hat{w}_t^{(k)} h(Z_t^{(k)})|Y_{1:t}] = \mathbb{E}[h(X_t)|Y_{1:t}] p(Y_t|Y_{1:t-1}).$$

Proof. First note that $\alpha_t^{(k)} = (q_{X_t^{(k)},Y_t} * g)(X_t^{(k)})$. Let h be an arbitrary integrable function and let $I = \mathbb{E}[\hat{w}_t^{(k)} h(Z_t^{(k)})|Y_{1:t}]$. By the law of total probability, $I = \mathbb{E}[\mathbb{E}[\hat{w}_t^{(k)} h(Z_t^{(k)}) \,|\, X_t^{(k)}, Y_{1:t}]]$. By the definition of Z_t and $\hat{w}_t$,

$$\begin{aligned} & \mathbb{E}[\hat{w}_t^{(k)} h(Z_t^{(k)}) \,|\, X_t^{(k)}, Y_{1:t}] \\ = & \int h(z) (q_{X_t^{(k)},Y_t} * g)(X_t^{(k)}) \frac{r(Y_t|z)}{q_{X_t^{(k)},Y_t}(z)} \\ & \frac{K(z|Z_{t-1}^{(k)\prime})}{K(X_t^{(k)}|Z_{t-1}^{(k)\prime})} \frac{q_{X_t^{(k)},Y_t}(z) g(X_t^{(k)} - z)}{(q_{X_t^{(k)},Y_t} * g)(X_t^{(k)})}\, dz \\ = & \int h(z) \frac{K(z|Z_{t-1}^{(k)\prime})}{K(X_t^{(k)}|Z_{t-1}^{(k)\prime})} r(Y_t|z)\, g(X_t^{(k)} - z)\, dz\ , \end{aligned}$$

and hence by Fubini's theorem

$$\begin{aligned} I &= \int\int h(z) \frac{K(z|Z_{t-1}^{(k)\prime})}{K(x|Z_{t-1}^{(k)\prime})} r(Y_t|z)\, g(x - z)\, dz \\ & \qquad K(x|Z_{t-1}^{(k)\prime})\, dx \\ &= \int h(z) K(z|Z_{t-1}^{(k)\prime}) r(Y_t|z)\, dz\ , \end{aligned}$$

which equals the posterior multiplied by $p(Y_t|Y_{1:t-1})$, finishing the proof. □

Similarly to Proposition 1, this statement shows that the two-step sampling step of LIS filter is unbiased. Hence, we can expect that LIS filters will enjoy similar theoretical properties as the bootstrap filter, or the more general SIR filters [7]. Building on our previous argument, it is not hard to show that LIS is more efficient than the bootstrap filter under conditions when LLS is more efficient than the bootstrap filter and when the proposal function $q_{x,y}$ fits $r(y|\cdot)$ around x for any x, y. Instead of developing such a theoretical result, the efficiency of the new algorithm will be demonstrated in the experimental section on a number of problems.

C. Using Gauss-Mixture Proposals in LIS filters

If in LIS the window function is chosen to be a Gaussian and the proposal function is chosen to be a mixture of Gaussians at the same time, we get a particularly attractive algorithm. First, since continuous densities can be well-approximated to any error by mixtures of Gaussians [14, 13] the resulting algorithm retains it generality. Further, as we show it now, the algorithm can also be implemented efficiently.

Let $u = (x, y)$ and choose $q_{x,y} = q_u$ to be a mixture of Gaussians with n components, having priors $p_{u,1}, \ldots, p_{u,n}$, means $\mu_{u,1}, \ldots, \mu_{u,n}$ and covariance matrices $\Sigma_{u,1}, \ldots, \Sigma_{u,n}$. Then

$$q_u(z) = \sum_{i=1}^n p_{u,i} \frac{e^{-1/2(z-\mu_{u,i})^T \Sigma_{u,i}^{-1} (z-\mu_{u,i})}}{((2\pi)^N |\Sigma_{u,i}|)^{1/2}}. \qquad (5)$$

Let the window function be a zero-mean Gaussian with variance Σ_g:

$$g(z) = ((2\pi)^N |\Sigma_g|)^{-1/2} e^{-1/2 z^T \Sigma_g^{-1} z}. \qquad (6)$$

In order to implement the LIS filter one needs to be able to draw samples from $q_u(\cdot) g(x - \cdot)$ and to evaluate $(q_u * g)(x)$. For the above specific choices, it turns out that $q_u(\cdot) g(x - \cdot)$ is a mixture of Gaussians, too with covariances and means defined by the following equations:

$$C_{u,i} = (\Sigma_{u,i}^{-1} + \Sigma_g^{-1})^{-1}, \qquad (7)$$

$$\nu_{u,i} = C_{u,i} \Sigma_g^{-1} x + C_{u,i} \Sigma_{u,i}^{-1} \mu_{u,i}, \qquad (8)$$

and unnormalized weights:

$$L_{u,i} = p_{u,i} \frac{e^{-1/2(\mu_{u,i}-x)^T \Sigma_{u,i}^{-1} C_{u,i} \Sigma_g^{-1} (\mu_{u,i}-x)}}{((2\pi)^N |\Sigma_{u,i}||\Sigma_g|/|C_{u,i}|)^{1/2}}. \quad (9)$$

Let $L_u = \sum_{i=1}^n L_{u,i}$. Notice that $(q_u * g)(x) = L_u$. Further, sampling from $q_u(\cdot)g(x-\cdot)$ can be implemented by first drawing an index from the normalized weights $(L_{u,1}/L_u, \ldots, L_{u,n}/L_u)$ and then drawing a sample from the appropriate Gaussian. The corresponding calculations are further elaborated on in Fig. 3 for a single particle and a single time-step (time indeces and conditioning on previous samples/observations are dropped).

Inputs: prediction density ($K(.|X)$), observation density ($r(.|Y)$), observation (Y), Gauss-mixture proposal q_u

- Draw the jth particle X_j from $K(.|X)$. Let $u = (X_j, Y)$.
- Calculate the Gauss-mixture parameters ($\nu_{u,i}$, $C_{u,i}$, $L_{u,i}$) of $q_u(\cdot)g(X_j = \cdot)$ using Equations (7)–(9).
- Draw an index k from $\left\{ \frac{L_{u,i}}{L_u} \right\}_{i=1}^n$
- Draw Z_j from a Gaussian with mean $\nu_{u,k}$ and covariance $C_{u,k}$.
- Calculate the weight

$$w_j = \left(\sum_{i=1}^n L_{u,i} \right) \frac{f(Z_j)}{q_u(Z_j)} \frac{K(Z_j|X)}{K(X_j|X)}.$$

Figure 3. Local Importance Sampling with mixture of Gaussians proposals and Gaussian window function ($q_{x,y}(z)$ is defined by (5), and $g(z)$ is defined by (6))

IV. Related Work

The Local Likelihood Sampling (LLS) algorithm, that was briefly overviewed in Section III.A, was introduced in [16]. Here we generalized the idea underlying LLS to obtain a new algorithm, the Local Importance Sampling (LIS) based particle filter. LIS filters are more flexible than LLS filters, whilst we believe that they retain the advantages of LLS filters.

One of the best known particle filter whose aim is to overcome the curse of reliability is the *Auxiliary Variable Particle Filter (AVPF)* introduced by Pitt and Shephard [11]. In addition to the bootstrap filter, AVPF is the baseline particle filter used in our experiments. AVPF is an instance of the generic SIR filter [7]. It uses a proposal density of the form $q(x_t|X_{t-1}^{(1)}, \ldots, X_{t-1}^{(N)}, Y_t) \propto \sum_{k=1}^N r(Y_t|\overline{X}_t^{(k)}) K(x_t|X_{t-1}^{(k)})$, where $\overline{X}_t^{(k)}$ is the expected next state for particle k: $\overline{X}_t^{(k)} = \mathbb{E}[a(X_{t-1}^{(k)}, \xi_t)|X_{t-1}^{(k)}]$.[2] Sampling is implemented by first doing a weighted resampling step using the weights $r(Y_t|\overline{X}_t^{(k)})$ and then drawing the next states using the transition density kernel K. Another trick is to compute the unnormalized weight of particle k by $r(Y_t|X_t^{(k)})/r(Y_t|\overline{X}_t^{(p_k)})$, where p_k is the index of the particle that was used to draw $X_t^{(k)}$ (see Section 13.3.2, [6]).

AVPF is similar to LLS/LIS filters in that it also allows the observation to influence the sampled positions of the particles. This makes AVPF somewhat more efficient than the bootstrap filter when the observation density is peaky. However, as this influence is still moderate (since the particles' positions are still sampled from the prediction density) highly peaked observation functions or when the prediction density is broad, AVPF's performance will still degrade unless the number of particles is kept high. Another issue arises when the prediction density is multi-modal. In this case using the expected next states does not make sense as the "predictor variables" $\overline{X}_t^{(k)}$. In such a case one must typically resort to sampling from the kernel K, further increasing the variance. The advantage of AVPF to LIS filters is that in LIS filters the user has to design the proposals, though we note that this choice can be made automatic by employing standard density estimation methods.

Another recent method aimed to increase efficiency in the face of reliable observations is *likelihood sampling* considered e.g. in details in [4]. In this approach it is the likelihood function $p(Y_t|\cdot)$ that is used as the proposal, whilst the prediction density is used to calculate the weights. The success of this method thus will depend on whether the likelihood is a good predictor of the true state. For multi-modal likelihoods (when aliasing effects are severe) a large number of particles can be generated away from the likely 'next' positions. Such particles will get low weights in the weighting process and thus will have no significant effect on the estimated posterior thereby increasing the variance of the estimator. Although LLS/LIS also make use of the likelihood density (or an approximation to it), they first sample from the prediction density, followed by sampling from a *localized* forms of the observation likelihood function. Hence LLS and LIS will not be adversely affected by multi-modal observation likelihoods.

The LS-N-IPS algorithm, introduced in [15], can be thought of as the precursor of LLS/LIS. The difference of LLS/LIS and LS-N-IPS is that LS-N-IPS modifies the position of particles in the second stage by letting them climb the observation likelihood in a deterministic way. Hence, this algorithm introduces some additional bias and relies on the availability of a method to climb the observation likelihood. Although, according to the well-known bias-variance dilemma, introducing bias is not necessarily 'bad', LLS/LIS filters can achieve the same variance reduction without introducing any additional bias.[3] Another advan-

[2] The generic AVPF leaves open the choice of $\overline{X}_t^{(k)}$. In practice, the choice shown here is one of the most frequently used ones.

[3] Note that weighted importance sampling itself yields biased estimates of the posterior.

tage of, LLS/LIS filters is that they do not rely on the availability of a hill-climbing algorithm.

Boosted particle filters (BOOPF) [10] are probably the closest to LLS/LIS filters in their spirit. BOOPF is another instance of SIR. Its proposal, using our notation can be written in the form:

$$q(\cdot|X_{t-1},Y_t) = \alpha(\overline{X}_t,Y_t)\, r(Y_t|\cdot)g(\overline{X}_t-\cdot) + \big(1-\alpha(\overline{X}_t,Y_t)\big)\; K(\cdot|X_{t-1}),$$

where $\overline{X}_t$ is the expected next state given X_{t-1}, g is a rectangular window function and

$$\alpha(\overline{x},y) = \begin{cases} A, & \text{if } r_0 < \max_{x':d(x',\overline{x})\le\lambda} r(y|x'); \\ B, & \text{otherwise.} \end{cases}$$

Here $0 < B \ll A < 1$, and r_0, λ are parameters to be chosen by the user. In effect, BOOPF will sample the next state from the localized version of the observation likelihood when the observation likelihood is sufficiently large in a neighborhood of the expected next state, whilst in the other case the prediction density is used to sample the next state.[4] In any case, the above proposal depends on the most recent observation and drawing samples from it can be done in an efficient manner. Unlike LLS/LIS, BOOPF's performance degrades when the prediction density is multi-modal or when it has a large variance as in such cases the expected value of the next state is a bad predictor of where the next state might be.

V. EXPERIMENTS

The purpose of this section is to present the results of a series of experiments the LIS filter is compared with the baseline bootstrap filter and AVPF, the purpose being to systematically compare LIS with these other algorithms in a controlled environment. We have chosen the bearings-only tracking problem, described in the section, as the tracking problem to be used. Actually, we used two versions of this problem: The standard single object version where the problem is to track a single ship by using angular measurements only and a version with more than one ship, where the ships move independently of each other and the observations carry information about the identity of the ships that they originate from. The purpose of considering this second problem was to study the scaling properties of the algorithms studied as a function of the dimensionality of the state space.

A. The Bearings-only Problem

The 'bearings-only tracking' problem has been considered previously by several authors [7, 11, 2, 1]. The aim is to track the motion of a ship, while observing only angles to it. The problem is illustrated on Fig. 4. Without the loss of generality the observer's position is fixed to the origin. The straight lines connecting the ship's position and the observer in subsequent time steps define the angles (with respect to the horizontal line). These angles corrupted by noise form the series of observations.

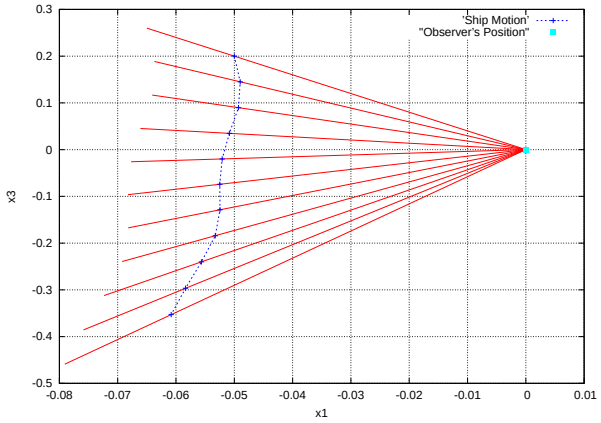

Figure 4. An example of the ship's motion. Shown are the lines connecting the observer and the ship's positions. The ship is moving in a top-down direction.

Formally, the ship's dynamics is second order, with the horizontal and vertical components developing independently of each other:

$$X_{t+1} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} X_t + \sigma_\eta \begin{pmatrix} \frac{1}{2} & 0 \\ 1 & 0 \\ 0 & \frac{1}{2} \\ 0 & 1 \end{pmatrix} \xi_t.$$

Here $X_t \in \mathbb{R}^4$, $\xi_t \in \mathbb{R}^2$, $\xi_{t1}, \xi_{t2} \sim \mathcal{N}(0,1)$. X_{t1} and X_{t3} represent the ship's horizontal and vertical positions, respectively, whilst X_{t2} and X_{t4} represent the ship's respective horizontal and vertical velocities. The initial state is sampled from a 4-dimensional Gaussian with a diagonal covariance matrix whose parameters will be given later.

The angle to the ship at time t is $\theta_t = \tan^{-1}(X_{t3}/X_{t1})$. The noise corrupting this angle has a wrapped Cauchy density:

$$r(y|\theta_t) = \frac{1}{2\pi}\frac{1-\rho^2}{1+\rho^2-2\rho\cos(y-\theta_t)}.$$

Provided that ρ is close one, the wrapped Cauchy density with its high peak around zero and heavy tails is thought to reflect a sonar's behaviour well: angular measurements are typically very reliable with occasional outliers.

The parameters of the system used in the experiments were as follows: When a single ship is tracked, the initial state is sampled from a Gaussian with mean $(-0.05, 0.001, 0.2, -0.055)$ and with a diagonal covariance matrix with diagonal entries $0.001 \times (0.5^2, 0.005^2, 0.3^2, 0.01^2)$. The magnitude of the process noise is given by $\sigma_\eta = 0.001$, whilst the parameter of the wrapped Cauchy distribution is $\rho = 1 - 0.005^2$.

[4] The algorithm as presented in [10] uses heuristically derived approximations to the observation likelihood and it is slightly more complicated than the one presented here, but their essence are the same.

In the experiments, where three ships are to be tracked, the initial state is sampled from three Gaussians. For all of these Gaussians we use the covariance matrix used in the single-ship experiments. The means of the Gaussian belonging to first ship are identical to the means used in the single-ship experiments, whilst the means of the of the other Gaussian were $(0.02, -0.01, 0.6, -0.055)$ and $(0.05, -0.01, -0.2, -0.02)$.

In the multi-ship experiments the three ships move independently of each other. Further, unlike in multi-target tracking, we assume that the observations are not unordered, i.e., we do not consider here the ambiguity of the assignment of observations to the objects. Formally, if y_1, y_2, y_3 are the observed angles, θ_1,θ_2 and θ_3 are the angles corresponding to the positions of ships one, two and three then

$$r_3(y_1, y_2, y_3|\theta_1, \theta_2, \theta_3) = r(y_1|\theta_1)r(y_2|\theta_2)r(y_3|\theta_3).$$

It should be clear that this model is limited in the sense that in many cases one would never know the correspondance between the observations and objects. In fact, the major source of difficulty for real-world multi-target tracking lies in resolving this ambiguity. However, our focus here are the scaling properties of the algorithms as a function of the dimensionality of the system and for this purpose the above problem is a just good enough.

B. The Choice of the Proposal for LIS

We first discuss the choice of the proposal $q_{x,y}$ for the single ship tracking task. The idea underlying the design is that of observation is simple: the predicted particle positions should be adjusted to fit closely the observed angles as the angular measurements are reliable. The actual choice of the importance function $q_{x,y}$ is a Gaussian with one of its axes parallel to the observation angle. The mean of $q_{x,y}$ was set to the particle's position projected to the observation angle line. To be more exact $q_{x,y}$ is a Gaussian with mean $(x_1 \cos^2(y) + x_3 \sin^2(y))$, where y is the observed angle, whilst the covariance is set to $U\Lambda U^T$ with

$$U = \begin{pmatrix} \cos(y) & -\sin(y) \\ \sin(y) & \cos(y) \end{pmatrix}$$

and

$$\Lambda = \begin{pmatrix} \kappa\sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}.$$

Here $\kappa > 0$ is a design parameter defining the ratio of the variance along the axis parallel to the observed angle to the variance along the orthogonal axis. Its value is chosen arbitrarily to be 100.[5] A reasonable value for σ^2, the variance along the orthogonal axis, is the value that makes the Gaussian fit the Wrapped Cauchy observation density the best. We used this value in our experiments.

As U is independent of the particles' positions, U can be precomputed at the beginning of the sampling steps, leaving only the calculation of the Gaussians' means in the body of the loop. The window function g is defined as a zero-mean Gaussian with covariance matrix

$$C = \begin{pmatrix} \delta_g & 0 \\ 0 & \delta_g \end{pmatrix}.$$

It is easy to see that these choices enable us to use the Gaussian-mixture LIS (cf. Fig. 2).

We must remark that the proposal function and the window function as defined above depend only on the position of the ship. Hence, when a particle's state is adjusted in the second sampling step, its velocity component must be appropriately adjusted (in a deterministic manner). Although, strictly speaking, this procedure does not fit the description of LIS, it can be shown by some limiting arguments that the it still gives unbiased samples.

C. Results for Single Ship Tracking

Particle filters have many qualities that must be taken into account for their meaningful comparison. These include (i) tracking performance, (ii) computation cost, (iii) memory needs, and (iv) ease of implementation. As far as implementation easiness is concerned, the bootstrap filter is a clear winner. The implementation easiness of LIS and AVPF are not that easy to compare, as both have a number of design parameters whose tuning typically requires experience and insight. Our experience is that LIS and AVPF are roughly equal as far as their implementation easiness is concerned.

In the rest of this section we compare these filters in a qualitative manner. The tracking error defined as the Euclidean distance between the mean predicted and the actual ship positions was used as the basis of the measurements. Measurements were made along trajectories of length 10, following the literature. The tracking errors were measured with 10 independent state-observation sequences, whilst keeping the initial position fixed. Unless otherwise noted, each result is the average of 100 runs for these 10 sequences.

The simplest way to compare particle filters is by their tracking error whilst the number of particles is kept the same. Fig. 5 shows such results as a function of time steps. Here the number of particles is kept at 100 for all algorithms. As expected, AVPF performs better than the baseline bootstrap filter. LIS improves upon the performance of both the bootstrap filter and AVPF quite significantly. In order to develop an understanding of what these performance differences mean, we present 'particle clouds' generated by the three algorithms for an arbitrary selected time-step in Fig. 6. It should be clear from the figure that the particle set generated by LIS is much better concentrated along the lines pointing towards the true state than the sets generated by both AVPF and the bootstrap filter. Literally, LIS makes a better use of the available information. Moreover, the particle sets generated by AVPF are more concentrated around the true state than those generated by the bootstrap filter.

[5] Lately, we have found that smaller values (up to a point) give better results.

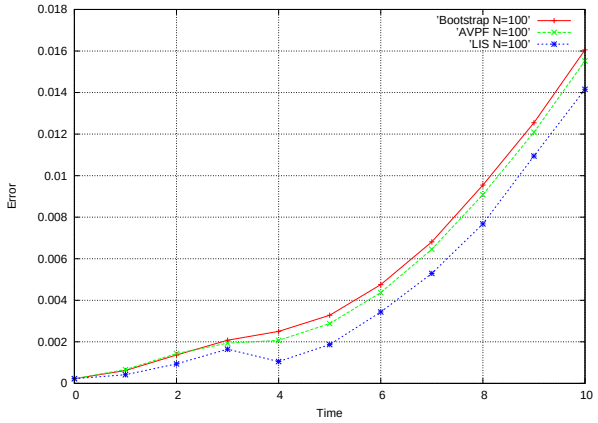

Figure 5. Tracking errors of the bootstrap filter, AVPF and LIS as a function of time. In these experiments the number of particles was kept fixed at the same value (here 100).

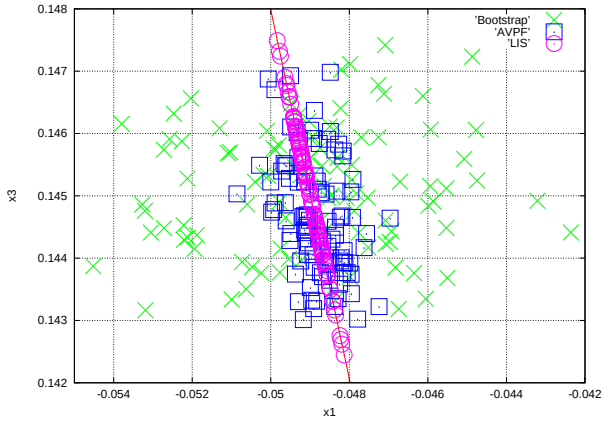

Figure 6. Particle clouds generated by AVPF ('□'), the bootstrap filter ('×') and LIS ('○') for an arbitrary selected time step

The running time of all the considered algorithms scales linearly with the number of particles. Hence in real-time applications where the per iteration time is limited, the running time will limit the number of particles that can be used. Since the per particle cost of the algorithms is different, the slower algorithms' performance will suffer more from real-time constraints. Comparing the computational cost of algorithms is not easy. Here we provide both a detailed analysis of the per particle computations for each of the algorithms, as well as the results of some empirical results. Table 1 shows a detailed account for the various elementary computational steps for the algorithms considered. This table can serve as the basis of predicting the running times of the various algorithms. For example, in computer vision applications the evaluation of the observation density is the far most expensive step as it involves calling the actual image processing routines. In other applications (like the one considered here) the various steps have roughly the same cost. In such a case, LIS with N particles can be expected to be cheaper to execute than the bootstrap filter with M particles or AVPF with K particles if $3.5N < M$, and respectively, $1.75N < K$. The next table (Table 2) shows

	BPF	**AVPF**	**LIS**	**IS**
Sampling from the dynamics	1	2	1	0
Evaluating the obs. density	1	2	1	1
Evaluating the dyn. density	0	0	2	1
Sampling from the proposal	0	0	1	1
Proposal Density Evaluation	0	0	1	1
Preprocessing	0	0	1	0

Table 1. Per-particle computation steps of the bootstrap filter (BPF), AVPF, LIS and Importance Sampling(IS). Preprocessing for LIS is the calculation of the means of two Gaussians. This boils down to computing two matrix-vector products, implementable with 8 multiplications.

the actual measured running times of the algorithms.[6] The codes of the three algorithms were written in C++ and all of them have reasonable implementations. No special effort was made, except those already mentioned, to optimize the codes of the algorithms. The table shows both the total CPU time when the number of particles is the same for all the three algorithms and the total CPU time when the particle numbers are set so that the errors of the algorithms are roughly equal (see Fig. 7). Given this table, we may conclude that LIS is roughly 12.5 times slower than the bootstrap filter and is roughly 2.6 times slower than AVPF, thus the theoretical predictions are off by a factor of 3.6 and 1.5, respectively. Despite this when the tracking errors are kept equal we get that LIS is the winner (in terms of execution time), followed by AVPF and the bootstrap filter.

	BPF	**AVPF**	**LIS**
CPU time with 10000 particles	122ms	590ms	1523ms
CPU time with equal error	36.6ms N=3000	29.5ms N=500	15.2ms N=100

Table 2. Measured running times of the bootstrap filter (BPF), AVPF and LIS with equal particle numbers and equal errors (N is the number of particles)

Fig. 8 shows the tracking error and the deviations of the errors of the three algorithms in the 10th time step for a

[6]The machine used was a Mobile Intel Celeron 2.5GHz with 256 MB RAM

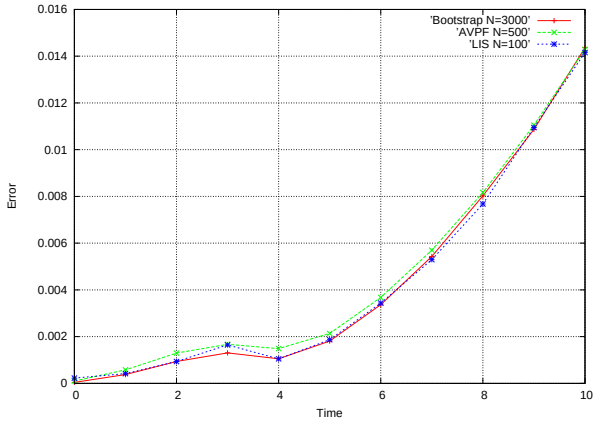

Figure 7. Tracking errors of the bootstrap filter, AVPF and LIS as a function of time. The particle sizes are set such that the errors are roughly equal. It can be seen from the figure that the performance of all three algorithms is the same uniformly in time.

number of sample sizes. As expected, the error decays (although not very rapidly) as the number of particles is increased for all the three algorithms. Interestingly, LIS keeps a considerable margin over the other algorithms over the range investigated, though its gain, by the nature of the problem studied, decreases when the number of particles is increased.

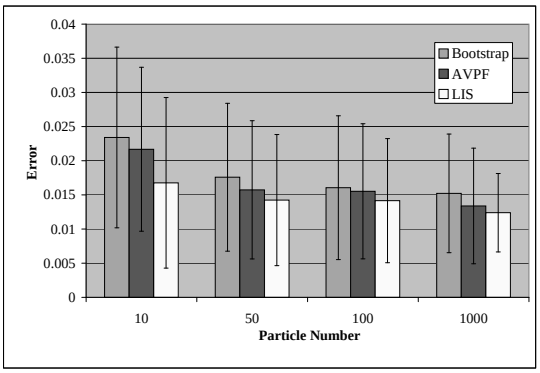

Figure 8. Tracking errors of the bootstrap filter, AVPF and LIS and the deviation of the error as a function of the number of particles

In the above experiments the standard deviation of the window function σ_g was set to 0.0005 and the ratio parameter κ of the proposal was set to 100. In order to test the sensitivity of LIS to these parameters we experimented with a number of values for these parameters. First, the window parameter was changed. This resulted in no significant changes in the performance as long as the parameter was kept in a reasonable range. It should be clear, however, that if the window size is too small then LIS degrades to the bootstrap filter. As remarked earlier, smaller values of κ, the parameter that governs how much we trust in the predicted distance of the ship, were found to yield to enhance performance. As κ and the window size both grow to infinity the algorithm degrades to likelihood sampling.

D. Results for Tracking Multiple Ships

Several real-world tasks require tracking of objects in high-dimensional spaces. In this section we study the performance of the algorithms for the simplified multi-object tracking problem that was described informally in Section V. *A.* . Formally, we assume that the observation likelihood for the observations $Y_1, \ldots, Y_M$ (assuming M ships) is of the product form:

$$r(Y_1, \ldots, Y_M|\theta_1, \ldots, \theta_M) = \prod_{k=1}^{M} r(Y_k|\theta_k).$$

Clearly, when M ships are tracked, the dimension of the state becomes $4M$. In reporting the errors the Euclidean distance of the mean predicted and actual positions is divided by the number of ships tracked, so as to allow a meaningful comparison of results obtained with differing ship-numbers. Exploiting the simple structure of the problem, the proposal of LIS is chosen to take a product form, just like the window functions.

In the first set of experiments 3 ships were tracked, resulting in a state-space of dimension 12. Fig. 9 shows the tracking errors of the algorithms as a function of time. These results were obtained with 10 tracking sequences and 100 runs for each sequences. Compared with Fig. 5, we observe that the advantage of LIS against the other algorithms increases considerably.

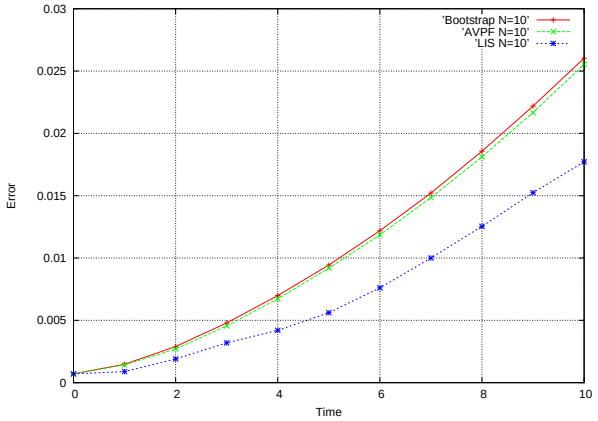

Figure 9. The average tracking error of the bootstrap filter, AVPF and LIS on the multi-object tracking problem with 10 particles

In order to gain further insight into the relative efficiency of these algorithms we present results for the equal CPU

time case, as well. Table 3 serves as the basis for computing the respective particle numbers. We note that as compared with the results for tracking a single ship, the execution times for the bootstrap filter and AVPF are doubled only, whilst that of for LIS are tripled. The better than expected execution times for the bootstrap filter and AVPF are a bit of surprising. We conjecture that some low-level mechanisms (caching, loop unrolling) might have caused this differences. Based on these results, the number of particles in

	BPF	**AVPF**	**LIS**
CPU time with 10000 particles	245ms	796ms	4597ms
CPU time with equal error	245ms N=10000	238ms N=3000	4.6ms N=10

Table 3. Measured running times of the bootstrap filter (BPF), AVPF and LIS with equal particle numbers and equal errors (N is the number of particles) for tracking 3 ships (N is the number of particles)

the subsequent experiments was set to 10, 50 and 100, respectively. Fig. 10 shows the resulting tracking errors as a function of time steps. LIS again clearly performs better than the other algorithms, despite that it uses 10 particles only. Figs. 11–13 plot the particle clouds for the three algo-

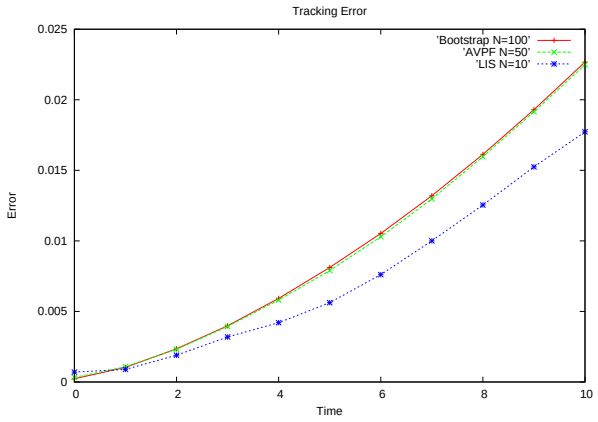

Figure 10. Average tracking error for the bootstrap filter, AVPF and LIS when tracking 3 ships. The number of particles are set to 100, 50 and 10, respectively so as to ensure that the algorithms' running times are the same.

rithms. Note that on these figures a single particle is represented by 3 points. We also remark that the horizontal and vertical scales are different in these figures. This creates the (wrong) impression that the estimated posterior's variance in the horizontal direction for both the bootstrap filter and AVPF were larger than the variance in the vertical direction.

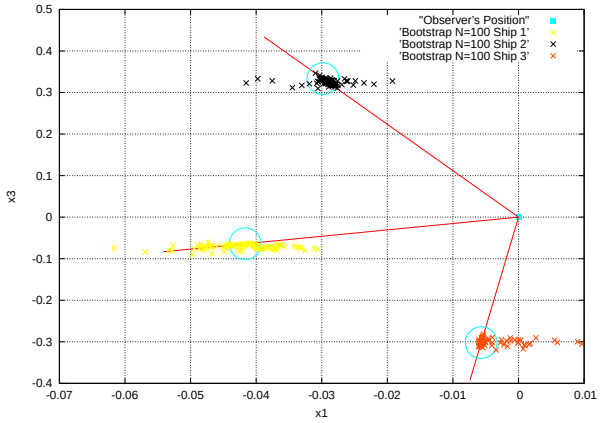

Figure 11. Illustration of the posterior representation of the bootstrap filter when 3 ships are tracked simultaneously. The number of particles is 100. The ships' positions are represented by larger circles. The figure also shows the straight lines connecting the observer's position with positions of the ships.

Figs. 14–16 show the sample paths of the ships together with the mean predicted positions for a given sequence. In these plots the number of particles are 1000 for the bootstrap filter, 200 for AVPF and 10 for LIS. Visual inspection reveals that LIS indeed makes a better use of the available information in the observations. In fact, the figures show that the error of tracking of the second ship becomes overly large for the bootstrap filter, whilst for AVPF the error becomes somewhat large both for the second and the first ship. Although the tracking error towards the last steps increases for ship 2 for LIS, LIS's errors are still much smaller than those obtained for the other algorithms for all ships and almost all time steps.

For the sake of completeness, the average tracking error with the corresponding standard deviations are plotted in Fig. 17 as a function of the number of particles. The figure confirms that tracking errors decrease with increasing the number of particles. Again, LIS is able to keep its margin over the range investigated.

Fig. 18 compares the tracking error and its standard deviation for LIS and the bootstrap filter when the number of objects to be tracked is increased from 2 to 20. As noted before, the Euclidean distance is normalized by the number of ships so as to allow a meaningful comparison between results of tracking when the object numbers are different.[7] In these experiments the ships' initial positions were set systematically with equal spacings along a circle with a fixed radius and setting the initial velocity direction to the tangent of the circle at the initial point. We remark that when the number of ships is 20, the state-space is 80 dimensional.

In these experiments only the bootstrap filter and LIS were compared. As it can be observed from the figure, the performance of the bootstrap filter degrades as the dimen-

[7]This normalization causes the decrease of the standard deviations.

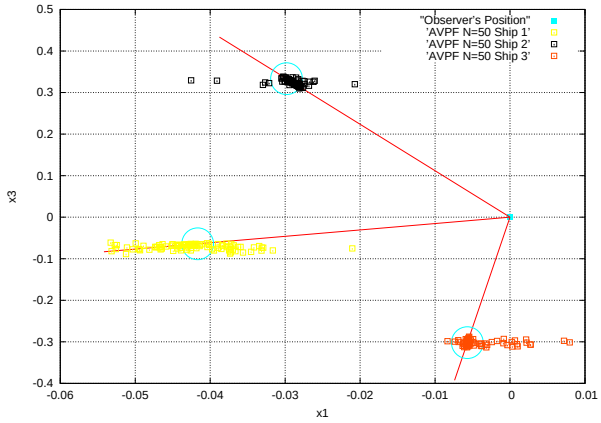

Figure 12. Illustration of the posterior representation of AVPF when 3 ships are tracked simultaneously. The number of particles is 50.

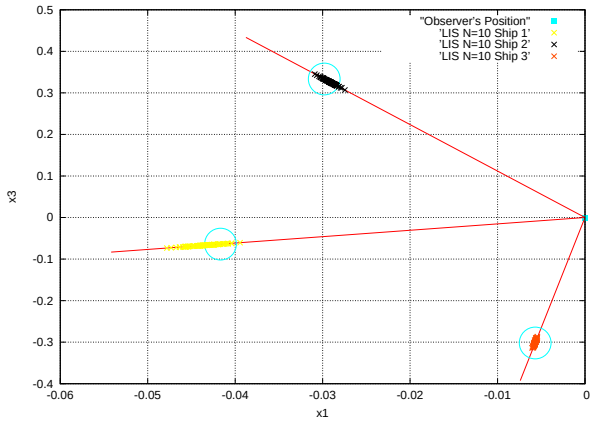

Figure 13. Illustration of the posterior representation of LIS when 3 ships are tracked simultaneously. The number of particles is 10.

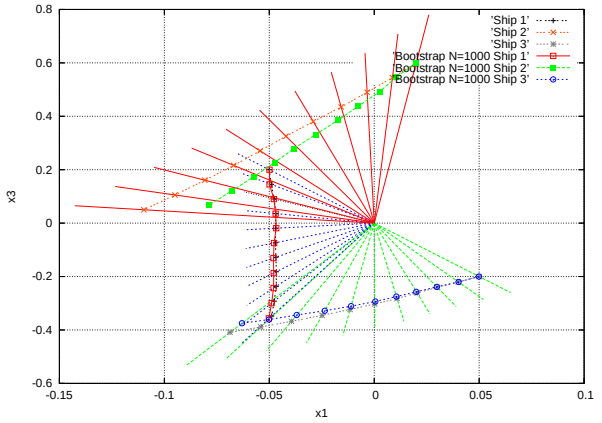

Figure 14. The ships' trajectories and the corresponding mean predicted positions of the 3 ships for the bootstrap filter with 1000 particles

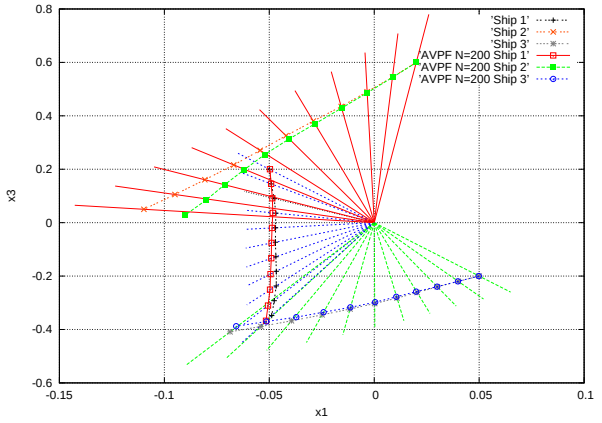

Figure 15. The ships' trajectories and the corresponding mean predicted positions of the 3 ships for AVPF with 200 particles

sionality is increased, whilst the performance of LIS stays steady. The degradation of the performance of the bootstrap filter is not as severe as one would expect due to the smoothness of the dynamics of the system (for less smooth systems the error might blow up exponentially).[8] We think that it is quite encouraging that the tracking error of LIS is not effected by the increased dimensionality, raising the hope that LIS could be used as a basis of efficient particle filters that are able to track very high-dimensional systems with a fairly high precision.

VI. Conclusions

The 'curse of reliable observations' refers to the well-known limitation of particle filters that increasingly reliable observations causes their performance to degrade. In this paper we have considered a family of particle filters that are designed to avoid this curse. These algorithms are modifications of the standard bootstrap filter whereas after the prediction step, the particles are randomly relocated in a second sampling step where sampling is restricted to a neighbourhood of the current position and is influenced by the most recent observation. Depending on whether this second step uses the observation density directly or an importance function, the resulting algorithms are called the Local Likelihood Sampling (LLS) or Local Importance Sampling (LIS) based particle filters. LLS was proposed earlier in [16], whilst LIS is proposed here. We have argued that LLS and LIS are more efficient than previous algorithms when the observations are reliable and that they can actually avoid the curse. Experiments with the standard bearings-only tracking problem and one multi-object version confirmed this expectation. Importantly, as compared to the bootstrap filter and AVPF, LIS was found to be increasingly more efficient

[8] In fact, we suspect that the errors at $N = 20$ are very close to the error level that can be obtained by pure prediction given the initial state and not taking into account the observations.

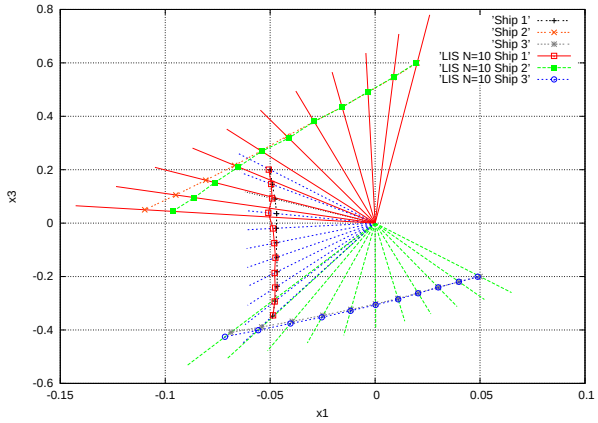

Figure 16. The ships' trajectories and the corresponding mean predicted positions of the 3 ships for LIS with 10 particles

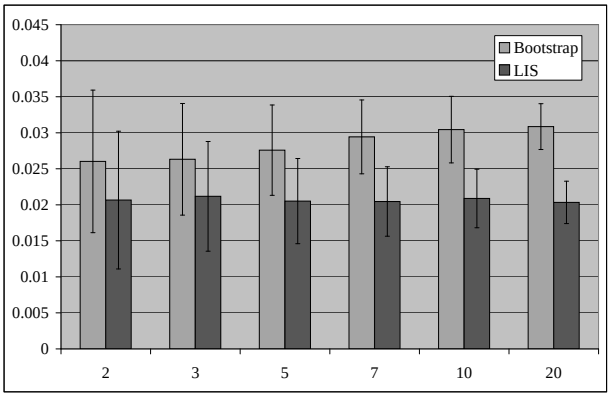

Figure 18. Tracking error for time step 10, as a function of the number of ships to be tracked for the bootstrap filter and LIS. Both algorithms use the same number of particles. The statistics is obtained by running 100 experiments for 10 independent realizations.

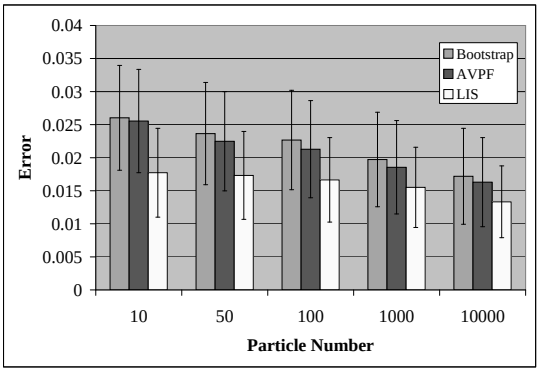

Figure 17. Tracking errors of the bootstrap filter, AVPF and LIS and the deviation of the error with different particle sizes when tracking 3 ships

as the dimensionality of the system to be tracked was increased. This raises the hope that based on LIS efficient particle filters might be designed for the open problem of tracking objects in very high-dimensional spaces.

REFERENCES

[1] N. Bergman. *Recursive Bayesian Estimation: Navigation and Tracking Applications*. PhD thesis, Department of Electrical Engineering, Linkoping, 2000.

[2] J. Carpenter, P. Clifford, and P. Fearnhead. An improved particle filter for nonlinear problems. *IEEE Proceedings-Radar Sonar and Navigation*, 146(1):2–7, 1999.

[3] D. Crisan and A. Doucet. A survey of convergence results on particle filtering for practitioners. *IEEE Trans. Signal Processing*, 50(3):736–746, 2002.

[4] W. B. D. Fox, S. Thrun and F. Dellaert. Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, New York, 2001. Springer.

[5] A. Doucet. On sequential simulation based methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 1998.

[6] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[7] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Proc. Inst. Elect. Eng. F*, 140(2):107–113, 1993.

[8] J.Vermaak, S. J.Godsill, and P. Perez. Monte carlo filtering for multi-target tracking and data association. *IEEE Transaction on Aerospace and Electronic System*, 2005.

[9] P. D. Moral and G. Salut. Non-linear filtering using monte carlo particle methods. *C.R. Acad. Sci. Paris*, pages 1147–1152, 1995.

[10] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *European Conference on Computer Vision*, volume 1, pages 28–39, 2004.

[11] M. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filter. *Journal of the American Statistical Association*, 94:590–599, 1999.

[12] M. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[13] D. Scott. *Multivariate Density Estimation. Theory, Practice, and Visualization*. J. Wiley & Sons, New York, London, Sydney, 1992.

[14] D. Titterington, A. Smith, and U. Makov. *Statistical Analysis of Finite Mixture Distributions*. J. Wiley & Sons, New York, London, Sydney, 1985.

[15] P. Torma and C. Szepesvári. LS-N-IPS: an improvement of particle filters by means of local search. *Proc. Non-Linear Control Systems(NOLCOS'01) St. Petersburg, Russia*, 2001.

[16] P. Torma and C. Szepesvári. Enhancing particle filters using local likelihood sampling. *ECCV2004, Prague. Lecture Notes in Computer Science*, pages 16–28, 2004.

[17] P. Torma and C. Szepesvári. On using likelihood-adjusted proposals in particle filtering: Local importance sampling. In *In S.Loncranic, H.Babic and M.Bellanger (eds.) Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pages 58–63, 2005.