18th International Conference on Information Fusion
Washington, DC - July 6-9, 2015

Langevin Monte Carlo Filtering for Target Tracking

Fernando J. Iglesias García, Mélanie Bocquel, Hans Driessen
Thales Nederland B.V. - Sensors Development System Engineering,
Hengelo, Nederland
Email: {Fernandojose.Iglesiasgarcia, Melanie.Bocquel, Hans.Driessen} @nl.thalesgroup.com

***Abstract*—This paper introduces the Langevin Monte Carlo Filter (LMCF), a particle filter with a Markov chain Monte Carlo algorithm which draws proposals by simulating Hamiltonian dynamics. This approach is well suited to non-linear filtering problems in high dimensional state spaces where the bootstrap filter requires an impracticably large number of particles.**

The simulation of Hamiltonian dynamics is motivated by leveraging more model knowledge in the proposal design. In particular, the gradient of the posterior energy function is used to draw new samples with high probability of acceptance. Furthermore, the introduction of auxiliary variables (the so-called momenta) ensures that new samples do not collapse at a single mode of the posterior density.

In comparison with random-walk Metropolis, the LMC algorithm has been proven more efficient as the state dimension increases. Therefore, we are able to verify through experiments that our LMCF is able to attain multi-target tracking using small number of particles when other MCMC-based particle filters relying on random-walk Metropolis require a considerably larger particle number.

As a conclusion, we claim that performing little additional work for each particle (in our case, computing likelihood energy gradients) turns out to be very effective as it allows to greatly reduce the number of particles while improving tracking performance.

I. Introduction

Filtering in dynamical systems with nonlinear time evolution and where the state of interest is only observable through nonlinear functions is a hard problem. A closed-form expression for the Bayesian posterior density is available only for a restricted class of dynamic systems. Consequently, it is oftentimes required to resort to approximate, suboptimal solutions given by numerical approximations. Amongst those, sequential Monte Carlo approaches (including particle filters) [1], [2] are especially attractive for two main reasons. First, they can in principle perform statistical inference no matter the mathematical form of the underlying system. Second, they feature convergence guarantees provided enough computational power.

The seminal particle filter [1] based on sequential importance sampling (SIS) and resampling can become computationally intractable as the dimension of the state space increases. In the context of multiple target tracking, this imposes a hard limitation in the number of objects we are able to track *jointly*. Furthermore, in order to process efficiently large amounts of data it is necessary to keep the number of samples low.

The combination of particle filters and Markov chain Monte Carlo (MCMC) methods is becoming increasingly more popular in the multi-object filtering domain [3], [4], [5]. The reason behind this growing interest being the efficiency of MCMC in high-dimensional, complex problems.

In the MCMC literature, the hybrid or Hamiltonian Monte Carlo (HMC) [6], [7] method is rather popular as it allows the sampler to perform an efficient search of the state space by leveraging model knowledge through first-order gradients. This is in contrast with random-walk MCMC techniques, such as Metropolis-Hastings [8], which perform an exhaustive, inefficient exploration of the state space by taking a succession of random steps. Furthermore, HMC is well-suited to sampling in high-dimensional state spaces [7].

This article introduces a novel approach to HMC-based particle filtering by means of the Langevin Monte Carlo (LMC) method, a special case of HMC. Theoretically, LMC features better asymptotic complexity with the dimension of the posterior than Metropolis-Hastings [7]. Thus, we expect that a particle filter relying on LMC requires fewer particles than an equivalent Metropolis-Hastings-based particle filter. We verify through example this hypothesis after describing an innovative fashion of introducing Hamiltonian dynamics in a multi-target particle filter through the Langevin equation.

II. State Space Model and Bayesian Filtering

Consider the following formulation for a general nonlinear dynamical system:

$$\mathbf{s}_{k+1} = f\left(\mathbf{s}_k, \mathbf{v}_k\right), \tag{1}$$

$$\mathbf{z}_k = g\left(\mathbf{z}_k, \mathbf{w}_k\right). \tag{2}$$

In the equations, $\mathbf{s}$ and $\mathbf{z}$ are used to denote the state of the system and the observations, respectively. The underlying processes describing the time evolution of the states and the observations of the system are assumed to follow the structure of a hidden Markov model.

The probability density function (PDF) of the process noise $\mathbf{v}$ together with the transition function f specify the transition distribution $\Pi\left(\mathbf{s}_{k+1}|\mathbf{s}_k\right)$. Likewise, the measurement noise $\mathbf{w}$ and the function g specify the likelihood $\Theta\left(z_k|\mathbf{s}_k\right)$ which defines the probability of observing $\mathbf{z}_k$ given that the system is at $\mathbf{s}_k$.

From a Bayesian perspective, the goal of filtering is specifying the PDF of the unobserved state conditioned on all evidence. Specifically, finding $p(\mathbf{s}_k|\mathbf{z}_{1:k})$. The solution essentially consists of the time and observation updates of the Bayes filter:

$$p(\mathbf{s}_k|\mathbf{z}_{1:k-1}) = \int \Pi(\mathbf{s}_k|\mathbf{s}_{k-1})\ p(\mathbf{s}_{k-1}|\mathbf{z}_{1:k-1})\ \mathrm{d}\mathbf{s}_{k-1} \quad (3)$$

$$p(\mathbf{s}_k|\mathbf{z}_{1:k}) \propto \Theta(z_k|\mathbf{s}_k)\ p(\mathbf{s}_k|\mathbf{z}_{1:k-1}) \quad (4)$$

III. MCMC-BASED PARTICLE FILTERING

In the particle filtering framework, MCMC can be introduced as a technique to improve the diversity of the particle cloud. This can be achieved by the simulation of a Markov chain starting from each of the surviving particles. In this way, MCMC is used in conjunction with importance sampling and resampling to rejuvenate degenerate particles. Nevertheless, a joint resampling-MCMC scheme renders a computationally expensive filter.

Another possibility is to replace importance sampling (thus, there is no need for resampling) with MCMC sampling. Henceforth, we will refer to such approach as MCMC-based particle filtering. From a practical point of view, such approach is interesting because a particle filter without resampling keeps its full parallelisation potential. In addition, the exponential growth of the number of particles with the dimension of the state space can be handled differently with an MCMC-based particle filter: thanks to its iterative structure, MCMC sampling is able to generate proposals modifying only a subset of the state variables. By contrast, a SIS particle filter requires to increase the number of particles; difficult design of the proposal distribution; or the addition of heuristics to post-process the particle cloud.

Previous work on MCMC-based particle filtering includes [3], [4], [5]. Although they are not exactly the same, all these methods rely on the Metropolis-Hastings algorithm.

In this paper, we introduce a novel MCMC-based particle filter that exploits model knowledge through the use first-order gradients. Our approach is appealing because the gradient of the posterior is included through a principled probabilistic fashion, well studied in the MCMC literature, known as Langevin Monte Carlo. Section IV describes the MCMC technique used by our filter, which is later introduced in Section V.

IV. LANGEVIN MONTE CARLO

This section describes MCMC methods that incorporate the gradient of the posterior into their sampling strategy. As it shall be detailed, the gradient is not simply introduced as in gradient descent, which would eventually collapse samples at the peaks of the posterior. Rather, the gradient is introduced through a well founded probabilistic framework.

HMC is a form of MCMC inspired by Hamiltonian mechanics [6], [7]. In particular, HMC is a sampling method whose proposal mechanism is based on the simulation of Hamiltonian dynamics through the introduction of auxiliary variables. Let us now recall Hamilton's equations and some other notation from mechanics necessary to convey the background of the HMC. Afterwards, we will make clear its connection with statistics and how it is formulated as an MCMC method. Firstly, Hamilton's equations in vectorial form are

$$\dot{\mathbf{q}} = \nabla_{\mathbf{p}}\mathcal{H}, \quad (5)$$

$$\dot{\mathbf{p}} = -\nabla_{\mathbf{q}}\mathcal{H}, \quad (6)$$

where $\mathbf{q}, \mathbf{p} \in \mathbb{R}^d$ are the so-called *position* and *momentum* variables and $\mathcal{H} \equiv \mathcal{H}(\mathbf{q}, \mathbf{p}) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is the *Hamiltonian*. We use the Nabla operator from calculus so that $\nabla_{\mathbf{x}}F$ denotes the gradient of the scalar field $F \equiv F(\mathbf{x})$ with respect to the vector $\mathbf{x}$. The time derivative of $\mathbf{x}$ is denoted by $\dot{\mathbf{x}}$.

The Hamiltonian is usually expressed as the sum of two scalar quantities, namely, the potential $\mathcal{U} \equiv \mathcal{U}(\mathbf{q})$ and kinetic energies $\mathcal{K} \equiv \mathcal{K}(\mathbf{p})$,

$$\mathcal{H}(\mathbf{q}, \mathbf{p}) = \mathcal{U}(\mathbf{q}) + \mathcal{K}(\mathbf{p}). \quad (7)$$

By making use of the *canonical distribution* concept, energy functions are associated with probability distributions via the expression

$$\mathcal{P} = \exp(-\mathcal{E}) \leftrightarrow \mathcal{E} = -\log(\mathcal{P}), \quad (8)$$

where $\mathcal{E}$ and $\mathcal{P}$ denote an arbitrary energy function and its associated probability distribution, respectively. In HMC, the usual choice of kinetic energy is

$$\mathcal{K}(\mathbf{p}) = \frac{1}{2}\mathbf{p}^{\mathrm{T}}\mathrm{M}^{-1}\mathbf{p}. \quad (9)$$

An interpretation of (5)-(6) is a joint mapping of $\mathbf{q}$ and $\mathbf{p}$ from time t to time $t+s$. Thus, Hamilton's dynamics specify a trajectory in an joint space of position and momentum. By choosing $\mathcal{U}$ to be the energy associated with the distribution of interest, Hamilton's equations specify a mechanism to explore a joint state space by generating samples ($\mathbf{q}$,$\mathbf{p}$), where $\mathbf{q}$ are in fact the samples of the distribution of interest and $\mathbf{p}$ are auxiliary variables.

Although the trajectory of $\mathbf{q}$ and $\mathbf{p}$ is continuous, it is in general not possible to compute it exactly since that requires solving exactly the differential equations (5)-(6). Therefore, it is necessary to resort to tools from numerical analysis to discretise and solve Hamilton's equations. Although not the only possible choice, in the HMC literature the most popular technique is the *leapfrog integrator*. Letting M be identity matrix in (9), a single step of the leapfrog method is

$$\mathbf{p}(t+\epsilon/2) = \mathbf{p}(t) - \epsilon/2\ \nabla_{\mathbf{q}}\ \mathcal{U}(t), \quad (10)$$

$$\mathbf{q}(t+\epsilon) = \mathbf{q}(t) + \epsilon\ \mathbf{p}(t+\epsilon/2), \quad (11)$$

$$\mathbf{p}(t+\epsilon) = \mathbf{p}(t+\epsilon/2) - \epsilon/2\ \nabla_{\mathbf{q}}\ \mathcal{U}(t+\epsilon). \quad (12)$$

HMC has two algorithm specifications: the leapfrog step size, ϵ, and the number of leapfrog steps, ℓ. Tuning ϵ and ℓ is difficult and therefore represents a practical barrier to the use of HMC. Suggestions for tuning ϵ and ℓ can be found in [7]. Tuning of ϵ on its own is also challenging. On the one hand, if ϵ is too large, the leapfrog integration becomes unstable. On the other hand, if ϵ is too small, the algorithm takes many small steps and becomes inefficient to explore the state space.

The main motivation behind HMC is to explore the state space efficiently, generating proposals whose probability of acceptance is large. As a matter of fact, Hamilton's equations conserve the value of the Hamiltonian. If proposals were generated by solving the equations exactly, then acceptance would happen with probability equal to one. The leapfrog integrator (or any other numerical method, for that matter) introduces some discretisation error which introduces some variation in the Hamiltonian at different points of the trajectory (when this variation grows unbounded the method is said to be unstable). Consequently, a Metropolis acceptance ratio is introduced to account for the variation in the Hamiltonian.

If the number of leapfrog steps is set to one, a special case of HMC called Langevin Monte Carlo (LMC) arises [7]. In fact, LMC is an iteration of HMC with one leapfrog step:

$$\mathbf{q}^* = \mathbf{q} - \epsilon^2/2\ \nabla_{\mathbf{q}}\, \mathcal{U}\left(\mathbf{q}\right) + \epsilon\ \mathbf{p} \tag{13}$$

$$\mathbf{p}^* = \mathbf{p} - \epsilon/2\ \nabla_{\mathbf{q}}\, \mathcal{U}\left(\mathbf{q}\right) - \epsilon/2\ \nabla_{\mathbf{q}}\, \mathcal{U}\left(\mathbf{q}^*\right) \tag{14}$$

The complete LMC algorithm is described in **Algorithm 1**.

```
Input  : A sample $\mathbf{q}$ of the posterior.
Output: A new sample $\mathbf{q}'$ of the posterior.
1  Draw momentum $\mathbf{p} \sim \mathcal{N}(\mathbf{0}, \mathrm{I})$.
2  $\mathbf{q}^* = \mathbf{q} - \epsilon^2/2\ \nabla_{\mathbf{q}}\, \mathcal{U}(\mathbf{q}) + \epsilon\ \mathbf{p}$.
3  $\mathbf{p}^* = \mathbf{p} - \epsilon/2\ \nabla_{\mathbf{q}}\, \mathcal{U}(\mathbf{q}) - \epsilon/2\ \nabla_{\mathbf{q}}\, \mathcal{U}(\mathbf{q}^*)$.
4  Compute Hamiltonians:
5      $\mathcal{H} = \mathcal{U}(\mathbf{q}) + 1/2\ \mathbf{p}^{\mathrm{T}}\mathbf{p}$.
6      $\mathcal{H}^* = \mathcal{U}(\mathbf{q}^*) + 1/2\ \mathbf{p}^{*\mathrm{T}}\mathbf{p}^*$.
7  Acceptance ratio:
8      $\alpha = \min\{1, \exp(-\mathcal{H}^* + \mathcal{H})\}$.
9      With probability $\alpha$, acceptance: $\mathbf{q}' = \mathbf{q}^*$;
10     otherwise, rejection: $\mathbf{q}' = \mathbf{q}$.
```

Algorithm 1: The Langevin Monte Carlo algorithm.

Even though LMC sampling does not keep all the advantages of HMC, its asymptotic complexity with the dimension of the state is lower than in random-walk Metropolis [7]. This is actually our main motivation behind proposing an LMC-based particle filter. As it is proven to scale better with dimensionality, we expect that a filter relying on LMC requires less number of particles to attain a certain performance standard than an equivalent filter using Metropolis-Hastings updates. Reducing the number of particles is critical when likelihood computations are very expensive since particle filters require to evaluate the likelihood for every particle. The main principle is that doing some extra work to propose better particles (extra work in this context means the evaluation of gradients) pays off if the necessary number of particles can be drastically reduced.

Remark: the term position used in this section does not mean the position of the objects under track in a sequential setting. In fact, when HMC is used in the context of tracking, the position variables include target position and velocity, possibly in addition to other state variables. Likewise, the time involved in the differential equations is not the time of the state and observation processes in the problem of filtering.

V. The Langevin Monte Carlo Filter

The Langevin Monte Carlo filter (LMCF) is summarised in **Algorithm 2**. This algorithm uses a state vector formed by the concatenation of the single-target state vectors (the so-called partitions, each of dimension $\mathrm{N}_{\mathbf{s}_j}$) , i.e.

$$\mathbf{s}_k = \left[\mathbf{s}_{k,1}^{\mathrm{T}} \quad \cdots \quad \mathbf{s}_{k,j}^{\mathrm{T}} \quad \cdots \quad \mathbf{s}_{k,\mathrm{N_t}}^{\mathrm{T}}\right]^{\mathrm{T}}. \tag{15}$$

where the number of targets $\mathrm{N_t}$ is fixed and known. In the algorithm, $\mathrm{N_p}$ denotes the number of particles.

The LMCF is inspired by the interacting population MCMC-PF (IP-MCMC-PF) [5], [9]. Proposals in the IP-MCMC-PF are drawn from the prediction density $p(\mathbf{s}_k|\mathbf{z}_{1:k-1})$ and accepted with probability

$$\alpha = \min\left\{1, \frac{\Theta\left(\mathbf{z}_k|\mathbf{s}_k^*\right)}{\Theta\left(\mathbf{z}_k|\mathbf{s}_k\right)}\right\}, \tag{16}$$

where $\mathbf{s}_k^*$ denotes the proposal and $\mathbf{s}_k$ the current state of the chain.

In comparison with **Algorithm 1**, the state vector $\mathbf{s}_k$ in **Algorithm 2** corresponds to the position variables $\mathbf{q}$ in Hamiltonian-based sampling. In both algorithms, the momentum is denoted by $\mathbf{p}$ and it serves to build proposals.

```
Input  : Particle approximation of $\mathrm{p}(\mathbf{s}_{k-1}|\mathbf{z}_{1:k-1})$,
         measurement $\mathbf{z}_k$.
Output: Particle approximation of $\mathrm{p}(\mathbf{s}_k|\mathbf{z}_{1:k})$.
1  Draw a seed $\mathbf{s}_k^{(0)} \sim \mathrm{p}(\mathbf{s}_k|\mathbf{z}_{1:k-1})$.
2  for $i \leftarrow 1$ to $N_p$ do
3      Draw $\mathbf{s}_k^{+} \sim \mathrm{p}(\mathbf{s}_k|\mathbf{z}_{1:k-1})$.
4      Draw partition index uniformly $j \sim \mathcal{U}\{1, \mathrm{N_t}\}$.
5      Apply Langevin step to the partition $j$ of $\mathbf{s}_k^{+}$, $\mathbf{s}_{k,j}^{+}$:
6          $\mathbf{p} \sim \mathcal{N}(\mathbf{0}, \mathrm{I})$ of dimension $\mathrm{N}_{\mathbf{s}_j}$.
7          $\mathbf{s}_{k,j}^{*} = \mathbf{s}_{k,j}^{+} - \epsilon^2/2\ \nabla_{\mathbf{s}_{k,j}}\, \mathcal{U}_\Theta(\mathbf{s}_k^{+}) + \epsilon\ \mathbf{p}$.
8          $\mathbf{p}^* = -1/\epsilon\left(\mathbf{s}_{k,j}^{(i-1)} - \mathbf{s}_{k,j}^{+} + \epsilon^2/2\ \nabla_{\mathbf{s}_k}\, \mathcal{U}_\Theta(\mathbf{s}_k^{+})\right)$.
9      Build the complete proposal:
10         $\mathbf{s}_k^{*} = \left[\mathbf{s}_{k,1}^{(i-1)}\ \ldots\ \mathbf{s}_{k,j}^{*}\ \ldots\ \mathbf{s}_{k,\mathrm{N_t}}^{(i-1)}\right]^{\mathrm{T}}$.
11     Compute Hamiltonians:
12         $\mathcal{H} = \mathcal{U}_\Theta\left(\mathbf{s}_k^{(i-1)}\right) + 1/2\ \mathbf{p}^{\mathrm{T}}\mathbf{p}$.
13         $\mathcal{H}^* = \mathcal{U}_\Theta(\mathbf{s}_k^{*}) + 1/2\ \mathbf{p}^{*\mathrm{T}}\mathbf{p}^*$.
14     Acceptance ratio:
15         $\alpha = \min\{1, \exp(-\mathcal{H}^* + \mathcal{H})\}$.
16         With probability $\alpha$, acceptance: $\mathbf{s}_k^{(i)} = \mathbf{s}_k^{(*)}$;
17         otherwise, rejection: $\mathbf{s}_k^{(i)} = \mathbf{s}_k^{(i-1)}$.
18 end
19 $\mathrm{p}(\mathbf{s}_k|\mathbf{z}_{1:k}) \approx \frac{1}{\mathrm{N_p}}\sum_{i=1}^{\mathrm{N_p}} \delta\left(\mathbf{s}_k - \mathbf{s}_k^{(i)}\right)$.
```

Algorithm 2: The Langevin Monte Carlo filter algorithm.

Motivation behind the proposal mechanism in the LMCF

Per-partition sampling: In the LMCF a proposed sample of the posterior differs from its corresponding seed in one, and only one, partition of the multi-target state. This is not

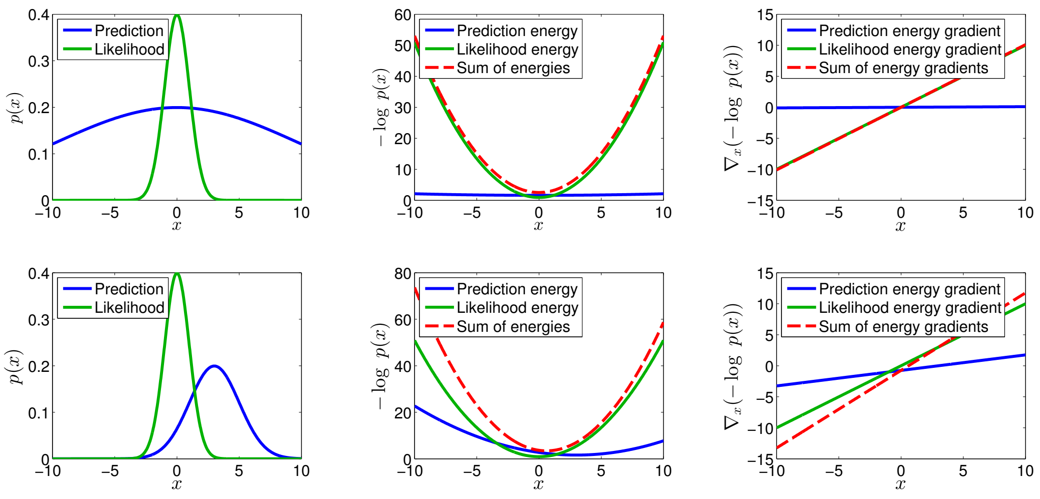


Fig. 1. Example illustrations of the impact of the gradient energy approximation exploited in the LMCF (see Section V). We note that due to the nature of the logarithm function, the gradient approximation is reasonable. First, for a trivial case where the prediction is rather uncertain and both likelihood and prediction have their modes at the same point (upper row). Second, for another case where there is mismatch between the modes of the likelihood and the prediction, and the uncertainty in the prediction is not much larger than the one in the likelihood (second row).

a novel decision in the proposal design of the algorithm as it is a common choice found in other MCMC-based particle filters [3], [5] and in the refinement step in [4]. The reason behind this decision lies in the avoidance of the curse of dimensionality which causes the acceptance ratio to drastically drop as the number of dimension increases. Intuitively, as the dimension of the space where proposals are drawn grows, its volume increases exponentially, thus requiring an intractable number of samples to ensure that regions of high probability are covered. We circumvent this problem by making proposals per-partition.

Posterior gradient approximation: Evaluation of the gradient of the posterior density energy in (17) is required to leverage Hamiltonian-based sampling. Applying Bayes theorem, the gradient in (18) is equal to the sum of the gradients of the likelihood energy and the predicted energy. The evaluation of the predicted energy's gradient has linear time complexity in terms of the number of particles. Therefore, the LMCF uses an alternative mechanism to generate proposals which only exploits the gradient of the likelihood energy, whose evaluation, for a single particle, does not depend on the total number of particles used in the filter. We argue for approximating the gradient of the posterior energy to the gradient of the likelihood energy (i.e. neglecting the gradient of the predictive energy). The argument is illustrated in Fig. 1. Recall that the predict stage of the Bayes filter (3) can be seen as a convolution of the belief at the previous time step s_{k-1} via the transition distribution $\Pi\left(s_k|s_{k-1}\right)$. The update stage of the Bayes filter (4) is a multiplication with the likelihood $\Theta\left(z_k|\mathbf{s}_k\right)$. Note that due to the nature of these two operations, the predict stage will result in a distribution with no less uncertainty than the update. As it can be appreciated from Fig. 1, when two distributions are multiplied, the energy (plots in the second column in the figure) of the resulting distribution is close to the energy of the distribution with less uncertainty. This is a consequence of the nature of the logarithm function. Therefore, a good approximation of the gradient of interest (plots in the third column) is the gradient of the energy of the least uncertain distribution.

$$\mathcal{U} = -\log \mathrm{p}\left(\mathbf{s}_k|\mathbf{z}_{1:k}\right) = \mathcal{U}_\Theta + \mathcal{U}_f \tag{17}$$

$$\nabla_{\mathbf{s}_k}\mathcal{U} = \nabla_{\mathbf{s}_k}\mathcal{U}_\Theta + \nabla_{\mathbf{s}_k}\mathcal{U}_f \tag{18}$$

Reverse momentum: It is worth taking a closer look at line 8 in **Algorithm 2**. This line corresponds to line 3 in **Algorithm 1**. In these lines, the momentum associated with the reverse move from proposal to the current state of the sampler is computed. In case of the LMCF, note that the momentum $\mathbf{p}^*$ is not computed using the proposal $\mathbf{s}^*_{k,j}$ and the current state $\mathbf{s}^{(i-1)}_{k,j}$, but the latter and the sample of the prediction $\mathbf{s}^+_{k,j}$. This is because the sampler of the LMCF does not use the momentum $\mathbf{p}$ to move from the current sample to the proposal (as in the general LMC outlined in the previous section), but to altere the sample of the prediction $\mathbf{s}^+_{k,j}$. Therefore, the reverse momentum $\mathbf{p}^*$ is computed by regarding a fictitious move from $\mathbf{s}^+_{k,j}$ to $\mathbf{s}^{(i-1)}_{k,j}$.

VI. Experiments

This section describes the example we have chosen to validate the performance of the novel Langevin-based particle filter introduced in this paper. We consider the motion of a fixed number of targets moving on a two-dimensional plane. The dynamics of the targets are modelled using the same nearly constant velocity motion model. The targets are tracked using

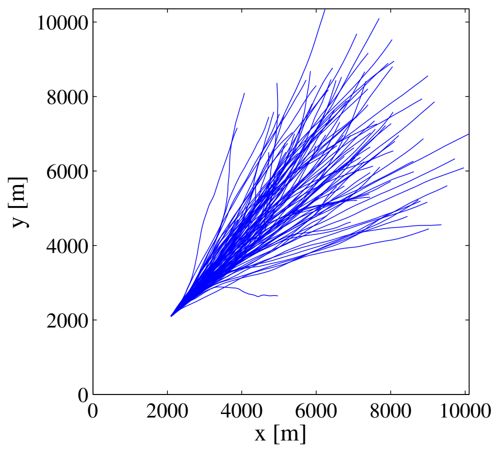


Fig. 2. Target trajectories in a scenario with 100 targets.

range and bearing plot measurements. An instance of a scenario like the ones used in the experiments is shown in Fig. 2.

The section is organised as follows. First, both the transition and observation models are introduced in VI-A and VI-B, respectively. Then, we briefly specify the particular values of the parameters used in our simulation in VI-C. Finally, the tracking performance obtained with our new LMC-based filter as well as other filters (including another MCMC-based particle filter) is shown in VI-D.

A. Motion model

The trajectories of the targets are simulated using the following equations:

$$\mathbf{a}_k \sim \mathcal{N}\left(\mathbf{0}, \sigma_a^2 \ \mathrm{I}_{2\mathrm{N}_\mathrm{t}}\right) \tag{19}$$

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \dot{\mathbf{x}}_{k-1} \ \Delta_t + \frac{1}{2} \ \mathbf{a}_k \ \Delta_t^2 \tag{20}$$

$$\dot{\mathbf{x}}_k = \dot{\mathbf{x}}_{k-1} + \mathbf{a}_k \ \Delta_t \tag{21}$$

I_n is the identity matrix of dimension n. Δ_t denotes the time difference between consecutive time steps. $\mathbf{x}$ is a vector formed by the concatenation of the two-dimensional positions of each target and $\dot{\mathbf{x}}$ is a vector constructed in a similar fashion containing the target velocities. $\mathbf{a}$ represents a random perturbation of the target velocities. Finally, k is used to denote the time step.

The motion model describes the (linear) dynamics of N_t targets moving independently with nearly constant velocities. The state vector of the j^{th} target ($j = 1, \ldots, \mathrm{N}_\mathrm{t}$) comprises four components:

$$\mathbf{s}_{k,j} = \begin{bmatrix} x_{k,j} & y_{k,j} & \dot{x}_{k,j} & \dot{y}_{k,j} \end{bmatrix}^{\mathrm{T}}, \tag{22}$$

corresponding to the position and velocity vectors. The state vector describing the dynamics of all targets is the concatenation of the individual target states as in (15).

The motion is specified by a near constant velocity model with Gaussian noise:

$$\mathbf{s}_k = \mathrm{F} \ \mathbf{s}_{k-1} + \mathbf{v}_k, \tag{23}$$

where

$$\mathrm{F} = \mathrm{I}_{\mathrm{N}_\mathrm{t}} \otimes \begin{bmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

corresponds to the deterministic component of the state transition, being $\otimes$ the Kronecker product. The noise is Gaussian distributed and uncorrelated,

$$\mathbf{v}_k \sim \mathcal{N}\left(\mathbf{0}, \Sigma_\Pi\right), \tag{25}$$

$$\Sigma_\Pi = \mathrm{I}_{\mathrm{N}_\mathrm{t}} \otimes \Sigma_\pi, \tag{26}$$

where Σ_π is the covariance of the single-target motion noise:

$$\Sigma_\pi = \Delta_t^2 \sigma_a^2 \begin{bmatrix} \frac{\Delta_t^2}{4} & 0 & 0 & 0 \\ 0 & \frac{\Delta_t^2}{4} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{27}$$

B. Observation model

This section describes the generation of the observations and the corresponding observation model. We consider a simplified observation model with perfect measurement-to-track association and where all the targets generate a single plot measurement at each time step. Needless to say, this approach is possible because we are simulating our own scenario (that is, both the trajectories and the observations). In a real application, one would need to solve the data association problem [10].

Similarly to the state vector, the complete observation is the concatenation of the individual target observations:

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{z}_{k,1}^{\mathrm{T}} & \cdots & \mathbf{z}_{k,j}^{\mathrm{T}} & \cdots & \mathbf{z}_{k,\mathrm{N}_\mathrm{t}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{28}$$

$$\mathbf{z}_{k,j} = \begin{bmatrix} r_{k,j} & b_{k,j} \end{bmatrix}^{\mathrm{T}}, \tag{29}$$

where $r_{k,j}$ and $b_{k,j}$ denote the observed range and bearing angle. The noise in the observations is additive, uncorrelated and Gaussian distributed with variance σ_r^2 for the range and σ_b^2 for the bearing.

Finally, the likelihood function is computed jointly using all the partitions of the state,

$$\Theta\left(\mathbf{z}_k | \mathbf{s}_k\right) = \prod_{j=1}^{\mathrm{N}_\mathrm{t}} \phi\left(r_{k,j}; g^r\left(\mathbf{s}_{k,j}\right), \sigma_r^2\right) \cdot \phi\left(b_{k,j}; g^b\left(\mathbf{s}_{k,j}\right), \sigma_b^2\right). \tag{30}$$

In the equation, $\phi\left(x; \mu, \sigma^2\right)$ denotes the PDF of a univariate Gaussian with mean μ and variance σ^2 evaluated at x; and the observation functions g^r and g^b are the usual ones:

$$g^r\left(\mathbf{s}_{k,j}\right) = \sqrt{x_{k,j}^2 + y_{k,j}^2}, \tag{31}$$

$$g^b\left(\mathbf{s}_{k,j}\right) = \arctan\left(\frac{y_{k,j}}{x_{k,j}}\right). \tag{32}$$

TABLE I
SIMULATION PARAMETERS.

Δ_t	σ_a^2	σ_r^2	σ_b^2
$1\,[\mathrm{s}]$	$50\,[\mathrm{m}^2\ \mathrm{s}^{-4}]$	$10^3\,[\mathrm{m}^4]$	$10^{-5}\,[\mathrm{rad}^4]$

TABLE II
VARIABLE STEP SIZE ϵ DEPENDING ON THE TIME STEP k.

$k \in$	$[1, 10]$	$[11, 20]$	$[21, 30]$	$[31, 40]$	$[41, 50]$
ϵ	20	25	30	35	40

C. Parameters

The value of the parameters we used in our simulations are outlined in Table I. The initial target positions $\mathbf{x}_0$ and velocities $\mathbf{v}_0$ are $2\,[\mathrm{km}]$ and $100\,\left[\mathrm{m\ s}^{-1}\right]$, respectively. The simulations start with targets at positions $\mathbf{x}_1$ and with velocities $\mathbf{v}_1$, obtained after applying (19)-(21) once, and last for 50 time steps.

A step size ϵ must be chosen for the LMCF. We choose ϵ carefully but not obsessively, by observing the results in a few preliminary runs. In particular, we use different values of ϵ depending on the time step. These values are reported in Table II.

All the particle filters used in the experiments are initialised by drawing particles from Gaussian densities centered around the ground truth positions and velocities. The standard deviations are equal to $500/3\,[\mathrm{m}]$ for target positions and $1\,\left[\mathrm{m\ s}^{-1}\right]$ for velocities. The initial means and covariances in the Kalman filter are also such Gaussian densities.

D. Results

Given the transition and observation models described in the previous paragraphs, we note we are dealing with a sequential estimation problem with a linear transition model and nonlinear observation model. The noise in both cases is zero-mean and Gaussian distributed. In spite of the nonlinearities in the observation model, a classical implementation of the Bayes filter such like the extended Kalman filter (EKF) is able to perform well in this problem. In fact, this enables us to have a baseline and we will regard the EKF performance as the best achievable one in this toy example. In this way, we can compare the minimum number of particles the particle filters require to attain a performance standard close to the EKF's.

Each time a scenario is simulated it will be different since the generation of the trajectories as described by (19)-(21) and the observations (28)-(29) are stochastic processes. In addition, the particle filters are random algorithms since they are based upon sampling which is inherently random. Consequently, there is a need of repeating the simulations in different scenarios many times in order to obtain robust and statistically reliable results. Let $\mathrm{N_{MC}}$ be the number of Monte Carlo simulations, then the measure to assess the performance of the filters is the average position error (APE), defined as:

$$\mathrm{APE}_k := \frac{1}{\mathrm{N_{MC}}\mathrm{N_t}} \sum_{n=1}^{\mathrm{N_{MC}}} \sum_{j=1}^{\mathrm{N_t}} \sqrt{\left(\hat{x}_{k,j}^n - x_{k,j}^n\right)^2 + \left(\hat{y}_{k,j}^n - y_{k,j}^n\right)^2} \tag{33}$$

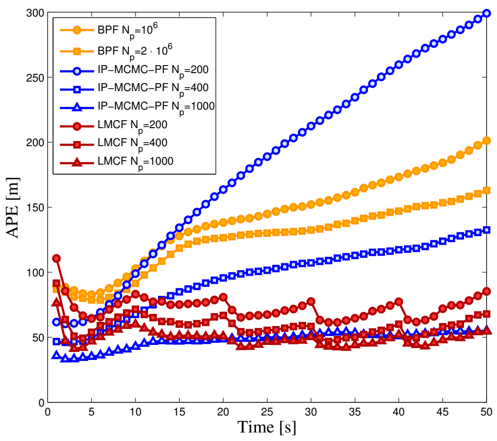
Together, $\hat{x}_{k,j}^n$ and $\hat{y}_{k,j}^n$ denote the estimated position of target j at time step k in the Monte Carlo simulation n, whereas $x_{k,j}^n$ and $y_{k,j}^n$ denote the ground truth target location. For the EKF, the estimated position is directly given by the update step of the filter. For the particle filters, the estimated positions are the sample average over the population of particles (which corresponds to the maximum likelihood estimate).

Results are obtained from $\mathrm{N_{MC}} = 100$ Monte Carlo simulations. The APE obtained with the bootstrap particle filter (BPF), the IP-MCMC-PF, and the novel LMCF are shown in Fig. 3a with $\mathrm{N_t} = 10$ (state space dimension equal to 40). In this case, the approximate computation times per time step[1] are 70 milliseconds for the LMCF with $\mathrm{N_p} = 400$, 120 milliseconds for the IP-MCMC-PF with $\mathrm{N_p} = 1000$, and 6.1 seconds for the BPF with $\mathrm{N_p} = 2 \cdot 10^6$. The APE at the last time step of the simulation for different choices of the number of particles is shown in Fig. 3b. This plot is useful to set a certain performance standard (i.e. fix a value along the y axis) and read from the x axis the number of particles that are required in order to obtain that performance for each of the particle filters. A similar plot for the case of $\mathrm{N_t} = 100$ (400-dimensional state space) is shown in Fig. 4a. The BPF is not included in these plots since the number of particles is much larger than for the MCMC-based filters (see Fig. 3a).

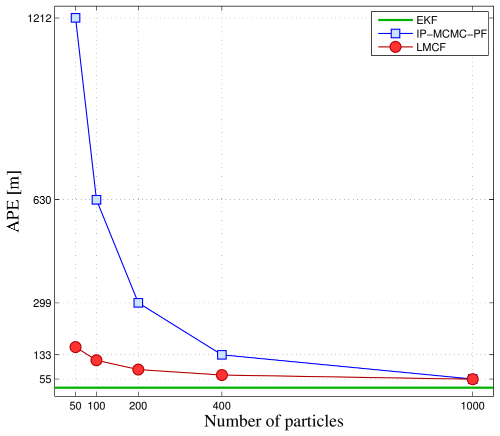
The results serve to validate our hypothesis that a particle filter leveraging Hamiltonian-based sampling, thus exploiting gradient information, requires less number of samples to successfully achieve a certain performance standard. In our case, such performance standard is given by the APE of the EKF. From Fig. 3b and Fig. 4a, it can be seen that an MCMC-based particle filter such as the IP-MCMC-PF, whose proposals are drawn from the prediction density, requires a considerably larger number of particles than the equivalent LMC-based particle filter that proposes by applying Langevin steps to samples of the prediction.

Fig. 4b shows box plots of the APE for the IP-MCMC-PF and the LMCF when $\mathrm{N_t} = 100$. $\mathrm{N_p} = 4000$ in both filters. From the boxes, we can see that the median error of the LMCF is stable. The small oscillations in the APE are given by the crude choice of step size, which is constant throughout several time steps (see Table II). In contrast with the bounded median error in the LMCF, the median error of the IP-MCMC-PF grows continuously over time.

[1]Matlab code executed on a 3.7 GHz Intel Xeon CPU E5-1620 with 16 GB RAM. The computation times should be interpreted with care since they are implementation dependent. For instance, the BPF implementation is vectorised, whereas the IP-MCMC-PF and LMCF are not. Also, none of the filters leverages parallel computing.

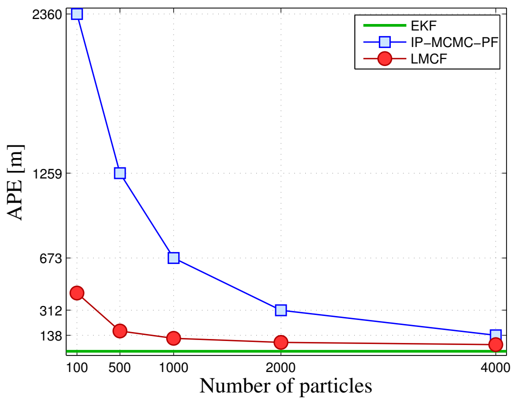


(a) Average position error as a function of the time step of the simulations.

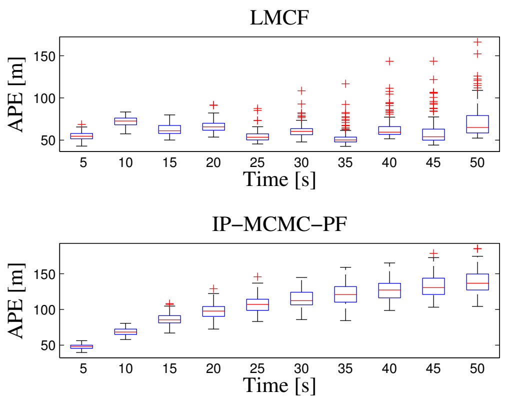


(b) Average position error at the last time step as a function of the number of particles.

Fig. 3. Results in the 10 targets scenario.



(a) Average position error at the last time step as a function of the number of particles.



(b) Average position error box plots with 4000 particles as a function of the time step.

Fig. 4. Results in the 100 targets scenario.

VII. Conclusion and Future Work

Conclusion

The work presented in this paper introduces a new MCMC-based particle filter leveraging Hamiltonian dynamics that we name the Langevin Monte Carlo filter (LMCF). This is not the first particle filter employing Hamiltonian Monte Carlo, hybrid Monte Carlo filtering was introduced in [11] by Choo et. al. Our approach however differs from Choo's in two aspects. Firstly, the LMCF is a pure MCMC-based particle filter, whereas Choo's filter also includes importance sampling and resampling. They include the MCMC algorithm as a way to improve the diversity of the particle cloud, which in turn makes it possible to reduce the number of particles. A drawback with Choo's approach is that resampling limits considerably the parallel computing potential of the filter. Secondly, the LMCF simulates Hamiltonian trajectories using a single leapfrog step (which results in a type of Langevin equation). To the best of our knowledge, this is the first MCMC-based particle filter that does not rely on resampling and leverages first-order gradient information in a principled manner through the use of Hamiltonian dynamics.

A combination of particle filtering and LMC has been recently introduced in [12] in the context of the particle MCMC (PMCMC) framework. This framework is nonetheless different from MCMC-based particle filtering. On the one hand, in PMCMC a particle filter is used in order to build efficient transition kernels for an MCMC sampler. On the other hand, we use MCMC as an efficient substitute of importance sampling and resampling in the particle filter.

Furthermore, the work presented here focusses on tracking a fixed and known number of objects. Of course, this is a very strong assumption far from the reality in many applications (such as radar surveillance). Nevertheless, we claim that this work is relevant for tracking an unknown and time-varying number of objects since approaches to deal with this more general problem often rely on filters designed for a fixed number of targets. Examples of these are the multiple hypotheses tracker (MHT) [13] and the multiple cardinality hypotheses tracker (MCHT) [14], which respectively rely on the Kalman filter and the interacting population MCMC PF (IP-MCMC-PF) [5].

In comparison with standard LMC sampling, we have to include certain modifications so that it is possible to use it successfully in our sequential setting. First of all, per-partition sampling. If samples were drawn using the Langevin equation from the joint state space, the acceptance ratio would be doomed to drop rapidly to zero as the number of partitions grows. Thus, resulting in a completely inefficient sampler. We show how to use per-partition sampling with the LMC and also verify experimentally its good performance even when the number of partitions is equal to one hundred. Secondly, given that the proposal mechanism in the LMCF is composed of two steps (sample the prediction density and then apply the Langevin step) we have to modify the computation of the momenta associated with the proposed samples to keep the posterior invariant (or, in other words, to maintain the detailed balance condition [15]).

Under certain assumptions on the posterior distribution, LMC can be shown to provide better scaling with the dimensionality of the state space than random-walk Metropolis in terms of the number of samples [7]. This is our main motivation to design a filter using the LMC. As we have proven by example, even for a problem with simple transition and observation models, the use of LMC has a considerable positive impact on performance: we are able to reduce the number of particles by a factor of 10 (see Fig. 3b and 4a) and still maintain the tracking performance standard (given by the EKF and measured through the average position error, see Section VI).

We expect this improvement to be especially relevant in data intensive applications where the computational cost per particle is large. For instance, integrated processing approaches like track-before-detect [16], wide-area surveillance applications, multi-sensor systems, amongst others.

Future Work

As discussed in VI-C, the step size in the LMCF needs to be manually adjusted. Nevertheless, this parameter could be tuned automatically. For instance, an algorithm based on the approach we followed in the experiments to find a suitable of value of ϵ would be to generate a small set of samples with different step sizes and choosing the value of ϵ for which the acceptance rate is closest to optimal. Another less engineered and more well-founded solution would consist of exploiting second-order derivative information through the Hessian [17].

In Section V we explained the per-partition proposal mechanism in the LMCF. Although this approach is suitable to avoid degeneracy as the number of partitions increases, it may not be always suitable if it is a requirement to keep intact target correlations in the transition model. By using HMC sampling with trajectories composed of several steps, it is expected that proposals can be made in the joint state space without resulting in an acceptance probability close to zero.

Finally, although stemming from completely different origins, both the Daum-Huang particle flow filter [18] and the HMC-based particle filters have in common that the transition between the predictive and the posterior densities are given by differential equations. Perhaps, insights into one of the methods could be used to gain more understanding of the other, and vice versa.

Acknowledgment

The research leading to these results has received funding from the EU's Seventh Framework Programme under grant agreement N^{o}607400. The research has been carried out within the **TRAX** project.

References

[1] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *IEE Proceedings F (Radar and Signal Processing)*. IET, 1993.

[2] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo methods in practice*. Springer-Verlag, 2001.

[3] Z. Khan, T. Balch, and F. Dellaert, "MCMC-based particle filtering for tracking a variable number of interacting targets," *Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[4] S. K. Pang, J. Li, and S. J. Godsill, "Models and algorithms for detection and tracking of coordinated groups," in *IEEE Aerospace Conference*, 2008.

[5] M. Bocquel, H. Driessen, and A. Bagchi, "Multitarget tracking with interacting population-based MCMC-PF," in *Proceedings of the 15th International Conference on Information Fusion*. IEEE, 2012.

[6] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Physics letters B*, 1987.

[7] R. M. Neal, "MCMC using Hamiltonian dynamics," in *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC, 2011.

[8] S. Chib and E. Greenberg, "Understanding the Metropolis-Hastings algorithm," *The American Statistician*, 1995.

[9] F. J. Iglesias García, M. Bocquel, and H. Driessen, "Advanced IP-MCMC-PF design ingredients," in *Proceedings of the 17th International Conference on Information Fusion*. IEEE, 2014.

[10] S. Blackman and R. Popoli, *Design and analysis of modern tracking systems*. Artech House, 1999.

[11] K. Choo and D. J. Fleet, "People tracking using hybrid Monte Carlo filtering," in *8th International Conference on Computer Vision (ICCV)*. IEEE, 2001.

[12] J. Dahlin, F. Lindsten, and T. Schon, "Particle Metropolis Hastings using Langevin dynamics," in *International Conference on Acoustics Speech and Signal Processing (ICASSP)*. IEEE, 2013.

[13] D. Reid, "An algorithm for tracking multiple targets," *Transactions on Automatic Control*, 1979.

[14] M. Bocquel, "Random finite sets in multi-target tracking. Efficient sequential MCMC implementation," Ph.D. dissertation, University of Twente, 2013.

[15] C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan, "An introduction to MCMC for machine learning," *Machine Learning*, 2003.

[16] Y. Boers and J. N. Driessen, "Multitarget particle filter track-before-detect application," in *Proceedings on Radar, Sonar and Navigation*. IEEE, 2004.

[17] Y. Zhang and C. A. Sutton, "Quasi-Newton methods for Markov chain Monte Carlo," in *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[18] T. Ding and M. J. Coates, "Implementation of the Daum-Huang exact-flow particle filter," in *Statistical Signal Processing Workshop*. IEEE, 2012.