Lattice Particle Filters

Christiane Lemieux* **Dirk Ormoneit†** **David J. Fleet‡**

*Department of Mathematics and Statistics, University of Calgary, Alberta, Canada
†Department of Computer Science, Stanford University, Stanford, CA 94305
‡Xerox Palo Alto Research Center, 3333 Coyote Hill Rd, Palo Alto, CA 94304

Abstract

A common way to formulate visual tracking is to adopt a Bayesian approach, and to use particle filters to cope with nonlinear dynamics and nonlinear observation equations. While particle filters can deal with such filtering tasks in principle, their performance often varies significantly due to their stochastic nature. We present a class of algorithms, called lattice particle filters, that circumvent this difficulty by placing the particles deterministically according to a Quasi-Monte Carlo integration rule. We describe a practical realization of this idea and discuss its theoretical properties. Experimental results with a synthetic 2D tracking problem show that the lattice particle filter yields a performance improvement over conventional particle filters that is equivalent to an increase between 10 and 60% in the number of particles, depending on their "sparsity" in the state-space. We also present results on inferring 3D human motion from moving light displays.

1 Introduction

Visual tracking and scene estimation are often formulated in computer vision as Bayesian filtering problems, for which *particle filters* (PF) have become a popular method [1, 2, 11, 16, 19, 21]. The PF approximates a desired probability distribution over the unknown state variables with a weighted particle set, and thereby provides a simple yet powerful approach to dealing with multimodal distributions, and nonlinear dynamics and observation equations [6, 8, 11, 15]. But despite its successes, the PF can be unstable when applied to complex, high-dimensional problems; statistically speaking, even though the PF produces *properly weighted* samples, the random variation of predictions based on these samples may be excessive and therefore the predictions may be unreliable. For visual tracking, this results in poor estimates of object location, and in more severe situations it may cause the algorithm to "lose track" of the object altogether.

This paper proposes the *Lattice Particle Filter* (LPF) as an alternative filter where particles are placed deterministically according to a *lattice rule*. Lattice rules are a subclass of *Quasi-Monte Carlo* (QMC) methods, which have been found useful for high-dimensional integration in computer graphics and finance [18, 3, 13]. An important theoretical advantage of QMC methods is that for N samples the error converges at the rate $O(N^{-1} \log^s N)$ as compared to $O(N^{-1/2})$ for traditional Monte Carlo (MC), where s is the dimension of the state space. From a practical viewpoint, *randomized* QMC methods can be used to construct unbiased estimators that have smaller variance than MC estimators.

After a brief introduction to the PF, we introduce QMC methods and then describe the lattice particle filter. We show quantitative results on two problems relevant to computer vision, namely, tracking 2D image patterns, and the inference of 3D human pose from a 2D binocular sequence of projected limb positions.

2 Previous Work

Consider a visual tracking problem where $\mathbf{x}_t$ denotes the unknown position of an object at time t, and $\mathbf{y}_t$ denotes the image observation at time t. In Bayesian filtering we are interested in computing the conditional distribution $p(\mathbf{x}_t \mid \mathbf{Y}_t)$, with $\mathbf{Y}_t \equiv (\mathbf{y}_t, ..., \mathbf{y}_1)$, which is also often called the *filtering distribution*. The PF is a method for Bayesian filtering for which $p(\mathbf{x}_t \mid \mathbf{Y}_t)$ is approximated by a set of weighted states (or particles). This approximation is updated recursively from one time step to the next as new observations become available. Gordon et al. [8] provide a clear description of the method which they call the *Bootstrap algorithm*. In computer vision it is well known as the *Condensation Algorithm* [11]. Other descriptions of the algorithm, along with a number of interesting extensions are given by Liu et al. [15], Doucet et al. [6], and Pitt and Shepard [20].

The goal of the particle filter is to approximate the filtering distribution $p(\mathbf{x}_t \mid \mathbf{Y}_t)$ with a set of samples. In most applications it is difficult to sample directly from $p(\mathbf{x}_t \mid \mathbf{Y}_t)$. Instead, using Bayes' rule and a Markov assumption, one can use the fact that the filtering distribution is proportional to the product of a *likelihood function*, $p(\mathbf{y}_t \mid \mathbf{x}_t)$, and a *prediction distribution* that summarizes the information from previous filtering steps, $p(\mathbf{x}_t \mid \mathbf{Y}_{t-1})$. The PF draws random proposals (states) from the prediction distribution, and then evaluates the likelihood function at each proposal; likelihood values serve as importance weights to account for the discrepancy between the true filtering distribution and the proposal from which the states were sampled.

In many vision applications the evaluation of the likelihood function dominates the computational cost of the algorithm. As a consequence the number of particles, and usually the dimensionality of the state space, should be kept relatively small. For example, in estimating 3D human motion, it has been useful to to design low-dimensional subspaces of the state-space within which to do the filtering [21]. One can also learn better models of the body's dynamics.

Of course, low dimensional representations may not exist for unconstrained motions, in which case the only way to improve the performance of the PF is the efficicent placement of particles. Towards this end MacCormick and Isard [16] used partitioned-sampling to track articulated objects. Importance sampling has been used to evaluate the current evidence multiple times in order to improve statistical efficiency of the sampling [6, 20]. It has also proved fruitful to use low-level visual detectors to seed the particle set with high likelihood states [2, 12].

This paper examines a Quasi-Monte Carlo (QMC) method as a general way to further reduce the number of samples. Similar approaches have only recently been suggested for filtering [7, 19], however, in these contributions, sampling techniques using non-randomized QMC methods are applied locally at each time step, without exploiting the global structure of the sampling scheme. Here we use a *randomized* QMC method, and this will allow us to get properly weighted samples that can be used to construct unbiased estimators in the same way the PF does. In addition, our choice of QMC point set and our ability to apply it to the global filtering problem imply that the quality of the sampling remains the same throughout time.

3 Bayesian Inference in Dynamical Systems

Before describing the lattice particle filter, we review the particle filter and the filtering equations. As mentioned above, let $\mathbf{x}_t$ denote the unknown state variable at time t, and let $\mathbf{y}_t$ denote the image observation at time t. The *transition model* characterizes the state dynamics; with a first-order Markov model it takes the form $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. The *observation model* specifies the probabilistic relation between the state and the visual appearance of the object in the image; this *likelihood function* takes the form $p(\mathbf{y}_t|\mathbf{x}_t)$. With these models, the filtering distribution $p(\mathbf{x}_t|\mathbf{Y}_t)$ is given by

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{Y}_t) &= \int\!...\!\int p(\mathbf{x}_t, ..., \mathbf{x}_0|\mathbf{Y}_t) d\mathbf{x}_1 ... d\mathbf{x}_{t-1} \\ &\propto \int\!...\!\int \prod_{s=1}^{t} p(\mathbf{y}_s|\mathbf{x}_s) p(\mathbf{x}_s|\mathbf{x}_{s-1}) d\mathbf{x}_s . \quad (1) \end{aligned}$$

The direct evaluation of (1) is difficult in practice because it involves an integral whose dimension grows with t. Hence it is advisable to solve (1) recursively using the *prediction* and *filtering* equations:

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{Y}_{t-1}) &= \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{Y}_{t-1}) d\mathbf{x}_{t-1} \quad (2) \\ p(\mathbf{x}_t|\mathbf{Y}_t) &\propto p(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{Y}_{t-1}). \quad (3) \end{aligned}$$

If the transition and observation models are linear with Gaussian noise, then (2) and (3) are also Gaussian and can be updated using the Kalman filter [9]. Otherwise, the representation and the computation of (2) and (3) can be difficult. One approach is to approximate (3) with a weighted set of particles,

$$\mathcal{S}_t = \{(\mathbf{x}_{t,i}, w_{t,i}) \,|\, i = 1, \ldots, N\}, \quad (4)$$

such that $\mathcal{S}_t$ approximates $p(\mathbf{x}_t|\mathbf{Y}_t)$ as well as possible. This idea can be made precise by requiring that $\mathcal{S}_t$ be *properly weighted* [15] in the sense that

$$\sum_{i=1}^{N} w_{t,i} f(\mathbf{x}_{t,i}) \overset{N\to\infty}{\longrightarrow} E[f(\mathbf{x}_t|\mathbf{Y}_t)] \quad (5)$$

for arbitrary integrable functions f. This is an important property of the approximation methods described below.

The PF generates properly weighted samples [15] and it encourages the exploration of paths $(\mathbf{x}_{1,i}, \ldots, \mathbf{x}_{t-1,i})$ that appear likely given $\mathbf{Y}_{t-1}$, hence discarding particles with small weights from (4). The PF can be described by:

$$\begin{aligned} a(i) &\sim \text{multinomial}(w_{t-1,1}, \ldots, w_{t-1,N}), \quad (6) \\ \mathbf{x}_{t,i} &:= g(\mathbf{u}_{t,i}, \mathbf{x}_{t-1,a(i)}), \quad (7) \\ w_{t,i} &:= \frac{p(\mathbf{y}_t \mid \mathbf{x}_{t,i})}{\sum_{k=1}^{N} p(\mathbf{y}_t \mid \mathbf{x}_{t,k})}. \quad (8) \end{aligned}$$

The index sequence generated in step (6) specifies the "surviving" particles, $\mathbf{x}_{t-1,a(i)}$, $i = 1, ..., N$. These particles are then propagated forward in step (7), and step (8) specifies the weights to be associated with each new particle. The points $\mathbf{u}_{t,i}$ are independent and $g(\mathbf{u}_{t,i}, \mathbf{x}_{t-1,i})$ is a *transformation function* that maps the uniform vector $\mathbf{u}_{t,i}$ and the previous state $\mathbf{x}_{t-1,i}$ onto an sample from $p(\mathbf{x}_t|\mathbf{x}_{t-1,i})$. This transformation is a component of any computer simulation since random variate generation for many continuous distributions are based on transformations of uniform random numbers [5]. For example, in the case of a univariate standard normal transition model, $g(\mathbf{u}_{t,i}, 0)$ would be the inverse standard normal distribution function.

The LPF provides an improved way to execute the filtering steps (6) – (7). The idea is to draw points $\mathbf{u}_{t,i}$ from a lattice rule instead of using random sampling. An important aspect of both the PF and the LPF is that particles evolve in

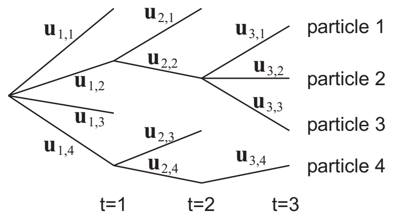

Figure 1. Tree associated with the particle filter for $N = 4$.

a tree structure in which "branches" die when their associated particle is not chosen in the resampling step. Figure 1 depicts the tree structure and shows how points $\mathbf{u}_{t,i}$ are used to augment branches from $\mathbf{x}_{t-1,a(i)}$ to $\mathbf{x}_{t,i}$ at each time.

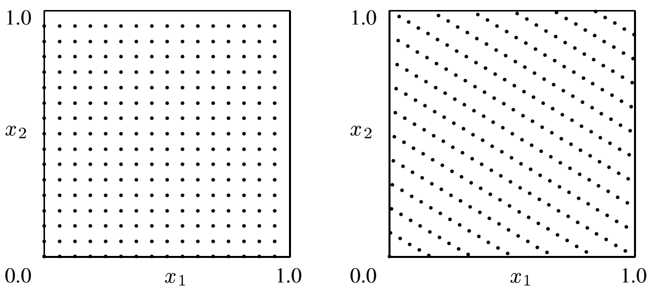

Figure 2. *Left:* Rectangle rule. *Right:* Korobov lattice rule with $N = 256$ and $a = 25$. The projection of the points on each axis results in N distinct values in this case.

4 Lattice Particle Filter

The PF described in Section 3 produces properly weighted samples that can be used to approximate the filtering distribution (1). However, the PF's performance may vary considerably in practice due to the random nature of the sampling. This is sometimes particularly harmful in the context of filtering because, even though the approximation error due to sampling at each step may be small, errors can accumulate over time in an exponential manner.

To see this, consider a simple example where the transition density $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is uniform on the state-space $[0, 1)$ and independent of $\mathbf{x}_{t-1}$. Let the observations be binary with $P(\mathbf{y}_t = 1|\mathbf{x}_t)$ being 1 if $\mathbf{x}_t < 0.2$ and 0 otherwise. That is, the system states are placed independently on the unit interval, but only those states in $[0, 0.2)$ trigger the response $\mathbf{y}_t = 1$ in the observations. Suppose furthermore that the *true* state trajectory of $\mathbf{x}_t$ happened to evolve entirely within $[0, 0.2)$ so that $\mathbf{y}_t = 1$ for all t, and that we use a PF with $N = 10$ particles to recover this trajectory. At each time step the number of particles within $[0, 0.2)$ is binomially distributed with parameters 10 and 0.2. Hence the probability of having no particles within $[0, 0.2)$ in any one of k time steps is $(1-(1-0.8^{10})^k)$, which converges to one exponentially with k. In other words, after a sufficiently long time there almost certainly occurs one time where the PF lost track of the object completely.

In this example, an alternative strategy would have been to choose only one particle at random at each time, and to place the remaining particles equidistantly around that particle. Then there would always be two particles in $[0, 0.2)$. This deterministic placement of particles according to a *low-discrepancy rule* is a special case of a QMC method. We next provide some general background on QMC methods and lattice rules, and then describe the LPF.

4.1 Quasi-Monte Carlo Methods

Consider a generic integration problem with respect to the uniform measure over the domain $[0, 1)^s$. There are several was to approach this task: MC integration amounts to averaging the values of the integrand at N random, uniform, independent points in $[0, 1)^s$. Its error has a convergence rate of $O(N^{-\frac{1}{2}})$ that is independent of s. However, the constant terms in the approximation error are typically quite large, which makes the algorithm too slow for many applications. A deterministic alternative might be to use a Cartesian product of one-dimensional point sets, as shown in Figure 2(left) for $s = 2$. The problem with this approach is that the number of points N needed to preserve a constant level of integration accuracy grows exponentially fast with the dimension s. This is caused by the poor projections of such point sets onto lower-dimensional subspaces; for example, note that in Figure 2(left), the 256 points project to only 16 points along each axis to integrate the x_1 and x_2 components of an integrand of the form $f(x_1, x_2)$.

By contrast, QMC methods aim to create point sets whose different projections onto coordinate subspaces always contain N distinct points. An overview of random number generation and QMC methods is given in the book by Niederreiter [17]. An example of a 2D QMC point set, called a Korobov lattice rule, is shown in Figure 2(right).

A second goal of QMC methods is to create point sets that are as close as possible to a uniform distribution. More precisely, QMC methods are based on so-called *low-discrepancy point sets* which give convergence rates of $O(N^{-1} \log^s N)$ for integration error, in contrast to the MC method that is $O(N^{-\frac{1}{2}})$. In practice, the performance of QMC integration is intimately related to the *effective dimension* of the integration problem, i.e., the ability to approximate the integrand by a sum of low-dimensional functions [18, 3]. Integrands having a small effective dimension can be integrated accurately by QMC point sets that have good projections over subspaces of low dimension. Superior performance of QMC over MC has been demonstrated in numerous applications [18, 13, 3].

As a special case of QMC methods we consider *lattice rules* here [22, 10, 14]. By comparison to the Sobol and Niederreiter sequences [17] used in [19], lattice rules are easy to implement, and can be built so that their projections onto low-dimensional subspaces are not only well distributed but also of equivalent quality for different sub-

$\log_2 N$	4	5	6	7	8	9
a_l	3	5	11	13	25	55
a_h	3	5	5	11	75	51
$\log_2 N$	10	11	12	13	14	15
a_l	43	259	307	699	2087	7243
a_h	139	519	1081	1289	2961	2149
$\log_2 N$	16	17	18	19	20	21
a_l	11035	27891	18373	21643	201579	431119
a_h	21553	27383	3597	120079	172565	232501

Table 1. Generators for dimensions up to 8, a_l, or problems with dimension up to 32, a_h.

spaces. In addition they can be randomized to produce unbiased estimators.

To build the LPF, we use a special construction of lattice rule called a *shifted Korobov lattice rule* that contains N points $\mathbf{U}_1, \ldots, \mathbf{U}_N$ given by

$$\mathbf{U}_i = \left(\frac{i-1}{N}(1, a, \ldots, a^{s-1}) + \mathbf{\Delta} \right) \bmod 1 \;, \quad (9)$$

for $i = 1, ..., N$, where the modulo 1 is applied componentwise. The integer $a \in [1, ..., N-1]$ is called the *generator* of the rule; its choice is crucial for the performance. A table of suitable values for a, depending on the sample size N and the state dimension, are given in Table 1 (see [14] for more details). The vector $\mathbf{\Delta}$ is a shift that is uniformly distributed over $[0,1)^s$, which implies that the points $\mathbf{U}_i$ are also uniform, but *dependent*. This shift is important as a means of obtaining error estimates via multiple simulations [4], and it is necessary to guarantee that the resulting approximations are *unbiased*.

4.2 LPF Algorithm

The LPF algorithm propagates N particles following the rules (6), (7), and (8). However, the uniform numbers $\mathbf{u}_{t,i}$ in the forward propagation (7) become components of the (shifted) lattice rule (9).

More precisely, we start with a shifted lattice rule $P_N = \{\mathbf{U}_i, i = 1, \ldots, N\}$ in nT dimensions, and we decompose P_N into components of the form $\mathbf{u}_{t,1}, \mathbf{u}_{t,2}, \ldots, \mathbf{u}_{t,N}$ for each time t. Note that, due to (9), these components can be computed recursively for $t = 1, ..., T$, without storing P_N explicitly. Next, we use $\mathbf{u}_{t,1}, \mathbf{u}_{t,2}, \ldots, \mathbf{u}_{t,N}$ for forward propagation in step (7). An important issue is the assignment of a particular uniform number $\mathbf{u}_{t,j}$ to the uniform variables $\mathbf{u}_{t,i}$ in (7). There is a danger of assigning components to particles that might depend on the outcome of the resampling, hence introducing a serious bias. Our approach is to generate at each time t a uniform and random permutation $\Gamma_t = (\gamma_{t,1}, \ldots, \gamma_{t,N})$ of the integers $[1 \ldots N]$, and then assign the point $\mathbf{u}_{t,\gamma_{t,i}}$ to particle i. This implies that the trajectory $\mathbf{x}_{1,i}, \ldots, \mathbf{x}_{t,i}$ of the i^{th} particle from time 1 to t is generated by the point

$$\tilde{\mathbf{U}}_{t,i} = (\mathbf{u}_{1,\gamma_{1,i}}, \mathbf{u}_{2,\gamma_{2,i}}, \ldots, \mathbf{u}_{t,\gamma_{t,i}}). \quad (10)$$

Note that the point set $\{\tilde{\mathbf{U}}_{t,1}, \ldots, \tilde{\mathbf{U}}_{t,N}\}$ does not form a lattice rule, which would be neccessary to obtain the $O(N^{-1}\log^s N)$ convergence rate *globally*. However, the point set used at each time t forms a lattice rule. In addition, it can be shown [14] that when $\gcd(a, N) = 1$, the point sets used at each time step differ in the implementation only because of the random shift. Hence the quality of the sampling remains the same throughout time, which is generally not the case for other types of QMC point sets such as those used in [19].

In summary, the LPF implementation only differs from the PF in the use of the points $\mathbf{u}_{t,i}$ used to generate the state $\mathbf{x}_{t,i}$. Instead of calling a pseudorandom number generator n times to define $\mathbf{u}_{t,i}$, the LPF instead generates *one* random shift Δ, one random permutation $\Gamma_t = (\gamma_{t,1}, \ldots, \gamma_{t,N})$ of the integers from 1 to N, and then use the point

$$\left(\frac{\gamma_{t,i}-1}{N}(1, a, \ldots, a^{n-1}) + \Delta \right) \bmod 1$$

to generate the state $\mathbf{x}_{t,i}$ of the i^{th} particle. More details and pseudo-code are provided in a technical report accompanying this paper, and we will make a version of the C code available on the internet.

5 Theoretical Aspects

In this section we discuss some theoretical results and we outline differences and similarities between the LPF and the ordinary PF. In terms of search methods, one can view the ordinary PF as an informed, random search where different nodes of a search tree are expanded or truncated at different time steps (see Fig. 1). Conversely, the LPF corresponds to an (almost) deterministic search algorithm. The observation model is used as a heuristic function to determine which "node" (i.e. branch of the filtering tree) is expanded next, and the transition model specifies how to carry out this expansion. In the LPF the expansion is designed to search the state space as *evenly* as possible.

From a mathematical perspective, we can show that the samples generated by the LPF are properly weighted in the sense of (5). This is suggested by the fact that the updating of the weights $w_{t,i}$ and the sampling with replacement in the LPF are done exactly as in the PF. The only difference is that the particles $\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,N}$ are not independent, and hence the LPF requires a slightly more sophisticated mathematical treatment. We formalize this intuition in the following theorem (proof omitted due to space limitations), which covers the LPF as a special case:

Theorem 1 *If each particle $\mathbf{x}_{t,i}$ is generated by a point $\mathbf{u}_{t,i}$ that has a uniform distribution over $[0,1)^n$ and such that $\mathbf{u}_{t,i}$ and $\mathbf{u}_{s,j}$ are independent for $1 \le s < t \le T$, $1 \le i, j \le N$, and if the resampling step at time t is done by sampling with replacement using weights $w_{t,i}$ proportional to $p(\mathbf{y}_t|\mathbf{x}_{t,i})$, then $\mathcal{S}_t = \{\mathbf{x}_{t,i}, w_{t,i}, i = 1, \ldots, n\}$ is a properly weighted sample, for $t = 1, \ldots, T$.*

Hence, at least asymptotically, the LPF produces a correct approximation of the filtering distribution (3), and furthermore, the quantity $\sum_{i=1}^{N} w_{t,i} f(\mathbf{x}_{t,i})/N$ is an unbiased estimator of $E(f(\mathbf{x}_t|\mathbf{Y}_t))$. Note that the random shift in (9) is essential to this result and is a principal difference between the LPF and previous work on QMC-based filtering where samples are generated deterministically, e.g. [19].

6 Experiments

To test the LPF in practice we compare it to a conventional PF using residual sampling [15]. Both filters observed the same data, had the same temporal dynamics (transition density), and the same likelihood function (observation density). The goal of the experiments is to compare the reliability with which the different filters provide an approximation to the filtering distribution (3). The quality of this approximation is measured by evaluating the mean of the filtering distribution. In particular, the deviation from the mean of the true posterior across many runs of the filter should be lower for the better filter. By applying the filters with varying numbers of particles we can also analyze the relationship between estimation errors and the computation time required by both algorithms. Computational requirements in both algorithms scale in almost identical ways with the number of particles.

6.1 Disk Tracking

The first experiment involves a circular disk undergoing a random walk. The center position of the disk at time t, $\mathbf{x}_t \in I\!R^2$, is given by

$$\mathbf{x}_t \;=\; \mathbf{x}_{t-1} \;+\; \eta_t, \tag{11}$$

where η_t is 2D i.i.d. zero-mean Gaussian noise; i.e., $\eta_t \;\sim\; N(\bar{0}, \sigma_x^2 \mathcal{I}_2)$, where $\mathcal{I}_2$ is the 2×2 identity matrix. We used a standard deviation of $\sigma_x = 3.0$ pixels.

The effective scene model $m(\mathbf{s}, t)$ is simply 1 if pixel $\mathbf{s}$ is within a disk of radius 16 pixels, centered at $\mathbf{x}_t$:

$$m(\mathbf{s}, t) \equiv \begin{cases} 1 & \text{for } \|\mathbf{s} - \mathbf{x}_t\| \le 16 \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

The image observation is constructed by adding i.i.d. zero-mean Gaussian noise $\nu_{t,\mathbf{s}} \sim N(0, \sigma_\nu^2)$ to $m(\mathbf{s}, t)$:

$$\mathbf{I}(\mathbf{s}, t) \equiv m(\mathbf{s}, t) + \nu_{t,\mathbf{s}} \tag{13}$$

with standard deviation $\sigma_\nu = 0.25$.

The image $\mathbf{I}_t$ is then interpreted as a vector of observations, $\mathbf{y}_t$, in the state-space model of Section 3. The transition density used is

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) \equiv N(\mathbf{x}_t - \mathbf{x}_{t-1}, \sigma_d^2 \mathcal{I}_2)\;. \tag{14}$$

We intentionally misspecified the model (14) by choosing the standard deviation $\sigma_d = 5$, which is about 60% larger

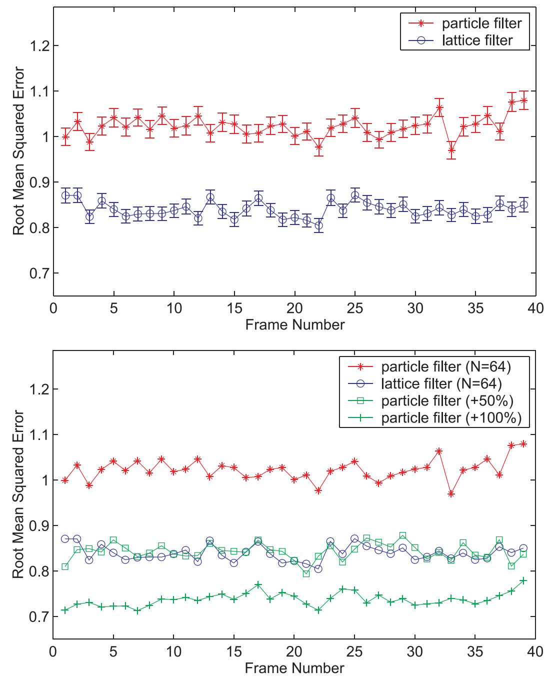

Figure 3. Root mean-squared errors (RMSE) in the estimated disk location for a PF with residual sampling and for the LPF. (TOP) RMSE with confidence levels based on 64 samples. (BOTTOM) RMSE for PF's with different sample sizes; with 50% more samples, the PF produces errors similar in magnitude to the LPF.

than the standard deviation in the "true" transition model (11). This is because the model parameters of the transition model are typically unknown in practice, and hence a relatively large value must be chosen to search a sufficiently large neighborhood of $\mathbf{x}_t$. By contrast, an accurate physical model of the observation model is available in many cases, and therefore the likelihood function is derived from (13).

We created 1000 image sequences, each with 40 frames, by simulating equations (11), (12), and (13). We then applied the PF and the LPF to each sequence, initialized by setting $\mathbf{x}_0$ to the true disk location, to track the disk location, $\mathbf{x}_t$. The disk location was estimated by taking the weighted mean of the particles, according to (5). This gives 1000 error sequences where the error at each time is simply the Eucledian distance between the estimated and the true disk location. As a summary statistic, we compute the root mean-squared prediction error (RMSE) of the 1000 trials at each time step.

We report results using 16, 32, 64, 128, 256, and 512 particles. The RMSE results of the experiment with 64 particles is shown in Figure 3 as a function of time. The blue circles represents the RMSE of the LPF and the other curves represent the performance of the ordinary PF for different numbers of particles. Figure 3 (top) compares the LPF to the PF with identical numbers of particles, showing RMSE

# particles	16	32	64	128	256	512
difference in Var	20%	21%	19%	11%	11%	10%
efficiency gain	50%	60%	50%	15%	20%	20%

Table 2. Experimental Results in the Disk Tracking Task

and standard error bars that provides a measure of our confidence in this expected deviation from the true disk location. The LPF produces location errors that are approximately 20% less than the PF with the same number of samples. This supports our hypothesis that the deterministic placement of particles improves the average performance of the estimate. In addition, although difficult to discern from this plot, the standard error bars from the LPF are also smaller.

Figure 3 (bottom) shows the same RMSE results for the LPF but compared against RMSE results for three PFs with different numbers of particles. This allows one to assess the computational gains due to the LPF; i.e., difference in the required number of particles to keep the expected errors equal. One can see that we need to increase the number of particles, and hence the computation time, for the PF by approximately 50% to obtain a performance comparable to that of the LPF. Even with 50% more particles the performance of each individual trial may still be worse for the PF due to the larger error bars.

Finally, these results hold over a relatively wide range of particle sizes. Figure 4 shows other experiments with from 16 particles up to 512 particles. The experimental setup is identical to that in Figure 3. Note that the average prediction error of both methods, as well as the variability of the predictions, measured in terms of the error bars, decreases with an increasing number of particles. A summary of the results from the complete set of experiments which produced similar results is given in Table 2.

Note that the LPF consistently outperforms the PF by a margin of at least 10%. The performance improvement seems to be less pronounced in the cases where we have relatively many particles (128, 256, and 512). This is to be expected because an optimized particle placement seems particularly relevant in the case where there are relatively few particles. Regarding the percentage of additional particles needed by the PF to match the performance of the LPF, which is reported in the third row of the table, the maximum difference of 60% occurs at a sample size of 32 particles for the LPF. Again, this difference becomes less pronounced as the overall number of particles increases.

6.2 Human Motion Tracking

In our second set of experiments, we apply both the LPF and the PF using residual sampling to a human motion tracking problem, where the task is to recover human body poses from 2D pose projections. That is, rather than using camera images as observations, we feed projections of the true locations of body markers (joints) into the system as input information. However, we only provide partial information to the filtering algorithm by using a single 2D

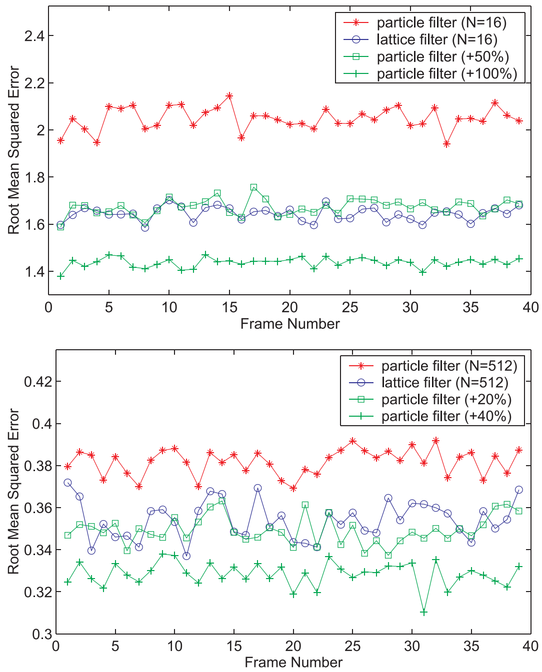

Figure 4. Root mean-squared errors (RMSE) in the estimated disk location for PF's with residual sampling and for the LPF. The number of samples for the different filters are show in the legends in each plot.

projection of each marker. This filtering problem is particularly challenging because the human body pose is described in terms of a 10 dimensional angular space, and the tracking occurs directly on this space without first applying algorithms for dimensionality reduction like principal component analysis [21].

The image observations are labeled 2D positions corresponding to markers on a human subject, the 3D locations of which were found with a commerical motion capture system. The 6 markers used for tracking an individual leg or the lower body are shown in Fig. 5. The observations were generated by projecting the 3D points onto the image plane and then adding 2D Gaussian noise. Given a camera center at (X_c, Y_c, Z_c), with the optical axis parallel to the Y-axis, a marker point (X_m, Y_m, Z_m) produces the observation

$$\mathbf{d} = \left(\frac{X_m - X_c}{Y_m - Y_c}, \frac{Z_m - Z_c}{Y_m - Y_c} \right) + \eta \qquad (15)$$

where η is isotropic 2D mean-zero Gaussian noise with variance σ^2. In the experiments below we used $\sigma = 0.002$, which is approximately 2% the length of an upright spine projected into the image plane.

The body is modeled as a hierarchical, articulated model, with the spine as the root. The the lower body model (i.e. the state vector) has 10 angular degrees of freedom (DOF): 2 DOFs for the pevis, 3 DOF joints for the hips, and 1 DOF joints for the knees. Given a state, $\mathbf{s}$, one can simply traverse

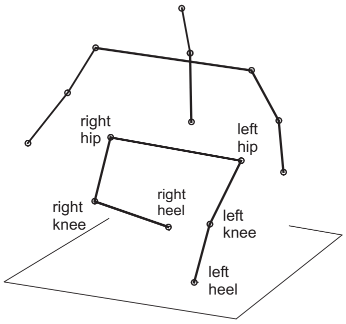

Figure 5. Locations of 6 markers on the lower body.

the kinematic tree, from the spine to the pelvis, and then to each leg, to generate the 3D marker locations. These 3D locations are then projected into the image to generate predicted observations from the model $\{\mathbf{q}_j\}_{j=1}^6$. Let $\hat{\mathbf{q}}_j(\mathbf{s})$ be the 2D location for the j^{th} marker that is predicted by state $\mathbf{s}$. With this and (15), we can write the likelihood function for state $\mathbf{s}$ and observations $D = \{\hat{\mathbf{q}}_j\}_{j=1}^6$ as

$$p(D \mid \mathbf{s}) = \left(\frac{1}{2\pi\sigma^2}\right)^6 \exp\left[-\frac{\sum_{i=1}^6 ||\mathbf{q}_j - \hat{\mathbf{q}}_j||^2}{2\sigma^2}\right] . \quad (16)$$

where σ^2 is the variance of the image noise.

The experiments below simulate a binocular observer that views the person from approximately 2.5m, with approximately 6cm between the two eyes. Since independent noise is added to the image observations from the two views, the joint likelihood is the product of the individual likelihoods, as given in (16). For a temporal prior over these state variables, we make the overly simplistic assumption that the states change slowly over time. In particular, the temporal prior for states $\mathbf{s}_t$ given $\mathbf{s}_{t-1}$ is an isotropric, mean-zero Gaussian, with a variance of σ_a^2. In the experiments reported below, we fix $\sigma_a = 0.1$ rad. (almost 6 deg.).

Finally, note that this model is quite weak in that we do not model limits on how far joints can rotate in practice. The tracker may therefore draw high-likelihood proposals that are anatomically impossible but are consistent with the 2D observations. Also, limb lengths are determined from the dimensions of the human subject before tracking begins and are then fixed during tracking. This is a source of error as real joints are more complicated than our model, causing some parts such as the pelvis to vary in width over time.

Comparing the performance of the LPF and PF is more difficult in this case. As the input is noisy, the true 3D structure of the human figure is not well constrained at every frame. As a consequence, the mean of the posterior is not, in general, equal to the true state. As the filters are attempting to approximate the posterior distribution, an appropriate measure of perforance is the difference between the true posterior mean state and the means states computed using the approximations provided by the LPF and the PF.

To extract the mean of the true posterior, we ran the PF on the input sequences with 10^3 more samples than we used in the experiments that follow. This was more than enough samples to have confidence that we found the true mean. We then ran the LPF and the PF 200 times on the same input, with different random seeds on every run. The means on each run were computed according to (5).

Figure 6 summarizes the results for individual state variables of the left leg, namely, two angles from the left hip, and one for the left knee. Notice from Fig. 6 (top) that the average of the mean estimates obtained over the 200 trials is very close to the posterior mean in both cases. This is not surprising since both are the LPF and the PF produced unbiased estimates; in part this shows that the LPF is working as expected. The error bars in Fig. 6 (top) show the standard deviation of the ensemable of means for the 200 runs, which is where we expect the LPF to show smaller variability of the estimates about the true posterior mean. Like the case with the disk tracker, the standard deviation for the LPF is usually 5% - 20% below that for the particle filter, with similar gains in computational efficiency.

Figure 6 (bottom) shows the expected (absolute) difference between the true posterior mean and the mean estimates obtained with the LPF and the PF. The error bars are larger than in Fig. 3 because we have only 200 instead of 1000 trials. The lower errors with the lattice method are clear nonetheless. Finally, note that although the variance reduction in individual states is relatively modest, reducing the variance in each of 10 dimensions can significantly reduce the volume of the 10-D search space that will need to be explored by the filter at each time step.

7 Conclusions

We presented an algorithm called the lattice particle filter to improve the reliability of particle filters in visual tracking. Specifically, the method places the particles deterministically according to a randomly shifted lattice rule, which reduces the variance of the particle-based approximation. Experiments demonstrate that the practical improvement from using the LPF amounts to savings in the number of particles of between 20% and 60%. The size of this effect depends on the "sparsity" of the particles in the state-space, being more pronounced when there are relatively few particles as it is typical for real applications. In our experiments involving a 10-d human motion tracker, the lattice particle filter also led to similar reductions in variance.

For future work, we plan alternative versions of the LPF that are based on a global lattice rule in nT dimensions. We believe that the global lattice property will be central to obtain exact convergence rates and to optimize the practical performance of the LPF. However, it involves the creation of particles satisfying a global lattice constraint. We will also explore alternative resampling methods that could, e.g., depend on the variability of the likelihoods obtained at each time step, such as those in [6, 15].

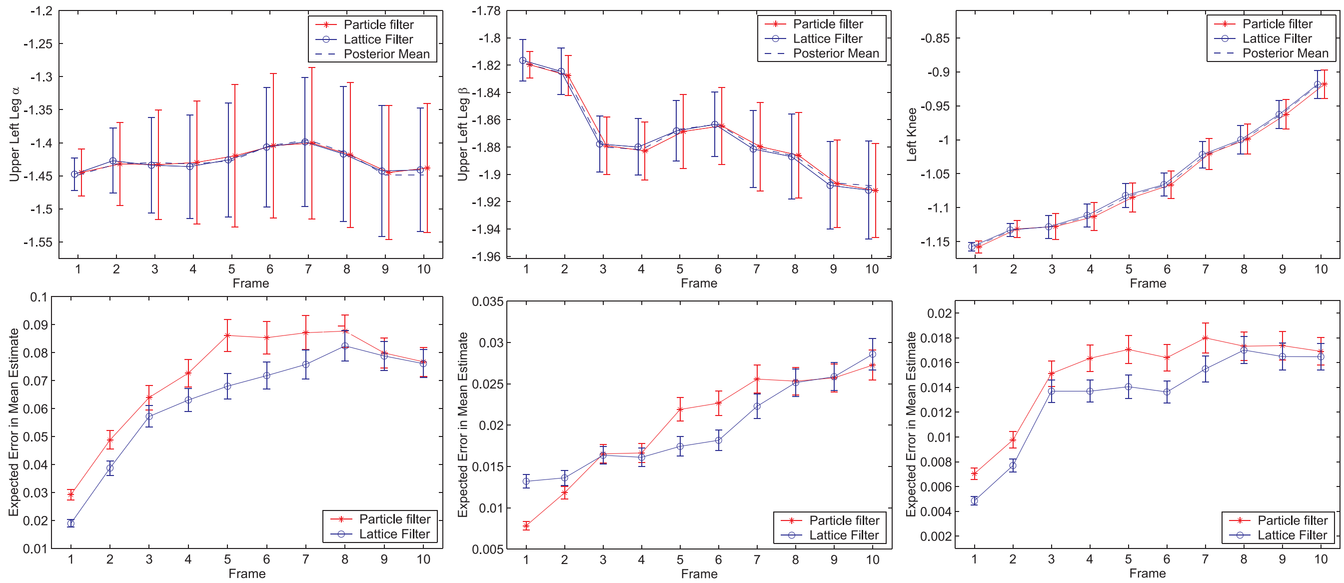

Figure 6. Summary of 200 trials of PF and LPF tracking of lower body. (Top) Individual state variables for the left leg, for which the LPF means are generally is closer to the posterior mean, and have significantly less variance. (Bottom) Expected difference between the posterior mean and the means computed by the PF and the LPF.

References

[1] M.J. Black. Explaining optical flow events with parameterized spatiotemporal models CVPR, Fort Collins, pp. 326–332, 1999.

[2] M.J. Black and D.J. Fleet. Probabilistic detection and tracking of motion discontinuities. *IJCV*, 38:229–243, 2000

[3] R. E. Caflish, W. Morokoff, and A. Owen. Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension. *J. Comp. Finance*, 1:27–46, 1997.

[4] R. Cranley and T. N. L. Patterson. Randomization of number theoretic methods for multiple integration. *SIAM J. Num. Anal.*, 13:904–914, 1976.

[5] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New-York, 1986.

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

[7] P. Fearnhead. *Sequential Monte Carlo Methods in Filter Theory*. PhD thesis, University of Oxford, 1998.

[8] N.J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. Radar, Sonar and Navig.*, 140:107–113, 1993.

[9] J. D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[10] F. J. Hickernell. Lattice rules: How well do they measure up? In P. Hellekalek and G. Larcher (eds), *Random and Quasi-Random Point Sets*, *Lecture Notes in Statistics*, Vol. 138, pp. 109–166. Springer, NY, 1998.

[11] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *IJCV*, 29:2–28, 1998.

[12] M. Isard and A. Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. ECCV, pp. 893–908, Freiburg, 1998.

[13] A. Keller. Quasi-Monte Carlo methods in computer graphics: The global illumination problem. *Lectures in Appl. Math.*, 32:455–469, 1996.

[14] P. L'Ecuyer and C. Lemieux. Variance reduction via lattice rules. *Management Science*, 46:1214–1235, 2000.

[15] J.S. Liu and R. Chen. Sequential monte carlo methods for dynamic systems. *JASA*, 93:1032–1044, 1998

[16] J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. ECCV, Dublin, Vol II, pp. 3–19, 2000.

[17] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*, Vol 63 of *SIAM CBMS-NSF Regional Conf. Series in Appl. Math.*. SIAM, Philadelphia, 1992.

[18] S. Paskov and J. Traub. Faster valuation of financial derivatives. *J. Portfolio Management*, 22:113–120, 1995.

[19] V. Philomin, R. Duraiswami, and L. Davis. Quasi-random sampling for condensation. ECCV, Vol II, pp. 134–149, Dublin, 2000.

[20] M. K. Pitt and N. Shepard. Filtering via simulation: Auxilary particle filters. *JASA*, 94:590–599, 1999.

[21] H. Sidenbladh, M. Black, and D. Fleet. Stochastic tracking of 3d human figures using 2d image motion. ECCV, Vol II, pp. 702–718, Dublin, 2000

[22] I. H. Sloan and S. Joe. *Lattice Methods for Multiple Integration*. Clarendon Press, Oxford, 1994.