Performance and Complexity Analysis of Adaptive Particle Filtering for Tracking Applications*

Miodrag Bolić, Sangjin Hong, and Petar M. Djurić
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, NY 11794-2350
{mbolic, snjhong, djuric}@ece.sunysb.edu

Abstract

This paper provides a performance and complexity analysis of particle filtering as applied to real-time object tracking. The number of particles and the sampling rate strongly influence the performance of particle filters, but more importantly they affect very much their complexity. In the paper, we also propose a particle filter that changes the number of used particles during filtering, where the number of particles is employed for making decisions about performing resampling. The performance of the proposed particle filters is demonstrated on the bearings-only tracking problem.

1 Introduction

Particle filters [2] are used in non-linear problems where the interest is in tracking of dynamic signals. It has been shown that the number of particles is very important in determining the performance of particle filtering. The performance of tracking, besides on number of particles, depends on the rate of acquiring new samples, which itself is dependent on the characteristics of the tracked object. The number of particles and the processing rate directly affect the hardware complexity of the implemented filter. For hardware efficient implementations in tracking applications, it is imperative to use as few particles as possible at any given time. Such minimization of the number of particles can be accomplished by dynamically varying the number of particles using the processed observations.

Some existing techniques for changing the number of particles during filtering include the Kullback-Leibler distance (KLD) sampling and the likelihood based adaptation [4, 6]. These methods are based on the idea of propagating small number of particles when the density is concentrated in a small region (i.e., in presence of small uncertainties) of the state space and propagating large number of particles in cases of high uncertainties. For hardware implementation, the KLD sampling method is computationally too intensive. In the likelihood based adaptation method, new particles are generated until the sum of non-normalized likelihoods exceeds a pre-specified threshold. With the restriction of minimum and maximum numbers of particles, this method is much simpler to implement.

A problem with the likelihood based adaptation is that some importance weights can be too large which could stop generation of new particles. This leads to inaccurate estimates in the bearings-only tracking problem (note that there the position error may be high, and yet the position can be compatible with the measured bearing). Another problem of the likelihood based adaptation is that it introduces data dependencies in the sampling and importance steps making them more difficult for parallel hardware implementation.

Another important issue is the resampling of the particles of the filter. There are several methods for assessing if resampling is necessary during particle filtering [1, 7]. In [7], the variance of the weight coefficients is proposed for that purpose, but this method introduces additional computational requirements. Methods where particles below a certain threshold are rejected have been proposed in [8]. Such methods are not suitable for real time VLSI implementations because the execution time itself is a random variable. Here we propose a method that decides if resampling is necessary based on the current number of particles of the filter.

2 Analysis of tracking with particle filters

In this section we analyze the sample importance resampling (SIR) filter applied to the bearings-only

*This work has been supported under Awards CCR-9903120 and CCR-0220011.

tracking (BOT) problem [5]. In brief, we have found that the performance of the particle filtering ceases to improve when the number of particles is greater than a certain number N. This number depends on the problem (object's trajectory, parameters that describe the dynamics of the object) and the initial point in the BOT problem. From our initial investigations, increased number of particles helps the reduction of lost tracks but does not improve the performance in terms of mean square error (MSE) (provided the number of particles is greater than N).

In the case when the likelihood is computed in regions where it has very small values, it could happen that all the importance weights are negligible. Then, we say that the track is lost. It has been noticed that particle filters lose tracks when there are significant deviations from the previous estimate. Figure 1 illustrates the percentages of cases when all importance weights have values close to zero versus the used number of particles. The starting point has the same mean as the actual starting point of the object, and the other parameters in the simulation model are the same as in [5]. As shown in the figure, the loss can be reduced by increasing the number of particles.

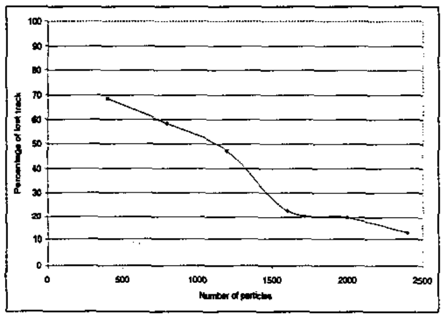


Figure 1: Percentage of cases when all weights have zero values versus number of used particles.

Particle filters must be designed in a way to avoid the loss of tracking. This problem can be addressed in several ways:

1. increased number of particles,
2. increased sampling rate,
3. change of the system variance, and
4. conversion of the model from a rectangular to polar coordinate system [3]. This solution is computationally more intensive, but it is less sensitive to large deviations of the estimates.

To reduce the complexity of particle filtering, it is necessary to operate with minimal computations. From the standpoint of implementation, it is not practical to keep large number of particles or high sampling rates to prevent the loss of tracks. A much more practical approach would be to adjust the sampling rate or the number of particles whenever necessary.

2.1 Effect of sampling rate

It is common in many DSP problems that the sampling rate of the input affects the performance. This is also true when the object movement is correlated with the sampling rate. When the object moves fast, it is advantageous to sample the input at a higher rate in order to avoid loss of tracks. Note that changing of the sampling rate requires changes in the parameters of the model. These changes in some cases may lead to deterioration in performance. Simulation results show that much less gain than expected is obtained when the sampling rate is increased.

2.2 Change of system variance

Up to this point, it was considered that the importance sampling density is the prior density of the state. Although the SIR filter with the prior as the importance density is easier to implement, it does not adapt very well to drastic changes in the system. Here, we are proposing simple modifications of the importance sampling density with the goal of improving the performance of the particle filter in presence of fast changes in the bearing angle.

The main idea is to have the importance sampling density π of the form of the prior density π_1 and/or the density π_2 with larger standard deviation $\sigma_2 > \sigma_1$. Two methods have been simulated:

1. $\pi = \pi_1$ if the particle filter tracks well, and $\pi = \pi_2$ otherwise. The criteria for the quality of tracking are described in Section 3.
2. $\pi = \lambda\pi_1 + (1-\lambda)\pi_2$, where λ is a weight coefficient (in the simulations λ is set to 0.8).

It was observed that when the system variance is large, the accuracy of tracking decreases while the tracking of sudden changes improves. However, low accuracy would cause losing the track which imposes the necessity for choosing the system variance very carefully. Thus, by increasing the system variance, the problem of losing track cannot be resolved.

3 Particle filters with variable number of particles

In this section, an algorithm for adaptively changing the number of particles is proposed. The main objective is to compute the likelihood at a set of particles that cover as large support as possible without increasing the variance of the importance sampling density.

This is achieved by increasing the number of particles when the algorithm determines that an increase is needed. In the case when the particle filter tracks well, the number of particles is decreased. In our simulations, the number of particles when the tracking is satisfactory is M, and when it requires more particles, it is $4M$. These numbers, in general, depend on the nature of the problems that is addressed.

In this implementation, the number of particles is changed (if necessary) during the resampling step. Several criteria for determining whether to alter the number of particles have been considered. The number of particles is increased if

1. the number of particles that have weights lower than a threshold T_l is above a certain number,
2. the number of particles that will be replaced after the resampling procedure is above a certain number, or
3. the difference between the predicted and the real observations is larger than a certain value.

The criteria for the decrease of number of particles are analogous, except that different thresholds and numbers are used.

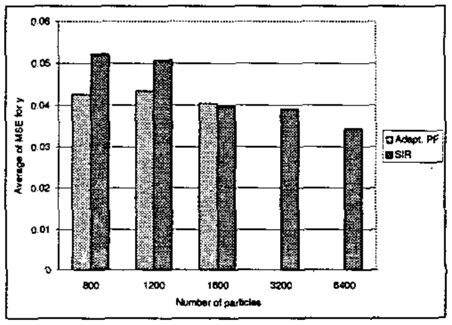
It was noted that particle filters start giving poorer estimates if the input observations change slightly in long time intervals. To correct the problem, the number of particles is increased is such situations. In our implementation, the input angle is monitored for T_f consecutive samples. If the input is still almost constant after T_f samples, the number of particles and/or the variance of the system noise are temporarily increased.

3.1 Complexity and performance analysis

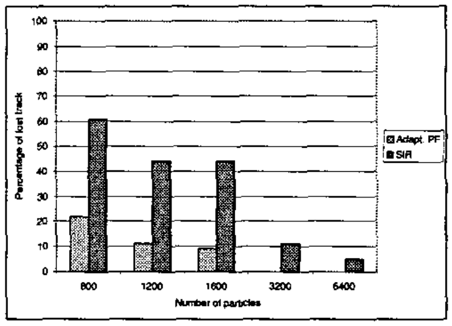
Criteria 1 and 2 allow for improved tracking following the changes in the input data (after the number of particles is increased), while the likelihood based methods [6] are designed to react to changes at the time instant when the changes are detected. The likelihood based methods are difficult for high speed parallel implementations. There the array length of states and weights is data dependent and in the case of parallel implementation the length is determined through communication among parallel elements. In order to simplify the hardware implementation, the criteria for determining the number of particles for processing the next observation are chosen in a way to produce fixed lengths of arrays.

The number of particles that will be replaced after resampling (criterion 2) can be obtained directly from resampling without increasing the computational complexity. However, the number of particles will be updated after the next resampling step. Criterion 3 yields worse results because it may happen that the predicted value of the observation is close to the real value but the tracking of the position is poor.

The average of the mean square error (MSE) and tracking loss rate of the adaptive particle filter and SIR are shown in Figure 2. In the simulations, the number of particles was either M or $4M$. For comparison, the MSE and percentage of cases when the track was lost are displayed for the SIR filter which performs resampling at every time instant. The SIR performances are also shown for higher numbers of particles which correspond to the numbers to which adaptive particle filter adapts. In other words, comparisons should also be made between the performances of the proposed filter that operates with M particles (and occasionally switches to $4M$ particles) and the SIR filer that uses $4M$ particles. The results suggest that the adaptive particle filters with variable number of particles can significantly improve the performance of particle filters.



(a)



(b)

Figure 2: (a) Average of MSE for position y versus the number of particles for the adaptive particle filter. (b) Percentage of lost tracks versus the number of particles.

Since there was a jump in the input observations in the performed simulations, the adaptive particle filter adapted its number of particles more frequently. The gain in speed in the simulations of the adaptive over the standard particle filter which operates on $4M$ particles (for sequential implementation) was about 50%.

4 Particle filter with decreasing number of particles

Instead of increasing the number of particles when necessary, as was described in the previous section, the algorithm presented in this section is based on reducing the number of particles in consecutive time instants. The number of particles reduces slowly when the object is tracked well. In the case of large changes in the input observations, the number of particles will be significantly reduced. If the number is below a pre-defined threshold, the algorithm performs resampling and increases the number of particles to its initial value. Thus, the decision for resampling is based on the number of currently used particles. This criterion does not require any additional calculations.

Filtering without resampling produces less particles with non-zero weights for each consecutive input observation. This means that the processing time for each following observation will be shorter. When the number of useful particles becomes less than a critical number N, resampling is performed on the remaining N or less particles. After resampling, the total number of particles is M. It was observed that the accuracy of tracking with particle filters decreases significantly when the number of used particles is below a certain number and that property is used for defining the critical number N. The algorithm of the particle filter with decreasing number of particles is shown next.

Initial step: Propose M particles and assign the value $1/M$ to all weights.

Particle filter step at time t:

1. Propose new particles.
2. Calculate the weights of the proposed particles, and sum their weights.
3. Normalize the weights.
4. Remove particles with weights less than a predefined threshold.
5. If the number of remaining particles is less than N, perform resampling (after renormalization[1]) so that the new random measure contains M particles. Assign to all M particles weights of $1/M$.
6. Go to the step 1.

When the computation of the weights requires only exponentiation, the number of times when exponentiation is executed can be reduced by modifying the algorithm and introducing an adaptive threshold for the exponent of the function. Here, step 4 of the algorithm is merged with step 2 so that particles are removed if the exponent of the weights $\alpha_t^{(m)}$ deviates significantly from the minimum exponent. Particles are considered to have negligible weights if they satisfy: $\alpha_t^{(m)} > \min(\alpha_t^{(1:M)}) + K$, for $m = 1, ..., M$, and where $K > 0$ is a predefined number. For example, in the simulations K was set to $\ln(0.1M)$, which means that particles with weights less than $\max(w_t^{(1:M)})/(0.1M)$ are considered negligible. This method has an advantage because only the exponents of the weights are compared and the exponential function is not computed for negligible weights.

[1]In an actual implementation, this step can be avoided.

4.1 Complexity and performance analysis

In Figure 3, the mean square error and the percentage of times when the track is lost are presented versus the number of particles. Several resampling methods have been compared, where resampling was performed (a) at every time instant, (b) at every n-th instant ($n = 5$), (c) when the effective number of particles was lower then a pre-defined threshold (two thresholds were used: $4M/5$, $M/5$), and (d) when the current number of particles was less than N ($N = 0.2M$). The last case was simulated for two values of the threshold $K = \{\ln(0.4M), \ln(0.1M)\}$.

The worst tracking performance is obtained when the resampling is performed periodically at every n-th instant. Such result is expected since this method does not exploit information from the processed data. The main advantage of periodic resampling in real-time applications is that the processing time of particle filters per input sample can be reduced if they allow for a delay of several samples of the output. The processing time of the input samples is $[(n-1)t_{wo} + t_w]/n$, where t_{wo} and t_w are the processing times of the particle filter per input sample without and with resampling, respectively, and the equation tends to t_{wo} as n increases.

The particle filter with decreasing number of particles had approximately the same frequency of resampling as the particle filter which performs resampling using an effective number of $M/5$ particles as a criterion. It is not surprising then that these two filters have similar tracking characteristics. However, the particle filter with decreasing number of particles has two very important advantages:

1. The generation of particles and the computation of weights are performed only for the particles with non-negligible importance weights.
2. Resampling is performed on a much smaller number of particles.

The simulation results show that the saving in computation time is about 30% in the sampling and importance computation steps. We must say here that this saving would be higher if the particle filter tracks the trajectory very well. The saving in number of operations in the importance step is very important since the importance step is computationally the most intensive of all operations of the particle filter.

The second advantage means that the input number of particles in the resampling procedure is less or equal to N, while the output number of particles is M. The complexity of the resampling algorithm is directly proportional to the input number of particles. We chose $N = 0.2M$ in the simulations, which means that the savings in the resampling step is almost 80%. This may be particularly useful in sequential implementations of particle filters.

We must note here that particle filters with decreasing number of particles have some overhead for processing which decreases its processing speed. The main disadvantage from a hardware implementation standpoint is that step 4 of the filter cannot be pipelined with previous steps which imposes usage of an additional loop of M iterations.

Methods where resampling is performed based on data dependent criteria such as effective number of particles cannot be used for increasing the processing speed in hardware implementation. However, they can be used to decrease the average power consumption on the chip. Besides, they are very useful for increasing the speed of simulations on a sequential machine.

5 Conclusions

In this paper various issues related to performance and complexity of particle filters in tracking applications have been addressed. They include a presentation of several ideas for designing particle filters with variable number of particles. In order to prevent track losses, one method increases the number of particles and the variance of the system model accordingly. Another method adjusts the number of particles based on the number of negligible weights. The first method significantly improves the performance of the filter, and the second method reduces the execution time while keeping the performance approximately the same.

References

[1] J. Carpenter, P. Clifford, P. Fearnhead,"An improved particle filter for non-linear problems," IEE Proceedings F: Radar, Sonar and Navigation, vol. 146, pp. 2-7,1999.

[2] A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*, New York: Springer Verlag, 2001.

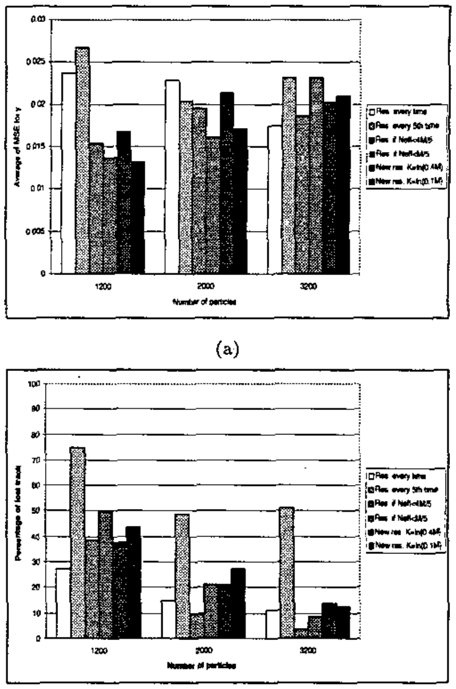


Figure 3: (a) Average mean square error for position and velocity for different resampling methods versus the number of particles. (b) Percentage of time when tracking is lost versus the number of particles

[3] P. Fearnhead, *Sequential Monte Carlo Methods in Filter Theory*, PhD Thesis, Merton College, University of Oxford, 1998.

[4] D. Fox,"KLD-sampling: Adaptive particle filter," Advances in Neural Information Processing Systems 14: Proceedings of the 2001 NIPS Conference, MIT Press, 2002.

[5] N. J. Gordon, D. J. Salmond, A. F. M. Smith, "A novel approach to nonlinear and non-Gaussian Bayesian state estimation," IEE Proceedings F, vol. 140, pp. 107-113, 1993.

[6] D. Koller, U. Lerner, "Using learning for approximation in stochastic processes," Proceedings of the Fifteenth International Conference on Machine Learning, pp. 287-295, July 1998.

[7] J. S. Liu, R. Chen, "Blind deconvolution via sequential imputations," Journal of the American Statistical Association, vol. 90, no. 430, pp. 567-576, 1995.

[8] J. S. Liu, R. Chen, W.H. Wong, "Rejection control and sequential importance sampling," Journal of the American Statistical Association, vol. 93, no. 443, pp. 1022-1031, 1998.