Niching in Monte Carlo Filtering Algorithms

Alexis Bienvenüe†, Marc Joannides‡, Jean Bérard⋆, Eric Fontenas‡ and Olivier François†

† LMC, BP 53, F-38041 Grenoble Cedex 9, France
‡ LABSAD, BP 47, F-38040 Grenoble Cedex 9, France
⋆ LaPCS, Université Lyon 1, F-69622 Villeurbanne Cedex, France

Abstract. Nonlinear multimodal filtering problems are usually addressed via Monte Carlo algorithms. These algorithms involve sampling procedures that are similar to proportional selection in genetic algorithms, and that are prone to failure due to genetic drift. This work investigates the feasibility and the relevance of niching strategies in this context. Sharing methods are evaluated experimentally, and prove to be efficient in such issues.

1 Introduction

In evolutionary computation, selection is often considered as a source of premature convergence. Given a problem with multiple solutions, a genetic algorithm (GA) will at best ultimately converge to a population containing only one of these solutions. This phenomenon, known as genetic drift has been observed in natural as well as in artificial evolution, and is undesirable in many applications (e.g. multi-objective optimization). To overcome the above problem, several methods have been proposed that take their inspiration from mathematical ecology [1]. GA were developed that are capable of forming and maintaining stable subpopulations, or *niches*. GAs which employ *niching* mechanisms become capable of finding multiple solutions to a problem, within a single population [2], [3], [4]. Among these methods, the most popular is *fitness sharing*, that works by modifying the objective function according to the presence of nearby individuals.

Beyond the field of evolutionary computation, similar phenomena have been observed in Monte Carlo strategies such as iterated bootstrap or particle filtering [5], [6]. Such algorithms are based on selection procedures as well, and are closely connected to the traditional GA framework (see Section 2). These strategies have proved their efficiency in high-dimensional nonlinear problems. In terrain navigation for instance, an aircraft measures its relative elevation sequentially, and the goal of filtering is to estimate the position and velocity of the aircraft. This problem is multimodal as several positions might correspond to a single relative elevation. While Monte Carlo strategies are theoretically able to simulate the true distribution of the aircraft position, selection often concentrates the solutions on a single mode leading to wrong decisions.

While the benefit of sharing methods has been intensively studied by the EC community, few efforts have been devoted to the other contexts. This work evaluates the feasibility of sharing methods in Monte Carlo nonlinear filtering algorithms. Section 2 presents an account on the filtering problem and Monte Carlo methods. Section 3 introduces niching strategies in sampling procedures for particle filters and discusses the choice of a sharing bandwidth. In Section 4, the algorithm is evaluated on a set of one-dimensional test problems similar to those encountered in terrain navigation. For these simple models, the solution to the filtering problems can be computed exactly. On this basis, the niching method is compared to the standard algorithm, and proves to be beneficial in this context.

2 Monte Carlo Filtering

Filtering addresses the issue of predicting an unknown signal (X_t) given noisy observations of this signal. Mathematically, the signal is modelled as a Markov process taking values in some measurable space $\mathcal{X}$

$$X_t = F(X_{t-1}) + V_t, \quad t \geq 1 \tag{1}$$

where F is a deterministic function and (V_t) is a sequence of independent identically distributed centered random variables. More generally, such dynamics can be specified according to some Markov kernel $Q(x, dx)$ that describes the transition probabilities between successive states. The distribution of X_t is often called the *prior distribution*.

In the filtering problem, the signal cannot be observed directly. Instead, data are (indirectly) gathered from the observation of a second signal

$$Y_t = H(X_t) + W_t, \quad t \geq 0, \tag{2}$$

where H is usually a nonlinear function and W_t a noisy variable independent from X_t.

The filtering problem consists of making predictions about the original signal X_t given the observations Y_0, Y_1, …, Y_t. This amounts to estimating (or computing) the conditional distribution of X_t given these observations. This distribution is called the *posterior distribution*.

Filtering has an old tradition that goes back to the seminal paper by Kalman [7]. The standard approach assumes that F and H are linear and that V_t and W_t are Gaussian random variables of known covariance matrices. In contrast, solving nonlinear filtering problems turns out to be particularly difficult, and the difficulty is even increased when the signal becomes unidentifiable (e.g., H not invertible). In such a case, the posterior distribution may be multimodal. Kalman filters would therefore lead to erroneous predictions since these methods always predict a single mode.

Kunita and Stettner [8] developed general recursion schemes that compute the exact solution of the filtering problem based on Bayes' formula. Despite their closed form, these equations are hardly of practical interest since numerical computations of high-dimensional integrals are involved.

Monte Carlo filtering is an algorithmic alternative to Kunita-Stettner recursion. It consists of a computer intensive technique, and can be useful where linear filtering fails. This method is based on a *particle system approach* [10], [11], [9] in which the posterior distribution is computed empirically. In this approach, a population of n particles is evolved in the signal space according to a two-stages procedure. More precisely, let $x_t = (x_1, \dots, x_n)$ be the population at time t (x_0 is randomly initialized). The two steps are iterated as follows.

1) **Prediction.** Create n new particles $x'_1, \dots, x'_n$ by sampling from the transition kernel $Q(x, dx)$

$$x'_i \sim Q(x_i, dx), \quad i = 1, \dots, n.$$

Conditional to $x_1, \dots, x_n$, the new particles are independent.

2) **Correction.** Resample the particles according to a proportional scheme taking the observation y_t (at time t) into account

$$x''_i \sim \frac{L_t(y_t - H(x'_i))}{\sum_j L_t(y_t - H(x'_j))}$$

where L_t is the likelihood function of the observation noise W_t. Define the population at time $t+1$ as being $x_{t+1} = x''$.

The convergence of this algorithm to the optimal solution of the filtering problem has been proved in [11], when the population size goes to infinity.

Turning to an evolutionary computation perspective, there is a close connection between Monte Carlo filters and the simple GA without recombination. This connection has been emphasized in previous works by [11], [12]. In the above algorithm, particles can be identified as individuals in a population, where the set of phenotypes corresponds to the possible states of the signal. The first step, called *prediction*, is similar to the mutation step in GAs. Each individual generates an offspring by mutation from the kernel $Q(x, dx)$ (and the offspring replaces its parent). In the second step, called *correction*, a random selection of the offspring is performed. The selection strategy is similar to the proportional selection scheme used in GAs. However, the fitness function is time-dependent as it must account for the information contained in the data at each instant. Mathematically, the fitness of offspring x_i can actually be defined as

$$f_t(x_i) = L_t(y_t - H(x_i)). \tag{3}$$

3 Niching in a filtering context

3.1 Niching algorithms

The reason why Monte Carlo filtering methods should work is that their infinite-population models correspond to Kunita's recursion scheme precisely. However, the shortcomings of infinite-population models are well-known. By their very nature, they may not truly reflect the finite-population properties that are of major interest to a practitioner. For instance, the effect of stochastic fluctuations during the correction step are neglected in this approach.

The same kind of remark also holds for the traditional GA. To overcome the above problem, Goldberg and Richardson proposed a method based on sharing [1]. These methods require that the objective *fitness* function be shared as a single resource among similar individuals in a population. Niching is achieved by degrading the objective function (i.e., the unshared fitness) of an individual according to the presence of nearby individuals. This type of niching requires a distance metric on the phenotype of the individuals. The objective fitness $f(i)$ of an individual i is degraded by first summing all of the shared values of individuals within a fixed bandwidth σ_{sh} of that individual and then dividing $f(i)$ by this sum, which is known as the *niche count*. More specifically, if two individuals are separated by distance $d(i,j)$, then a shared value is added

$$\mathrm{sh}(d(i,j)) = \begin{cases} 1 - (\frac{d(i,j)}{\sigma_{\mathrm{sh}}})^{\alpha} & \text{if } d(i,j) < \sigma_{\mathrm{sh}} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

to the niche count

$$m(i) = \sum_{j=1}^{n} \mathrm{sh}(d(i,j)). \tag{5}$$

The parameters σ_{sh} and α are chosen by the user of the niched GA based on some a priori knowledge of the fitness landscape. The parameter α is often set to one, yielding the triangular sharing function. Finally, the shared fitness is defined as

$$f'(i) = \frac{f(i)}{m(i)}. \tag{6}$$

The actual fitness of each individual is modulated according to the density of the population around it: the fitness of isolated individuals is increased, while that of individuals in well represented regions of the search space is decreased.

3.2 Sharing methods in Monte Carlo filters

In this paper, we investigate the maintenance of stable subpopulations in Monte Carlo filtering algorithms using the method of sharing function. Since the available computational resources do not allow the number of particles to be arbitrary large, standard Monte Carlo filters often suffer from a loss of diversity due to the stochastic nature of resampling. In real-world applications (e.g. real-time target tracking algorithms), this premature loss of diversity implies loosing the signal for some time by concentrating all individuals in a possibly wrong region of the search space. Maintaining stable niches in Monte Carlo filter is therefore a crucial point, since these niches actually correspond to existing modes of the posterior distribution. A niching procedure can be included in Monte Carlo filters as follows.

1) **Prediction (unchanged).** Create n new particles x_1', ..., x_n' by sampling from the transition kernel $Q(x, dx)$

$$x_i' \sim Q(x_i, dx), \quad i = 1, \ldots, n.$$

Conditional to $x_1, \ldots, x_n$, the new particles are independent.

2) **Correction.** Resample the particles according to a proportional scheme taking the observations into account

$$x_i'' \sim \frac{f_t'(x_i')}{\sum_j f_t'(x_j')}$$

where $f_t'(x_i')$ is the shared fitness of $f_t(x_i') = L_t(y_t - H(x_i'))$.

In implementing this algorithm, choosing the bandwidth σ_{sh} is a critical step. Deb's rule sets this parameter by taking into account distances between peaks, and relative fitnesses. Specifically,

$$\sigma_{\mathrm{sh}} = \min_{i,j} \left\{ \frac{d(x_i', x_j')}{1 - r_{ij}} \right\} \tag{7}$$

where

$$r_{ij} = \min \left(\frac{f_t(x_j')}{f_t(x_i')}, \frac{f_t(x_i')}{f_t(x_j')} \right) \tag{8}$$

and the metric d corresponds to the Euclidean Distance. In one-dimensional filtering problems, a better-supported rule is given by Silverman [13] inspired from density estimation

$$\sigma_{\mathrm{sh}} = 0.9 \min\left(\mathrm{sd}(x'), \mathrm{iqr}(x')/1.34\right) n^{-0.2}.$$

This rule is based on the minimum of the standard deviation of x' and its interquartile range divided by 1.34. Using this rule is quite natural in a niching context. Sharing indeed degrades the fitness function by dividing by the density of nearby particles and actually involves an estimation of this density. Note that a proper use of this rule requires that the sharing method be build upon a Gaussian kernel instead the triangular function. Similar rules also exist in higher dimensions.

4 Experiments

4.1 Test problems

Evaluating the impact of the sharing method in Monte Carlo filters is difficult in general. A number of test problems have been chosen to assess the performances empirically. The selection of six test problems is inspired from target tracking issues. Three noisy dynamical systems describe the motion of a target in one dimension. The first motion is the classical $AR(1)$ dynamics [7]

$$X_t = 0.9X_{t-1} + V_t, \quad V_t \sim \mathcal{N}(0,1) \tag{9}$$

where the (random) initial condition X_0 is sampled according to the Gaussian distribution $\mathcal{N}(0, 5.26315)$ (so that the signal is stationary). The second motion is called the *piecewise linear dynamics*, and can be described as

$$X_t = X_{t-1} - 0.1\,\mathrm{sign}(X_{t-1}) + V_t, \quad V_t \sim \mathcal{N}(0,1) \tag{10}$$

and $X_0 = 0$. The third motion is called the *double well dynamics* [14]

$$X_t = X_{t-1} - 0.04X_{t-1}(X_{t-1}^2 - 1) + V_t, \quad V_t \sim \mathcal{N}(0, 0.01 \times q) \tag{11}$$

where q is a parameter set to 0.24 in [14], and $X_0 = 0$. In addition, these three motions are observed through different functions. The first observation function is a symmetric one

$$H(x) = |x|, \tag{12}$$

and the second is non-symmetric

$$H(x) = \begin{cases} 2x & \text{if } x \geq 0\,; \\ -x/2 & \text{if } x \leq 0\,. \end{cases} \tag{13}$$

The observation noise is standard Gaussian $W_t = \mathcal{N}(0, \sigma^2)$ (σ is often set to 1). The length of simulation runs is equal to $T = 100$. Regarding the symmetric observation function, the posterior distribution is bimodal while this is not always the case for the non-symmetric function. For the six problems, the posterior distribution can be computed exactly using Kunita's recursions. Knowing the exact solution will be useful in assessing the accuracy of the filtering procedures during the experiments.

Figure 1 displays a typical trajectory from the double well dynamics (a) and the exact posterior distributions computed via Kunita's recursions under a symmetric observation function (b). A population of 20 individuals is evolved using the classical Monte Carlo procedure (c) and the niching algorithm (d).

4.2 Experimental design

Simulation runs contain simulated trajectories of the signal motion and the corresponding observations. For each of the six models, simulations are repeated 100 times so that the performances can be evaluated statistically. In the experimental design, the following parameter settings are experimented.

NS	No Sharing
Sil	Silverman's rule for the sharing bandwidth
Deb	Deb's rule for the sharing bandwidth
σ_{sh}	constant sharing bandwidth (values = 0.1, 1, 10)

The variable NS means that no sharing is used. The algorithm corresponds to the classical Monte Carlo filtering method. The next levels indicate how the sharing bandwidth has been set up: Silverman's rule, Deb's rule or constant. Except for Silverman's rule, the triangular function is chosen (parameter sh = 1 in Figure 2). Other choices were tested but the results did not change significantly.

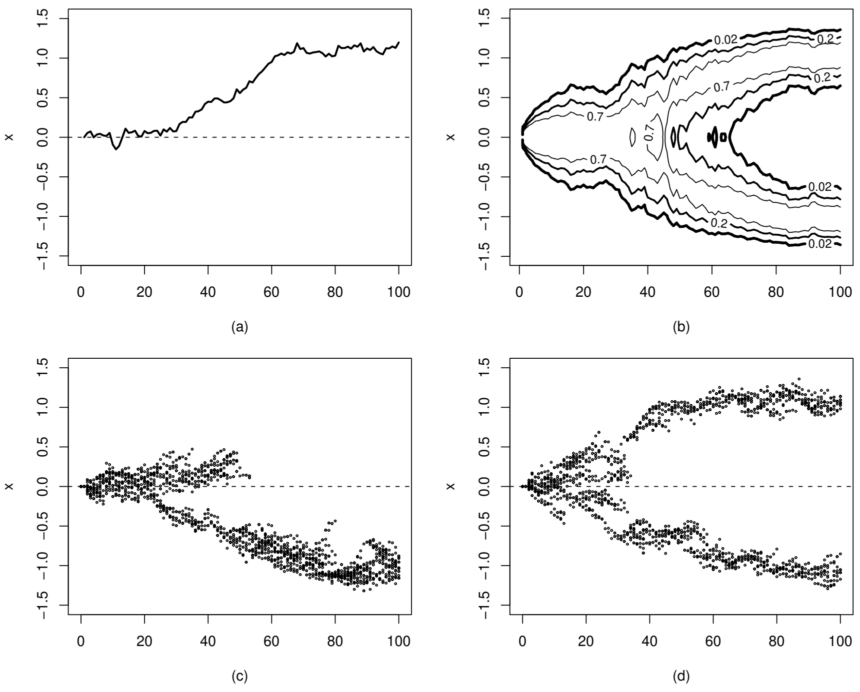

Fig. 1. (a) A simulated trajectory from the double well dynamics. (b) Contour plot of the posterior distribution densities. The observation function is symmetric $H(x) = |x|$. (c) Classical Monte Carlo filtering plot (population size = 20). (d) Monte Carlo + Niching simulation plot.

4.3 Performance measures

As shown by Figure 1, sharing can be helpful in combating genetic drift in Monte Carlo filtering algorithms. Indeed, the distribution of individuals seems closer to the true distribution than the population corresponding to the classical procedure. To quantify these observations, several performance measures can be introduced. Such measures assess the distance of the empirical distribution of the population from the true distribution. The ratio of good decisions taken by the algorithms can be compared.

KS distance. The Kolmogorov-Smirnov metric is a standard measure of the distance between two probability distributions μ_1 and μ_2. It is defined as

$$D_{KS}(\mu_1, \mu_2) = \sup_t |F_1(t) - F_2(t)| \in [0, 1]$$

where F is the cumulative distribution function of μ. Here, μ_1 denotes the empirical distribution of the population computed from Monte Carlo algorithms and μ_2 denotes the true distribution.

Measure of symmetry: MS. In the case where H is symmetric (half of the simulation runs), the posterior distribution of X_t is bimodal (and symmetric). MS is the fraction of time during which two niches (modes) subsist. (A niche is considered active when more than 10 percent of the population are present.)

Ratio of good decisions: RGD. In the case where H is non symmetric, we say that an algorithm takes a decision when all individuals are located at the same side of zero during 5 time steps. Good decisions correspond to all individuals evolving in the same region as the signal.

4.4 Results

This Section presents the most significant results obtained after the series of experiments. Symmetric observation functions are discussed first. Figure 2 reports the values of MS and the KS distance for the double well dynamics (DW). Similar results have been obtained for AR(1) and the piecewise linear dynamics. These results are summarized in Table 1.

Figure 2 shows that the fraction of time (MS) during which two niches subsist averages to 0.346 in the Monte Carlo filtering algorithm. This fraction increases to 0.922 when sharing is used together with Deb's rule (the best bandwidth is however $\sigma_{\rm sh} = 1$). Simulation runs have also been performed with a population size of 100 individuals. The improvement due to sharing is more significant when the population size is small. Similar remarks can be made regarding the KS distance. For large population sizes, the goodness-of-fit seems better when Silverman's rule is used.

Table 1 reports the relative gain in using Deb's rule computed for each measure (MS and KS). This gain represents the difference between measures with sharing / without sharing averaged over 100 runs. The results are given as percentages (MS and KS are floating point numbers between 0 and 1). Numbers in brackets represent the best ratio obtained from constant bandwidth rules (when this ratio is significantly better than Deb's rule). The star means that MS reached the maximal value (100%). The following set of comments can be made about these results.

1) For this set of bimodal problems, sharing always improves Monte Carlo filters (the improvement may sometimes be a minor one).
2) Deb's rule is competitive (and has the advantage to be adaptive). This explains why this rule is chosen as a reference rule in Table 1.

3) Silverman's rule outperforms the other rules for large population sizes. (In some sense, this rule is optimal in density estimation when n grows to infinity.)
4) Sharing is less efficient when the observation noise is small.

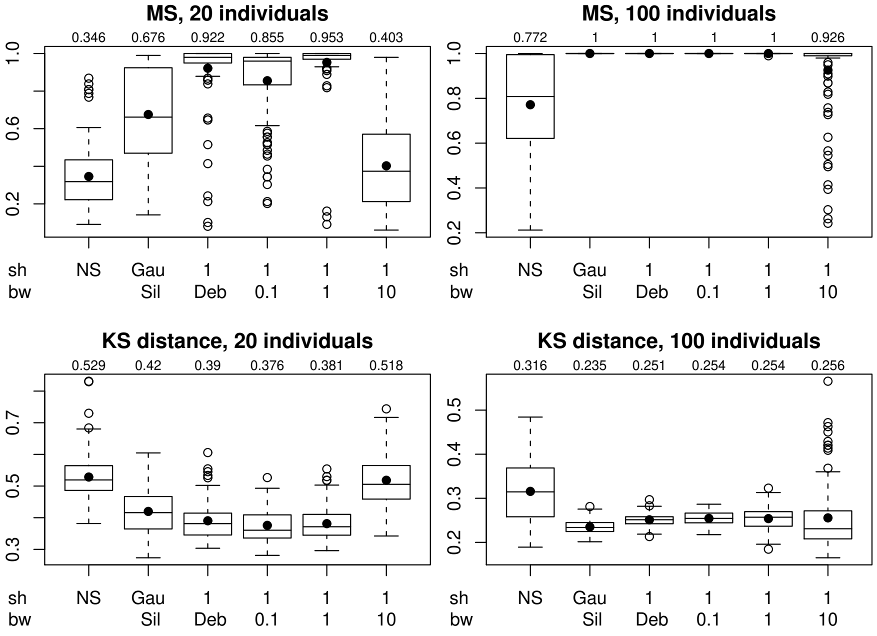

Fig. 2. Performance measures corresponding to the double well dynamics ($q = 0.24$, $\sigma = 1$) and the symmetric observation function. sh refers to the sharing function (Gaussian or triangular) and bw to the bandwidth parameter.

Turn now to the experimental results in the non-symmetric observation context. These results are summarized in Table 2 using the same notations as before. In the non-symmetric context, the measure of symmetry has been replaced by the ratio of good decisions (RGD). The posterior distribution is indeed multimodal only during a short interval of time, after that it concentrates on a single mode.

In contrast to bimodal problems, high gains can hardly be expected. The aim of sharing is maintaining individuals in niches when such niches truly exist. Note that this method do not create artificial niches. Significant gains can nevertheless be observed when the posterior distribution remains multimodal within a long period before concentrating on a single mode. In the period during which the filtering problem is multimodal, maintaining subpopulations in all niches is crucial, as the algorithm should be capable of tracking the mode that will subsist.

	gains (%)			
	20 individuals		100 individuals	
dynamics	MS	KS	MS	KS
$AR(1)$, $\sigma = 1$	24	5	4*	3
$AR(1)$, $\sigma = 0.1$	0	0	*(15)* 2	*(7)* 1
piecewise linear, $\sigma = 1$	17	6	13*	8
piecewise linear, $\sigma = 0.2$	1	1	*(20)* 7	4
DW, $q = 0.05$, $\sigma = 1$	*(49)* 35	*(15)* 8	25*	6
DW, $q = 0.24$, $\sigma = 1$	57	14	23*	6
DW, $q = 0.1$, $\sigma = 1$	62	17	21*	6
DW, $q = 0.24$, $\sigma = 0.05$	*(21)* 12	4	40	15

Table 1. Improvement obtained from sharing with Deb's rule (symmetric observation function).

	gains (%)			
	20 individuals		100 individuals	
dynamics	RGD	KS	RGD	KS
$AR(1)$, $\sigma = 1$	2	-1	2*	-3
$AR(1)$, $\sigma = 0.1$	*(1)* -5	0	-2	0
piecewise linear, $\sigma = 1$	*(2)* -4	1	0	-5
piecewise linear, $\sigma = 0.2$	-2	2	4	3
DW, $q = 0.05$, $\sigma = 1$	11	9	17*	*(1)* -5
DW, $q = 0.24$, $\sigma = 1$	*(19)* 6	6	10*	*(1)* -12
DW, $q = 0.24$, $\sigma = 0.05$	-2	-1	5	-3

Table 2. Performances of sharing for the non-symmetric observation function.

5 Discussion

This paper presented a new paradigm in Monte Carlo filtering algorithms: the method of likelihood sharing. While niching methods are widely accepted in evolutionary computation, the benefit of these techniques remains unexplored in several neighboring domains.

Our results give evidences that sharing methods can improve Monte Carlo filtering algorithms significantly. These methods are dedicated to problems for which posterior distributions are multimodal and standard algorithms are not efficient. Adding a sharing method allows population sizes to be reduced by a large factor, and contributes to the global efficiency of the algorithm.

The method can be beneficial as well for problems that are not purely multimodal. This occurs in tracking a specific mode among several others that would be prominent after a while.

The empirical results presented in this paper have been obtained for one-dimensional problems, for which the solution can be computed by standard numerical methods. Further work is needed to extend this contribution to real-world problems (such as those arising in terrain navigation) and higher dimensional issues.

Acknowledgements

This work is supported by the projects IMAG-SASI and AlPB.

References

1. D. E. Goldberg and J. Richardson. *Genetic Algorithms with Sharing for Multimodal Function Optimization.* In J. J. Grefenstette, editor, Genetic Algorithms and Their Applications: Proc. of the 2nd international Conference on Genetic Algorithms, July 28-31, Cambridge MIT, L. Erlbaum Associates, Hillsdale, MA, pp. 41–49, 1987.
2. K. Deb and D.E. Goldberg. *An investigation of niche and species formation in genetic function optimization.* In J.D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, pp. 42–50. Morgan Kaufmann, Fairfax VA, 1989.
3. C.M. Fonseca and P. J. Fleming. *An overview of evolutionary algorithms in multiobjective optimization.* Evolutionary Computation, **3**:1 (1995), pp. 1–16.
4. J. Horn. The nature of niching: Genetic algorithms and the evolution of optimal, cooperative populations. PhD Thesis, Univ. Illinois at Urbana-Champaign, 1997.
5. C. Musso and N. Oudjane. *Particle methods for multimodal filtering.* In Proc. of the second international conference on Information Fusion, Silicon Valley, CA, July 6-8, IEEE Press, pp. 785–792, 1999.
6. C. Musso, N. Oudjane, F. Legland. *Improving regularized particle filters.* In A. Doucet, J.F.D. freitas and N.J. Gordon, eds: Sequential Monte Carlo methods in practice, Springer-Verlag, New-York, 2001.
7. R. E. Kalman. *A New Approach to Linear Filtering and Prediction Problems*, Transaction of the ASME–Journal of Basic Engineering (1960), pp. 35-45.
8. H. Kunita. *Asymptotic behavior of non-linear filtering errors of markov processes*, J. Multivariate Analysis, **1**:4 (1971), pp. 365–393.
9. A. Doucet, J.F.D. freitas and N.J. Gordon, Eds. *Sequential Monte Carlo methods in practice*, Springer-Verlag, New-York, 2001.
10. N. Gordon, D. Salmond and A. Smith. *Novel approach to nonlinear/non-Gaussian Bayesian state estimation.* IEEE Proceedings, Part F, **140** (1993), pp. 107–113.
11. P. Del Moral. *Non linear filtering: Interacting particle solution.* Markov Process and Related Fields, **2**:4 (1996), pp. 555–580.

12. P. Del Moral, L. Kallel and J. Rowe. *Modeling Genetic Algorithms with Interacting Particle Systems.* In L. Kallel, B. Naudts, A. Rogers, Eds. Natural Computing Series: Theoretical Aspects of Evolutionary Computing, pp. 10–67, Springer-Verlag, Berlin, 2001.
13. B. W. Silverman. Density estimation. Chapman and Hall, New-York, 1986.
14. R.N. Miller, E.F. Carter, S.T. Blue. *Data Assimilation into nonlinear stochastic models*, Tellus, **51A** (1999), pp. 167–194.