# Parallelisation of the Particle Filtering Technique and Application to Doppler-Bearing Tracking of Maneuvering Sources

V. Teulière and O. Brun

*Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) du CNRS, 7, avenue du Colonel Roche, 31077 Toulouse Cedex, France.*

**Abstract**

Particle filtering may cope with nonlinear and/or non-Gaussian models. As a counterpart, this technique suffers from an heavy computation cost and can not always satisfy the real time constraints of applications. A data parallel algorithm is proposed to achieve real-time particle filtering. Extensive results are presented in the passive SONAR tracking context for maneuvering targets.

*Key words:* Particle filtering, parallel algorithm, Doppler-Bearing tracking, maneuvering target.

## 1 Introduction

Nonlinear filtering is the natural framework to state global estimation problems where a dynamic stochastic process is partially observed by a noise-corrupted nonlinear measuring device. It is well known that in general the solution of such problems can not be derived in closed form due to its infinite dimensional character, with the noticeable exception of linear Gaussian state space models [22]. Therefore, approximate solutions have been proposed. Traditional solutions are based on local linearisation of system and measuring equations and/or moment truncation of density probability. However, such approaches can not maintain guaranteed performance or even stability.

*Email address:* {`teuliere,brun`}`@laas.fr` (V. Teulière and O. Brun).

An efficient approach in the general case of non linear and/or non Gaussian state space model is particle filtering (PF) [6,7,4,11,13,3] (for a survey see [5]). The PF technique yields optimal solutions for problems with higly non-linear models and non-Gaussian dynamic and observation noises. It recursively constructs the conditional probability density of the state variables, with respect to all available measurements, through a random exploration of the state space by entities called particles. A weight is assigned to each particle by a Bayesian correction term based on measurements. Its main advantage relies on global properties of the procedure, which leads to guaranteed convergence.

The PF technique has shown on both real and simulated data astonishing gains in comparison to classical techniques, in particular in weak signal to noise ratio contexts. Examples of applications include RADAR [14,15], LORAN C [16] and GPS [10]. The main advantage of the particle technique is that it takes into account all dynamic aspects and nonlinearities of the problem. However, despite successful applications to various signal processing problems, the PF technique suffers from an heavy computation cost and can seldom cope with the hard real-time constraints of the most challenging application (online computation).

Passive SONAR processing is an example of an application for which online PF processing can not always be applied. Classical Bearing-only tracking (BOT) and Doppler bearing tracking (DBT) are mainly restricted to constant velocity motion. Both problems have been well investigated [17,20,18,21]. Target motion analysis for a maneuvering target is an harder problem and existing methods present severe drawbacks [19,23]. Particle technique provides a convenient framework for maneuvering target tracking in passive SONAR context since source maneuvers can be modeled by means of a non Gaussian stochastic process which provides a very realistic behavior. Introducing a new particle technique [9], called Gaussian Particle Filter (GPF), the computation cost can be meaningfully reduced. As sampling rate is relatively low (around 0.1 $Hz$), on-line GPF can be computed most of the time. However, in some difficult situations (especially when target moves almost radially), an efficient tracking can only be achieved with a very large number of particles, forbidding sequential real time computation.

Despite these difficulties, on-line computation of PF can be achieved using the inherent parallel structure of these algorithms. This article addresses the parallelization of the PF technique. It is organized as follows. In section 2, we describe the general discrete time nonlinear filtering problem, the PF algorithm and the GPF algorithm. In section 3, we present the parallel PF that we have developed. Sections 4.1 presents the stochastic model of a of maneuvering target in the passive SONAR context. This non linear and non Gaussian filtering problem provides a convincing benchmark to assess the performance of the parallel PF algorithm. For this application, extensive results obtained with an Origin2000 parallel computer and with a PC cluster are given in section 4.2. Finally, some concluding remarks are drawn in section 5.

## 2 Particle filtering technique

### *2.1 Discrete-time nonlinear filtering*

Let the following equation represents the evolution of the state $x_k \in \mathbb{R}^n$ of a discrete time dynamic system:

$$x_{k+1} = f_k(x_k, w_k) \tag{1}$$

where $f_k$ is the system transition function and $w_k \in \mathbb{R}^p$ is a dynamic noise which has a known probability density function (PDF). It is assumed that the PDF $p(x_0)$ of the initial state is known. At discrete times, measurements (or observations) $y_k \in \mathbb{R}^p$ of the state $x_k$ become available and are related to the state through the following observation equation:

$$y_k = h_k(x_k, v_k) \tag{2}$$

where $h_k$ is the measurement function and $v_k \in \mathbb{R}^p$ is an observation noise of known PDF. At time $k$ the set of measurements $Y_k = \{y_\tau, \tau = 1, \ldots, k\}$ is available.

The optimal estimator $\widehat{x}_k$ (in the minimum variance sense) is provided by the conditional expectation $E[x_k|Y_k]$. The problem of assessing this conditional expectation is obviously related to the computation of the conditional PDF $p(x_k|Y_k)$ which is classically obtained in two stages: prediction and update.

- **Prediction :** Assume that the required PDF $p(x_{k-1}|Y_{k-1})$ is known at time $k-1$. The prediction step yields $p(x_k|Y_{k-1})$ using the Chapman-Kolmogorov integral equation:

$$p(x_k|Y_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|Y_{k-1})dx_{k-1} \tag{3}$$

- **Update :** At time $k$, when $y_k$ becomes available, the prior PDF $p(x_k|Y_{k-1})$ is updated via Bayes' rule:

$$p(x_k|Y_k) = \frac{p(y_k|x_k)p(x_k|Y_{k-1})}{p(y_k|Y_{k-1})} \tag{4}$$

where the normalization factor $p(y_k|Y_{k-1})$ is the integral of the numerator. Substituting (3) in (4), we obtain the following recursive solution:

$$p(x_k|Y_k) = \frac{p(y_k|x_k) \int p(x_k|x_{k-1})p(x_{k-1}|Y_{k-1})dx_{k-1}}{p(y_k|Y_{k-1})} \tag{5}$$

Equation (5) is the formal solution to the Bayesian recursive estimation. Unfortunately, this solution is, in general, infinite-dimensional, with the noticeable exception of the well known linear Gaussian filter (Kalman filter). A rather crude approximation is the extended Kalman filter (EKF). A different and efficient approach to this problem is PF.

### *2.2 Particle filtering*

PF is a procedure to solve the general nonlinear and/or non-Gaussian estimation problem. The most rigorous interpretation of the PF method is as a birth and death process. However, for sake of clarity, we recall here the weight interpretation of the PF algorithm (the only one used in practice).

- **Initialization**
  The initial PDF $p(x_0)$ is approximated by a Dirac comb of $N$ random particles :

  $$p(x_0) = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_0^i}(x_0) \tag{6}$$

  where $\delta_x$ denotes the Dirac's measure with pointwise support $x$ and $N$ is the sample size. The initial set $\{x_0^i\}_{i=1,\dots,N}$ of particles is obtained by $N$ random draws according to $p(x_0)$.

- **Prediction - a priori exploration**
  Each particle $x_k^i$ evolves independently according to (1) :

  $$x_k^i = f_k(x_{k-1}^i, w_{k-1}^i) \tag{7}$$

  where $w_{k-1}^i$ is drawn from the PDF of the dynamic noise $p(w_{k-1})$. This procedure yields an approximation of the a priori PDF:

  $$p(x_k) = \frac{1}{N}\sum_{i=0}^{N} \delta_{x_k^i}(x_k) \tag{8}$$

- **Update - weighting**

The update step consists in using the measurement $y_k$ for weighting particles. Substituting the a priori PDF of the state (8) in (5) yields:

$$p(x_k|Y_k) = \sum_{i=1}^{N} \rho_k^i \delta_{x_k^i}(x_k) \tag{9}$$

where $\rho_k^i$ is the weight of particle $i$ at time $k$. This weight can be computed in a recursive way:

$$\rho_k^i = \frac{p(y_k|x_k^i)}{\sum_{j=1}^{N} p(y_k|x_k^j)\rho_{k-1}^j}\rho_{k-1}^i \tag{10}$$

In the sequel, measurement noise is assumed to be additive and Gaussian (which is the case of most applications), i.e:

$$y_k = h_k(x_k) + v_k \text{ with } E[v_k] = 0 \text{ and } E[v_k v_k^T] = R \tag{11}$$

The measurement PDF $p(y_k|x_k)$ is then given by:

$$p(y_k|x_k^i) = \Gamma_{y_k}\left(h\left(x_k^i\right), R_k\right) \tag{12}$$

where $\Gamma_x(\bar{x}, V)$ denotes an $n$-Gaussian distribution of mean $\bar{x}$ and variance $V$.

A priori exploration and weighting step are depicted on figure 1 (dashed lines are the particle trajectories and arrow's amplitude represents particle weight). Most likely particles, i.e. those corresponding to most probable states, get "heavier" weights than others.

- **Estimation**
  The estimator $\widehat{x}_k$ is given by a weighted sum of the particle states $x_k^i$:

$$\widehat{x}_k = \sum_{i=1}^{N} \rho_k^i x_k^i \tag{13}$$

and the error covariance $\widetilde{P}_k$ (which provides a confidence interval on the estimate $\widehat{x}_k$) by:

$$\widetilde{P}_k = \sum_{i=1}^{N} \rho_k^i x_k^i {x_k^i}^T - \quad \widehat{x}_k \widehat{x}_k^T \tag{14}$$

- **Resampling**
  The above steps of prediction and update form a single iteration of the PF algorithm (fed by initialization). However, this crude approach leads to

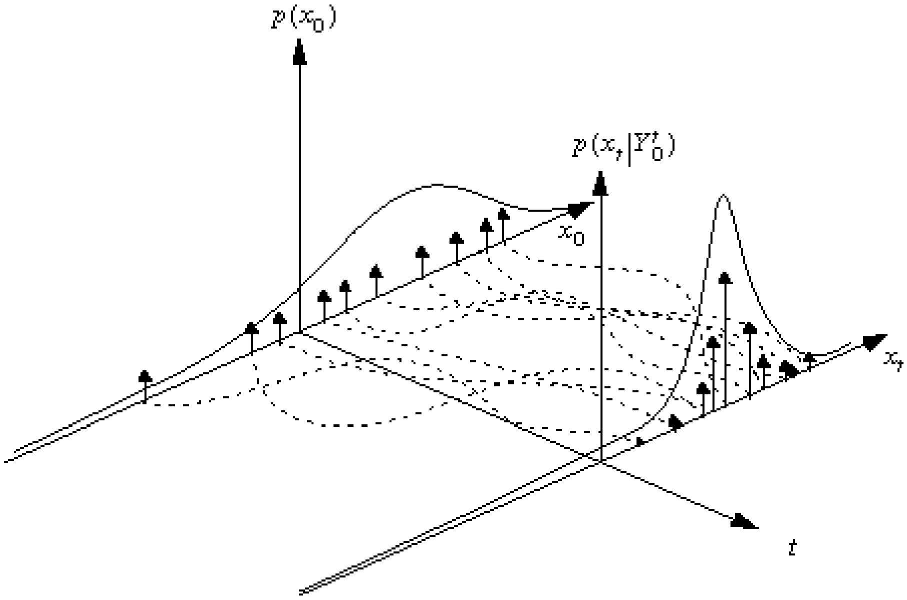



Fig. 1. Exploration of the state space by particles.

degeneration of the algorithm. Indeed, the weights $\rho_k^i$ will numerically degenerate due to the fact that likelihoods for different sample path diverge. We would then expect, asymptotically, null weights for almost all particles but a very few, due to normalization. Consequently, this degenerate discrete representation of the PDF asymptotically bears no useful information about the process, and some regularizing action has to be done.

A sampling/resampling technique, known as particle redistribution, can cope with this degeneration problem. A new set of particles $\{x_k^i\}_{i=1,\ldots,N}$ is obtained by redistributing the particles positions (in the state space) according to the discrete representation of the PDF. Several efficient techniques can achieve this resampling step. In the parallel context, proportional redistribution is the most suitable one. This approach is fast and efficient, but as a counterpart the sample size $N$ varies. Each particle $i$ of weight $\rho_k^i$ is duplicated in $n_k^i$ particles, where:

$$n_k^i = round\left[\rho_k^i N_c\right] \tag{15}$$

where *round* denotes the nearest integer value and $N_c$ is the wanted number of particles after each redistribution (equal to the initial number of particles $N$). The weight $\rho_k^{j=1,\ldots,n_k^i}$ of each duplicated particle is set to:

$$\rho_k^j = \frac{\rho_k^i}{n_k^i} \quad j = 1, \ldots, n_k^i \tag{16}$$

Obviously, if the particle weight is lower than $1/(2Nc)$, $n_k^i = 0$. The resulting number of particles $\sum_{i=1}^N n_k^i$ differs from $N_c$ and varies after each redistribution (in practice, variation with respect to the initial sample size is lower than 5%).

This procedure has a regulating effect on the algorithm. Indeed, most probable states, corresponding to "heavy" particles, give birth to more particles, at the expense of light particles which die. This procedure guarantees an adaptive occupation of the state space regions according to their probability masses.

However, it is very important to note that particle resampling is not applied at each iteration. For a finite number $N$ of particles, systematic resampling might have disastrous consequences on the behavior of the PF. Indeed, redistribution induces a "loss of diversity" and it is thus preferable to let particles acquire a significant weight before resampling.

Particle redistributions can be done periodically or adaptively. In the latter case, redistribution occurrence is governed by the weight's deviations which provides a degeneration measure. An efficient heuristic indicator is the effective sample size $N_{eff}$ [1] which can be approximated by:

$$N_{eff} \approx \frac{1}{\sum_{i=1}^{N} \left(\rho_k^i\right)^2} \tag{17}$$

Redistribution is applied if $N_{eff}$ gets lower than a given threshold.

### *2.3 Gaussian particle filtering (GPF)*

When dynamic perturbations are weak, the exploration of the state space during the prediction step is insufficient. Due to redistribution, the whole set of particles tends to concentrate on the neighborhood of a few particle positions (the most likely). This problem can be solved by increasing the number $N$ of particles. However, to reduce the algorithmic cost, a suitable solution is to enhance the exploratory properties of the particles. At time $k$, the conditional PDF of the state is given by:

$$p(x_k|Y_k) = \sum_{i=1}^{N} \rho_k^i \Gamma_{x_k}(x_k^i, P_k^i) \tag{18}$$

where

$$P_k^i = E\left[\left(x_k - x_k^i\right)\left(x_k - x_k^i\right)^T\right] \tag{19}$$

Each particle corresponds to a Gaussian distribution $\Gamma_{x_k}(x_k^i, P_k^i)$ of weight $\rho_k^i$. The computation of the conditional PDF of the state $p(x_k|Y_k)$ is performed via local linearization of the model (system and measurements). For sake of simplicity we consider the following model :

$$\begin{cases} x_{k+1} = f_k(x_k, w_k) \\ y_k = h_k(x_k) + v_k \end{cases} \tag{20}$$

where $v_k \sim \Gamma_{v_k}(0, R_k)$. The functions $f_k$ and $h_k$ are differentiable with respect to $x_k$. We recall here the main features of GPF (details can be found in [9]).

- **Initialization**
  Initialization of the Gaussian particle set $\{x_k^i, P_k^i\}_{i=1,\ldots,N}$ aims to approximate a priori PDF of the state $p(x_0)$ and to ensure validity of linearization. This is provided by mean of a regular metric suited to the problem geometry. As the number of particles increases, the initial covariance gets smaller and the linearization assumption gets more relevant.

- **Prediction**
  Suppose that the conditional PDF $p(x_{k-1}|Y_{k-1})$ is known at time $k$:

  $$p(x_{k-1}|Y_{k-1}) = \sum_{i=1}^{N} \rho_{k-1}^i \Gamma_{x_{k-1}}(x_{k-1}^i, P_{k-1}^i) \tag{21}$$

  Prediction of the particle state $x_{k|k-1}^i$ is performed via a random sampling of $w_{k-1}^i$ according to $p(w)$:

  $$x_{k|k-1}^i = f_k(x_{k-1}^i, w_{k-1}^i) \tag{22}$$

  The prediction covariance matrix $P_{k|k-1}^i$ is computed from the linearized dynamic equation. This leads to a classical prediction step of a parametric EKF (i.e without any additive dynamic term):

  $$P_{k|k-1}^i = F_{k-1}^i P_{k-1}^i {F_{k-1}^i}^T \tag{23}$$

  where $F_k^i$ denotes the Jacobian of $f_k$ with respect to $x_k$ computed at point $(x_k^i, w_k^i)$. One can notice at this point, that the linearization assumption is ruled by initialization. If initial particle covariance $P_0^i$ is small (which is true since $N$ is large), covariance $P_{k|k-1}^i$ can only decrease according to equation (23).

- **Update**
  Using Bayes' rule (4) and linearizing measurement equation yields $x_k^i$ and $P_k^i$ which are updated as an EKF:

$$x_k^i = x_{k|k-1}^i + K_k^i \left( y_k - h_k \left( x_{k|k-1}^i \right) \right) \tag{24}$$

$$K_k^i = P_{k|k-1}^i {H_k^i}^T \left( H_k^i P_{k|k-1}^i {H_k^i}^T + R_k \right)^{-1} \tag{25}$$

$$P_k^i = P_{k|k-1}^i - K_k^i H_k^i P_{k|k-1}^i \tag{26}$$

where $H_k^i$ denotes the Jacobian of $h$ with respect to $x$ in $x_{k|k-1}^i$. Recursive computation of weights is given by:

$$\rho_k^i = \frac{\rho_{k-1}^i \Gamma_{y_k}(h_k(x_{k|k-1}^i), H_k^i P_{k|k-1}^i {H_k^i}^T + R_k)}{\sum_{j=1}^{N} \rho_{k-1}^j \Gamma_{y_k}(h_k(x_{k|k-1}^j), H_k^j P_{k|k-1}^j {H_k^j}^T + R_k)} \tag{27}$$

- **Estimate and its variance**
  State estimate $\widehat{x}_k$ is provided as in the PF case:

$$\widehat{x}_k = \sum_{i=1}^{N} \rho_k^i x_k^i \tag{28}$$

and estimation error covariance is given by:

$$\widetilde{P}_k = \sum_{i=1}^{N} \rho_k^i \left[ \left( x_k^i - \widehat{x}_k \right) \left( x_k^i - \widehat{x}_k \right)^T + P_k^i \right] \tag{29}$$

- **Resampling**
  The degeneration of the algorithm can not be avoided. To enhance exploratory properties, particles are distributed according to the local Gaussian distribution $\Gamma_{x_k}(x_k^i, P_k^i)$. This “spot redistribution“ limits meaningfully the loss of “diversity“ due to heavy weight particles which are selected statistically many times.

## 3 Parallel particle filtering algorithm

From the previous description of the PF and GPF algorithms, it is obvious that these algorithms are inherently parallel since they essentially do a random exploration of the state space by independent entities. As previously noticed, particles only interact via weight normalization, which is only needed for estimate computation and for particle redistribution. Particles are completely independent during the other steps.

From this point of view, the most straightforward way to take advantage of this inherent parallelism is to use a synchronous data parallel approach [2].

It consists in partitioning the set of $N$ particles into several independent separate subsets of nearly the same size. Each subset is assigned to one processor.

In order to describe the parallel algorithm developed, let:

- $P + 1$ be the number of processors.
- $T_j$ denotes the task on the processor $j = 0 \dots P$.
- $n_k^j$ be the number of particle assigned to task $T_j$ at time $k$ ($n_0^j \approx N/P$ at time $k = 0$).
- $\left\{ x_k^{i,j}, \rho_k^{i,j} \right\}_{i=1,\dots,n_j^k}$ be the particle subset of task $T_j$ at time $k$. $x_k^{i,j}$ is the particle state and $\rho_k^{i,j}$ is its weight.

The simplest and most efficient way to compute the estimate is to use the master/slave parallel programming model. Therefore, only one task, hereafter called master task $T_0$, is devoted to this computation. The other tasks $T_{j=1\dots P}$, which are referred to as slave tasks in the sequel, independently proceed to state prediction and unnormalized weight computation. The key point of the algorithm is that only a few scalar exchanges are needed between slave tasks and master task in order to compute the estimate $\widehat{x}_k$ and control degeneration. In fact, as shown below, the data exchanged correspond to aggregated informations on the local subsets of particles. The notations used and the principle of this data parallel approach are depicted on figure 2.

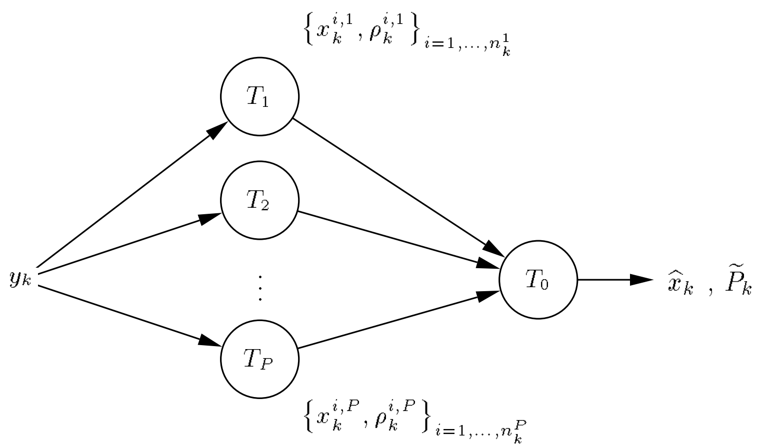



Fig. 2. The data parallel approach principle.

Obviously, this approach efficiently exploits the inherent parallelism of the evolution and update steps of the PF algorithm. Indeed, the slave tasks independently realize the same computations. Therefore a linear speedup can be expected if the communication overhead is kept negligible.

In the following, we examine the two main issues in the parallelization of the PF algorithm, namely the parallel computation of the estimate and the control of degeneration. The former point constitutes the main loop of the algorithm and is discussed in section 3.1, while the latter is discussed in section 3.3.

### *3.1 Algorithm main loop*

At time $k = 0$, each slave task $T_j$ independently initialize its particle subset $\left\{x_k^{i,j}, \rho_k^{i,j}\right\}_{i=1,\ldots,n_j^k}$ according to $p(x_0)$.

Then, all tasks enter in the algorithm main loop. The computations done by the different tasks during one iteration (say iteration $k$) of this main loop are detailed in the following paragraphs. They are also depicted in figure 3 (dashed lines represent communication flows).

#### *3.1.1 Prediction and unnormalized update*

All slave tasks $T_{j=1\ldots P}$ independently proceed to the prediction step according to (1):

$$x_k^{i,j} = f_k(x_{k-1}^{i,j}, w_k^{i,j}) \tag{30}$$

When measurement $y_k$ becomes available, unnormalized weights are recursively updated:

$$\rho_k^{i,j} = \exp\left[-\frac{1}{2}\left(y_k - h(x_k^{i,j})\right)^T R_k^{-1}\left(y_k - h(x_k^{i,j})\right)\right]\rho_{k-1}^{i,j} \tag{31}$$

Then all slave tasks $T_{j=1\ldots P}$ compute the following quantities:

$$Z_k^j = \sum_{i=1}^{n_k^j}\rho_k^{i,j} \quad , \quad X_k^j = \sum_{i=1}^{n_k^j}\rho_k^{i,j}x_k^{i,j}$$
$$\widetilde{P}_k^j = \sum_{i=1}^{n_k^j}\rho_k^{i,j}x_k^{i,j}x_k^{i,j\,T} \text{ and } Q_k^j = \sum_{i=1}^{n_k^j}\left(\rho_k^{i,j}\right)^2$$

These quantities represent aggregated data on the local subset of particles. $Z_k^j$ is the unnormalized weight of task $T_j$ and $X_k^j$ is its unnormalized local estimate. $\widetilde{P}_k^j$ and $Q_k^j$ are respectively used to compute the estimate error covariance $\widetilde{P}_k$ and to control degeneration. All slave tasks $T_j$ send these data to the master task.

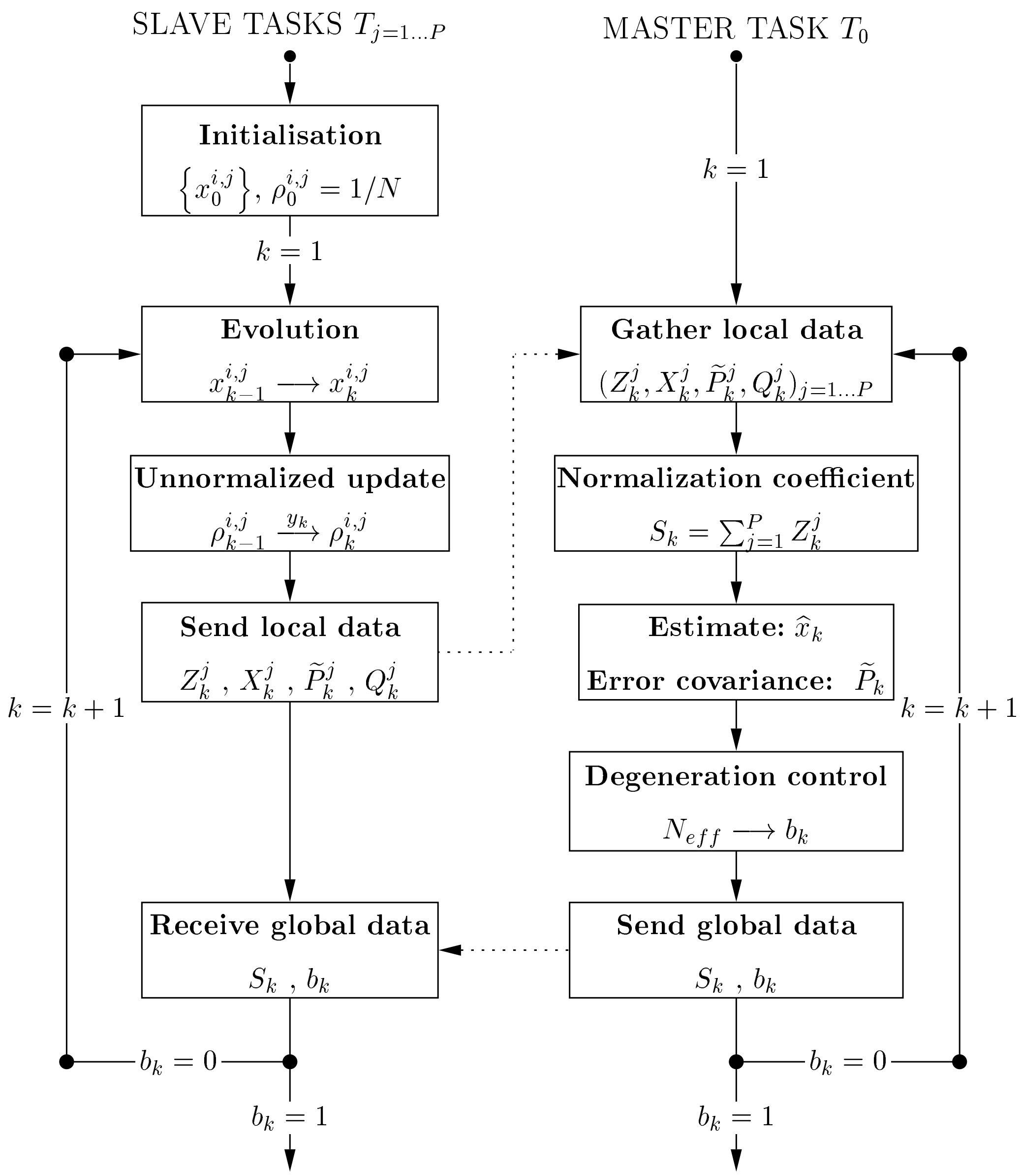



Fig. 3. The parallel estimate computation.

### *3.2 Estimate and error covariance*

The master task $T_0$ gathers the local informations $(Z_k^j, X_k^j, \widetilde{P}_k^j, Q_k^j)$ sent by slave tasks $T_j$. Then, it computes the normalization coefficient of weights:

$$S_k = \sum_{j=1}^{P} Z_k^j \tag{32}$$

and it computes the estimate $\widehat{x}_k$ and its error covariance $\widetilde{P}_k$ as follows (see eq. (13) and eq. (14)):

$$\widehat{x}_k = \frac{\sum_{j=1}^{P} X_k^j}{S_k} \qquad \text{and} \qquad \widetilde{P}_k = \frac{\sum_{j=1}^{P} \widetilde{P}_k^j}{S_k} - \widehat{x}_k^T \widehat{x}_k \tag{33}$$

### *3.2.1 Degeneration control and weight normalization*

To check the degeneration of the set of particles, master task $T_0$ computes the heuristic effective sample size indicator (see eq.(17)):

$$N_{eff} \approx \frac{(S_k)^2}{\sum_{j=1}^{P} Q_k^j} \tag{34}$$

If the effective sample size gets lower than a given threshold, a redistribution of the whole set of particles has to be performed. In this case, the master task sets a boolean flag $b_k$ to 1. Then, it sends $S_k$ and $b_k$ to all slave tasks $T_j$.

Upon reception of this message, slave tasks check the value of $b_k$. If $b_k = 0$, no redistribution is needed and the tasks proceed to the next iteration of the main loop. Otherwise, a redistribution is performed as described in section 3.3.

## *3.3 Redistribution and load balancing*

If a redistribution is needed ($b_k = 1$), a naive solution would be to proceed as follows. Each slave task $T_j$ first normalize its local weights $\rho_k^{i,j}$ by dividing it by $S_k$ and then the particle subsets $\left\{x_k^{i,j}, \rho_k^{i,j}\right\}$ are locally (i.e. without any communication) redistributed as detailed in 2.2. The major drawback of this crude approach is that the size $n_k^j$ of the subsets obtained on different processors will not be balanced. Furthermore, a single task will asymptoticaly gathers the whole particle set.

A simple and efficient scheme to achieve a balanced load after redistribution is to exchange particles between slave tasks. The load balancing procedure, detailed in the following, is illustrated in figure 4.

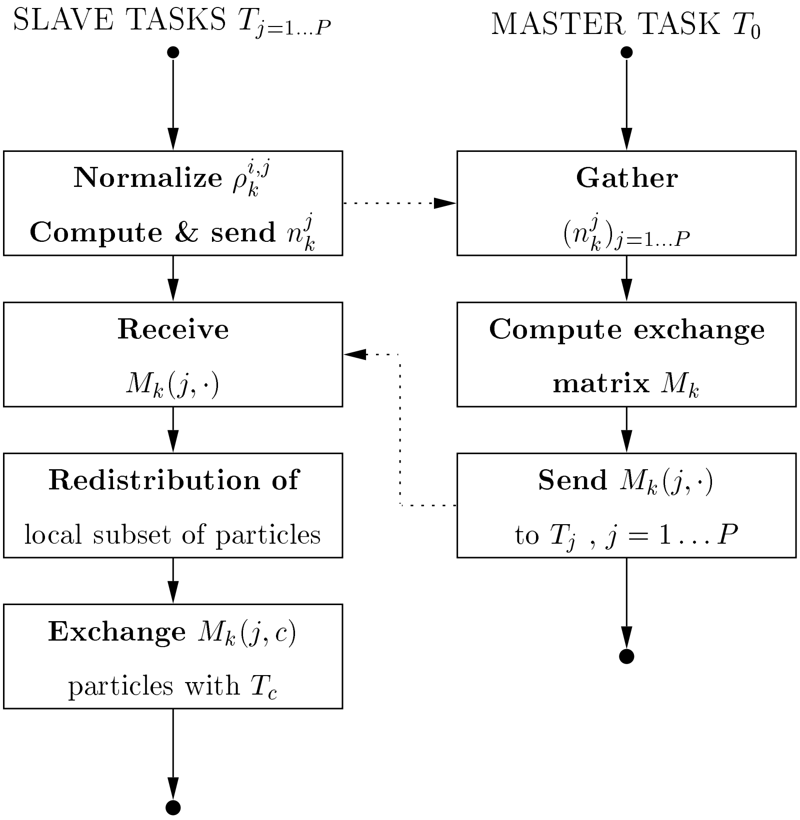



Fig. 4. Load balancing procedure.

The number of particles to be exchanged between each couple of slave tasks is computed by the master task. To this end, each slave task $T_j$ computes the size $n_k^j$ of the particle subset that would be obtained after redistribution, i.e:

$$n_k^j = \sum_{i=1}^{n_{k-1}^j} round\left[\rho_k^{i,j} N\right] \tag{35}$$

and send it to the master task.

These numbers are gathered by the master task which computes the total number $N_r = \sum_{j=1}^{P} n_k^j$ of particles obtained after redistribution, and the deviation $\Delta n_k^j$ of each slave task $T_j$ from the ideal load $round(N_r/P)$:

$$\Delta n_k^j = n_k^j - round(N_r/P) \tag{36}$$

Using these deviations, an exchange matrix $M_k$ is computed by the master task. $M_k(l,c)$ denotes the number of particles exchanged between task $T_l$ and task $T_c$. If $M_k(l,c) > 0$, the particles flow goes from $T_l$ to $T_c$. Obviously, $M_k(l,c) = -M_k(c,l)$ as the amount of particles sended and received have to be equal. The matrix $M_k$ is computed by the master task according to the following algorithm.

**Algorithm 1 (Computation of the exchange matrix)**
**for** $l = 1 \ldots P$ **do**
  **if** $(\Delta n_k^l > 0)$ **then** /* $T_l$ *sends particles* */
    **for** $c \neq l$ **do**
      **if** $(\Delta n_k^c < 0)$ **then** /* $T_c$ *receive particles from* $T_l$ */
        **if** $(\Delta n_k^l + \Delta n_k^c \leq 0)$ **then** /* $T_l$ *send all to* $T_c$ */
          $M_k(l,c) = \Delta n_k^l$ *and* $M_k(c,l) = -M_k(l,c)$
          $\Delta n_k^c = \Delta n_k^c + \Delta n_k^l$
          $\Delta n_k^l = 0$
        **else** /* $T_c$ *receive all from* $T_l$ */
          $M_k(l,c) = -\Delta n_k^c$ *and* $M_k(c,l) = -M_k(l,c)$
          $\Delta n_k^l = \Delta n_k^l + \Delta n_k^c$
          $\Delta n_k^c = 0$
        **end if**
      **end if**
    **end for**
  **end if**
**end for**

Once this matrix $M_k$ has been computed, the master task sends to each slave task $T_j$ the $j^{\text{th}}$ row of the exchange matrix, denoted in the sequel $M_k(j,\cdot)$. Upon reception of the row $M_k(j,\cdot)$, the slave task $T_j$ performs the local redistribution of its subset of particles. Then, it exchanges particles with other slave tasks according to the following algorithm.

**Algorithm 2 (Exchange policy of Task $T_j$)**
**for** $c = 1 \ldots P$ , $c \neq j$ **do**
  **if** $M_k(j,c) > 0$ **then**
    **send** $M_k(j,c)$ *particles to* $T_c$
  **else if** $M_k(j,c) < 0$ **then**
    **receive** $M_k(j,c)$ *particles from* $T_c$
  **end if**
**end for**

After particle exchanges, the number of particles of task $T_j$ is:

$$n_k^j = \sum_{i=1}^{n_{k-1}^j} round\left[\rho_k^{i,j} N\right] \; + \; \sum_{c=1}^{P} M_k(c,j) \tag{37}$$

This exchange procedure leads to a balanced load ($n_k^j$ nearest from $N/p$) after redistribution. This procedure of course induces a greater complexity of the redistribution step. However, since redistribution are infrequent, its cost is very low with respect to the benefits of a balanced load.

A further improvement of this procedure, which minimizes the communication cost, is to reverse the exchange and redistribution procedures. Thus, instead of exchanging a large amount of duplicated particles of light weights (around $1/Nc$), it is possible to exchange a lower number of particles, but having heavier weights.

## 4 Application of parallel GPF to Doppler-Bearing Tracking

In this section, we report results obtained with a MPI implementation of the parallel PF algorithm on the passive SONAR tracking problem. Section 4.2 details the stochastic target motion model and related measurements equations. This non linear and non Gaussian problem provides a convincing benchmark for evaluating parallel GPF performance. Three scenarios are studied and GPF's efficiency is illustrated on the most challenging one. Time performance of parallel GPF on different multi-processors architectures for these scenarios is detailed in section 4.1

### *4.1 Doppler-Bearing Tracking for maneuvering target*

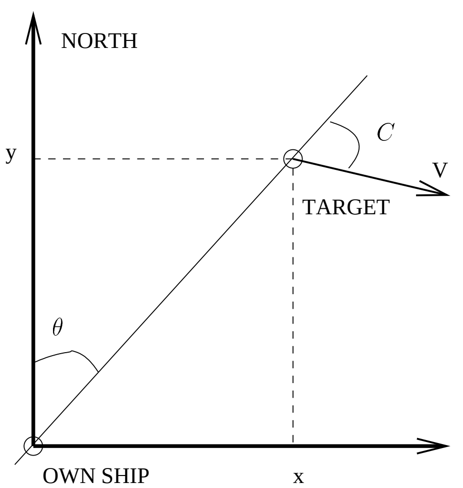



Fig. 5. Doppler-Bearing tracking geometry

Consider the source-observer encounter depicted in figure 5. The source, located at the coordinates $(x_k, y_k)$, moves at speed $V_k$, course $C_k$ and is supposed to emit a constant acoustic frequency denoted $F^0$. Its state is then defined by the following vector:

$$X_k = \left[ x_k \; y_k \; V_k \; C_k \; F_k^0 \right]^T \tag{38}$$

Own ship motion is not needed to ensure observability as long as the target is not moving radially, and in the sequel, observer is assumed motionless. Each component of the target state vector evolves according to:

$$\begin{aligned}
x_k &= x_{k-1} + TV_k \sin C_k && (39)\\
y_k &= y_{k-1} + TV_k \cos C_k && (40)\\
V_k &= V_{k-1} + TA_k && (41)\\
C_k &= C_{k-1} + T\Omega_k && (42)\\
F_k^0 &= F_{k-1}^0 && (43)
\end{aligned}$$

where $T$ is the sampling rate, $A$ and $\Omega$ are respectively acceleration and rotation control. These controls could be modeled as Gaussian noise. But that kind of model is unrealistic since Brownian motion can not fit a ship trajectory. Most of the time, target travels with a constant velocity vector and sometimes a maneuver occurs. Thus, the trajectory is a collection of consecutive legs (constant velocity motion) separated by a maneuver (change of course or speed). A very realistic behavior of source maneuver is provided by modeling these controls using two Poisson process $N_k^1$ and $N^2$ defined by:

$$\begin{cases} P(N_i = 1) = \lambda_i \\ P(N_i = 0) = 1 - \lambda_i \end{cases} \quad \text{for } i = 1, 2 \tag{44}$$

where $\lambda_i$ denotes the rate of the Poisson process $N_i$. Moreover, each control belongs to a finite set: $[-\Omega_{max}, \Omega_{max}]$ for rotation and $[-A_{max}, A_{max}]$ for acceleration. Stochastic evolutions of these controls are given by:

$$\Omega_k = \left(N_1 U_{\Omega_{max}} + (1 - N_1)\Omega_{k-1}\right)(1 - N_2) \tag{45}$$

and

$$A_k = \left(N_1 U_{A_{max}} + (1 - N_1)A_{k-1}\right)(1 - N_2) \tag{46}$$

where $U_{u_m}$ denotes an uniform distribution on the set $[-u_m, u_m]$. $\lambda_1$ corresponds to the average frequency of maneuver and $\lambda_2$ to maneuver duration. A typical control trajectory and the corresponding target motion are depicted in figure 6 (frequencies were set for an average of 10 minutes maneuver each hour).

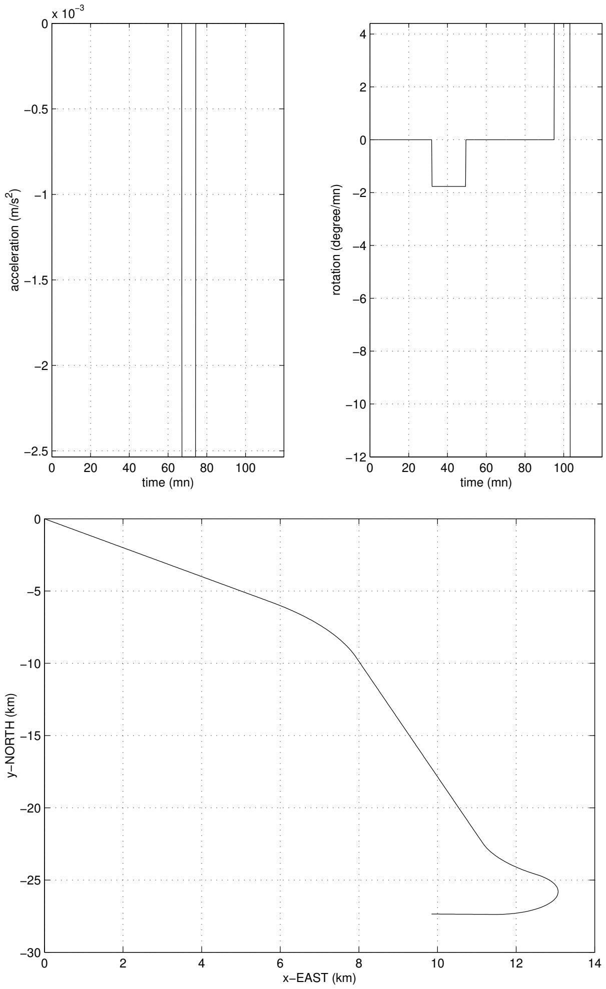



Fig. 6. Stochastic motion of target

The measurement vector is composed of the bearing relative to the y-axis (north) of the observer and the Doppler shifted frequency (related to radial speed), i.e:

$$\begin{bmatrix} y_k^\theta \\ y_k^f \end{bmatrix} = \begin{bmatrix} \arctan \dfrac{x_k}{y_k} \\ F_k^0 \left( 1 - \dfrac{V_k \left( x_k \sin C_k + y_k \cos C_k \right)}{c\sqrt{x_k^2 + y_k^2}} \right) \end{bmatrix} + \begin{bmatrix} v_k^\theta \\ v_k^f \end{bmatrix} \tag{47}$$

where $c$ is the speed of sound in water, $v^\theta$ and $v^f$ are two uncorrelated zero-mean Gaussian noise of respective variance $R_\theta$ and $R_f$. This non linear and non Gaussian filtering problem is solved using a GPF. Computation of the GPF is a straightforward application of section 2.3 and the crucial initialization

Table 1
Parameters of the 3 scenarios.

| | $SCNA$ | $SCNB$ | $SCNC$ |
|---|---|---|---|
| $N$ | 500 | 5000 | 100 000 |
| Duration (hour) | 1 | 1 | 2 |
| Sample rate (seconds) | 1 | 10 | 10 |
| Number of sample | 3600 | 360 | 720 |
| Initial state | | | |
| $x_0\,(km)$ | 14.1 | 10 | 29.5 |
| $y_0\,(km)$ | -14.1 | 0 | 5.2 |
| $V_0\,(m/s)$ | 7.9 | 10.3 | 10.3 |
| $C_0\,(^o)$ | 0 | 0 | 270 |
| $F_0\,(Hz)$ | 300 | 300 | 300 |
| Maneuver | none | rotation | rotation |
| Initial time (mn) | | 20 | 20 |
| Final time (mn) | | 29 | 57 |
| Rotation rate ($^o/mn$) | | 10 | 3 |
| Course after maneuver | | | |
| $C_0\,(km)$ | | 270 | 0 |

details can be found in [9].

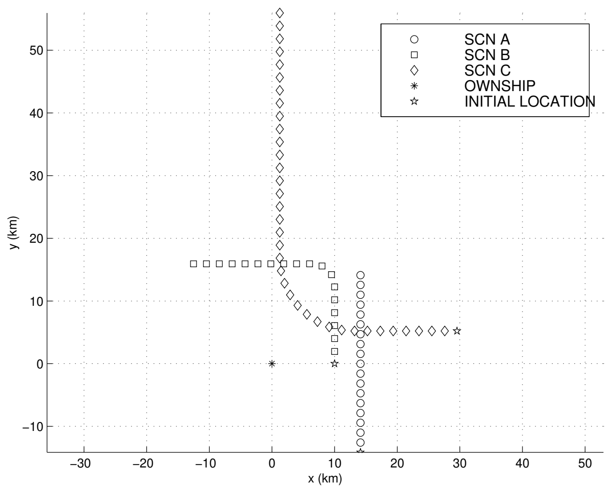



Fig. 7. Scenarios

Performance of the parallel algorithm is studied on three scenarios ($SCNA$, $SCNB$ and $SCNC$) depicted on figure 7 and detailed in table 1.

For all scenarios, standard deviations of bearing and frequency noises are respectively $R_\theta = 1^o$ and $R_f = 0.5Hz$. Scenario A and B can be treated on-line as $N$ is relatively small. The goal of these two scenarios is to study the speedup of the parallel GPF for fine grain applications (especially Scenario A). Last scenario C is far harder as target mainly moves radially except during maneuver: PF has to perform track and detect simultaneously when maneuvers

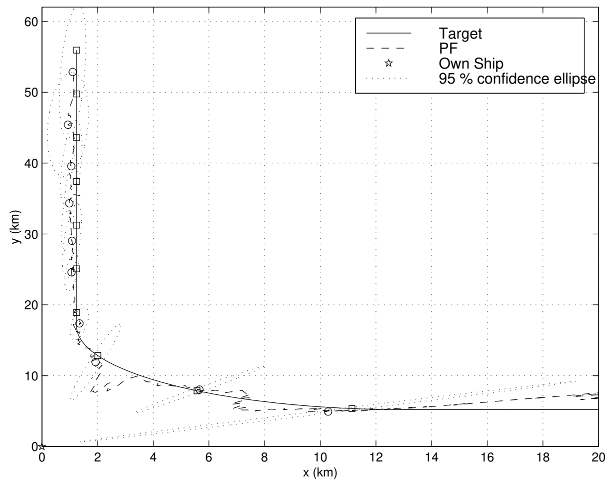



Fig. 8. PF location estimation on SCN C

occur. Consequently, an efficient estimate can only be delivered with $100K$ particles and GPF can not be run online. Indeed, during track and detect stages, the state PDF is multi modal. Thus, for a lower number of particles, some modes disappear prematurely due to redistributions and the resulting estimate can be biased.

The PF's efficiency is illustrated on scenario C (see figures 8 and 9). As one can see on figure 8, before maneuver, location uncertainty is important. Despite this difficulty, PF correctly performs tracking : real target location (square) always lies within the 2-sigma ellipse confidence interval deduced from GPF estimate (circle) and error covariance (dotted). Moreover, after maneuver, estimability gets weaker since target is moving almost radially and PF's error covariance increases consequently. Figure 9 details estimate (dashed) of target state (plain) and 95 % confidence interval (dotted). Course tracking is particularly relevant.

Results obtained on scenario A and B are obviously better. Scenario A perfectly mets Cramér-Rao lower bound, and PF converges before maneuver which is perfectly estimated on scenario B.

### *4.2 Time performances*

The results presented in the following have been obtained on the following parallel computers (PE is the acronym of Processing Element):

- **SGI/Cray Origin2000**. The Origin2000 is a 32-PE CC-NUMA multiprocessor architecture. The processors are 64-bit R10000 (250 MHz). Nodes

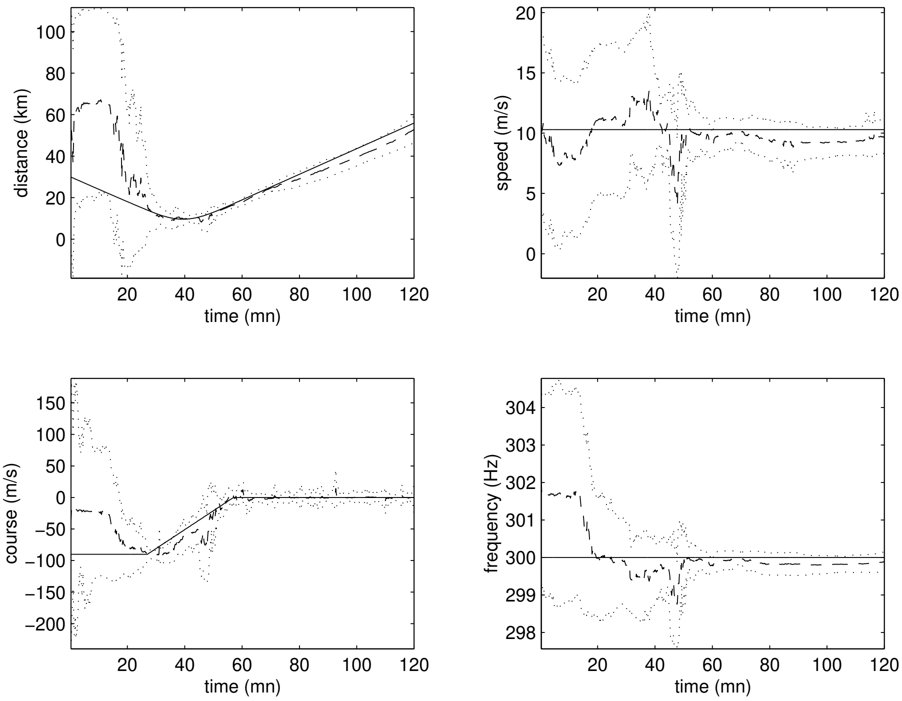



Fig. 9. PF state estimation on SCN C

Table 2
Average time per iteration (ms) on the Origin2000 computer

| Scenario | SCN A | | | | | SCN B | | | | | SCN C | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # PE | 4 | 8 | 16 | 24 | 32 | 4 | 8 | 16 | 24 | 32 | 4 | 8 | 16 | 24 | 32 |
| Time | 18.8 | 9.4 | 5.2 | 3.7 | 3.0 | 186.5 | 92.8 | 47.5 | 32.5 | 25.1 | 3763 | 1877 | 949 | 631 | 485 |

are interconnected by CrayLinks switches (hypercube topology, 800 MB/s full duplex).

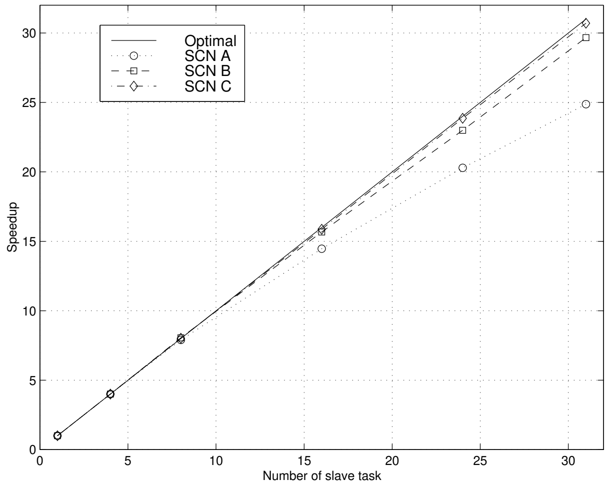
- **Cluster of PC**. This architecture is a cluster of 5 dual Intel-PII-233 (10 PEs) interconnected by a 100Mbps Ethernet TCP/IP network. The goal is to investigate the capabilities of this kind of cheap PC clusters to handle efficiently such parallel methods.

Each scenarios were simulated 50 times on different measurements sequences in order to obtain average time and speedup performance of the parallel algorithm on both computers. Table 2 and 3 present the average number of milliseconds needed to complete an iteration of the parallel PF algorithm for all scenarios. As expected, the performance of the Origin 2000 is far greater than PC cluster (around 60 %). However, PC cluster can also handle online computation of the GPF for scenario C with only 4 slave tasks.

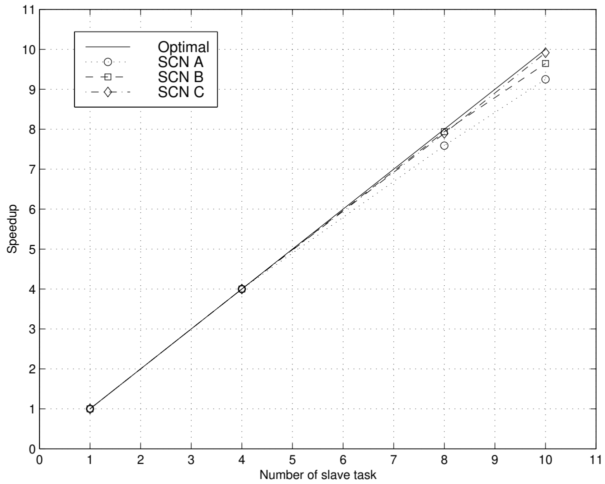
Finally,the speedups obtained have been plotted on figure 10 (they are computed with respect to the number of slave tasks). The SGI Origin2000 has near

Table 3
Average time per iteration (ms) on the PC cluster

| Scenario | SCN A | | | SCN B | | | SCN C | | |
|---|---|---|---|---|---|---|---|---|---|
| # PE | 4 | 8 | 10 | 4 | 8 | 10 | 4 | 8 | 10 |
| Time | 29.6 | 15.6 | 12.8 | 301.7 | 152.0 | 125.1 | 5660 | 2867 | 2283 |




*Origin* 2000




*PC cluster*

Fig. 10. Average speedup with respect to the number of slave tasks.

linear speedups up to 32 PE (the efficiency with 32 PE is greater than 80% for scenario A). PC cluster has been tested up to 10 slave tasks. The speedup of the small grain scenario A incurves earlier in comparison to the Origin 2000 computer. This is mainly due to the poor performances of Ethernet networks.

Several conclusions can be drawn from these results:

- Passive SONAR tracking using a GPF can be computed on-line on a cheap computer architecture, i.e PC cluster.

- For the fine grain scenario A, with 32 PE, 15 particles are assigned to each task and efficiency remains higher than 80 %. One would then expect, an increasing speedup up to a far larger number of PE.

- Speedup can not increase indefinitely. If the computing grain gets really low, other versions of the parallel algorithm have to be considered in order to minimize the number of communications between slave and master tasks:

(1) The exchange matrix could be computed “a priori“ by the master task without any communication using the following approximation (see formula 35):

$$n_k^j \approx round\left[\frac{Z_k^j}{S_k} Nc\right]$$

With this approximation, the master task doesn't need to gather the values $n_k^j$, $j = 1, \ldots, P$. Hence the communication cost is significantly reduced. Of course, loads would not be perfectly balanced due to the errors introduced by this approximation. However, for a large number of PE, it probably yields significant savings in computing time.

(2) In practice, there is no need to deliver the estimate at each iteration. Hence, communication cost can be reduced if slave tasks send their aggregated data periodically and not at each iteration. But, the period has to be chosen wisely as no redistribution can be done between two successive periods.

## 5 Conclusion

We have shown in this article that a straightforward data parallel approach can be used to efficiently exploit the inherent parallelism of the particle filtering method.

In the passive SONAR context, the algorithm developed has a very good scalability and time performances are compatible with the real-time constraints. Moreover, cheap PC clusters (i.e 4 or 5 biprocessors) with high-speed communication links (Ethernet 100 MHz) can efficiency handle on-line computation. We are currently building a PC cluster constituted of 800 MHz Pentium III and 1 Gigabit switch (Myrinet). Such an architecture will obviously be much faster than the one tested and one can expect reasonably as good performances as the SGI Origin 2000 for the parallel PF. It can be expected that similar results can be obtained for more complex systems since this complexity will mainly results in coarser grain applications.

We are currently developing new approaches to extend the scalability of the method to hundreds of processors. The goal is to handle the larger number of particles (greater than 50 K) necessary in very low signal to noise ratio and high sampling rate applications (such as RADAR signal processing where sampling rate is around 200 $\mu s$).


## Acknowledgement

This research has been supported by the CALMIP supercomputing center and CICT of Paul Sabatier University, Toulouse.


## References


[1] J.S LIU and R. CHEN *Blind deconvolution via sequential imputations* Journal of the American Statistical Association, Vol. 430, 1995.

[2] O. BRUN, J. M. GARCIA *Real-Time Parallel Particle Filtering* Fourteenth International Symposium of Mathematical Theory of Networks and Systems, Perpignan, France, 2000.

[3] G. RIGAL. *Filtrage non-linéaire, résolution particulaire et applications au traitement du signal.* PhD Thesis, Université Paul Sabatier - LAAS/CNRS, 1993.

[4] H. AKASHI and H. KUMAMOTO *Random Sampling Approach to State Estimation in Switching Environments.* Automatica, 13, 429-434, 1977.

[5] A. DOUCET, S. J. GODSILL and C. ANDRIEU *On Sequential Monte Carlo Sampling Methods for Bayesian Filtering* Statistics and Computing, vol. 10, no. 3, pp. 197-208, 2000.

[6] J. E. HANDSHIN and D. Q. MAYNE *Monte Carlo Techniques to Estimate the Conditional Expectation in Multi-stage Non-linear Filtering.* International Journal of Control, 9, 547-559, 1969.

[7] J. E. HANDSHIN *Monte Carlo Techniques for Prediction and Filtering of Non-linear Stochastic Processes.* Automatica, 6, 555-563, 1970.

[8] P. DEL MORAL *Résolution particulaire des problèmes d'estimations et d'optimisations non-linéaires* PhD Thesis, Université Paul Sabatier - LAAS/CNRS, 1994

[9] V. TEULIERE *Contribution au filtrage de Volterra et à l'estimation particulaire - Application aux transmissions électroniques et acoustiques* PhD Thesis, Université Paul Sabatier - LAAS/CNRS, 2000.

[10] H. CARVALHO and P. DEL MORAL and A. MONIN and G. SALUT *Optimal nonlinear filtering in GPS/INS integration* IEEE Transactions on Aerospace and Electronic Systems, Vol. 33, No 3, 1997.

[11] N.J GORDON and D.J SALMOND and A.F.M SMITH *Novel approach to nonlinear/non-Gaussian bayesian state estimation* IEE-Proceedings-F, Vol. 140, No 2, 1993.

[12] J.C. NOYER *Traitement non-linéaire du signal radar par filtrage particulaire* PhD Thesis, Université Paul Sabatier - LAAS/CNRS, 1996.

[13] T. HUILLET and G. SALUT *Interprétation des équations du filtrage non-linéaire* Séance du GdR Automatique du CNRS (Pôle non linéaire), 1989.

[14] M.A. CHAMON *Filtrage particulaire et ouverture synthétique inverse sur cibles radar non coopératives* PhD Thesis, Ecole Nationale Supérieure de l'Aéronautique et de l'espace - LAAS/CNRS, 1996.

[15] J-C NOYER, G. SALUT *Optimal non-linear radar tracking of non-Gaussian manoeuvering targets* to appear in IEEE Aeorospace and Electronic Systems

[16] A. MONIN, G. RIGAL, P. CHARRON, *A New Technology for LORAN-C Receivers* International Symposium on Integration of LORAN-C/Eurofix and EGNOS/Galileo, Bonn, Germany, 22-23 march 2000.

[17] J. AIDALA & S. HAMMEL. *Utilisation of modified polar coordinates for bearing only-tracking.* IEEE Transactions on Automatic Controls, vol. 28, no. 3, pages 283–293, 1983.

[18] Y.T CHAN & S.W. RUDNICKI. *Bearing-only tracknig using instrumental variables.* IEEE Transactions on Aerospace and Electronic Systems, vol. 28, no. 4, pages 1076–1082, octobre 1992.

[19] J.P LE CADRE & O. TREMOIS. *Bearing-only tracking for manoeuvring sources.* IEEE Transactions on Aerospace and Electronic Systems, vol. 34, no. 1, pages 179–192, janvier 1998.

[20] S.C. NARDONE, A.G. LINDGREN & K.F. GONG. *Fundamental properties and performance of conventional Bearing-only target motion analysis.* IEEE Transactions on Automatic Control, vol. AC-29, no. 9, sept 1984.

[21] PASSERIEUX, PILLON, BLANC-BENNON & JAUFRET. *Target motion analysis with bearings and frequencies measurements.* Dans Proceedings of the 22nd Asilomar Conference, Pacific Grove,CA, Nov 1988.

[22] JAZWINSKI. Stochastic processes filtering theory. Academic Press, 1970.

[23] S.K. RAO. *Algorithm for detection of manoeuvring targets in bearing-only passive target tracking.* IEE Proceedings Radar, Sonar and Navigation, vol. 146, no. 3, pages 141–146, juin 1999.