# A Particle Filter to Estimate non-Markovian States

Bas Terwijn, Josep M. Porta and Ben J.A. Kröse
*IAS Group, Informatics Institute, University of Amsterdam*
Kruislaan 403, 1098SJ, Amsterdam, The Netherlands
{bterwijn,porta,krose}@science.uva.nl

**Abstract.** Common particle filter algorithms are unable to differentiate between sensor noise and kidnap situations. These two problems are usual in realistic applications and, in general, ad-hoc heuristic are developed to deal with them. In this paper, we present a particle filter algorithm where the state is enriched with information about the coherence of the sensor readings. This additional information allows us to tackle non-Markovian environments and, thus to deal in a unified way with sensor noise and kidnap situations. The experiments we report show the advantages of this new particle filter approach w.r.t. existing ones.

## 1 Introduction

Probabilistic formalisms are very useful to represent the interaction between autonomous systems and complex environments. Recently, particle filters [1] have been widely used to implement probabilistic frameworks [4, 10, 3, 11]. In all these fields, particle filters are applied in the context of a Markov chain trying to estimate the state of a given system using the information provided by observations probabilistically related with that state.

Particle filters have many advantages over other probabilistic state-estimation methods (such as Kalman Filters): they are easy to implement, they are robust to random noise in the observations, and they can track many hypothesis simultaneously. However, particle filters also have some well-know weak points as, for instance, their high computational cost. Another drawback is that particle filter algorithms rely on the Markov assumption and this assumption does not hold in many realistic environments. The consequence is that particle filter algorithms can, in some cases, provide wrong state estimations.

In this paper, we present a particle filter algorithm that is able to deal with non-Markovian environments. In Section 2, we review the basic particle filter algorithm and, in Section 3, we describe the problems of this approach to deal simultaneously with *kidnap* (or *tele-portation*) situations and with noisy sensor readings. We show that this is due to the Markov assumption on the environment, that is explicitly used in the particle filter framework. In Section 4, we present a particle filter algorithm where the state representation is extended to include information about previous observations. This makes our algorithm able to deal with non-Markovian environments and, thus, to deal with the kidnap situations and with the noisy sensors in a unified way. In section 5, we present experiments to compare our proposal with existing algorithms and we finish the paper, summarizing our work and extraction some conclusions out of it. Along the paper, we use the robot localization problem as an example, but all the reasonings can be easily translated to other fields.

## 2 Markov State Estimation: Particle Filtering

In many cases, we need to estimate the stochastic (hidden) state of a given system at time $t$, $x_t$ using observations (stochastically) related with that state $\{y_1, \ldots, y_t\}$ and the control actions issued to (stochastically) change the state $\{u_1, \ldots, u_t\}$[1]. Formally, what we want to estimate the posterior $p(x_t|\{u_1, y_1, \ldots, u_t, y_t\})$. The Markov assumption says that observations only depend on the current state and that the current state only depend on the previous state and the last executed action, Thus, the current state can be updated from the previous state estimation taking into account only the last executed action $u_t$ and the current observation $y_t$. Applying Bayes

$$p(x_t|u_t, y_t) \propto p(y_t|x_t)\, p(x_t|u_t), \tag{1}$$

where the probability $p(x_t|u_t)$ can be computed propagating from $p(x_{t-1}|u_{t-1}, y_{t-1})$

$$p(x_t|u_t) = \int p(x_t|u_t, x_{t-1})\, p(x_{t-1}|u_{t-1}, y_{t-1})\, dx_{t-1}. \tag{2}$$

Equations 1 and 2 define a recursive system to estimate the state from a uniform initial distribution $p(x_0)$.

The probability $p(x_t|u_t, x_{t-1})$ is called the *action model* and $p(y_t|x_t)$ is the *sensor model*. In our robot localization system [5, 11], the state is the pose (position and orientation) of the robot, the actions are the movements of the robot, and the observations are images taken by the robot's camera. The action model is inferred from odometry and we compute the sensor model using an appearance map of the environment.

The shape of the prior $p(x_t|u_t, y_t)$ can vary along time and, consequently, we have use a method able to approximate arbitrary probability distributions. One of these methods is the particle filtering. In this approach, the continuous posterior $p(x_{t-1}|u_{t-1}, y_{t-1})$ is approximated by a set of $I$ random samples, called particles, that are positioned at points $x^i_{t-1}$ and have weights $\pi^i_{t-1}$. Thus, the posterior is

$$p(x_{t-1}|u_{t-1}, y_{t-1}) = \sum_{j=1}^{J} \pi^j_{t-1}\, \delta(x_{t-1}|x^j_{t-1}),$$

where $\delta(x_{t-1}|x^j_{t-1})$ represents the delta function centered at $x^j_{t-1}$.

The central issue in the particle filter approach is how to obtain a set of particles to approximate $p(x_t|u_t, y_t)$ from the set of particles $x^j_{t-1}$, $j \in [1, J]$ approximating $p(x_{t-1}|u_{t-1}, y_{t-1})$. In the auxiliary particle filter [8, 11] the probability of a particle to survive is proportional to $\pi^j_{t-1}\, p(y_t|x_t)$. In this way, we favor the selection of particles with high prior and also with high likelihood.

## 3 The Kidnap-Noise Dilemma

A (real) *kidnap* (or *teleportation*) situation occurs when the state of the system is suddenly changed by an action not considered in the action model.

A *virtual kidnap* situations can occur even with perfect action and sensor models. Suppose we use $n$ particles to track two hypotheses, $x_1$ and $x_2$, that are equally valid according to the

[1]In our notation, the system goes through the following sequence $x_0 \xrightarrow{u_1} (x_1, y_1) \xrightarrow{u_2} \ldots \xrightarrow{u_t} (x_t, y_t)$.

sensor model and that at time $t$ we have $z_t^1$ particles at around hypotheses $x_1$ and $z_t^2 = n - z_t^1$ particles at $x_2$. After sampling new particles, the probability of getting a specific value for $z_{t+1}^1$ given $z_t^1$ and $n = z_t^1 + z_t^2$ follows a binomial distribution

$$p(z_{t+1}^1 | z_t^1, n) = \left(\frac{z_t^1}{n}\right)^{z_{t+1}^1} \left(\frac{n - z_t^1}{n}\right)^{n - z_{t+1}^1} \frac{n!}{z_{t+1}^1!\,(n - z_{t+1}^1)!}. \tag{3}$$

According to this distribution, $p(z_{t+1}^1|z_t^1, n)$ with $z_t^1 = 0$ is not zero, but $p(z_{t+1}^1|z_t^1, n)$ for $z_{t+1}^1 > 0$ and $z_t^1 = 0$ is zero. In other words, the sampling can produce states with $z_{t+1}^1 = 0$ (or $z_{t+1}^2 = 0$) but it is unable to escape from these states. Therefore, if we repeat the sampling long enough, $z_t^1$ (or $z_t^2$) would become 0. For instance, with $n = 200$ and $z_0^1 = 100$, after 52 sampling steps, the more likely particle distributions are those with $z_t^1 = 0$ and $z_t^1 = 200$. Thus the system would eventually converge to a single hypothesis and there is 50% of changes to converge to the wrong one. This situation can be regarded as a *virtual kidnap*. Virtual kidnap can be minimized by enlarging the number of particles but this results in high computational costs.

A complete particle filter algorithm must be able to deal with real/virtual kidnap situations. In those situations, there is a mismatch between the state estimation and the areas with high probability according to the sensor model. As the number of particles goes to infinite, there would be always one particle close enough to the areas with high probability according to the sensor model, and the particle filter would be able to recover from the kidnap. However, for efficiency reasons, we are not likely to use so many particles and, thus special heuristic strategies have to be developed to deal with the kidnapping problem. A couple of such heuristics can be found in the literature: the *re-sample* strategy and the *sample-from-likelihood* one.

The *re-sampling* strategy [9] consists in re-sampling all particles from scratch after a given period without overlap between the prior and the likelihood. This strategy has the inconvenient that a series of completely noisy sensor reading triggers the *re-sampling* without any need. Additionally, when re-sampling particles, there is no guarantee that we use a valid sensor reading and not a noisy one. In this second case, a new re-sample needs to be executed few time steps later slowing down the convergence of the particle filter.

Another simple solution to the kidnap problem is the *sample-from-likelihood* (or *sensor resetting localization*) strategy that consists on sampling part of the new particles directly from the likelihood [7, 6]. This strategy is able to solve the kidnap problem, but it causes other problems when there is noise in the sensor readings.

Noise in the sensor readings can produce wrong values for the sensor model (i.e., $p(x|y)$ distributions with high values on points where the robot is not). In [2], outliers in the sensors readings are filtered out but this clearly limits capacity of reaction in kidnap situations.

If the *sample-from-likelihood* strategy is used, then two coherent (i.e., giving high probably to the same area of the state space) consecutive noisy readings would translate all particles to a wrong state. This can be regarded as an efficient response to a kidnap situation, but in general this produces *false kidnaps*: moving all particles where the robot is not. The system would recover from that false kidnap when new observations are obtained, but the result is an undesired oscillation in the estimation of the state.

None of the heuristics used to enhance particle filters offers a proper way to differentiate between a kidnap situation and a sequence of coherent noisy sensor readings.

## 4 A Particle Filter with Memory

The kidnap-noise confusion is due to the Markov assumption. That assumption states that sensor readings are conditionally independent given the state. Therefore we have that

$$p(y_t|x_t, x_{t-1}, y_{t-1}) = p(y_t|x_t).$$

Thus, the noise in consecutive sensor readings is uncorrelated. Using this, we can easily differentiate between a kidnap situation and noise: noise has a random distribution and in a kidnap situation the sensor model consistently indicates a new state for the system. However, in many problems, if we obtain a noisy observation $y^n$, it is likely to be observed again next time slices. So, in those cases we have that $p(y^n|x_t, x_{t-1}, y^n) > p(y^n|x_t)$ and, thus, the Markov assumption does not hold. For example, in the context of robot localization this situation happens when the robot is moving in dynamic environments: new objects in the environment, people moving around, etc are likely to corrupt the robot's observations for many time slices with highly correlated noise.

The usual way to deal with non-Markovian environments is to augment the state description with some memory (i.e., information about previous time slices) so that the problem becomes Markovian. Following this line, what we propose is to enrich the state description storing for each particle, its position and its weight (as usual), but also its *coherence*. The *coherence* gives us information about whether or not the corresponding particle has been recently in agreement with the sensor model. Particles with high *coherence* provide very confident estimations on the state that should not be forgotten just because of few noisy sensor readings. The coherence of particles created due to noisy sensor readings would never grow and these particles would soon be deleted. In the case of a kidnap, the coherence of the particles sampled on the new state would grow until they become the best estimation for the state.

Our approach has some points in common with the *sample-from-likelihood* strategy described above. This strategy can be regarded as a two time slices memory strategy since two consecutive coherent observation around a given state produce the displacement of all particles to that new state. What we propose in this paper can be seen as an extension of this mechanism where we can adjust the number of coherent observations necessaries to change the state estimation.

### *4.1 The Algorithm*

As mentioned, the state to be estimated is represented by $n$ particles and each particle $j$ has a state $x_j$, a weight $\pi_j$, and a coherence value $c_j \in [\min_c, \max_c]$. Initially, particles are spread uniformly in the space of states.

When defining a new set of particles we sample $n_l$ particles directly from the likelihood, $n_i$ particles from the prior-likelihood intersection, and $n_c$ particles using the coherence index (with $n_l + n_i + n_c = n$). Sampling from the likelihood introduces $n_l$ new particles (with minimum coherence, $\min_c$) into the system. Sampling from the prior-likelihood is the process described in section 2. The only addition is that the coherence of particles sampled using this method is increased by $\Delta_c$ (up to a maximum $\max_c$). Finally, sampling using the coherence is achieved using the coherence of particles as weighting factor. The weight of particles selected that way is set in the same way as particles sampled from the prior-likelihood intersection and

```
Particle Filter with the Coherence mechanism
  Input:   A set of particles $n$ $\{(x_j, \pi_j, c_j)\ j \in [1, n]\}$ approximating $p(x_{t-1}|u_{t-1}, y_{t-1})$
           Last executed action $u_t$.
           The current observation $y_t$.
  Output:  A set $n$ particles approximating $p(x_t|u_t, y_t)$
  Process:
    Move the particles according to $u_t$.
    Sample new particles:
        from likelihood: Create $n_l$ new particles from $p(y_t|x_t)$.
        from prior-likelihood intersection: Create $n_i$ new particles using $(\pi_j\ p(y_t|x_j))$ as weight.
        from the coherence: Create $n_c$ new particles using $c_j$ as a weight.
    Add noise to particles according to $u_t$.
    Update the weights:
        $\pi_j = 1$ for particles sampled from the likelihood.
        $\pi_j = p(y_t|x_j)/p(y_t|\mu_j)$ for rest of particles ($\mu_j$ is the state of the particle before adding noise)
        Normalize weights.
    Update the coherence:
        Set $c_j = \min_c$ for particles sampled from the likelihood.
        Increase the coherence of particles sampled from prior-likelihood:
            $c_j \leftarrow c_j + \Delta_c$      (only if $c_j < \max_c$).
```

Figure 1: The particle filter algorithm including the coherence sampling mechanism.

its coherence remains the same (so, it decreases w.r.t. particles sampled using the previous method).

Figure 1 summarizes the steps of this algorithm. Observe that if $n_l$ and $n_c$ are set to 0 we get the standard auxiliary particle filter algorithm. If only $n_c$ is set to 0 then the algorithm implements the *sample-from-likelihood* strategy described in section 3.

The result is a particle filter that attempts to provide the most coherent state estimation possible taking into account the recent sequence of observations.

## 5 Experiments

In our experiments, we use a unibot: a simulated robot moving along a segment (of size 10000 in our examples). Using a simple simulation has the advantage that we know the ground truth about the state of the system and we can easily analyze the results of the experiments. In this system, the state is unidimensional (i.e., the position of the unibot in the segment). In our experiments the unibot is placed at position $x_0 = 2000$, and the sensor model is defined as a Gaussian with $\sigma = 10$ that, in the absence of noise, is centered at the unibot position. No action model is considered because the unibot is static (except when there is a kidnap).

Using this setup, we compare five different algorithms:

- The standard particle filter algorithm described in section 2.
- The particle filter with the *sample-from-likelihood* heuristic described in section 3 with $n_l$ equal to $0.1 \times n$.

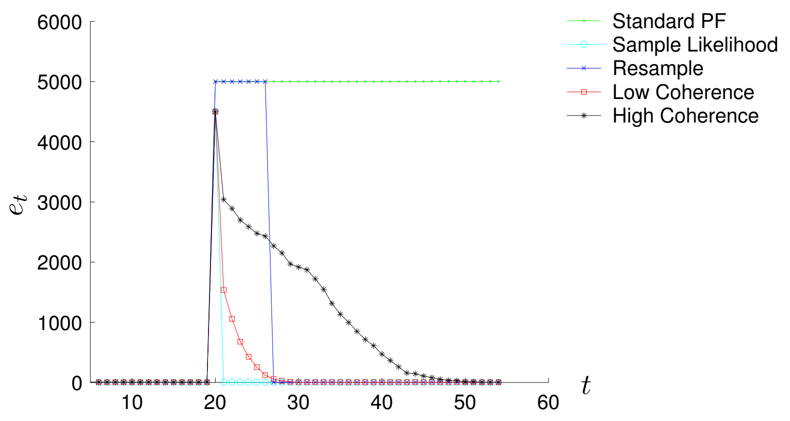

Figure 2: Evolution of error with a kidnap occurs at time $t = 20$.

- The particle filter with the *re-sample* heuristic also described in section 3. In this case the re-sample threshold is set to 8 time slices.
- Our algorithm with $n_l = 0.1 \times n$, $n_i = 0.6 \times n$, $n_c = 0.3 \times n$, $\min_c = 0$, $\max_c = 5$, and $\Delta_c = 1$.
- Our algorithm extending $\max_c$ to 15 and setting $n_i = 0.3 \times n$ and $n_c = 0.6 \times n$,

We compare the performance of these four systems in three different conditions:

- In a kidnap situation. In this experiment the unibot is suddenly displaced to position 7000 (5000 units far away from its original position).
- When there is non-Markovian noise in the observations. This kind of noise causes the Gaussian representing the sensor model to be displaced at random to a point in the uni-dimensional environment (far away from the real position of the unibot) and to remain there for several time slices. In this way, there is a dependency between two consecutive noisy sensor readings.
- When there is Markovian noise in the observations. In this case, the sensor model is displaced at random every time slice (in a given interval of time). Thus, the sensor noise in this interval of time does not depend on previous sensor readings.

In the three cases, the error at each time slice is computed as $e_t = \sum_{j=1}^{n} \pi_t^j \, |x_t^j - x_t|$, with $t$ the time slice, $\pi_t^j$ and $x_t^j$ the weights and position of particle $j$ at that time slice, and $x_t$ the correct position of the unibot at time $t$. In all the experiments, the results are averaged over 30 runs, we use $n = 80$ particles and we initialize the system with all particles on the correct position ($x_0$).

Figure 2 shows the average error when we kidnap the system at time slice 20. We can see that the standard particle filter is unable to react to the kidnap. The other four systems are able to detect and recover from the kidnap, with the *sample-from-likelihood* strategy being the most efficient one. The *re-sample* strategy waits 8 time slices before executing the recovery strategy and our algorithm performs a smooth transition toward the new state of the system. The speed of this transition depend on the value for the $\max_c$ parameter: the larger the smoother the transition. During the transition, particles are clustered around $x_t$ for $t < 20$

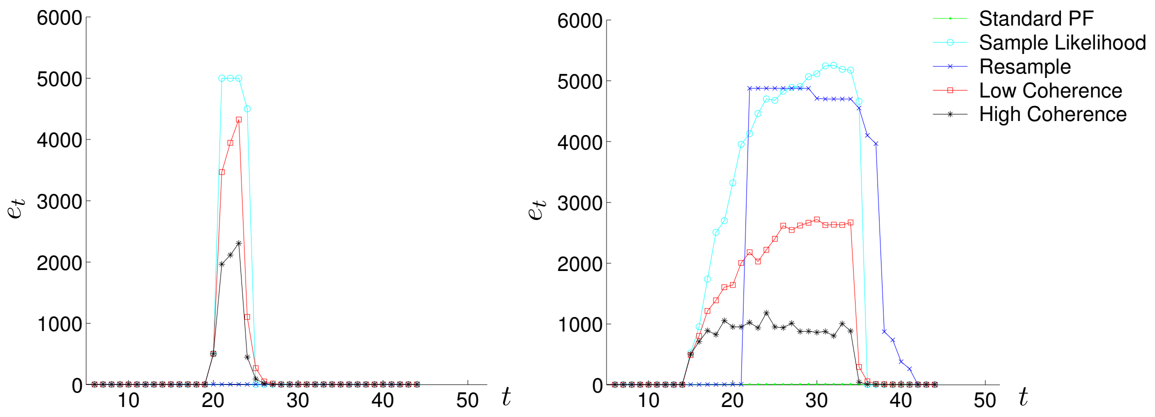

Figure 3: Experiments with non-Markovian noise (left) and with Markovian noise (right).

| | *Standard PF* | *Sample from likelihood* | *Re-sample strategy* | *Low Coherence* | *High Coherence* |
|---|---|---|---|---|---|
| Kidnap | 175096 | 4717 | 35210 | 8889 | 41571 |
| Non-Markovian Noise | 213 | 20169 | 216 | 13880 | 7693 |
| Markovian Noise | 301 | 84112 | 77538 | 42210 | 18352 |

Table 1: Summary of the error for the different systems and experiments.

and $x_t$ when $t \geq 20$. The first cluster of particles becomes less and less *coherent* over time and thus, it ends up disappearing.

Figure 3-left shows the average error when there is non-Markovian (i.e., correlated) noise in the sensor model in between times slices 20 and 24. The standard particle filter and the *re-sample* heuristic are completely immune to Markovian noise. If the noise was present for longer time, the *re-sample* heuristic would be triggered resulting in large error in the state estimation. The *sample-from-likelihood* heuristic (that was the best for the kidnap situation) performs the worst in this experiment. Our algorithm is also influenced by this kind of noise but we can reduce this influence by setting a larger maximum confidence ($\max_c$ parameter).

Figure 3-right shows the average error with Markovian noise in the sensor readings from time slice 20 to time slice 35. Standard particle filter is, again, insensitive to noise. Both the *re-sample* strategy (that performed well in the previous experiment) and the *sample-from-likelihood* heuristic perform poorly with this kind of noise. The sensitivity of our algorithm to this kind of noise can be adjusted: by increasing $\max_c$ we decrease the sensitivity.

Table 1 summarizes the results, with $e_t$ integrated over time. A robust particle filter must be able to deal both with the sensor noise and with the kidnap problem. The table shows that none of the existing techniques solves the two problems. However, when the confidence mechanism is used, the two problems can be addressed properly. The $\max_c$ parameter is proportional to the length of a sequence of non-Markovian noise to be considered a kidnap. So, it determines the trade of between the amount of non-Markovian noise we can deal with and the efficiency in reacting to kidnap situations.

## 6 Conclusions

The Markov assumption, explicitly used in the particle filtering approach, implies that the environment is static and that sensors readings are conditionally independent between them. In many applications, these assumptions are unrealistic and, as a consequence, the performance of the particle filter degrades.

We have introduced a new particle filter algorithm that, as shown in the experiments, can deal in a unified way with the kidnap problem and with sensor noise (even if this noise is, to certain degree, non-Markovian). We have applied this particle filter approach to robot localization in robot soccer getting a more robust estimation of the position of the robot, but we don't have yet quantitative results on these experiments.

The relation of the state estimation produced by our algorithm with that that would be obtained using an Expectation-Maximization algorithm on the whole sequence of observations and actions should be analyzed. Additionally, we have to study more thoughtfully the sensitivity of the algorithm to the different parameters and to determine principled ways to set them.

## Acknowledgments

Nikos Vlassis largely contributed to the development of the work reported in this paper. This work has been partially supported by the European (ITEA) project *"Ambience: Context Aware Environments for Ambient Services"*.

## References

[1] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo in Practice*. Springer-Verlag, New York, 2001.

[2] D. Fox, W. Burgard, and S. Thrun. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[3] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. *Sequential Monte Carlo Methods in Practice*, chapter Particle Filters for Mobile Robot Localization. Springer-Verlag, New York, 2000.

[4] M. Isard and A. Blake. Condensation - Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[5] B.J.A. Kröse, N. Vlassis, R. Bunschoten, and Y. Motomura. A Probabilistic Model for Appearance-based Robot Localization. *Image and Vision Computing*, 19(6):381–391, April 2001.

[6] S. Lenser and M. Veloso. Sensor Resetting Localization for Poorly Modelled Mobile Robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2000.

[7] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Image-Based Monte-Carlo Localisation without a Map. In *Proceedings Conference of the Italian Association of Artificial Intelligence*, 2003.

[8] M.K. Pitt and N. Shephard. Filtering Via Simulation: Auxiliary Particle Filters. *J. Amer. Statist. Assoc.*, 94(446):590–599, June 1999.

[9] J.M. Porta, J.J. Verbeek, and B.J.A. Kröse. Active Appearance-Based Robot Localization Using Stereo Vision. *Submitted to* Autonomous Robots, 2003.

[10] J. Vermaark, C. Andrieu, A.Doucet, and S.J. Godsil. Particle Methods for Bayesian Modeling and Enhancement of Speech Signals. *IEEE Transactions on Speech and Audio Processing*, 10(3):173–185, 2002.

[11] N. Vlassis, B. Terwijn, and B.J.A. Kröse. Auxiliary Particle Filter Robot Localization from High-Dimensional Sensor Observations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Washington D.C., USA*, pages 7–12, May 2002.