# An Optimized DBN-Based Mode-Focussing Particle Filter

Séverine Dubuisson and Christophe Gonzales
Laboratoire d'Informatique de Paris 6, Université Pierre et Marie Curie, France
firstname.name@lip6.fr

## Abstract

*We propose an original particle filtering-based approach combining optimization and decomposition techniques for sequential non-parametric density estimation defined in high-dimensional state spaces. Our method relies on Annealing to focus on the correct distributions and on probabilistic conditional independences defined by Dynamic Bayesian Networks to focus samples on their modes. After proving its theoretical correctness and showing its complexity, we highlight its ability to track single and multiple articulated objects both on synthetic and real video sequences. We show that our approach is particularly effective, both in terms of estimation errors and computation times.*

## 1. Introduction

Articulated object tracking is an important task in computer vision. Its applications include in particular gesture recognition, human tracking and event detection. Unfortunately, tracking articulated structures with accuracy and within a reasonable time is computationally challenging due to the high dimensionality of the state and observation spaces. In this paper, we tackle this problem using Sequential Monte Carlo methods (a.k.a. Particle Filter – PF). Essentially, this framework [3] aims at estimating a state sequence $\{\mathbf{x}_t\}_{t=1,\ldots,T}$, whose evolution is given by $\mathbf{x}_{t+1} = \mathbf{f}_{t+1}(\mathbf{x}_t, \mathbf{n}^{\mathbf{x}}_{t+1})$, from a set of observations $\{\mathbf{y}_t\}_{t=1,\ldots,T}$ related to the state by $\mathbf{y}_{t+1} = \mathbf{h}_{t+1}(\mathbf{x}_{t+1}, \mathbf{n}^{\mathbf{y}}_{t+1})$. Usually, $\mathbf{f}_{t+1}$ and $\mathbf{h}_{t+1}$ are nonlinear functions, and $\mathbf{n}^{\mathbf{x}}_{t+1}$ and $\mathbf{n}^{\mathbf{y}}_{t+1}$ are i.i.d. noise sequences. This problem is naturally represented by the Markov chain of Fig. 1.(a). From a probabilistic point of view, it amounts to estimate, for any $t$, either i) $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$ or ii) $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, where $\mathbf{x}_{1:t}$ denotes the tuple $(\mathbf{x}_1, \ldots, \mathbf{x}_t)$. The first quantity can be computed by iteratively using Eq. (1) and (2), which are referred to as a prediction step and a correction step respectively.

$$p(\mathbf{x}_{1:t+1}|\mathbf{y}_{1:t}) = p(\mathbf{x}_{t+1}|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t}) \quad (1)$$

$$p(\mathbf{x}_{1:t+1}|\mathbf{y}_{1:t+1}) \propto p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1})p(\mathbf{x}_{1:t+1}|\mathbf{y}_{1:t}) \quad (2)$$

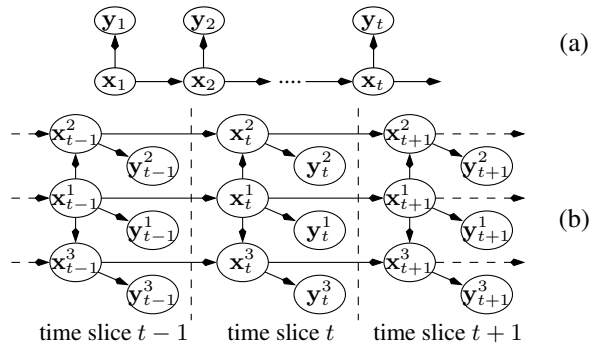



Figure 1. (a) A Markov chain for state estimation. (b) A DBN.

with $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$ the transition corresponding to function $\mathbf{f}_{t+1}$ and $p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1})$ the likelihood corresponding to $\mathbf{h}_{t+1}$. Probability $p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t+1})$ can be computed similarly with prediction and correction steps defined by:

$$p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t}) = \int_{\mathbf{x}_t} p(\mathbf{x}_{t+1}|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t})d\mathbf{x}_t \quad (3)$$

$$p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t+1}) \propto p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1})p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t}). \quad (4)$$

Basically, PF aims at approximating the above distributions using weighted samples. Thus, Eq. (3) and (4) are estimated by samples $\{\mathbf{x}^{(i)}_{t+1}, w^{(i)}_{t+1}\}$ of $N$ possible realizations of the state $\mathbf{x}^{(i)}_{t+1}$ called *particles*. In the prediction step (Eq. (3)), PF propagates the particle set $\{\mathbf{x}^{(i)}_t, w^{(i)}_t\}$ using a proposal function $q(\mathbf{x}_{t+1}|\mathbf{x}^{(i)}_{1:t}, \mathbf{y}_{t+1})$; in the correction step (Eq. (4)), PF weights the particles using a likelihood function, so that $w^{(i)}_{t+1} \propto w^{(i)}_t p(\mathbf{y}_{t+1}|\mathbf{x}^{(i)}_{t+1}) \frac{p(\mathbf{x}^{(i)}_{t+1}|\mathbf{x}^{(i)}_t)}{q(\mathbf{x}^{(i)}_{t+1}|\mathbf{x}^{(i)}_{1:t}, \mathbf{y}_{t+1})}$, with $\sum_{i=1}^N w^{(i)}_{t+1} = 1$. The particles can then be resampled: those with the highest weights are duplicated, while the others are eliminated. The estimation of the posterior density $p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t+1})$ is then given by $\sum_{i=1}^N w^{(i)}_{t+1} \delta_{\mathbf{x}^{(i)}_{t+1}}(\mathbf{x}_{t+1})$, where $\delta_{\mathbf{x}^{(i)}_{t+1}}$ are Dirac masses centered on particles $\mathbf{x}^{(i)}_{t+1}$. Of course, PF can approximate Eq. (1) and (2) in a similar way using weighted samples of tuples $(\mathbf{x}^{(i)}_1, \ldots, \mathbf{x}^{(i)}_{t+1})$.

As shown in [6], the number of particles necessary for a good estimation of the above densities grows exponentially with the dimension of the state space, hence making PF's

basic scheme unusable in real-time. In the literature, different variants of PF have been proposed to cope with high-dimensional spaces. They can be roughly divided into three categories: i) those that reduce the state space dimension by approximating it; ii) those that reduce it by making local searches; and iii) those that exploit natural independences in the state space to decompose it into a set of sub-spaces of reasonable sizes where PF can be applied. Here, for accuracy reasons, we focus on the last two approaches. Among the local search-based approaches, let us cite the popular *Annealed Particle Filter* (APF) [2] that consists of adding annealing iterations (or layers) to PF's resampling step in order to better diffuse particles in the state space. Among the third type of approaches, *Partitioned Sampling* (PS) [5] decomposes the state space as a Cartesian product of conditionally independent subspaces and iterates PF over all of them. A similar idea based on Dynamic Bayesian Networks (DBN) [7] is exploited in [4]: the proposal function $q$ is decomposed as the product of the conditional densities in each node of the network, and PF is applied sequentially on each node following a topological order of the DBN. These two approaches are combined in [9] to define an algorithm for PF totally integrated into a DBN. The work described in [1] is probably the closest to ours: a parallel algorithm of PF is described that uses the same joint probability in the DBN to reduce the number of particles. The state space is divided into subspaces in which the particles are independently generated by several proposal densities $q$. This approach enables to easily choose $q$ to sample each subspace. However, it requires a specific independence structure in the DBN that limits the generalization of the algorithm.

In this paper, we propose to significantly refine both PS and APF by fully exploiting conditional independences in the state space defined by DBNs. More precisely, we will show that, using DBNs, some permutations of PS's substates can be performed that focus the samples on the modes of the distributions. When combined with APF, this scheme proves particularly effective, both to significantly reduce estimation errors and to keep computation times low. The paper is organized as follows. In Section 2, we recall the basics of APF and PS. In Section 3, we describe our approach, prove its correctness and show its computational complexity. Using experiments on synthetic and real video sequences, Section 4 highlights the efficiency of the method both in terms of estimation errors and computation times. Finally, we conclude and give perspectives in Section 5.

## 2. Basics of PS and APF

The key idea of PS is that the state and observation spaces $\mathcal{X}$ and $\mathcal{Y}$ can often be naturally decomposed as $\mathcal{X} = \mathcal{X}^1 \times \cdots \times \mathcal{X}^P$ and $\mathcal{Y} = \mathcal{Y}^1 \times \cdots \times \mathcal{Y}^P$ where conditional independences between subspaces $(\mathcal{X}^i, \mathcal{Y}^i)$ can be exploited so that the sequential application of PF on each of them provides samples over $(\mathcal{X}, \mathcal{Y})$ estimating $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. As these subspaces are "smaller", the distributions to estimate have fewer parameters than those defined on $(\mathcal{X}, \mathcal{Y})$, which significantly reduces the number of particles needed for their estimation and, thus, speeds up the computations. More precisely, let $(\hat{\mathbf{x}}_t^{(i),j}, \mathbf{x}_t^{(i),-j})$ denote the particle with the same state as $\hat{\mathbf{x}}_t^{(i)}$ on part $j$ and the same as $\mathbf{x}_t^{(i)}$ on the other parts. Then, given a sample $\{\mathbf{x}_t^{(i)}\}$ at time $t$, PS first uses PF to compute sample $\{\hat{\mathbf{x}}_{t+1}^{(i),1}, \mathbf{x}_t^{(i),-1})\}$ where only the first part is propagated/corrected. Then, with this new sample, PF is applied on the second part, and so on (see [5] for details). As an example, to track a human body, $\mathcal{X}$ can be naturally decomposed as $\mathcal{X}^{\text{torso}} \times \mathcal{X}^{\text{left arm}} \times \mathcal{X}^{\text{right arm}}$ where, given the position of the torso, the left and right arm positions are independent. PS then first applies PF on the torso, then on the left arm and finally on the right arm. However, by multiplying the number of subspaces, we also multiply the resampling steps which, in turn, increases the noise in the estimation process and decreases its accuracy.

Quite differently, APF increases the accuracy of PF by exploiting a local search process in order to find the modes of the distributions. More precisely, after applying PF on the whole space $\mathcal{X}$, it explores the neighborhood of the particles to move them toward the modes. To do so, it alternates some form of weighted resampling that guarantees some survival rate of the particles with iterations of PF. This process is called *annealing*. Of course, APF can be combined with PS by substituting each iteration of PF over $\mathcal{X}$ by one of PS. This algorithm is denoted by PS-APF.

## 3. A Particle Filter with substate permutation

The algorithm we propose in this paper heavily relies for its correctness on the DBN's independence model, a.k.a. $d$-separation [8]. An example of a DBN is shown in Fig. 1.b. Basically, each node of a DBN represents a random variable and is assigned its probability distribution conditionally to its parents in the graph. The DBN represents the joint distribution of all its nodes as the product of all their conditional distributions. Hence, two nodes, say $\mathbf{x}_t^i$ and $\mathbf{x}_s^j$ are dependent conditionally to a set of nodes $Z$ if and only if there exists a chain $\{\mathbf{c}_1 = \mathbf{x}_t^i, \ldots, \mathbf{c}_n = \mathbf{x}_s^j\}$ linking $\mathbf{x}_t^i$ and $\mathbf{x}_s^j$ in the DBN such that i) for every node $\mathbf{c}_k$ such that the DBN arcs are $\mathbf{c}_{k-1} \rightarrow \mathbf{c}_k \leftarrow \mathbf{c}_{k+1}$, either $\mathbf{c}_k$ or one of its descendants is in $Z$; and ii) none of the other nodes $\mathbf{c}_k$ belongs to $Z$. Such a chain is called *active*. This is the $d$-separation criterion [8]. In our body tracking problem, given the position of the body in the past, as both arms are independent conditionally to the position of the torso, $p(\text{torso}, \text{left arm}, \text{right arm}) = p(\text{torso}) \times p(\text{left arm}|\text{torso}) \times p(\text{right arm}|\text{torso})$, justifying that the DBN of Fig. 1.b models this tracking problem if $\mathbf{x}_t^1, \mathbf{x}_t^2, \mathbf{x}_t^3$ represent the torso and the left and right arms respectively.

PS, as introduced in [5], did not rely on DBNs. However, the latter can be used to justify its correctness. Actually, if, by $d$-separation, for any $j \in \{1, \ldots, P\}$, $\mathbf{x}_t^j$ is independent of $(\mathbf{x}_{t-1}^{j+1}, \ldots, \mathbf{x}_{t-1}^P)$ conditionally to $(\mathbf{x}_t^1, \ldots, \mathbf{x}_t^{j-1}, \mathbf{x}_{t-1}^j)$, then each iteration of PF over one substate $\mathbf{x}_t^j$ samples over a distribution which is independent of $(\mathbf{x}_{t-1}^{j+1}, \ldots, \mathbf{x}_{t-1}^P)$. This is precisely what is needed to ensure that, after iterating over the $P$ parts of the object, PS and PS-APF have both sampled over distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. For instance, in the DBN of Fig. 1.b, $\mathbf{x}_t^1, \mathbf{x}_t^2, \mathbf{x}_t^3$ satisfy the above property and, thus, PS-APF can be used to perform PF first on the torso, then on the left arm and finally on the right arm.

Our algorithm exploits $d$-separation, first, to identify independent tracking subproblems where PF can be performed in parallel and, second, to mix their results in a most efficient way. More formally, we will assume that, *within each time slice*, the DBN structure is a directed tree (or a forest if we track multiple objects), i.e., there do not exist nodes $\mathbf{x}_t^i, \mathbf{x}_t^j, \mathbf{x}_t^k$ with the graph topology $\mathbf{x}_t^i \rightarrow \mathbf{x}_t^j \leftarrow \mathbf{x}_t^k$. In addition, we will assume that arcs across time slices link similar nodes, i.e., there exist no arc $\mathbf{x}_{t-1}^i \rightarrow \mathbf{x}_t^j$ with $j \neq i$. For articulated object tracking, these requirements are rather mild. Fig. 1.b satisfies both of them. For any set $R = \{i_1, \ldots, i_r\} \subseteq \{1, \ldots, P\}$, by abuse of notation, $\mathbf{x}_t^R$ denotes tuple $(\mathbf{x}^{i_1}, \ldots, \mathbf{x}_t^{i_r})$. Let $\mathbf{Pa}(\mathbf{x}_t^i)$ and $\mathbf{Pa}_t(\mathbf{x}_t^i)$ denote the set of parents of node $\mathbf{x}_t^i$ in the DBN in all time slices and in only time slice $t$ respectively. For instance, in Fig. 1.b, $\mathbf{Pa}(\mathbf{x}_t^2) = \{\mathbf{x}_t^1, \mathbf{x}_{t-1}^2\}$ and $\mathbf{Pa}_t(\mathbf{x}_t^2) = \{\mathbf{x}_t^1\}$. In addition, let $\{P_1, \ldots, P_K\}$ denote the partition of $\{1, \ldots, P\}$ defined by: i) $P_1 = \{j : \mathbf{Pa}_t(\mathbf{x}_t^j) = \emptyset\}$ (in Fig. 1.b, $P_1 = \{1\}$ because only $\mathbf{x}_t^1$ has no parent in time slice $t$); ii) for all $i \neq 1$, $P_i = \{j : \mathbf{Pa}_t(\mathbf{x}_t^j) \subseteq \{\mathbf{x}_t^k : k \in \cup_{r<i} P_r\}\}$ (in Fig. 1.b, $P_2 = \{2, 3\}$). Intuitively, $P_1$ is the central part of the object, $P_2$ are the subparts linked to $P_1$, $P_3$ those linked to $P_2$ and so on. Then, the following holds:

**Proposition 1** *The distribution estimated by applying PF iteratively over each $\mathbf{x}_t^i$, $i = 1, \ldots, P$, as PS-APF does, is the same as that estimated by applying PF in parallel for each set of parts $P_i$, $i = 1, \ldots, K$.*

**Proof:** When PF is applied on node $\mathbf{x}_t^j$, it propagates particles using $p(\mathbf{x}_t^j|\mathbf{Pa}(\mathbf{x}_t^j))$ and correct them using $p(\mathbf{y}_t^j|\mathbf{x}_t^j)$, overall modifying them using $p(\mathbf{x}_t^j, \mathbf{y}_t^j|\mathbf{Pa}(\mathbf{x}_t^j))$. By $d$-separation and the definition of the $P_i$'s, for each pair of parts $(j, k) \in P_i$, $(\mathbf{x}_t^j, \mathbf{y}_t^j)$ is independent of $(\mathbf{x}_t^k, \mathbf{y}_t^k)$ conditionally to $\mathbf{Pa}(\mathbf{x}_t^j)$. Hence, applying PF in parallel on each $\mathbf{x}_t^j$ of a given set $P_i$ results in a correct estimation of the joint a posteriori distribution and the proposition follows. $\square$

Proposition 1 allows parallelizing PS-APF. For instance, in the DBN of Fig 1.b, PF can be used first on $\mathbf{x}_t^1$ and, then, in parallel on $\mathbf{x}_t^2$ and $\mathbf{x}_t^3$. But its proof also highlights an important property: if we extract from sample $\{\mathbf{x}_t^{(i)}\}$ subsample $\{\mathbf{x}_t^{(i),j}\}$ restricted to the $j$th part, the latter estimates, up to a normalizing constant:

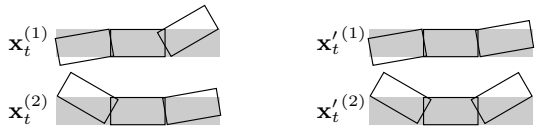
$$p(\mathbf{x}_t^j, \mathbf{y}_{1:t}^j|\mathbf{Pa}_t(\mathbf{x}_t^j)) = \int p(\mathbf{x}_t^j, \mathbf{y}_t^j|\mathbf{Pa}(\mathbf{x}_t^j)) p(\mathbf{x}_{t-1}, y_{1:t-1}^j) d\mathbf{x}_{t-1}.$$

Therefore, sample $\{\mathbf{x}_t{}^{(\sigma_j(i)),j}\}$ resulting from permuting the elements of $\{\mathbf{x}_t^{(i),j}\}$ by $\sigma_j$ also estimates distribution $p(\mathbf{x}_t^j, \mathbf{y}_{1:t}^j|\mathbf{Pa}_t(\mathbf{x}_t^j))$. If the subsamples are permuted independently for all parts $j$, the overall resulting sample $\{\mathbf{x}_t^\sigma\}$ will not in general represent $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. It turns out that, to enforce this property, we cannot use a particle filter estimating Eq (3) and (4), but we need one estimating Eq (1) and (2). So, from now on, assume that samples at time $t$ are of the form $\{\mathbf{x}_{1:t}^{(i)}, w_{1:t}^{(i)}\}$ instead of $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$. Similarly to states $\mathbf{x}_t$, weights $w_t$ can be decomposed as $w_t^j = p(\mathbf{y}_t^j|\mathbf{x}_t^j)$, i.e., the likelihood of part $j$. The global weight $w_t$ of a particle is then defined as $w_t \propto \prod_{j=1}^P w_t^j$ and $\{\mathbf{x}_{1:t}^{(i),j}, w_{1:t}^{(i),j}\}$ samples $p(\mathbf{x}_{1:t}^j, \mathbf{y}_{1:t}^j|\mathbf{Pa}(\mathbf{x}_{1:t}^j))$, where $\mathbf{Pa}(\mathbf{x}_{1:t}^j)$ represents $\bigcup_{s=1}^t \{\text{parents of nodes } \mathbf{x}_s^j \text{ in the DBN}\}$. For instance, in the DBN of Fig. 1.(b), the sample of the left arm positions over all time slices $\{\mathbf{x}_{1:t}^{(i),2}, w_{1:t}^{(i),2}\}$ estimates $p(\mathbf{x}_{1:t}^2, \mathbf{y}_{1:t}^2|\mathbf{x}_{1:t}^1)$, the distribution of all variables on top line conditionally to the middle ones. Similarly, $\{\mathbf{x}_{1:t}^{(i),1}, w_{1:t}^{(i),1}\}$ and $\{\mathbf{x}_{1:t}^{(i),3}, w_{1:t}^{(i),3}\}$ estimate $p(\mathbf{x}_{1:t}^1, \mathbf{y}_{1:t}^1)$ and $p(\mathbf{x}_{1:t}^3, \mathbf{y}_{1:t}^3|\mathbf{x}_{1:t}^1)$ respectively. The joint distribution is then equal to:

$$p(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) = \prod_{j=1}^P p(\mathbf{x}_{1:t}^j, \mathbf{y}_{1:t}^j|\mathbf{Pa}(\mathbf{x}_{1:t}^j)).$$

Although it is often not the case that, if $j, k \in P_i$, $\mathbf{x}_t^j$ is independent of $\mathbf{x}_t^k$ conditionally to $\mathbf{Pa}_t(\mathbf{x}_t^j)$, it is the case that $\mathbf{x}_{1:t}^j$ is independent of $\mathbf{x}_{1:t}^k$ conditionally to $\mathbf{Pa}(\mathbf{x}_{1:t}^j)$. For instance, in Fig. 1.(b), $\mathbf{x}_t^2$ is not independent of $\mathbf{x}_t^3$ conditionally to $\mathbf{x}_t^1$ because $\{\mathbf{x}_t^2, \mathbf{x}_{t-1}^2, \mathbf{x}_{t-1}^1, \mathbf{x}_{t-1}^3, \mathbf{x}_t^3\}$ is an active chain. However, there exists no active chain between $\mathbf{x}_{1:t}^2$ and $\mathbf{x}_{1:t}^3$ conditionally to $\mathbf{x}_{1:t}^1$. Now, consider two subparticles $(\mathbf{x}_{1:t}^{(i),3}, w_{1:t}^{(i),3})$ and $(\mathbf{x}_{1:t}^{(j),3}, w_{1:t}^{(j),3})$ such that $\mathbf{x}_{1:t}^{(i),1} = \mathbf{x}_{1:t}^{(j),1}$, then permuting them affects neither the estimation of $p(\mathbf{x}_{1:t}^2, \mathbf{y}_{1:t}^2|\mathbf{x}_{1:t}^1)$ (by conditional independence) nor that of $p(\mathbf{x}_{1:t}^3, \mathbf{y}_{1:t}^3|\mathbf{x}_{1:t}^1)$ (permutations within samples do not affect the probability they estimate). A fortiori, the estimated joint distribution is unchanged.

This example can be generalized as follows: for all $j \in \{1, \ldots, P\}$, let $\mathbf{Desc}(\{\mathbf{x}_t^j\})$ denote the set of descendants of $\{\mathbf{x}_t^j\}$ in time slice $t$, and $\mathbf{Desc}(\{\mathbf{x}_{1:t}^j\}) = \cup_{s=1}^t \mathbf{Desc}(\{\mathbf{x}_s^j\})$. For every $j \in \{1, \ldots, P\}$, let $\sigma_j$ be any permutation of sample $\{\mathbf{x}_{1:t}^{(i),j}\}$ such that only sub-particles with the same value of $\mathbf{Pa}(\mathbf{x}_{1:t}^j)$ are permuted. For every $P_i$, let the permutation operation $\circlearrowleft^{P_i}$ be defined as: for all $j \in P_i$, permute the subparticles belonging to parts $\{\mathbf{x}_{1:t}^j\} \cup_{k \in \mathbf{Desc}(\{\mathbf{x}_{1:t}^j\})} \{\mathbf{x}_{1:t}^k\}$ w.r.t. $\sigma_j$. Then the following proposition holds:




a) before permutation b) after permutation

Figure 2. The permutation operation.

**Proposition 2** *Applying operators $\circlearrowleft^{P_j}$, $j = 1, \ldots, P$, on $\{\mathbf{x}_{1:t}^{(i)}, w_{1:t}^{(i)}\}$ does not change the distribution estimated.*

**Proof:** After applying permutation $\sigma_j$ on subsample $\{\mathbf{x}_{1:t}^{(i),j}\} \cup_{k \in \mathbf{Desc}(\{\mathbf{x}_{1:t}^j\})} \{\mathbf{x}_{1:t}^{(i),k}\}$, the resulting subsample estimates $p(\mathbf{x}_{1:t}^{\{j\} \cup \mathbf{Desc}(\{\mathbf{x}_{1:t}^j\})}, y_{1:t}^{\{j\} \cup \mathbf{Desc}(\{\mathbf{x}_{1:t}^j\})} | \mathbf{Pa}(\mathbf{x}_{1:t}^j))$. Moreover, by $d$-separation, $\{\mathbf{x}_{1:t}^j\} \cup_{k \in \mathbf{Desc}(\{\mathbf{x}_{1:t}^j\})} \{\mathbf{x}_{1:t}^k\}$ is independent of the rest of the DBN conditionally to $\mathbf{Pa}(\mathbf{x}_{1:t}^j)$. Hence, applying $\sigma_j$ on $\{\mathbf{x}_{1:t}^j\} \cup_{k \in \mathbf{Desc}(\{\mathbf{x}_{1:t}^j\})} \{\mathbf{x}_{1:t}^k\}$ does not change the estimation of the joint distribution. Applying inductively this argument on every $j$ proves the result. □

The purpose of $\circlearrowleft^{P_i}$ is to focus the particles on the modes of the distributions. As an example, consider Fig. 2.(a), where 2 particles are represented, $\mathbf{x}_t^1, \mathbf{x}_t^2, \mathbf{x}_t^3$ being the central, left and right parts of the object respectively, and the gray areas being the actual position of the object. These particles have the same value of $\mathbf{x}_t^1$, so, by the above proposition and according to Fig. 1.(b), their values on $\mathbf{x}_t^3$ can be permuted. Note that, before this permutation, the right (resp. left) part of the first (resp. second) particle was far from the true state, which induced overall small weights to both particles. On the contrary, after permutation, the first particle is very close to the true state (and thus has a high weight) and the second one is far away (and thus has a low weight). After resampling, the latter will probably be discarded and only the best particle will remain.

Note that Proposition 1 also holds when samples are of the form $\{\mathbf{x}_{1:t}^{(i)}, w_{1:t}^{(i)}\}$. So this suggests a new particle filter in which samples are $\{\mathbf{x}_{1:t}^{(i)}, w_{1:t}^{(i)}\}$ and in which PF is not applied sequentially on each $\mathbf{x}_{1:t}^i$ but rather in parallel on each $\mathbf{x}_{1:t}^{P_i}$ followed by some permutation $\circlearrowleft^{P_i}$ that focuses particles on the modes of the distribution. Now, remark that, by nature, state space $\mathcal{X}$ is continuous (position, angle of the object, etc). Therefore, when two particles $\mathbf{x}_{1:t}^{(i)}$ and $\mathbf{x}_{1:t}^{(j)}$ have the same value on $\mathbf{Pa}_t(\mathbf{x}_t^k)$, the parents of some node $\mathbf{x}_t^k$ in time slice $t$, there is a high probability that these two particles result from the duplication of another one during a resampling step. In this case, it follows that both particles have the same value on $\mathbf{Pa}(\mathbf{x}_{1:t}^k)$ and Proposition 2 can be applied to permute their $k$th part. Hence, in practice, it is only a slight approximation to permute particle parts only when their parents in time slice $t$ are equal. The advantage is that we get a particle filter estimating Eq. (3) and (4) since all operations are related to time $t$ instead of $1 : t$. This is much more efficient since it consumes less memory

**Input**: $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$, partition $\{P_1, \ldots, P_K\}$, $L$ layers, image $\mathcal{I}$
**Output**: $\{\mathbf{x}_{t+1}^{(i)}, w_{t+1}^{(i)}\}$ at time $t+1$
$\{\mathbf{x}^{(i)}, w^{(i)}\} \leftarrow \{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$
**for** $n = 0$ **to** $L$ **do**
  **if** $n \neq 0$ **then**
    $\{\mathbf{x}^{(i)}, w^{(i)}\} \leftarrow$ APF weighted resampling $(\{\overline{\mathbf{x}}^{(i)}, \overline{w}^{(i)}\})$
  **for** $j = 1$ **to** $K$ **do**
    **for** $k = 1$ **to** $|P_j|$ **do**
      $\{\hat{\mathbf{x}}^{(i),k}\} \leftarrow$ propagation $(\{\mathbf{x}^{(i),k}\})$
      $\{\hat{w}^{(i),k}\} \leftarrow$ correction $(\{\hat{\mathbf{x}}^{(i),k}, w^{(i),k}\}, \mathcal{I})$
    $\{\overline{\mathbf{x}}^{(i)}, \overline{w}^{(i)}\} \leftarrow \circlearrowleft^{P_i}$ $(\{\hat{\mathbf{x}}^{(i)}, \hat{w}^{(i)}\})$
$\{\mathbf{x}^{(i)}, w^{(i)}\} \leftarrow$ resampling $(\{\overline{\mathbf{x}}^{(i)}, \overline{w}^{(i)}\})$
**return** $\{\mathbf{x}^{(i)}, w^{(i)}\}$

**Algorithm 1:** PS-APF with permutations: $APF^{\circlearrowleft}$.

and provides more accurate results (the joint distribution on time $t$ having fewer parameters than that on slices 1 to $t$, its estimation requires fewer particles). Overall, this leads to Algorithm 1.

Of course, some permutations are better than others to focus on the modes of the distribution. A "good" one can easily be defined: from the preceding paragraph, two particles with the same value on $\mathbf{Pa}_t(\mathbf{x}_t^j)$ have most probably been generated from the duplication of the same particle during a resampling step. In this case, for all $k \in P_i$, they also have a same value of $\mathbf{Pa}(\mathbf{x}_t^k)$. We can then partition the sample of $N$ particles into subsets $N_1, \ldots, N_R$ such that the particles of a set $N_r$, $r = 1, \ldots, R$, have the same value of $\mathbf{Pa}(\mathbf{x}_t^j)$ for some $j \in P_i$. For each $N_r$, all possible permutations are eligible. Let $\{r_1, \ldots, r_s\}$ be the elements of $N_r$. For each $j \in P_i$, let $\sigma_j$ be the permutation that sorts weights $w_t^{(r_h),j}$, $h = 1, \ldots, s$, in decreasing order. By applying $\sigma_j$ for all $j \in P_i$, we get a permutation operation $\circlearrowleft^{P_i}$ that assigns to the first particle the set of the highest weights, to the second one, the set of second best weights, and so on. Thus, the first particles have the highest weights, and the last ones the lowest (they will thus be discarded at the next resampling step).

The time complexity of such an algorithm for all $P_i$ is in $O(PN(E + \log N))$, where $E$ is the size of variables $\mathbf{x}_t^j$. Actually, for a given $P_i$, by using a hash table, the complexity of determining $N_r$ is in $O(N)$. For each $N_r$, we have to sort $|P_i|$ lists, which can be globally done in $|P_i| N \log N$. Finally, applying permutations modify at most $P|\mathbf{x}_t^j|$ per particle and is then performed in $O(NPE)$.

## 4. Experimental results

We have tested our approach on articulated objects tracking and have studied its ability to estimate high-dimensional densities both on synthetic and real video sequences. In particular, we provide computation times and estimation errors for varying state space dimensions and number of parts

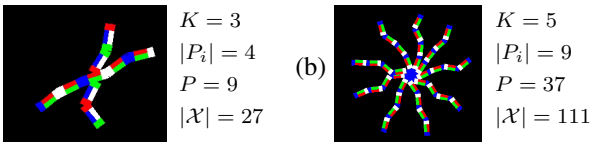



Figure 3. Zooms on synthetic video sequences.

treated in parallel. The objects we track are moving and deforming over time. They are modeled by a set of $P$ polygonal parts (or regions): a central one $P_1$ to which are linked $|P_i|$ arms of length $K-1$, $i = 2, \ldots, K$ (see Section 3). The polygons are manually positioned in the first frame. The state vector contains the parameters describing all parts, and is defined by $\mathbf{x}_t = \{\mathbf{x}_t^1, \ldots, \mathbf{x}_t^P\}$, with $\mathbf{x}_t^i = \{x_t^i, y_t^i, \theta_t^i\}$, where $(x_t^i, y_t^i)$ is the center of part $i$, and $\theta_t^i$ its orientation, $i = 1, \ldots, P$. We thus have $|\mathcal{X}| = 3P$. A particle $\mathbf{x}_t^{(j)} = \{\mathbf{x}_t^{(j),1}, \ldots, \mathbf{x}_t^{(j),P}\}$, $j = 1, \ldots, N$, is a possible spatial configuration, i.e., a realization, of the articulated object. Particles are propagated using a random walk whose variance has been empirically fixed for all tests ($\sigma_x = 1$, $\sigma_y = 1$ and $\sigma_\theta = 0.025$). The particle weights are computed using the color information of the current observation (image), and are given by $w_{t+1}^{(j)} = w_t^{(j)} p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}^{(j)}) \propto w_t^{(j)} e^{-\lambda \mathbf{d}^2}$, with $\lambda = 50$ and $\mathbf{d}$ the Bhattacharyya distance between target (prior) and reference (previously estimated) 8-bin histograms. PS-APF with $l$ layers (denoted by APF$_l$) and our APF with permutation with 0 or 1 layer, denoted by APF$_0^{\circlearrowleft}$ and APF$_1^{\circlearrowleft}$ respectively, were compared w.r.t. two criteria: computation times and estimation errors. The latter are given by the sum of the Euclidean distances between each corner of the estimated parts and its corresponding corner in the ground truth. All the results presented are a mean over 60 runs performed on a MacBook Pro with a 2.66 GHz Intel Core i7 processor.

**Quantitative tests on synthetic sequences.** We generated synthetic video sequences of 300 frames of $800 \times 600$ pixels. Despite their visual simplicity (five colors), they are very challenging because the densities to estimate are very high-dimensional. Fig. 3 shows examples of synthetic objects. Table 1 shows estimation errors and computation times for the density estimations of objects with various numbers of arms $|P_i|$ and arm's lengths $K$ (see Fig. 3). Tests were made with $N = 50$ and $N = 300$ particles. For all of them, the permuted versions of APF always give better estimations. In particular, APF$_1^{\circlearrowleft}$ is more robust than APF$_2$: a single permutation after an optimization step is more accurate than 2 annealing layers. Note that our approach does not increase computation times. Actually, the greater the values of $K$ and $|P_i|$, the more APF$_l^{\circlearrowleft}$ outperforms APF$_l$. This results from the simultaneous treatment of the $|P_i|$ parts that reduces the number of resampling steps. We will quantify this below on real video sequences.

Table 1. Estimation errors ($e$, in pixels) and computation times ($t$, in sec.), for an object with (i) first three rows $|P_i| = 4$, depending on $K$, and (ii) last three rows $K = 2$, depending on $|P_i|$.

| | | APF$_0^{\circlearrowleft}$ | | APF$_1$ | | APF$_2$ | | APF$_1^{\circlearrowleft}$ | | APF$_2^{\circlearrowleft}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | 50 | 300 | 50 | 300 | 50 | 300 | 50 | 300 | 50 | 300 |
| $K = 4$ | $e$ | 309 | 209 | 251 | 204 | 231 | 196 | 208 | 184 | 203 | 181 |
| $\|P_i\| = 4$ | $t$ | 6.2 | 37.9 | 12.6 | 77.4 | 18.9 | 115.9 | 12.3 | 76.1 | 18.5 | 114.5 |
| $K = 6$ | $e$ | 735 | 471 | 508 | 405 | 542 | 357 | 391 | 321 | 350 | 299 |
| $\|P_i\| = 4$ | $t$ | 10.3 | 63.8 | 21.2 | 134.6 | 32.1 | 206.0 | 20.5 | 130.2 | 31.0 | 195.2 |
| $K = 8$ | $e$ | 1680 | 1392 | 1203 | 1174 | 1178 | 1081 | 1152 | 1062 | 1098 | 978 |
| $\|P_i\| = 4$ | $t$ | 15.1 | 96.9 | 31.3 | 207.8 | 47.7 | 315.6 | 31.1 | 196.8 | 47.2 | 299.4 |
| $\|P_i\| = 6$ | $e$ | 96 | 70 | 85 | 66 | 78 | 63 | 67 | 62 | 64 | 61 |
| $K = 2$ | $t$ | 7.1 | 41.8 | 13.7 | 85.7 | 20.5 | 133.5 | 13.5 | 76.9 | 20.0 | 120.4 |
| $\|P_i\| = 8$ | $e$ | 162 | 134 | 168 | 135 | 149 | 127 | 125 | 117 | 112 | 112 |
| $K = 2$ | $t$ | 8.2 | 48.5 | 16.2 | 95.5 | 24.2 | 141.4 | 15.6 | 86.2 | 23.4 | 133.6 |
| $\|P_i\| = 10$ | $e$ | 185 | 157 | 193 | 156 | 183 | 151 | 155 | 142 | 150 | 140 |
| $K = 2$ | $t$ | 9.3 | 55.2 | 18.8 | 109.5 | 27.4 | 164.6 | 17.6 | 100.7 | 26.3 | 152.7 |

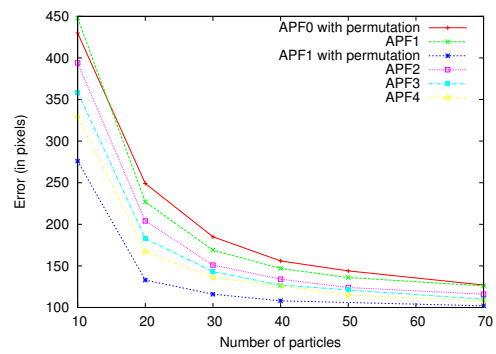



Figure 4. Convergence study for tracking object of Fig. 3.(a) for different approaches.

We now study the effect of one permutation compared to different layers of annealing on the estimation of the density of the object of Fig. 3.(a). Fig. 4 shows that APF$_0^{\circlearrowleft}$ and APF$_1$'s convergence rates are almost similar. This highlights the mode-focusing feature of APF$_0^{\circlearrowleft}$. But, more importantly, APF$_1^{\circlearrowleft}$ converges much faster than all the other methods: with only $N = 30$ particles, our approach converges ($e = 108$, $t = 7.5$), whereas APF$_4$ requires $N = 70$ particles to converge ($e = 109$, $t = 42.8$).

**Quantitative and qualitative tests on real sequences.** We tested our approach on sequences from the UCF50 dataset[1], to demonstrate the efficiency of our permutation operation to make the particle set better focus on the modes of the densities to estimate. This feature holds even when there are wide movements over time and when images have a low resolution. Qualitative results are given by superimposing on the frames of the sequences a red articulated object corresponding to the estimation derived from the weighted sum of the particles.

We present here results on the 242 $320 \times 240$ frames of

[1] http://server.cs.ucf.edu/∼vision/data/UCF50.rar

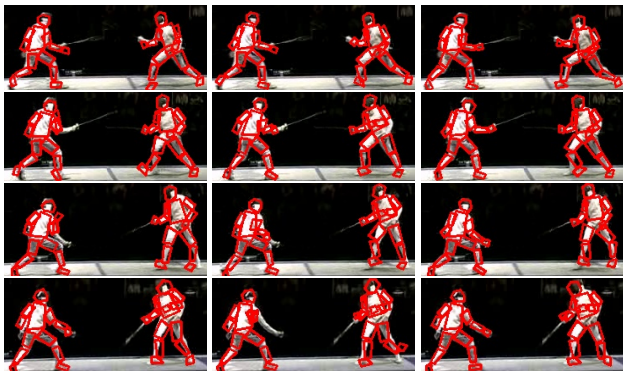


Figure 5. The `Fencing` sequence. Tracking results (zooms on parts of the images, $N = 1000$): from left to right, $\text{APF}_0^{\circlearrowleft}$, $\text{APF}_1$ and $\text{APF}_1^{\circlearrowleft}$ (frames 25, 65, 80, 120).

the `Fencing` sequence of this dataset. This sequence is challenging because it contains two articulated objects deforming and moving quickly (see the relative positions of the fencers w.r.t. the gray line on the floor). The fencers were modeled using $P = 27$ parts, resulting in $|\mathcal{X}| = 81$. We manually annotated this sequence to get a ground truth in order to compute estimation errors. This sequence is well suited to highlight the efficiency of our approach that processes in parallel both the different objects and their independent parts. We compared single filters (one for the two objects) and 2 independent filters (one per object). Qualitative tracking results are given in Fig. 5 for 3 single filters and $N = 1000$. Here again, the permutation operation improves tracking results. Table 2 confirms this qualitative analysis. As observed for synthetic sequences, our approach reduces the estimation errors. Note again that $\text{APF}_1^{\circlearrowleft}$ as a single filter outperforms both $\text{APF}_1$ (single or parallel filter) and $\text{APF}_2$. Table 2 also provides computation times. Our permutation operation is not time consuming compared to the resampling steps. In addition, the parallel processing within the $P_i$'s significantly reduces the number of resamplings in our approach. Overall, the latter is faster than PS-APF. For instance, for $N = 2000$, $\text{APF}_1^{\circlearrowleft}$ reduces resampling and total computation times by $60\%$ and $43\%$ respectively compared to $\text{APF}_1$, and this with an error decreased by $18\%$. Our tests also show that the results given by our approach (both in terms of total computation times and estimation errors) are equivalent whether we use 1 filter for the 2 objects or 1 filter per object. This shows its ability to correctly consider independent parts and work in the appropriate subspaces.

## 5. Conclusion

We have presented a new approach for sequential estimation of densities that (i) exploits the probabilistic independencies encoded into DBNs to apply particle filter computations on smaller subspaces; and that (ii) permutes some subsets of particles so that they concentrate around modes of the densities. We proposed a sound theoretical framework that guarantees that distributions are correctly estimated. In addition, we provided the time complexity of our approach. Experiments showed that our permutation operation is not time-consuming and, combined with the parallel processing of conditionally independent parts that reduces significantly the number of resamplings, it induces overall computation times that are often smaller than PS-APF. Moreover, we have shown that this gain of computation time increases with the state space dimension.

To conclude, our current works concern the choice a better criterion for optimizing permutations. In particular, this leads us to study new definitions of what a good particle set should be.

Table 2. `Fencing` sequence. Estimation errors ($e$ - in pixels) and computation times ($t$ = total times, $r$ = resampling times, and $p$ = permutation times - in seconds).

| | | $\text{APF}_0^{\circlearrowleft}$ | $\text{APF}_1$ | $\text{APF}_1^{\circlearrowleft}$ | 2 $\text{APF}_1$ | 2 $\text{APF}_1^{\circlearrowleft}$ | $\text{APF}_2$ | $\text{APF}_2^{\circlearrowleft}$ |
|---|---|---|---|---|---|---|---|---|
| $N = 1000$ | $e$ | 420 | 418 | 335 | 382 | 327 | 360 | 302 |
| | $t$ | 99.3 | 379.3 | 231.6 | 253.1 | 224.2 | 583.6 | 368.4 |
| | $r$ | 21.1 | 218.4 | 77.3 | 119.5 | 84.8 | 360.6 | 136.9 |
| | $p$ | 7.7 | - | 24.1 | - | 14.3 | - | 42.7 |
| $N = 2000$ | $e$ | 280 | 275 | 225 | 245 | 225 | 229 | 196 |
| | $t$ | 217.8 | 943.4 | 536.3 | 612.1 | 513.8 | 1469.6 | 892.6 |
| | $r$ | 53.6 | 570.1 | 214.7 | 314.9 | 217.1 | 950.8 | 398.5 |
| | $p$ | 16.9 | - | 54.5 | - | 33.4 | - | 94.6 |

## References


[1] E. Besada-Portas, S. Plis, J. Cruz, and T. Lane. Parallel subspace sampling for particle filtering in dynamic Bayesian networks. In *ECML PKDD*, pages 131–146, 2009. 2

[2] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61:185–205, 2005. 2

[3] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. of Radar and Signal Processing*, 140(2):107–113, 1993. 1

[4] K. Kanazawa, D. Koller, and S. Russell. Stochastic simulation algorithms for dynamic probabilistic networks. In *UAI*, pages 346–35, 1995. 2

[5] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *ICCV*, pages 572–587, 1999. 2, 3

[6] J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *ECCV*, pages 3–19, 2000. 1

[7] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, 2002. 2

[8] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, 1988. 2

[9] C. Rose, J. Saboune, and F. Charpillet. Reducing particle filtering complexity for 3D motion capture using dynamic Bayesian networks. *AAAI*, pages 1396–1401, 2008. 2