Sub-sample swapping for sequential Monte Carlo approximation of high-dimensional densities in the context of complex object tracking

Severine Dubuisson, Christophe Gonzales, Xuan Son Nguyen

Sub-sample swapping for Sequential Monte Carlo approximation of high-dimensional densities in the context of complex object tracking

Séverine Dubuisson, Christophe Gonzales, Xuan Son Nguyen

Laboratoire d'Informatique de Paris 6 (LIP6/UPMC)
4 place Jussieu, 75005 Paris, FRANCE

Abstract

In this paper, we address the problem of complex object tracking using the particle filter framework, which essentially amounts to estimate high-dimensional distributions by a sequential Monte Carlo algorithm. For this purpose, we first exploit Dynamic Bayesian Networks to determine conditionally independent subspaces of the object's state space, which allows us to independently perform the particle filter's propagations and corrections over small spaces. Second, we propose a *swapping* process to transform the weighted particle set provided by the update step of the particle filter into a "new particle set" better focusing on high peaks of the posterior distribution. This new methodology, called *Swapping-Based Partitioned Sampling*, is proved to be mathematically sound and is successfully tested and validated on synthetic video sequences for single or multiple articulated object tracking.

Keywords: Density approximation, Object tracking, Particle Filter, Dynamic Bayesian Networks, *d*-separation

1. Introduction

Object tracking is an important computer vision task for a wide variety of applications including gesture recognition [**?**], human tracking [**?**] and event detection [**?**]. However, tracking articulated structures with accuracy and within a reasonable time is challenging due to the high complexity of the problem to solve. Indeed, assessing the state of such objects requires the estimation of numerous parameters of their state vector (depending on the application, these correspond to models of the object's shape, appearance, *etc.*). This inevitably induces a representation of the tracking problem into a high-dimensional space. Nowadays, dealing with high-dimensional spaces has become a major concern for many research communities and is not only restricted to computer vision. In this paper, to solve single or multiple articulated object tracking problems, we focus on the particle filter methodology. It consists of estimating the probability distribution over the states of the tracked object(s) using weighted samples

whose elements are possible realizations of the object state. We propose to improve this method by exploiting conditional independences naturally present in the state space in order to transform by *swapping* processes these weighted samples into "new samples" better focusing on high peaks of the distribution to estimate. This enables to deal with high-dimensional spaces by reducing the number of elements in the samples required for accurate density estimations. In addition, for fixed sample size, swapping increases the accuracy of these estimations.

The paper is organized as follows. Section **??** recalls the particle filter's methodology and its main features. Section **??** then presents a short overview of the existing approaches that try to solve the high-dimensionality problem. Section **??** recalls the partitioned sampling approach and justifies its correctness using d-separation, the independence property at the core of dynamic Bayesian networks (DBN). Then, in Section **??**, we present our approach, which fully exploits d-separation both to parallelize some computations and to focus particles on the modes of the posterior distribution. Section **??** gives some experimental results on challenging synthetic video sequences. Finally, concluding remarks and perspectives are given in Section **??**. All the proofs are given in an appendix in Section **??**.

2. Theoretical framework: Particle Filter (PF)

Particle filters are an effective solution to the optimal filtering problem, in which the goal of tracking is to estimate a state sequence $\{\mathbf{x}_t\}_{t=1,\dots,T}$ whose evolution is specified by a dynamic equation $\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{n}_t^{\mathbf{x}})$ given a set of observations $\{\mathbf{y}_t\}_{t=1,\dots,T}$. Those are related to $\mathbf{x}_t$ by $\mathbf{y}_t = \mathbf{h}_t(\mathbf{x}_t, \mathbf{n}_t^{\mathbf{y}})$. Usually, $\mathbf{f}_t$ and $\mathbf{h}_t$ are vector-valued and time-varying functions and $\mathbf{n}_t^{\mathbf{x}}$ and $\mathbf{n}_t^{\mathbf{y}}$ are independent and identically distributed noise sequences. The evolution of such system can be modeled in a probabilistic way by a Markov chain as in Fig. **??**.

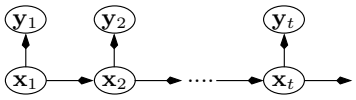


Figure 1: Evolution of a dynamic system by a Markov chain.

All these equations are usually considered in a probabilistic way although they have been extended to other uncertainty models such as Belief Functions [**?** **?**] or Fuzzy Logic [**?**]. The optimal filtering problem is then solved in two main steps. First, in a so-called *prediction step*, the density function $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ is computed, where $\mathbf{y}_{1:t-1}$ denotes tuple $(\mathbf{y}_1, \dots, \mathbf{y}_{t-1})$:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}, \tag{1}$$

with $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ the prior density related to transition function $\mathbf{f}_t$. Then, a

correction step is applied, which computes $p(\mathbf{x}_t|\mathbf{y}_{1:t})$:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1}), \tag{2}$$

with $p(\mathbf{y}_t|\mathbf{x}_t)$ the likelihood density related to the measurement function $\mathbf{h}_t$. Note that this precisely corresponds to the computations one would do in the Markov chain of Fig. **??** to compute efficiently $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

When functions $\mathbf{f}_t$ and $\mathbf{h}_t$ are linear, or linearizable, and when distributions are Gaussian or mixtures of Gaussians, sequence $\{\mathbf{x}_t\}_{t=1,\dots,T}$ can be computed analytically by Kalman, Extended Kalman or Unscented Kalman Filters [**?**]. Unfortunately, most vision tracking problems involve nonlinear functions and non-Gaussian distributions. In such cases, tracking methods based on Particle Filters (PF) [**? ?**], also called Sequential Monte Carlo (SMC) methods, can be applied under very weak hypotheses: their principle is to approximate the above non-parametric distributions by weighted samples $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$, $i = 1, \dots, N$, where each $\mathbf{x}_t^{(i)}$ corresponds to a hypothetical state realization and is called a *particle*. As optimal filtering approaches do, PF consists of two main steps. First, a *prediction* of the object state in the scene (using previous observations) is computed. It consists of propagating the set of particles $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$ according to a *proposal function* q:

$$\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_t).$$

A common practice is to set $q(\mathbf{x}_t|\mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_t)$ equal to $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$ and we do it as well in the rest of the paper. The propagation step is followed by a *correction* of this prediction (using a new available observation), that consists of *weighting* the particles w.r.t. a *likelihood function*, so that:

$$w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{x}_t^{(i)}) \frac{p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t|\mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_t)},$$

with $\sum_{i=1}^{N} w_t^{(i)} = 1$. Particles can then be resampled, so that those with the highest weights are duplicated, and those with the lowest weights are discarded. The estimation of the posterior distribution is then given by:

$$\sum_{i=1}^{N} w_t^{(i)} \delta_{\mathbf{x}_t^{(i)}}(\mathbf{x}_t),$$

where $\delta_{\mathbf{x}_t^{(i)}}$ are Dirac masses centered on particles $\mathbf{x}_t^{(i)}$. There exist many models of PF, each one having its own advantages. Unfortunately, the computational cost of PF highly depends on the size of the state and observations spaces. Indeed, as particle sets are nothing but samples of densities over state and observation spaces, the larger the latter, the higher the number of particles needed to estimate them. When dealing with articulated objects, things get even worse as the state and observation spaces are then roughly equal to the

Cartesian product of the spaces representing each part of the objects, hence leading to state and observation spaces of an exponential size. In this case, a naive PF requires an unrealistically large number of particles, thus inducing too large computation times to be usable in practice. More precisely, it has been shown in [**?**] that the number of particles needed to track an object grows exponentially with the dimension of the state space of this object. Being able to deal with high-dimensional state and observation spaces is thus a major concern for the vision tracking research community, especially when using the PF framework for articulated object tracking. This has led the researchers of the domain to propose several sophisticated methods aiming at reducing the number of necessary particles or the number of computations. The next section discusses some of them.

3. Dealing with high-dimensional state spaces

The algorithms dealing with high-dimensional state spaces can be roughly divided into three main classes: those that reduce the space's size by adding constraints to the model, those that perform local search and those that exploit some decomposition of the state space. Here, we briefly review the first two and, as our article belongs to the last class, we describe this one in more details.

The first class of approaches adds some constraints to the mathematical models in order to reduce the exploration of the state space. In many papers, this is achieved by introducing into the tracking model the physical constraints of the articulated objects [**?** **?**]. This approach is popular and very effective for human tracking problems. Thus, in [**?**], the constraints are introduced during the simulation step, while in [**?** **?**] they are included in a proposal density dedicated to human body tracking. Other approaches that add constraints include the addition of object priors [**?** **?** **?**], the exploitation of knowledge about the object's behaviors [**?**], about how the objects should look [**?** **?** **?**] and about their interactions with the environment [**?**].

The second class of approaches, often referred to as the optimization-based approaches, is the set of algorithms that combine PF with local search techniques [**?**]. Due to PF's stochastic nature and to the combinatorial size of the articulated object state spaces, PF is never guaranteed to produce particle sets nearby the modes of the densities. Consequently, combining it with local search techniques can significantly improve it by better focusing the particles on those modes. This explains why optimization approaches are popular among the community working on object tracking. Among them, gradient descent-based methods have received some attention. For instance, stochastic gradient-based descents have been successfully combined with PF [**?**] and new stochastic meta-descent (SMD) approaches have been proposed to work into a constrained space [**?**], leading to an efficient *Smart Particle Filter* [**?**]. Particle Swarm Optimization has also been used in conjunction with PF [**?** **?** **?** **?**]. Here, the idea is to apply evolutionary algorithms inspired from social behaviors observed in wildlife (birds, fishes, bees, ants, *etc.*) to make the particles evolve following their own experience and the one of their neighborhood (for a complete review on

this topic, see [**?**]). Similarly, the inclusion of population-based metaheuristics and genetic algorithms into PF has been investigated [**?**]. Simulated Annealing has also been introduced into PF. This has resulted in the very popular *Annealed Particle Filter* (APF) [**?**]. APF consists of adding annealing iterations (or layers) to PF's resampling step in order to better diffuse particles in the state space. Of course, there exist many other optimization-based methods (like scatter-search [**?**], *etc.*) that we will not cite here because this is not the purpose of this article. However, all those methods have in common that they rely on a subtle compromise between the quality of the approximation of the density they try to estimate and their speed of convergence. Actually, by their *local* nature, they converge quickly in the neighborhoods of their starting point but they require much more time to escape these neighborhoods and converge with some guarantee toward the modes of the densities. To avoid such problems, some of these approaches include hierarchical search strategies, in which the search spaces are refined progressively, starting from a coarse description of the state space and ending up into the complete description of state space $\mathcal{X}$. The *Progressive Particle Filter* [**?**] is an example of such a strategy. Finally, note that all these methods often suppose that all the needed observations are available at each time slice, which may not necessarily be the case in practice.

The third class of approaches, to which belongs our article, exploits natural decompositions of the state and observation spaces into a set of subspaces of reasonable sizes where PF can be applied. Partitioned Sampling (PS) [**?**] is most popular among these methods. It exploits the fact that, in many problems, both the system dynamics and the likelihood function are decomposable over small subspaces. The key idea, that will be detailed in Section **??**, is then to substitute the application of one PF over the whole state space by a sequence of applications of PF over these small subspaces, thus significantly speeding-up the process. Despite recent improvements [**?** **?** **?**], PS still suffers from numerous resampling steps that increase noise and decrease the tracking accuracy over time. The same kind of decomposition is exploited in [**?**] in the context of a general PF for Dynamic Bayesian Networks (DBN). Here, the proposal distributions of the prediction step are decomposed as the product of the conditional distributions of all the nodes of the current time slice in the DBN. The prediction step is then performed iteratively on each node of the network (following a topological order of the DBN) using as the proposal distribution the probability of the node conditionally to its parents in the DBN. In [**?**], the sampling idea of [**?**] is combined with the resampling scheme proposed in [**?**] in order to create a PF algorithm well-suited for DBNs. This algorithm can be seen as a generalization of PS. By following a DBN topological order for sampling and by resampling the particles each time an observed node is processed, particles with low likelihood for one subspace are discarded just after the instantiation of this subspace, whereas particles with high likelihood are multiplied. This has the same effect as *weighted resampling* in PS. Another approach inspired from the Bayesian network community is the nonparametric Belief Propagation algorithm [**?** **?**]. It combines the PF framework with the well-known Loopy Belief Propagation algorithm [**?** **?**] for speeding-up computations (but at the expense

of approximations). It has been successfully applied on many problems of high dimensions [**?** **?** **?** **?** **?**]. Another popular approach is the *Rao-Blackwellized Particle Filter for DBN* (RBPF) [**?**]. By using a natural decomposition of the joint probability, RBPF decomposes the state space into two parts that fulfill the following condition: the distribution of the second part conditionally to the first part can be estimated using classical techniques such as Kalman filter. The distribution of the first part is then estimated using PF and the conditional distribution of the second part given the first one is estimated using Kalman filter. As the dimension of the first part is smaller than that of the whole state space, the sampling step of particle filter for the first part needs fewer particles and the variance of the estimation can be reduced. Though RBPF is very efficient at reducing the high dimension of the problem, it cannot be applied on all DBNs because the state space cannot always be decomposed into two parts fulfilling the condition. The framework introduced in [**?**] is a parallel PF for DBNs that uses the same decomposition of the joint probability as that of the DBN to reduce the number of particles required for tracking. The state space is divided into several subspaces that are in some respect relatively independent. The particles for these subspaces can then be generated independently using different proposal densities. This approach offers a very flexible way of choosing the proposal density for sampling each subspace. However the definition of the different subspaces requires the DBN to have a particular independence structure, limiting the generalization of this algorithm. In our paper, we address more general problems where no such independences hold. We focus on PS [**?** **?**] for its simplicity and generalization potential. In [**?**], PS was proved to be comparable to the Annealed Particle Filter [**?**], which is one of the best algorithms for tracking problems of high dimensions. We believe that PS can be significantly improved by better exploiting the independences in DBNs. This idea will be developed in the next sections.

4. Partitioned Sampling and Dynamic Bayesian Networks

Our approach relies both on the exploitation of a PS-like scheme and on that of *d*-separation, the independence property at the core of DBN. Therefore, in Subsection **??**, we describe in details PS and, in Subsection **??**, we present Dynamic Bayesian Networks and show that PS's soundness is actually related to *d*-separation.

4.1. Partitioned Sampling (PS)

PS is an effective PF designed for tracking complex objects with large state space dimensions using only a reduced number of particles. Its key idea is to partition the state space into an appropriate set of subspaces and to apply sequentially PF on each subspace. PS uses a tailored sampling scheme, called "weighted resampling", which ensures that the sets of particles resulting from the sequential applications of PF actually represent the joint distribution of the whole state space and are focused on its peaks.

Weighted resampling is defined as follows: let $g : \mathcal{X} \mapsto \mathbb{R}$ be any strictly positive continuous function, where $\mathcal{X}$ denotes the state space. Given a set of particles $\mathcal{P}_t = \{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ with weights $w_t^{(i)}$, weighted resampling proceeds as follows: let ρ_t be defined as $\rho_t(i) = g(\mathbf{x}_t^{(i)}) / \sum_{j=1}^N g(\mathbf{x}_t^{(j)})$ for $i = 1, \ldots, N$. Select independently indices $k_1, \ldots, k_N$ according to probability ρ_t. Finally, construct a new set of particles $\mathcal{P}'_t = \{\mathbf{x}'^{(i)}_t, w'^{(i)}_t\}_{i=1}^N$ defined by $\mathbf{x}'^{(i)}_t = \mathbf{x}_t^{(k_i)}$ and $w'^{(i)}_t = w_t^{(k_i)} / \rho_t(k_i)$. MacCormick [**?**] shows that $\mathcal{P}'_t$ represents the same probability distribution as $\mathcal{P}_t$ while focusing the particles on the peaks of g.

PS's key idea is to exploit some natural decomposition of the system dynamics w.r.t. subspaces of the state space in order to apply PF only on those subspaces. This leads to a significant reduction in the number of particles needed for tracking. So, assume that state space $\mathcal{X}$ and observation space $\mathcal{Y}$ can be partitioned as $\mathcal{X} = \mathcal{X}^1 \times \cdots \times \mathcal{X}^P$ and $\mathcal{Y} = \mathcal{Y}^1 \times \cdots \times \mathcal{Y}^P$ respectively. For instance, a system representing a hand could be defined as $\mathcal{X}^{\text{hand}} = \mathcal{X}^{\text{palm}} \times \mathcal{X}^{\text{thumb}} \times \mathcal{X}^{\text{index}} \times \mathcal{X}^{\text{middle}} \times \mathcal{X}^{\text{ring}} \times \mathcal{X}^{\text{little}}$. Assume in addition that the dynamics of the system follows this decomposition, i.e., that:

$$f_t(\mathbf{x}_{t-1}, n_t^{\mathbf{x}}) = f_t^P \circ f_t^{P-1} \circ \cdots \circ f_t^2 \circ f_t^1(\mathbf{x}_{t-1}), \tag{3}$$

where $\circ$ is the usual function composition operator and where each function $f_t^i : \mathcal{X} \mapsto \mathcal{X}$ modifies the particles' states only on subspace $\mathcal{X}^i$ [1].

The PF scheme consists of resampling particles, of propagating them using proposal function f_t and, finally, of updating their weights using the observations at hand. Here, the same result can be achieved by substituting the f_t propagation step by the sequence of applications of the f_t^i as given in Eq. (**??**), each one followed by a weighted resampling that produces new particles sets focused on the peaks of a function g. To be effective, PS thus needs g to be peaked with the same region as the posterior distribution restricted to $\mathcal{X}^i$. When the likelihood function decomposes as well on subsets $\mathcal{Y}^i$, i.e., when:

$$p(\mathbf{y}_t|\mathbf{x}_t) = \prod_{i=1}^{P} p^i(\mathbf{y}_t^i|\mathbf{x}_t^i), \tag{4}$$

where $\mathbf{y}_t^i$ and $\mathbf{x}_t^i$ are the projections of $\mathbf{y}_t$ and $\mathbf{x}_t$ on $\mathcal{Y}^i$ and $\mathcal{X}^i$ respectively, weighted resampling focusing on the peaks of the posterior distribution on $\mathcal{X}^i$ can be achieved by first multiplying the particles' weights by $p^i(\mathbf{y}_t^i|\mathbf{x}_t^i)$ and, then, by performing a usual resampling. Note that Eq. (**??**) naturally arises when tracking articulated objects. This leads to the condensation diagram given in Fig. **??**, where operations "$*f_t^i$" refer to propagations of particles using proposal function f_t^i as defined above, "$\times p_t^i$" refers to the correction steps where particle weights are multiplied by $p^i(\mathbf{y}_t^i|\mathbf{x}_t^i)$ (see Eq. (**??**)), and "$\sim$" refers to usual resamplings. MacCormick and Isard showed that this diagram produces mathematically correct results [**?**].

[1] Note that, in [**?**], functions f_t^i are more general since they can modify states on $\mathcal{X}^i \times \cdots \times \mathcal{X}^p$. However, in practice, particles are often propagated only one $\mathcal{X}^j$ at a time.

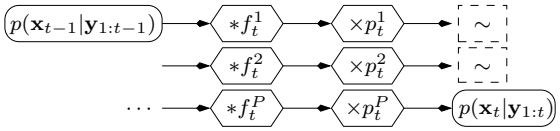


Figure 2: Partitioned Sampling condensation diagram: PS starts with a particle set estimating $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$. It first propagates particles using proposal function f_t^1 (applied on $\mathcal{X}^1$), then it corrects them using function p_t^1 and it resamples them. Second, it propagates and corrects the second part of the resulting particle set (over $\mathcal{X}^2$) using f_t^2 and p_t^2 respectively, and it resamples the result. And so on. After iterating over the P parts of the particles, the particle set estimates $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

4.2. d-separation: the independences assumed by PS

The hypotheses used by PS can best be explained by a DBN representing the conditional (in)dependences between random variables of states and observations [**?**]:

Definition 1 (Bayesian network (BN)). *A BN is a pair* $(G, \mathbf{P})$ *where* $G = (\mathbf{V}, \mathbf{A})$ *is a directed acyclic graph (DAG) and each node* $X \in \mathbf{V}$ *corresponds to a random variable*[2]. $\mathbf{P}$ *is the set of the probability distributions of each node* $X \in \mathbf{V}$ *conditionally to its parents* $\mathbf{pa}(X)$ *in* G*, i.e.,* $p(X|\mathbf{pa}(X))$*. The joint probability over all the random variables in* $\mathbf{V}$ *is then the product of all these conditional distributions:*

$$p(\mathbf{V}) = \prod_{X \in \mathbf{V}} p(X|\mathbf{pa}(X)).$$

BNs [**?**] can model the uncertainties inherent to object tracking problems: in this case, $\mathbf{V}$ is the set of state variables $\{\mathbf{x}_t\}$ and observation variables $\{\mathbf{y}_t\}$. The probabilistic dependencies between these variables are then encoded by arcs in the network. Fig. **??** thus represents a generic BN designed for object tracking. For articulated objects, we have seen previously that state space $\mathcal{X}$ can be partitioned into $\mathcal{X}^1 \times \cdots \times \mathcal{X}^P$. In this case, instead of having only one node $\mathbf{x}_t$ per "time slice", the BN contains one node $\mathbf{x}_t^i$ per $\mathcal{X}^i$ and per time slice. The probabilistic relationships between these variables are again encoded by arcs and the BN looks like that of Fig. **??**. DBNs, or more precisely 2TBNs, are an extension of BNs specifically designed to cope with time evolving systems:

Definition 2 (DBN: 2-slice Temporal Bayesian Network (2TBN)). *A 2TBN is a pair* $(B_1, B_\rightarrow)$ *of BN.* B_1 *represents the BN at time slice* $t = 1$ *and represents the joint probability* $p(\mathbf{x}_1, \mathbf{y}_1)$*. The BN* $B_\rightarrow$ *defines the transition between different time slices* $p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{x}_{t-1}, \mathbf{y}_{t-1})$*. It is assumed that the BN in each time slice* $t > 1$ *is identical to* $B_\rightarrow$*.*

[2]By abuse of notation, since there is a one-to-one mapping between nodes in $\mathbf{V}$ and random variables, we will use interchangeably $X \in \mathbf{V}$ (resp. $\mathbf{V}$) to denote a node in the network (resp. all the nodes) and its corresponding random variable (resp. all the random variables).

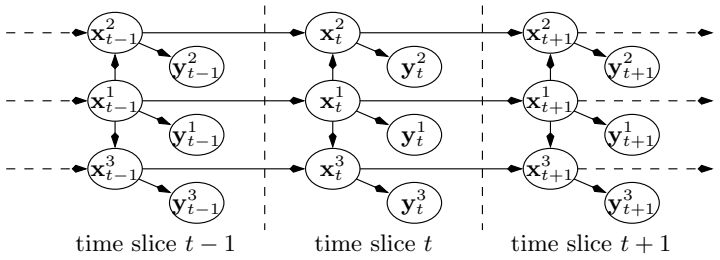


Figure 3: A Dynamic Bayesian Network for body tracking.

For instance, Fig. **??** represents an unrolled 2TBN: B_1 is constituted by a BN $\mathbf{x}_1 \rightarrow \mathbf{y}_1$, i.e., the BN at time slice 1, and $B_{\rightarrow}$ corresponds to the BN of the second time slice (including the arcs from time slice 1 to time slice 2). For articulated object tracking, the BN in each time slice can have several nodes. For instance, assume that an object to be tracked is composed of 3 parts: a torso, a left arm and a right arm. Let $\mathbf{x}^1_t, \mathbf{x}^2_t, \mathbf{x}^3_t$ represent these parts respectively and $\mathbf{y}^1_t, \mathbf{y}^2_t, \mathbf{y}^3_t$ represent the observations on these nodes. Then this body tracking can be represented by the DBN of Fig. **??**. Indeed, in this figure, the arcs actually represent the probabilistic dependences between the nodes. For instance, there is a direct dependence between torso $\mathbf{x}^1_t$ and right arm $\mathbf{x}^2_t$, and the position of the torso at time t has a direct influence on its position at time $t+1$, hence the arc $\mathbf{x}^1_t \rightarrow \mathbf{x}^1_{t+1}$.

From a particle filtering perspective, the key feature of BNs and DBNs is their graphical representation of probabilistic independences, which is called the d-separation criterion [**?**].

Definition 3 (d-separation [?]). *Two nodes $\mathbf{x}^i_t$ and $\mathbf{x}^j_s$ of a DBN are dependent conditionally to a set of nodes $\mathbf{Z}$ if and only if there exists a chain $\{\mathbf{c}_1 = \mathbf{x}^i_t, \ldots, \mathbf{c}_n = \mathbf{x}^j_s\}$ linking $\mathbf{x}^i_t$ and $\mathbf{x}^j_s$ in the DBN such that the following two conditions hold:*

1. *for every node $\mathbf{c}_k$ such that the DBN's 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 $\mathbf{Z}$;*
2. *none of the other nodes $\mathbf{c}_k$ belongs to $\mathbf{Z}$.*

Such a chain is called active *(else it is called* blocked*). If there exists an active chain linking two nodes, these nodes are dependent and are called d-connected, otherwise they are independent conditionally to $\mathbf{Z}$ and are called d-separated.*

For instance, conditionally to states $\mathbf{x}^i_t$, observations $\mathbf{y}^i_t$ of Fig. **??** are independent of the other random variables. Consequently, Eq. (**??**) implicitly holds in this DBN. In addition, by Definition **??**, $\mathbf{x}^1_t$ is independent of $\mathbf{x}^2_{t-1}$ and $\mathbf{x}^3_{t-1}$ conditionally to $\{\mathbf{x}^1_{t-1}\}$. Similarly, $\mathbf{x}^2_t$ is independent of $\mathbf{x}^3_{t-1}$ conditionally to $\{\mathbf{x}^1_t, \mathbf{x}^2_{t-1}\}$ and $\mathbf{x}^3_t$ is independent of $\mathbf{x}^2_t$ conditionally to $\{\mathbf{x}^1_t, \mathbf{x}^3_{t-1}\}$. As a consequence, the condensation diagram of Fig. **??** can be exploited to track this object since the "probabilistic" propagations/corrections of each part of

the object depend only on this part and its parents in the DBN. Therefore, by their d-separation property, DBNs provide a sound mathematical framework for proving the correctness of PS. But they can do more, as we will prove in the next section, which describes our main contribution.

5. Improving Particle Filtering by exploiting DBNs

The first step of our approach, which is presented in Subsection **??**, consists of parallelizing PS's propagations/corrections among carefully selected sets of object parts $\mathbf{x}_t^i$. This allows us to reduce the number of necessary resamplings, hence increasing the tracking accuracy. Then, and it is the main contribution of this paper, we show in Subsection **??** how permutations over some subsamples of the object parts processed in parallel can improve the estimation of the posterior density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ by focusing the particle set near the modes of this density while still guaranteeing that the density is correctly estimated.

5.1. A d-separation-based parallelization of PS

By d-separation, it can be proved that, in Fig. **??**, $\mathbf{x}_t^3$ is independent of $\mathbf{x}_{t-1}^2$ conditionally to $\{\mathbf{x}_t^1, \mathbf{x}_{t-1}^3\}$. Combining this independence property with those mentioned at the end of the preceding section, this implies that the propagations/corrections of $\mathbf{x}_t^2$ and $\mathbf{x}_t^3$ can be performed in parallel since none of them has any impact on the other one. Thus, this suggests the new condensation diagram of Fig. **??** where object parts $\mathbf{x}_t^2$ and $\mathbf{x}_t^3$ are processed in parallel.

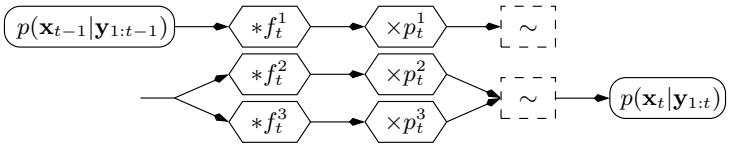


Figure 4: Condensation diagram exploiting conditional independences.

Of course, this illustrative example can be easily extended to more general 2TBNs. In this paper, we will focus on DBNs in which observations depend only on their corresponding state and in which the proposal transition function of a given part $\mathbf{x}_t^i$ of the object at time t depends only on that part in time $t-1$ (and possibly on other parts in time t). Note that these assumptions are rather mild for object tracking and they hold in most applications. For instance, they hold in Fig. **??**. More formally, this leads to the following definition:

Definition 4 (Object Tracking DBN – OTDBN). *An OTDBN is a 2TBN defined over random variables $\{\mathbf{x}_t^i, \mathbf{y}_t^i\}$, where $\mathbf{x}_t^i$ and $\mathbf{y}_t^i$ represent the state and observation of part i in time slice t, and satisfy the following four conditions:*

1. *there does not exist any arc $\mathbf{x}_s^i \rightarrow \mathbf{x}_t^j$ with $s < t-1$ or $s > t$;*
2. *for every i and $t > 1$, there exists an arc $\mathbf{x}_{t-1}^j \rightarrow \mathbf{x}_t^i$ if and only if $j = i$;*
3. *for each node $\mathbf{x}_t^i$, there exists a node $\mathbf{y}_t^i$ whose only parent is $\mathbf{x}_t^i$;*

4. *nodes* $\mathbf{y}_t^i$ *have no children.*

Now, let us extend the condensation diagram of Fig. **??** for arbitrary OTDBNs. Let X_t denote a generic node of an OTDBN $\mathcal{G}$ in time slice t (so either $X_t = \mathbf{x}_t^i$ or $X_t = \mathbf{y}_t^i$ for some i). Let $\mathbf{pa}(X_t)$ and $\mathbf{pa}_t(X_t)$ denote the set of parents of X_t in $\mathcal{G}$ in all time slices and in time slice t only respectively. For instance, in Fig. **??**, $\mathbf{pa}(\mathbf{x}_t^2) = \{\mathbf{x}_t^1, \mathbf{x}_{t-1}^2\}$ and $\mathbf{pa}_t(\mathbf{x}_t^2) = \{\mathbf{x}_t^1\}$. Similarly, let $\mathbf{an}(X_t)$ and $\mathbf{an}_t(X_t)$ be the set of ancestors of X_t in all time slices and in time slice t only respectively. Assume that the tracked object state space $\mathcal{X}$ is decomposed as $\mathcal{X} = \mathcal{X}^1 \times \cdots \times \mathcal{X}^P$ and that the probabilistic dependences between all random variables $\mathbf{x}_t^i$ and $\mathbf{y}_t^i$, $i = 1, \ldots, P$, are represented by an OTDBN $\mathcal{G}$. Let $\{P_1, \ldots, P_K\}$ be a partition of $\{1, \ldots, P\}$ defined by:

- $P_1 = \{k \in \{1, \ldots, P\} : \mathbf{pa}_t(\mathbf{x}_t^k) = \emptyset\}$;
- for any $j > 1$, $P_j = \{k \in \{1, \ldots, P\} \backslash \cup_{h=1}^{j-1} P_h : \mathbf{pa}_t(\mathbf{x}_t^k) \subseteq \bigcup_{h=1}^{j-1} \bigcup_{r \in P_h} \{\mathbf{x}_t^r\}\}$.

Intuitively, P_1 is the set of indices k of the object parts $\mathbf{x}_t^k$ that have no parent in time slice t (in Fig. **??**, $P_1 = \{1\}$); P_2 is the set of indices of the object parts whose parents in time slice t all belong to P_1 (in Fig. **??**, $P_2 = \{2, 3\}$), and so on.

It is not hard to see that, by d-separation, all the nodes $\mathbf{x}_t^k$ of a given P_j are independent conditionally to their parents $\mathbf{pa}(\mathbf{x}_t^k)$. Consequently, PS can propagate/correct all these nodes independently (in parallel) and produce a correct estimation of the *a posteriori* joint density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. This suggests the condensation diagram of Fig. **??** where, for every $j \in \{1, \ldots, K\}$, $P_j = \{i_{P_j}^1, \ldots, i_{P_j}^{k_j}\}$ and each object part $\mathbf{x}^{i_{P_j}^h}$ has its own proposal function $f_t^{i_{P_j}^h}$ and its own correction function $p_t^{i_{P_j}^h}$ (as described in Eq. (**??**)). In this diagram, all the propagations and corrections of the object parts in a given set P_j are thus performed in parallel and, subsequently, a resampling is performed over all these parts. The correctness of the diagram follows from Proposition **??**.

Proposition 1. *The set of particles resulting from the diagram of Fig.* **??** *represents probability distribution* $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

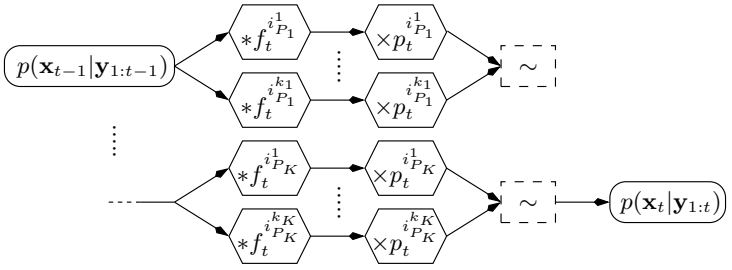


Figure 5: Parallelized Partitioned Sampling condensation diagram.

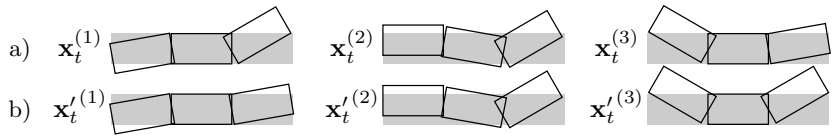


Figure 6: The particle swapping scheme: a) before swapping; b) after swapping

5.2. Swapping-Based Partitioned Sampling

There are two major differences between the diagrams of Fig. **??** and Fig. **??**: the latter performs fewer resamplings, thus it introduces less noise in the particle set and, more importantly, it enables to produce better fitted particles by swapping their subparts. Actually, consider again our body tracking example and assume that we generated the 3 particles $\mathbf{x}_t^{(i)}$ of Fig. **??**.a where $\mathcal{X}^1$ is the middle part of the object and $\mathcal{X}^2$ and $\mathcal{X}^3$ are its left and right parts respectively, and where the shaded areas represent the object's true state. According to the DBN of Fig. **??**, for fixed values of $\mathbf{x}_{1:t}^1$, the sets of left and right parts of the particles represent $p(\mathbf{x}_t^2, \mathbf{y}_{1:t}^2 | \mathbf{x}_{1:t}^1)$ and $p(\mathbf{x}_t^3, \mathbf{y}_{1:t}^3 | \mathbf{x}_{1:t}^1)$ respectively (summing out variables $\mathbf{x}_{1:t-1}^2, \mathbf{x}_{1:t-1}^3$ from the DBN). Hence, after permuting the values of the particles on $\mathcal{X}^2$ (resp. $\mathcal{X}^3$) for a fixed value of $\mathbf{x}_{1:t}^1$, distribution $p(\mathbf{x}_t^2, \mathbf{y}_{1:t}^2 | \mathbf{x}_{1:t}^1)$ (resp. $p(\mathbf{x}_t^3, \mathbf{y}_{1:t}^3 | \mathbf{x}_{1:t}^1)$) remains unchanged. A fortiori, this does not affect the representation of the joint posterior distribution $\int p(\mathbf{x}_{1:t}^1, \mathbf{y}_{1:t}^1) p(\mathbf{x}_t^2, \mathbf{y}_{1:t}^2 | \mathbf{x}_{1:t}^1) p(\mathbf{x}_t^3, \mathbf{y}_{1:t}^3 | \mathbf{x}_{1:t}^1) d\mathbf{x}_{1:t-1}^1 = p(\mathbf{x}_t, \mathbf{y}_{1:t})$. On Fig. **??**.a, particles $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(3)}$ have the same state on $\mathcal{X}^1$. Thus their right parts can be permuted, resulting in the new particle set of Fig. **??**.b. Remark that we substituted 2 particles, $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(3)}$, which had low weights due to their bad estimation of the object's right or left part states, by one particle $\mathbf{x}_t'^{(1)}$ with a high weight (due to a good estimation of all the object's parts) and another one $\mathbf{x}_t'^{(3)}$ with a very low weight. After resampling, the latter will most probably be discarded and, therefore, swapping will have focused particles on the peaks of the posterior distribution. Note however that not all permutations are allowed: for instance, none can involve particle $\mathbf{x}_t^{(2)}$ because its center part differs from that of the other particles.

The SBPS algorithm introduces these swappings into the diagram of Fig. **??**, which leads to that of Fig. **??**, where operations "$\rightleftharpoons^{P_j}$" refer to the particle subpart swappings briefly described above. Remark that, after the resampling operation of part P_j, the particles with high weights will be duplicated, which will enable swapping when processing next part P_{j+1}.

Swappings need however to be further formalized. For this purpose, we need more precise notations. As SBPS does not process all the object parts at the same time, we will denote by $Q_j = \cup_{h=1}^{j} P_h$ and $R_j = \cup_{h=j+1}^{K} P_h$ the set of parts already processed at the jth step by SBPS and those still to be processed respectively. In addition, let $Q_0 = R_K = \emptyset$. Thus $(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{1:t-1}^{(i),R_j})$ now represents the state of the ith particle after j steps of SBPS and, as shown in the proof of Proposition **??**, $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j}), w^{(i)}\}$ estimates $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. Sim-

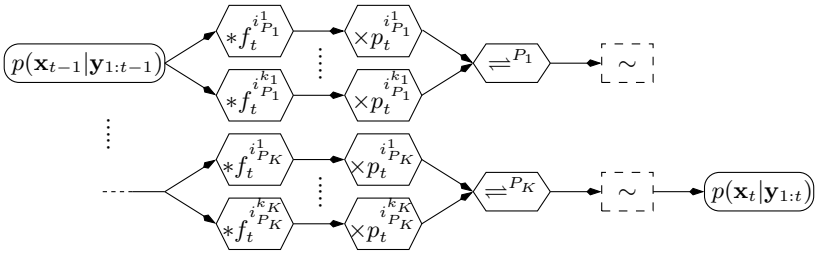


Figure 7: Swapping-based Partitioned Sampling (SBPS) condensation diagram.

ilarly, when needed, for any set $H \subseteq \{1, \ldots, P\}$, $\mathbf{x}_{1:t}^{(i),H}$ will refer to the whole trajectory of the subparts in H of the ith particle, i.e., their values from time slice 1 to time slice t. Now, to guarantee that swapping operations $\rightleftharpoons^{P_j}$ do not alter the estimated distributions, it is not sufficient to permute only the subparts in P_j. The reason why can be easily understood using the OTDBN: the network encodes the joint distribution using conditional probabilities of the type $p(\mathbf{x}_t^h|\mathbf{pa}(\mathbf{x}_t^h))$, hence permuting only the elements of some subpart $\mathbf{x}_t^k$ in sample $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}$ will change the parents' values of any child $\mathbf{x}_t^r$ of $\mathbf{x}_t^k$, and thus the sample will no longer estimate correctly $p(\mathbf{x}_t^r|\mathbf{pa}(\mathbf{x}_t^r))$ nor the joint probability $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}|\mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. To avoid this, we need to permute $\{\mathbf{x}_t^{(i),r}\}$ similarly to $\{\mathbf{x}_t^{(i),k}\}$. More generally, it is compulsory to permute all the nodes that are linked to $\mathbf{x}_t^k$ by chains that do not pass through $\mathbf{x}_t^k$'s ancestors. More formally, to guarantee that the estimation of the densities is unaltered by swappings, the set of subparts to permute similarly to $\mathbf{x}_t^k$ is called a "*swapping set*":

Definition 5 (Swapping set). *Let $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}$ be the particle set at the jth step of SBPS. Let $k \in P_j$ and let $Link_t(\mathbf{x}_t^k)$ denote the set $\{\mathbf{x}_t^k\} \cup \{\mathbf{x}_t^r$: there exists a chain between $\mathbf{x}_t^k$ and $\mathbf{x}_t^r$ passing only by nodes in time slice t and by no node in the ancestor set $\mathbf{an}_t(\mathbf{x}_t^k)\}$. In addition, let $Link_t^{Q_j}(\mathbf{x}_t^k) = \{\mathbf{x}_t^r \in Link_t(\mathbf{x}_t^k) : r \in Q_j\}$ and $Link_{t-1}^{R_j}(\mathbf{x}_t^k) = \{\mathbf{x}_{t-1}^r : \mathbf{x}_t^r \in Link_t(\mathbf{x}_t^k)$ and $r \in R_j\}$. Then, the set $Link_t^{Q_j}(\mathbf{x}_t^k) \cup Link_{t-1}^{R_j}(\mathbf{x}_t^k)$ is called a swapping set.*

For instance, in the OTDBN of Fig. **??** (where observation nodes have not been displayed to simplify the figure), the gray nodes represent the swapping set of node $\mathbf{x}_t^k$. We introduced swappings for fixed values of the parents of some nodes. As shown below, the d-separation criterion imposes that they correspond to the values of the nodes in Link_t over *all time slices*. In other words, only *admissible permutations* guarantee that densities are correctly estimated:

Definition 6 (admissible permutation). *Let $k \in P_j$ and $A_t^{Q_j} = \mathbf{an}_t(\mathbf{x}_t^k) \cap (\cup_{\mathbf{x}_t^h \in Link_t^{Q_j}(\mathbf{x}_t^k)} \mathbf{pa}_t(\mathbf{x}_t^h))$ and $A_{t-1}^{R_j} = \mathbf{an}_{t-1}(\mathbf{x}_{t-1}^k) \cap (\cup_{\mathbf{x}_{t-1}^h \in Link_{t-1}^{R_j}(\mathbf{x}_t^k)} \mathbf{pa}_{t-1}(\mathbf{x}_{t-1}^h))$. Finally, let $A = \{x_s^h : x_t^h \in A_t^{Q_j}, 1 \leq s \leq t\} \cup \{x_s^h : x_{t-1}^h \in A_{t-1}^{R_j}, 1 \leq s \leq t-1\}$. A permutation $\sigma : \{1, \ldots, N\} \mapsto \{1, \ldots, N\}$ is said to be admissible if and only if $\mathbf{x}_t^{(i),h} = \mathbf{x}_t^{(\sigma(i)),h}$ for all $h \in A$ and all $i \in \{1, \ldots, N\}$.*

In Fig. **??**, sets $A_t^{Q_j}$ and $A_{t-1}^{R_j}$ correspond to the black nodes and A to the two lines of nodes in the dotted polygon at the bottom of the figure.

Proposition 2. *For any $j \in \{1, \ldots, K\}$, let $\rightleftharpoons^{P_j}$ be a set of admissible permutations σ_k of subparts $k \in P_j$ applied to the swapping set of $\mathbf{x}_t^k$. Then the set of particles resulting from SBPS represents $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.*

5.3. SBPS in practice

It is important to note that, to be admissible, a permutation can only swap particles having identical values for *every* object part in A, i.e., on the parts in $A_s^{Q_j}$ and $A_s^{R_j}$ for *all* time slices s, not just those in $A_t^{Q_j} \cup A_{t-1}^{R_j}$, i.e, on the current time slice. Actually, in theory, only considering the current time slice may lead to incorrect results. For instance, in the DBN of Fig. **??**, $P_2 = \{2, 3\}$, i.e., object parts $\mathbf{x}_t^2$ and $\mathbf{x}_t^3$ are propagated/corrected in parallel and, therefore, are good candidates for swapping. Now, $\text{Link}_t(\mathbf{x}_t^2) = \{\mathbf{x}_t^2\}$, $A_t^{Q_2} = \{\mathbf{x}_t^1\}$ and $A_{t-1}^{R_2} = \emptyset$. By d-separation, $\text{Link}_t(\mathbf{x}_t^2)$ is not independent from $\mathbf{x}_t^3$ conditionally to $A_t^{Q_2} \cup A_{t-1}^{R_2}$ because the chain $\{\mathbf{x}_t^2, \mathbf{x}_{t-1}^2, \mathbf{x}_{t-1}^1, \mathbf{x}_{t-1}^3, \mathbf{x}_t^3\}$ is active. Hence, swapping over $\text{Link}_t(\mathbf{x}_t^2)$'s parts some particles that have the same value on $A_t^{Q_2}$ and $A_{t-1}^{R_2}$ but not on A, can modify the estimation of the joint posterior density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ since the $\mathbf{x}_t^3$ part is not swapped accordingly.

However, in practice, whenever two particles have the same value on $A_t^{Q_j} \cup A_{t-1}^{R_j}$, the continuous nature of the state space make it highly probable that one particle is a copy of the other due to resampling. Hence, their values on the whole of A should also be identical. In other words, due to the continuous nature of the state space, whenever two particles have precisely the same values on $A_t^{Q_j} \cup A_{t-1}^{R_j}$, they should also have the same values on the whole of A. This leads to the following kind of permutations:

Definition 7 (almost admissible permutation). *Let $k \in P_j$. A permutation $\sigma : \{1, \ldots, N\} \mapsto \{1, \ldots, N\}$ is said to be almost admissible if and only if $\mathbf{x}_t^{(i),h} = \mathbf{x}_t^{(\sigma(i)),h}$ for all $h \in A_t^{Q_j} \cup A_{t-1}^{R_j}$ and all $i \in \{1, \ldots, N\}$.*

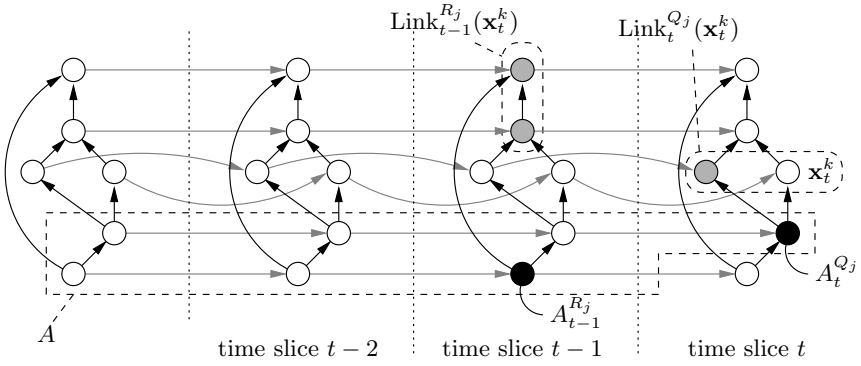


Figure 8: Swapping sets and admissible permutations.

Our implementation approximates the posterior distributions by performing swappings $\rightleftharpoons^{P_j}$ using almost admissible permutations. The advantage is that we do not need to keep track of the trajectories of the particles from time slice 1 to t. Another improvement can be made in the case where, considering only a single time slice, the OTDBN is a tree or a forest. This is precisely the case in Fig. **??** and it is often the case in articulated objects tracking. In such setting, it is easy to see that $\text{Link}_t^{Q_j}(\mathbf{x}_t^k) = \{\mathbf{x}_t^k\}$, $\text{Link}_{t-1}^{R_j}(\mathbf{x}_t^k) = \{\text{descendants of } \mathbf{x}_t^k \text{ in time slice } t\}$, $A_t^{Q_j} = \mathbf{pa}_t(\mathbf{x}_t^k)$ and $A_{t-1}^{R_j} = \emptyset$. So we shall permute only particles with the same value of $\mathbf{pa}_t(\mathbf{x}_t^k)$. Note that, when the single time slice subnetworks of OTDBNs are not trees or forest, there may exist some nodes $\mathbf{x}_t^h \in \text{Link}_t^{Q_j}(\mathbf{x}_t^k)$ such that $h \in P_j$. This is for instance the case of the node at the left of $\mathbf{x}_t^k$ in Fig. **??**. In this case, $\text{Link}_t^{Q_j}(\mathbf{x}_t^h) = \text{Link}_t^{Q_j}(\mathbf{x}_t^k)$ and $\text{Link}_{t-1}^{R_j}(\mathbf{x}_t^h) = \text{Link}_{t-1}^{R_j}(\mathbf{x}_t^k)$, so it is useless to consider permuting particles first w.r.t. $\mathbf{x}_t^k$ and, then, w.r.t. $\mathbf{x}_t^h$, the second permutation canceling the first one. Such a case cannot happen when single time slice subnetworks are trees or forests.

Finally, let us show how $\rightleftharpoons^{P_j}$ can determine attractive swappings, i.e., how high-peaked regions can be discovered. From the preceding paragraph, two particles with the same value on some $\mathbf{pa}_t(\mathbf{x}_t^r)$ have most probably been generated from the duplication of the same particle during a resampling step. In this case, for all $k \in P_j$, they also have the same value of $\mathbf{pa}_t(\mathbf{x}_t^k)$. We can then partition sample $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}$, $i = 1, \ldots, N$, 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 $\mathbf{pa}_t(\mathbf{x}_t^k)$ for some $k \in P_j$. For each N_r, all possible permutations are eligible. Let $\{r_1, \ldots, r_s\}$ be the elements of N_r. For each $k \in P_j$, let σ_k be the permutation that sorts weights $w_t^{(r_h),k}$, $h = 1, \ldots, s$, in decreasing order. By applying σ_k for all nodes in $\text{Link}_t^{Q_j}(\mathbf{x}_t^k) \cup \text{Link}_{t-1}^{R_j}(\mathbf{x}_t^k)$ and all $k \in P_j$, we get a permutation operation $\rightleftharpoons^{P_j}$ that assigns to the first particle the set of the highest weights, to the second one, the set of second best weights, *etc.* 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 following proposition shows that this process results in the permutation that increases the more the sum of the particles (when function f is linear) and that this tends to increase more the weights of the particles with high weights (when function f is strictly convex). This justifies that our swappings tend to move the particles toward the modes of the distributions.

Proposition 3. *Let $f : \mathbb{R} \mapsto \mathbb{R}$ be a strictly increasing convex function. Let $P_j = \{i_1, \ldots, i_{|P_j|}\}$ be a set of conditional independent subparts and let $\{w^{(i),k}\}$, $i \in 1, \ldots, N$, $k \in P_j$, be the set of weights of N particles on subpart k. Let $\{v^{(i),k}\}$, $i \in 1, \ldots, N$, $k \in P_j$, be the weights resulting from the application of $\rightleftharpoons^{P_j}$ on $\{w^{(i),k}\}$. Finally, let $\mathcal{S}_{i_r}$, $r \in \{1, \ldots, |P_j|\}$, denote the set of almost admissible permutations over $\mathbf{x}_t^{i_r}$ and let $\mathcal{S} = \mathcal{S}_{i_1} \times \cdots \times \mathcal{S}_{i_{|P_j|}}$. Then $\rightleftharpoons^{P_j}$ is the unique set of permutations such that:*

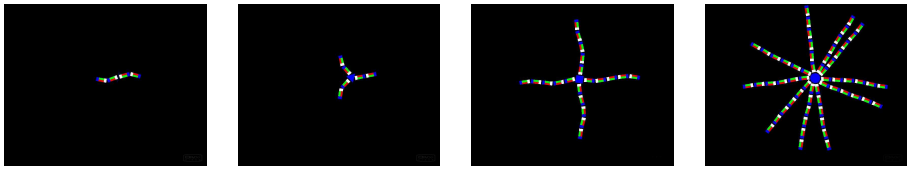
$$\sum_{i=1}^{N} f\left(\prod_{k \in P_j} v^{(i),k}\right) = \max_{(\sigma_{i_1}, \ldots, \sigma_{i_{|P_j|}}) \in \mathcal{S}} \sum_{i=1}^{N} f\left(\prod_{k \in P_j} w^{(\sigma_k(i)),k}\right). \tag{5}$$

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

6. Experimental results

6.1. Video sequences

Our experiments are performed on synthetic video sequences because they allow not to have to take into account specific properties of images (noise, *etc.*) and, additionally, they make it possible to simulate specific motions. This thus allows to compare particle filters on the sole basis of their estimation accuracy and, moreover, to focus comparisons on their different features. Therefore, we have generated our own synthetic video sequences composed of 300 frames of 800×640 pixels. Each video displays an articulated object randomly moving and deforming over time, subject to weak or strong motions. Some examples are given in Fig. **??**. With various numbers of parts, articulated objects are designed to test the capacity of particle filters to deal with high-dimensional state spaces. Our experiments are conducted to compare our approach against PS.



$K=3$, $\lvert P_j\rvert=2$	$K=3$, $\lvert P_j\rvert=3$	$K=6$, $\lvert P_j\rvert=4$	$K=7$, $\lvert P_j\rvert=10$
$P=5$, $\lvert\mathcal{X}\rvert=15$	$P=7$, $\lvert\mathcal{X}\rvert=21$	$P=21$, $\lvert\mathcal{X}\rvert=63$	$P=61$, $\lvert\mathcal{X}\rvert=183$

Figure 9: Examples of frames extracted from our synthetic video sequences (300 frames of 800×640 pixels), and the features of the corresponding articulated objects (number of arms $|P_j|$, $j>1$, length of arms $K-1$, total number of parts P, and dimension of state space $\mathcal{X}$).

6.2. Experimental setup

The tracked objects are modeled by a set of P polygonal parts (or regions): a central one P_1 (containing only one polygon) to which are linked $|P_j|$, $j>1$, arms of length $K-1$ (see Section **??**, and Fig. **??** for some examples). The polygons are manually positioned in the first frame. State vectors contain the parameters describing all the parts and are defined by $\mathbf{x}_t=\{\mathbf{x}_t^1,\ldots,\mathbf{x}_t^P\}$, with $\mathbf{x}_t^k=\{x_t^k,y_t^k,\theta_t^k\}$, where (x_t^k,y_t^k) is the center of part k, and θ_t^k is its orientation, $k=1,\ldots,P$. We thus have $|\mathcal{X}|=3P$. A particle $\mathbf{x}_t^{(i)}=\{\mathbf{x}_t^{(i),1},\ldots,\mathbf{x}_t^{(i),P}\}$, $i=1,\ldots,N$, 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 $\sigma_\theta = 0.025$ rad).

A classical approach to compute particle weights consists of integrating the color distributions given by histograms into particle filtering [**?**]. In such cases, we measure the similarity between the distribution of pixels in the region of the estimated part of the articulated object and that of the corresponding reference region. This similarity is determined by the Bhattacharyya distance $\mathbf{d}$ [**?**] between the histograms of the target and the reference regions. The particle weights are then computed by $w_{t+1}^{(i)} = w_t^{(i)} p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}^{(i)}) \propto w_t^{(i)} e^{-\lambda \mathbf{d}^2}$, with, in our tests, $\lambda = 50$ and $\mathbf{d}$ the Bhattacharyya distance between the target (prior) and the reference (previously estimated) 8-bin histograms. In both PS and SBPS, the articulated object's joint distribution is estimated starting from its central part P_1. Then, PS propagates and corrects particles polygon after polygon to derive a global estimation of the object (see the condensation diagram of Fig. **??**). Quite differently, SBPS considers the object's arms as independent conditionally to the central part, and thus it propagates, corrects and swaps simultaneously the P_j's parts, the jth joints of all the arms, for all $j = 2, \ldots, K$ (see Fig. **??**). PS and SBPS are 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 here are a mean over 60 runs performed on a MacBook Pro with a 2.66 GHz Intel Core i7.

6.3. Qualitative tracking results

This subsection is dedicated to visual tracking evaluation. Figure **??** shows some trackings resulting from PS and SBPS on two different video sequences: the estimations made by both approaches are drawn as yellow mesh polygons on top of the "real" colored objects. These estimations correspond to the average (weighted sum) of all the particles. For space reasons, only frames 50, 150 and 250 are displayed and, moreover, zooms focusing on the objects are made to better distinguish the discrepancy between the estimated and the real objects. As can be seen, SBPS is more robust for tracking these objects: visually, its estimation seems to be better. In Figure **??**, the five best particles (i.e. those with the highest weights) are drawn instead of the average mesh polygons. Here again, SBPS concentrates more the best particles around the object (i.e., around the modes of the distribution to estimate) than PS. All these qualitative results highlight the fact that, by embedding a swapping step, SBPS, seems more robust than PS. In the next subsections, quantitative results will confirm these first observations.

6.4. Quantitative tracking results

In this subsection, specific synthetic sequences have been generated in which the object is more strongly deforming and moving during time intervals $[50-150]$ and $[200-250]$. Figure **??** shows the tracking errors over time for two objects described in state spaces whose dimensions are 27 and 33 respectively. Recall

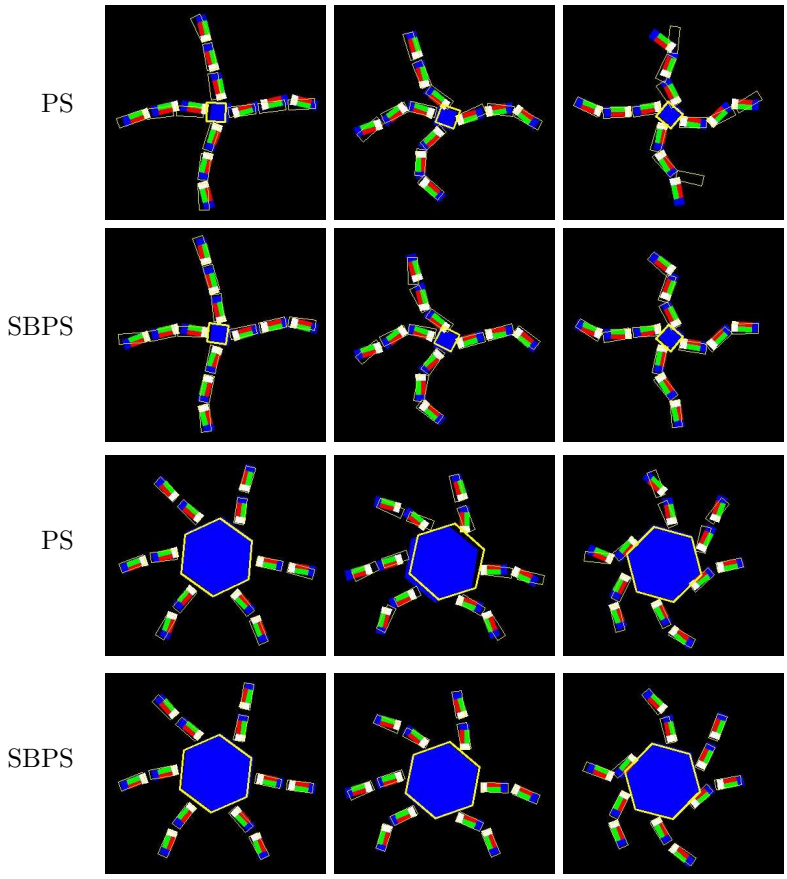


Figure 10: Qualitative tracking results on two sequences on frames 50, 150 and 250: the estimated object (particles' average) is superposed in yellow. First two rows: object with $K = 4$, $P_j = 4$ ($|\mathcal{X}| = 39$), tracked with $N = 50$ particles. Last two rows: object with $K = 3$, $P_j = 6$ ($|\mathcal{X}| = 39$), tracked with $N = 100$ particles.

that the errors correspond to the sum of the Euclidean distances between the polygon's corners of the real object and those of the estimated object. As can be seen, our filter is less affected than PS by strong motions of the object. This demonstrates the robustness of the proposed approach.

One of the main features of our approach is its ability to simultaneous deal with independent parts, making it robust for tracking in high-dimensional spaces. To test this feature, we have performed experiments varying the two factors that, when combined, produce these space high-dimensions: the number of parts that can be simultaneous treated (given by parameter $|P_j|$), and the length of the object's arms (given by parameter K). Tables **??** and **??** show comparative estimation errors resulting from PS and SBPS. In the first table,

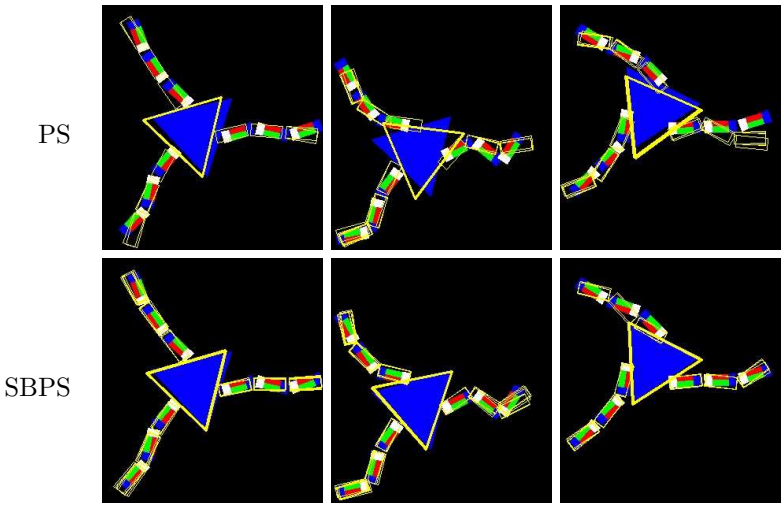


Figure 11: Qualitative tracking results on a sequence containing one object with $K = 4$, $P_j = 3$ ($|\mathcal{X}| = 30$). The five best particles (i.e. those with highest weights) are superposed on the frame 50, 150 and 250 (zooms).

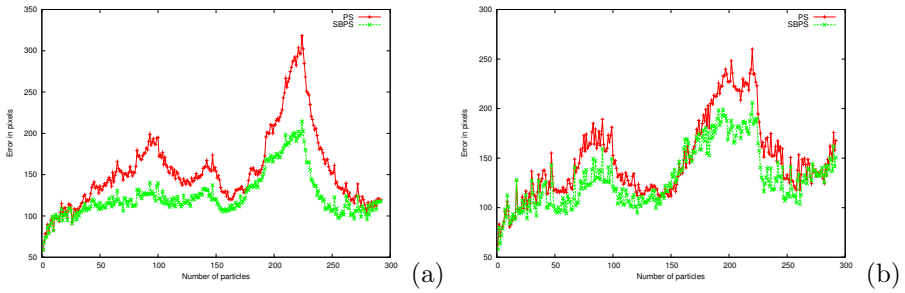


Figure 12: Estimation errors for PS and SBPS: interval times $[50 - 150]$ and $[200 - 250]$ are subject to strong motions and deformations of the object. (a) Object with $K = 5$, $|P_j| = 2$ ($P = 9$, $|\mathcal{X}| = 27$), $N = 80$. (b) Object with $K = 3$, $|P_j| = 5$ ($P = 11$, $|\mathcal{X}| = 33$), $N = 100$.

$|P_j|$ is fixed to 4 (i.e., objects have 4 arms), and K varies from 2 to 8. In the second table, K is fixed to 3 and P_j varies from 2 to 10. Experiments were performed with various numbers of particles ($N = 50$, $N = 300$ and $N = 600$). In all experiments, SBPS always outperforms PS. In addition, the difference of estimation errors between both approaches increases with K and $|P_j|$: this shows the interest of the simultaneous independent treatments. As we can see, the proposed approach is also more stable in terms of standard deviation (displayed inside parentheses in Tables **??** and **??**), especially for high-dimensional state spaces (i.e. high values for K or $|P_j|$). For example, for $K = 7$ and $|P_j|$=4, standard deviation with $N = 50$ particles reaches 65 with Partitioned

Sampling, and 18 with our approach. We can then state our approach produces more stable of results, despite its stochastic estimation, over a lot of runs.

We will show in the next section that SBPS also outperforms PS in terms of computation times, so that the gain in accuracy is not made at the expense of response times.

Table 1: Estimation errors in pixels (standard deviations are displayed inside parentheses), for tracking an object with $|P_j| = 4$, depending on K.

	$K =$	3	4	5	6	7
PS	$N = 50$	185(9)	309(14)	441(20)	735(42)	1245(65)
	$N = 300$	122(2)	220(3)	316(6)	511(10)	850(23)
	$N = 600$	116(1)	209(4)	292(7)	471(9)	767(15)
SBPS	$N = 50$	141(2)	251(7)	398(6)	508(10)	945(18)
	$N = 300$	109(1)	198(2)	288(6)	415(9)	752(14)
	$N = 600$	105(1)	194(3)	263(3)	405(9)	612(11)

Table 2: Estimation errors in pixels (standard deviations are displayed inside parentheses), for tracking an object with $K = 3$, depending on $|P_j|$.

	$\|P_j\| =$	3	5	7	9
PS	$N = 50$	150(10)	202(12)	326(14)	485(19)
	$N = 300$	91(3)	126(6)	262(9)	343(12)
	$N = 600$	80(2)	120(3)	246(7)	305(10)
SBPS	$N = 50$	111(2)	151(5)	234(6)	269(7)
	$N = 300$	71(2)	116(3)	205(2)	253(5)
	$N = 600$	66(1)	114(1)	197(2)	200(3)

Finally, we studied the convergence of the two approaches in terms of the evolution of the tracking errors w.r.t. the number of particles. The results are reported in Figure **??** for two different tracked objects. For both of them, we can observe that SBPS converges faster than PS. For the first object (left graph, $|\mathcal{X}| = 27$), convergence is reached by SBPS with only $N = 40$ particles (tracking error of 156 pixels, total computation time of 4 seconds) whereas PS needs $N = 70$ particles to converge (tracking error of 164 pixels, total computation time of 8 seconds). Thus, SBPS converges 50% faster than PS. For the second object (right graph, $|\mathcal{X}| = 48$), SBPS converges with $N = 120$ particles (tracking error of 124 pixels, total computation time of 21 seconds) whereas PS requires $N = 200$ particles to converge (tracking error of 122 pixels, total computation time of 33 seconds). Here, SBPS converges 36% faster than PS. Overall, SBPS always converges faster than PS.

To conclude this subsection, we have shown that our approach, consisting in swapping independent subparts, decreases significantly the estimation errors compared to PS. In the next subsection, we will show that this is not achieved at the expense computation times.

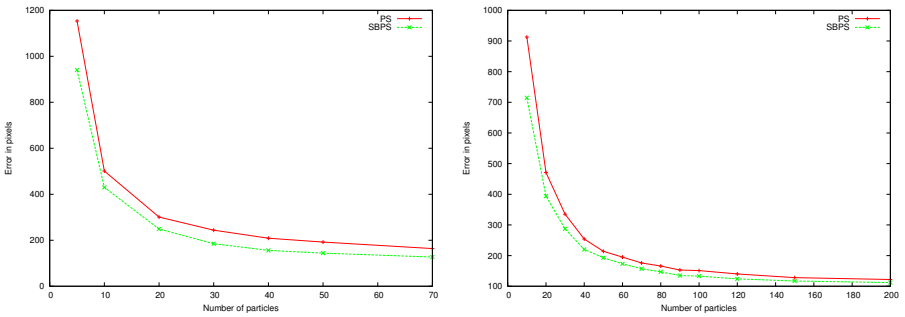


Figure 13: Convergence study for PS ans SBPS. From left to right, an object with $K = 3$, $|P_j| = 4$ ($P = 9$, $|\mathcal{X}| = 27$), and an object with $K = 4$, $|P_j| = 5$ ($P = 16$, $|\mathcal{X}| = 48$).

6.5. Computation times

For measuring the global computation times, essentially three steps are important to take into account: propagations, corrections and resamplings. Propagations and corrections (likelihood computations) have the same computation times for both approaches. Thus, only resampling and swapping (for SBPS) computation times are studied in this subsection. In Section **??**, we argued that, by simultaneously processing independent parts of the objects, our approach reduces the number of resamplings. However, it also introduces a new *swapping* step that may potentially increase the global computation times. To show that this is not the case, as in the previous subsection, we studied the behaviors of SBPS and PS in terms of total computation times, resampling computation times, and swapping computation times, depending on K, N and $|P_j|$.

Fig. **??** reports results depending on K, i.e., the length of the $|P_j| = 4$ arms of an object tracked with $N = 600$ particles. Total computation times of SBPS are always lower than those of PS. Moreover, this difference increases with K. If we only focus on resampling and swapping times, we can again see that the difference between PS and SBPS increases with K, while swapping times seems to be a linear function of K. We can conclude that adding the swapping step does not increase the total computation times.

Figure **??** shows the error curves for a similar test except that the varying feature is $|P_j|$, number of arms of the tracked object instead of K. For both curves, K is fixed to $K = 3$ and $N = 600$ particles are used. Here again, SBPS's total computation time increases more slowly than that of PS. For PS, resampling's computation times increase exponentially with $|P_j|$ while they only increase linearly for SBPS. Note that swapping times also tend to increase linearly. Overall, although, compared to PS, SBPS has an additional swapping step, SBPS clearly outperforms PS in terms of resampling's and swapping's computation times. This is due to the fact that, by processing many different parts of the object simultaneously, SBPS reduces significantly the number of re-

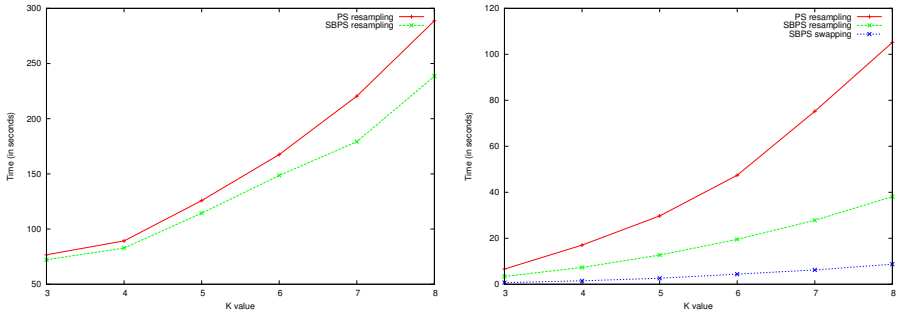


Figure 14: Computation times for PS and SBPS (in seconds) for tracking an object with $|P_j| = 4$ arms, depending on K ($N = 600$). From left to right, total computation times and resampling and swapping computations times.

samplings performed. As swapping is a fast operation, the latter is not sufficient to make SBPS's computations longer than those of PS.

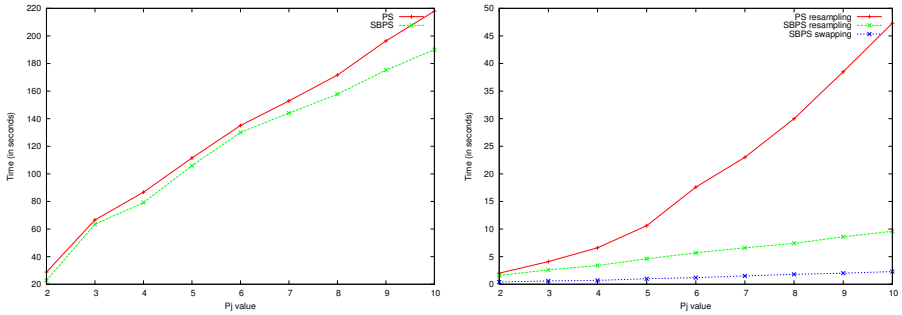


Figure 15: Computation times for PS and SBPS (in seconds) for tracking an object with $K = 3$ depending on $|P_j|$ ($N = 600$). From left to right, total computation times and resampling's and swapping's computation times.

Finally, Fig. **??** highlights how the number of particles used to track an object affects the computation times. In this experiment, the object has $|P_j| = 4$ arms of length 7 ($K = 8$), hence the state space dimension is $|\mathcal{X}| = 87$. Remark that swapping times are linearly related to N, and that the differences of resampling times between PS and SBPS increase with N, as well as, of course, total computation times.

6.6. Extension to multiple articulated object tracking

In this section we address the multiple object tracking problem. There are two general ways to deal with such a task: either one filter can be used for all the tracked objects but, then, this filter has to deal with very high-dimensional state spaces, e.g., for M objects, the number of dimensions of the state space

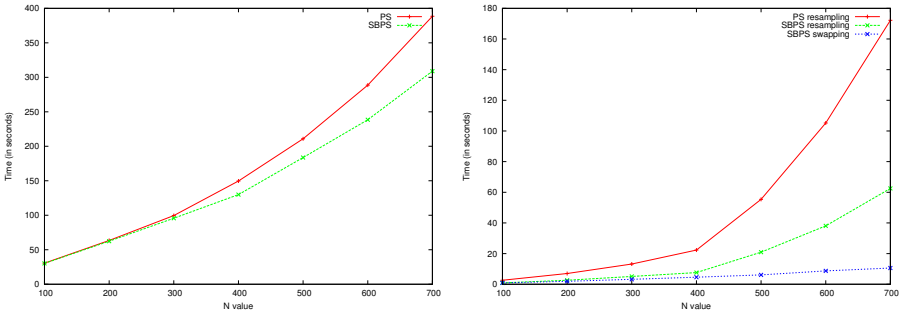


Figure 16: Computation times for PS and SBPS (in seconds) for tracking an object with $K = 8$, $|P_j| = 4$ depending on N. From left to right, total computation times and resampling's and swapping's computation times.

is multiplied by M; or one filter can be used per object, and each such filter just works in the reduced state space corresponding to the object it tracks. The goal of this subsection is to show that using one SBPS filter for all the objects is at least as efficient as using one PS filter per object: this will demonstrate the capacity of our approach to deal with high-dimensional subspaces by taking into account all the independences in the tracking problem.

Table **??** provides comparative results concerning the estimation errors and the computation times for both PS and SBPS. Here, each tracked object is defined by $|P_j| = 4$ and $K = 3$, and all objects are moving and deforming independently over time. We tested cases with $M = 2$ or $M = 3$ objects. When only one filter is used to estimate M objects, this one always uses $N = 100 \times M$ particles. When M filters are used (one per object), two configurations are tested: the case where each filter uses $N = 100 \times M$ particles and that where $N = 100$ particles are used per filter. As each such filter tracks only one object, it should actually need fewer particles for an accurate tracking. First, note that the lowest estimation errors result from SBPS (either one filter or M filters). Remark that using M PS filters is more interesting than using one for all the objects. This holds for all the numbers N of particles tested and it follows from the fact that the "lower" the dimension of the state space, the more robust PS is known to be. Conversely, for SBPS, the estimation errors are lower when using only one filter instead of M. This follows from the fact that increasing state space dimensions also increase the efficiency of swappings (because the products of the best weights $w^{(i),k}$ also tend to increase).

As for the response times, for a fixed number of particles, PS's resampling times are divided by M when dealing with one filter per object. This follows from the fact that, although the number of resamplings is identical whether one or M filters are used, the dimension of the state space for the single filter case is M times that of the multiple filter case. Conversely, SBPS's resampling times are equivalent when using one or M filters because the M filters perform M times the number of resamplings of the single filter but the latter are made in

Table 3: Estimation errors (e, in pixels) and computation times (in seconds, t total computation time, r resampling time and s swapping time), for M objects with $K = 3$ and $|P_j| = 4$ (N is the number of particles used per filter).

		1 PS	M PS		1 SBPS	M SBPS	
	$N =$	$M \times 100$	$M \times 100$	100	$M \times 100$	$M \times 100$	100
$M = 2$	e	356	281	311	211	236	260
	t	36.6	34.9	16.4	36.3	34.9	16.3
	r	3.0	1.7	0.6	0.9	1.0	0.3
	s	-	-	-	1.1	0.5	0.2
$M = 3$	e	611	312	370	242	253	308
	t	86.9	78.6	37.4	82.1	79.8	37.8
	r	12.3	4.5	1.6	2.2	2.5	0.8
	s	-	-	-	3.0	1.3	0.5

a space M times larger than those used by the M filters. For the same reason, swapping times are equivalent for one and M filters. Finally, when N decreases, all the computation times decrease, but the estimation errors increase, so that this induces a trade-off between response time and accuracy.

Overall, our approach produces more accurate results than PS and is also faster. In addition, from the accuracy point of view, one single SBPS filter is better than one SBPS filter per object. However, the latter requires much fewer particles and can thus be significantly faster. This can prove to be particularly useful when dealing with large-scale state spaces.

7. Conclusion

We have presented a new approach for sequential estimation of densities that has two main advantages. First, it exploits the probabilistic independences encoded into DBNs to apply particle filter computations on smaller subspaces. Second, it swaps 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. The combination of this swapping step with the parallel processing of conditionally independent parts significantly reduces the number of required resampling steps, inducing overall computation times that are often smaller than PS. Moreover, we have shown that this gain of computation time increases with the state space dimension (number of parts of articulated objects, number of objects), as well as with the number of particles.

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

8. Appendix: proofs

Proof Proposition ??: For any $j = 1, \ldots, K-1$, let $Q_j = \cup_{h=1}^{j} P_h$ and $R_j = \cup_{h=j+1}^{K} P_h$, i.e., Q_j and R_j represent the set of parts processed up to the processing of parts P_j and those still to be processed respectively. Let $Q_0 = R_K = \emptyset$. By abuse of notation, for any set $H \subseteq \{1, \ldots, P\}$, let $\mathbf{x}_t^H$ denote the set of nodes $\{\mathbf{x}_t^k : k \in H\}$.

First, let us prove that, for every $j \in \{1, \ldots, K\}$ and every $k \in P_j$, we have:

$$\mathbf{x}_t^k \perp\!\!\!\perp \bigcup_{h \in P_j \setminus \{k\}} \{\mathbf{x}_t^h\} \cup \mathbf{x}_t^{Q_{j-1}} \cup \mathbf{x}_{t-1}^{R_{j-1}} \cup \mathbf{y}_{1:t-1} \cup \mathbf{y}_t^{Q_{j-1}} \mid \mathbf{pa}(\mathbf{x}_t^k). \tag{6}$$

If there existed in the OTDBN an active chain $\{\mathbf{c}_1 = \mathbf{x}_t^k, \ldots, \mathbf{c}_n\}$ between $\mathbf{x}_t^k$ and one of the nodes in the independent part of Eq. (**??**), its first arc would necessarily be $\mathbf{c}_1 \rightarrow \mathbf{c}_2$ since $\mathbf{c}_1$'s parents are the conditioning set. Let $\mathbf{c}_V$ denote the last node such that, for all $2 \leq h \leq V$, the arcs of the chain are $\mathbf{c}_{h-1} \rightarrow \mathbf{c}_h$. By definition of sets P_j's, none of the nodes $\mathbf{x}_t^h \in \mathbf{x}_t^{P_j} \setminus \{\mathbf{x}_t^k\}$ is a descendant of $\mathbf{x}_t^k$. In addition, by definition of OTDBNs, nodes in time slice $t-1$ cannot be some descendant of $\mathbf{x}_t^k$. Consequently, $\mathbf{c}_V \neq \mathbf{c}_n$ and, thus, the active chain contains arcs $\mathbf{c}_{V-1} \rightarrow \mathbf{c}_V \leftarrow \mathbf{c}_{V+1}$. As $\mathbf{c}_V$ is a descendant of $\mathbf{x}_t^k$, neither it nor its descendants are in the conditioning set and, by d-separation, the chain cannot be active and (**??**) holds.

Assume now that, before processing parts P_j, the particle set represents probability distribution $p(\mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}})$. Note that this is the case just before processing part P_1 since this distribution is equal to $p(\mathbf{x}_{t-1} | \mathbf{y}_{1:t-1})$. Let us show that, after the parallel propagations "$*f_t^{i_{P_j}^k}$", the particle set represents probability distribution $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}})$. Operation "$*f_t^{i_{P_j}^k}$" corresponds to multiplying the distribution represented by the particle set by $p(\mathbf{x}_t^{i_{P_j}^k} | \mathbf{pa}(\mathbf{x}_t^{i_{P_j}^k}))$ and integrating out variable $\mathbf{x}_{t-1}^{i_{P_j}^k}$. Overall, the parallel propagations in P_j correspond to computing:

$$\int p(\mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \prod_{k \in P_j} p(\mathbf{x}_t^k | \mathbf{pa}(\mathbf{x}_t^k)) \, d\mathbf{x}_{t-1}^{P_j} \tag{7}$$

because (**??**) implies that no operation "$*f_t^k$" depends on $\mathbf{x}_t^{P_j \setminus \{k\}}$. Now, since for all $k \in P_j$, $\mathbf{pa}(\mathbf{x}_t^k) \subseteq \mathbf{x}_t^{Q_{j-1}}$, Eq. (**??**) implies that:

$$p(\mathbf{x}_t^k | \mathbf{pa}(\mathbf{x}_t^k)) = p(\mathbf{x}_t^k | \bigcup_{h \in P_j \setminus \{k\}} \{\mathbf{x}_t^h\}, \mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}).$$

By the well-known chain rule formula, the above equation thus implies that:

$$\begin{aligned} \prod_{k \in P_j} p(\mathbf{x}_t^k | \mathbf{pa}(\mathbf{x}_t^k)) &= p(\mathbf{x}_t^{i_{P_j}^1}, \ldots, \mathbf{x}_t^{i_{P_j}^{k_j}} | \mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \\ &= p(\mathbf{x}_t^{P_j} | \mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}). \end{aligned}$$

Therefore, Eq. (**??**) is equivalent to:

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

As $Q_j = Q_{j-1} \cup P_j$ and $R_{j-1} = P_j \cup R_j$, the above equation is equivalent to $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}})$.

Now, let us show that after the parallel corrections "$\times p_t^{i_{P_j}^k}$", the particle set estimates distribution $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. These operations correspond to performing, up to a constant:

$$p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \times \prod_{k \in P_j} p(\mathbf{y}_t^k | \mathbf{x}_t^k). \tag{8}$$

By definition of OTDBNs, nodes $\mathbf{y}_t^k$ have no children and only one parent: $\mathbf{x}_t^k$. Hence, by d-separation, they are independent of the rest of the network conditionally to this parent. Therefore, as in the preceding paragraph, we can easily prove that:

$$\prod_{k \in P_j} p(\mathbf{y}_t^k | \mathbf{x}_t^k) = p(\mathbf{y}_t^{P_j} | \mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}).$$

Therefore, Eq. (**??**) is equivalent to $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_t^{P_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}})$, which, when normalized, is equal to $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}, \mathbf{y}_t^{P_j}) = p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. The last step of the processing of P_j is a resampling that does not alter this distribution. Finally, note that this probability is precisely that assumed at the beginning of the proof. Hence, after processing all the P_j's, the particle set estimates $p(\mathbf{x}_t^{Q_K}, \mathbf{x}_{t-1}^{R_K} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_K}) = p(\mathbf{x}_t | \mathbf{y}_{1:t})$. □

Proof Proposition ??: Let $k \in P_j$ and let σ_k be an admissible permutation for $\mathbf{x}_t^k$ as described above. To prove the proposition, it is sufficient to show that, after applying σ_k on all the nodes in the swapping set of $\mathbf{x}_t^k$, the particle set still estimates $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. Let us partition $(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j})$ into the following sets:

- $L = \text{Link}_t^{Q_j}(\mathbf{x}_t^k) \cup \text{Link}_{t-1}^{R_j}(\mathbf{x}_t^k)$, i.e., the set of nodes to permute;
- $A_t^{Q_j} \cup A_{t-1}^{R_j}$ as described in Definition **??**, i.e., the set of nodes that separate L from the rest of the OTDBN in time slice $t-1:t$;
- $B_t^1 = \mathbf{x}_t^{Q_j} \backslash (\text{Link}_t^{Q_j}(\mathbf{x}_t^k) \cup A_t^{Q_j})$ and $B_{t-1}^2 = \mathbf{x}_{t-1}^{R_j} \backslash (\text{Link}_{t-1}^{R_j}(\mathbf{x}_t^k) \cup A_{t-1}^{R_j})$.

Duplicate these nodes over all times slices:

$$\begin{aligned}
\text{Link} &= \cup_{s=1}^t \{\mathbf{x}_s^h : \mathbf{x}_t^h \in \text{Link}_t^{Q_j}(\mathbf{x}_t^k)\} \cup_{s=1}^{t-1} \{\mathbf{x}_s^h : \mathbf{x}_{t-1}^h \in \text{Link}_{t-1}^{R_j}(\mathbf{x}_t^k)\}, \\
A &= \cup_{s=1}^t \{\mathbf{x}_s^h : \mathbf{x}_t^h \in A_t^{Q_j}\} \cup_{s=1}^{t-1} \{\mathbf{x}_s^h : \mathbf{x}_{t-1}^h \in A_{t-1}^{R_j}\}, \\
B &= \cup_{s=1}^t \{\mathbf{x}_s^h : \mathbf{x}_t^h \in B_t^1\} \cup_{s=1}^{t-1} \{\mathbf{x}_s^h : \mathbf{x}_{t-1}^h \in B_{t-1}^2\}.
\end{aligned}$$

Finally, consider the observations on these sets of nodes: let $\mathbf{y}^A = \{\mathbf{y}_s^h : \mathbf{x}_s^h \in A\}$, let $\mathbf{y}^B = \{\mathbf{y}_s^h : \mathbf{x}_s^h \in B\}$ and let $\mathbf{y}^{\text{Link}} = \{\mathbf{y}_s^h : \mathbf{x}_s^h \in \text{Link}\}$. Then:

$$
\begin{aligned}
p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j}) &\propto p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j}) \\
&= p(L, A_t^{Q_j}, A_{t-1}^{R_j}, B_t^1, B_{t-1}^2, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j}) \\
&= \int p(L, A, B_t^1, B_{t-1}^2, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})\ d(A \backslash (A_t^{Q_j} \cup A_{t-1}^{R_j})) \\
&= \int p(L, A, B_t^1, B_{t-1}^2, \mathbf{y}^{\text{Link}}, \mathbf{y}^A, \mathbf{y}^B)\ d(A \backslash (A_t^{Q_j} \cup A_{t-1}^{R_j})) \\
&= \int p(A, \mathbf{y}^A) p(L, \mathbf{y}^{\text{Link}} | A) p(B_t^1, B_{t-1}^2, \mathbf{y}^B | L, y^{\text{Link}}, A) d(A \backslash (A_t^{Q_j} \cup A_{t-1}^{R_j})) \quad (9)
\end{aligned}
$$

because, in OTDBNs, conditioning by A is equivalent to conditioning by $A, \mathbf{y}^A$ due to conditions 3 and 4 of Definition **??**.

Let us now prove that $(B_t^1 \cup B_{t-1}^2 \cup \mathbf{y}^B) \perp\!\!\!\perp L \cup y^{\text{Link}} | A$. Again by conditions 3 and 4 of Definition **??**, it is sufficient to show that there exists no active chain between one node of Link and one node of B. Suppose that there exists such an active chain $\{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$ with $\mathbf{c}_1 \in \text{Link}$ and $\mathbf{c}_n \in B$. First, note that the chain cannot pass through any node in time slice $s > t$ because, since $\mathbf{c}_1$ and $\mathbf{c}_n$ belong to time slices $\leq t$ and due to condition 1 of Definition **??**, there would exist a node $\mathbf{c}_h = \mathbf{x}_s^w$ such that $\mathbf{c}_{h-1} \rightarrow \mathbf{c}_h \leftarrow \mathbf{c}_{h+1}$ which would block the chain since neither $\mathbf{x}_s^w$ nor its descendants are in the conditioning set A. For the same reason, it is impossible that the chain passes through a node in $\text{Link}_t^{R_j}(\mathbf{x}_t^k) = \text{Link}_t(\mathbf{x}_t^k) \backslash \text{Link}_t^{Q_j}(\mathbf{x}_t^k)$. Now, let $\mathbf{c}_h = \mathbf{x}_s^v$ be the first node in the chain belonging to B. Assume that $\mathbf{c}_{h-1} = \mathbf{x}_r^w \in \text{Link}$. Then, by definition of the arcs of OTDBNs, either $(r = s) \wedge (v \neq w)$ or $(r \neq s) \wedge (v = w)$.

First case: if $r \neq s$, then $\mathbf{c}_h = \mathbf{x}_s^w$ is a duplicate of $\mathbf{c}_{h-1}$ in another time slice and, by definition of Link, either $\mathbf{c}_h \in \text{Link}$, which leads to a contradiction since $\mathbf{c}_h \in B$ by hypothesis, or $\mathbf{c}_h = \mathbf{x}_t^w \in \text{Link}_t^{R_j}(\mathbf{x}_t^k)$ which is also impossible by the preceding paragraph. Second case: if $r = s$, then, by definition of Link, $\mathbf{x}_t^w \in \text{Link}_t(\mathbf{x}_t^k)$ and, by definition of OTDBNs, the arc between $\mathbf{x}_t^w$ and $\mathbf{x}_t^v$ exists in the OTDBN. Hence, $\mathbf{x}_t^v \in \text{Link}_t(\mathbf{x}_t^k)$ and either $\mathbf{c}_h \in \text{Link}$ or $\mathbf{c}_h = \mathbf{x}_t^v \in \text{Link}_t^{R_j}(\mathbf{x}_t^k)$ and, similarly to the first case, both conditions imply a contradiction. So $\mathbf{c}_{h-1} \notin \text{Link}$ and $\mathbf{c}_{h-1} \notin B$. Therefore, either $\mathbf{c}_{h-1} \in \text{Link}_t^{R_j}(\mathbf{x}_t^k)$ or $\mathbf{c}_{h-1} \in A$. We saw above that the first case leads to a contradiction, so $\mathbf{c}_{h-1} \in A$.

So, for the chain to be active, since we condition on the nodes in A, necessarily the chain has the following arcs: $\mathbf{c}_{h-2} \rightarrow \mathbf{c}_{h-1} \leftarrow \mathbf{c}_h$. Let $\mathbf{x}_z^u = \mathbf{c}_{h-2}$. Again, by definition of OTDBNs, either $(r = z) \wedge (u \neq w)$ or $(r \neq z) \wedge (u = w)$. In the second case, $\mathbf{c}_{h-2}$ is a duplicate of $\mathbf{c}_{h-1}$ in another time slice, hence $\mathbf{c}_{h-2} \in A$ and, by d-separation, the arc $\mathbf{c}_{h-2} \rightarrow \mathbf{c}_{h-1}$ blocks the chain. In the first case, by definition of A, $\mathbf{c}_{h-1} = \mathbf{x}_r^w \in \mathbf{an}_r(\mathbf{x}_r^k)$ and, since $\mathbf{c}_{h-2}$ is a parent of $\mathbf{c}_{h-1}$, $\mathbf{c}_{h-2} \in \mathbf{an}_r(\mathbf{x}_r^k)$. This implies that $\mathbf{c}_{h-2} \notin \text{Link}$. But, by assumption, $\mathbf{c}_h$ is the first node in the chain belonging to B. Hence, necessarily, $\mathbf{c}_{h-2} \in A$ and arc

$\mathbf{c}_{h-2} \rightarrow \mathbf{c}_{h-1}$ blocks the chain. Overall, $\{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$ cannot be an active chain. Hence $(B_t^1 \cup B_{t-1}^2 \cup \mathbf{y}^B) \perp\!\!\!\perp L \cup y^{\text{Link}} | A$.

Now, exploiting this independence, Eq. (**??**) becomes:

$$p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j}) = \int p(A, \mathbf{y}^A) p(L, \mathbf{y}^{\text{Link}} | A) p(B_t^1, B_{t-1}^2, \mathbf{y}^B | A) d(A \backslash (A_t^{Q_j} \cup A_{t-1}^{R_j})).$$

Particles' permutations over subparts L for fixed values of A do not affect distribution $p(L, \mathbf{y}^{\text{Link}}|A)$ since estimations by samples are insensitive to the order of the elements in the samples. In addition, the estimations of $p(A, \mathbf{y}^A)$ and $p(B_t^1, B_{t-1}^2, \mathbf{y}^B|A)$ are neither affected since conditionally to A, B is d-separated from L. Consequently, applying permutation σ_k over L does not alter the estimation of $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. □

Proof Proposition ??: First, consider 2 particles $a, b \in 1, \ldots, N$, whose weights, say $t^{(a)}$ and $t^{(b)}$, are such that $\prod_{k \in P_j} t^{(a),k} \geq \prod_{k \in P_j} t^{(b),k}$ and such that there exists $h \in P_j$ such that $t^{(b),h} > t^{(a),h}$. Let $\rho = t^{(b),h} - t^{(a),h}$ and denote:

$$\begin{array}{ll} \alpha = \prod_{k \in P_j, k \neq h} t^{(a),k} \times t^{(a),h} & \beta = \prod_{k \in P_j, k \neq h} t^{(a),k} \times \rho, \\ \gamma = \prod_{k \in P_j, k \neq h} t^{(b),k} \times t^{(a),h} & \delta = \prod_{k \in P_j, k \neq h} t^{(b),k} \times \rho. \end{array}$$

Then, let us show that $f(\alpha+\beta)+f(\gamma) > f(\alpha)+f(\gamma+\delta)$ or, in other words, that swapping the hth weight between particles (a) and (b) strictly increases the sum over these particles of f(weight). By hypothesis, $\prod_{k \in P_j} t^{(a),k} \geq \prod_{k \in P_j} t^{(b),k}$ and $t^{(b),h} > t^{(a),h}$, which implies that $\prod_{k \neq h} t^{(a),k} > \prod_{k \neq h} t^{(b),k}$. Hence $\beta > \delta$. As f is strictly increasing, $f(\alpha + \beta) > f(\alpha + \delta)$. Therefore, $f(\alpha + \beta) + f(\gamma) > f(\alpha + \delta) + f(\gamma)$. Now, let us show that $f(\alpha + \delta) + f(\gamma) \geq f(\alpha) + f(\gamma + \delta)$. This is equivalent to showing that $f(\alpha + \delta) - f(\alpha) \geq f(\gamma + \delta) - f(\gamma)$. It is well known that, for any convex function g, and any $x_1 < x_2 < x_3$, we have: $\frac{g(x_3)-g(x_2)}{x_3-x_2} \geq \frac{g(x_2)-g(x_1)}{x_2-x_1}$. Now, $\gamma < \gamma + \delta \leq \alpha < \alpha + \delta$ because all these quantities are strictly positive and $\gamma + \delta$ and α are the weights of particles (b) and (a) respectively. Hence, if $\gamma + \delta = \alpha$, then

$$\frac{f(\alpha + \delta) - f(\alpha)}{\alpha + \delta - \alpha} \geq \frac{f(\alpha) - f(\gamma)}{\alpha - \gamma} = \frac{f(\gamma + \delta) - f(\gamma)}{\gamma + \delta - \gamma},$$

else (i.e., when $\gamma + \delta < \alpha$):

$$\frac{f(\alpha + \delta) - f(\alpha)}{\alpha + \delta - \alpha} \geq \frac{f(\alpha) - f(\gamma + \delta)}{\alpha - \gamma - \delta} \geq \frac{f(\gamma + \delta) - f(\gamma)}{\gamma + \delta - \gamma}.$$

Overall $f(\alpha+\delta)-f(\alpha) \geq f(\gamma+\delta)-f(\gamma)$ and, by transitivity, $f(\alpha+\beta)+f(\gamma) > f(\alpha)+f(\gamma+\delta)$.

Let σ be a permutation maximizing $\sum_{i=1}^{N} f(\prod_{k \in P_j} w^{s_k(i),k})$ over all permutations $(s_1, \ldots, s_{|P_j|}) \in \mathcal{S}$. Denote by $v^{(i),k}$ the weights $w^{\sigma_k(i),k}$ resulting from

the application of σ on the original particle weights $\{w^{(i),k}\}$. Without loss of generality, assume that σ sorts the "total" particle weights in decreasing order, i.e., that $\prod_{k\in P_j} v^{(a),k} \geq \prod_{k\in P_j} v^{(b),k} \iff a \leq b$. By the preceding paragraph, there cannot exist $h \in P_j$ such that there exists $i \geq 2$ such that $v^{(i),h} > v^{(1),h}$, else by permuting $v^{(i),h}$ and $v^{(1),h}$, we would get a permutation strictly increasing the sum of f(weights). For the same reason, there cannot exist $h \in P_j$ such that there exists $i \geq 3$ such that $v^{(i),h} > v^{(2),h}$. By induction, we have that, for any r, there cannot exist $h \in P_j$ such that there exists $i > r$ such that $v^{(i),h} > v^{(r),h}$. This precisely correspond to the permutation operation $\rightleftharpoons^{P_j}$. Therefore, $\rightleftharpoons^{P_j}$ is the unique permutation satisfying Equation (**??**). □