Noname manuscript No.
(will be inserted by the editor)

Combinatorial Resampling Particle Filter: an Effective and Efficient Method for Articulated Object Tracking

Christophe Gonzales · Séverine Dubuisson

Received: date / Accepted: date

Abstract Particle Filter (PF) is a method dedicated to posterior density estimations using weighted samples whose elements are called *particles*. In particular, this approach can be applied to object tracking in video sequences in complex situations and, in this paper, we focus on articulated object tracking, i.e., objects that can be decomposed as a set of subparts. One of PF's crucial step is a resampling step in which particles are resampled to avoid degeneracy problems. In this paper, we propose to exploit mathematical properties of articulated objects to swap conditionally independent subparts of the particles in order to generate new particle sets. We then introduce a new resampling method called *Combinatorial Resampling* that resamples over the particle set resulting from all the "admissible" swappings, the so-called *combinatorial set*. In essence, Combinatorial Resampling (CR) is quite similar to the combination of a crossover operator and a usual resampling, but there exists a fundamental difference between CR and the use of crossover operators: we prove that CR is sound, i.e., in a Bayesian framework, it is guaranteed to represent *without any bias* the posterior densities of the states over time. By construction, the particle sets produced by CR better represent the density to estimate over the whole state space than the original set and, therefore, Combinatorial Resampling produces higher quality samples. Unfortunately, the combinatorial set is generally of an exponential size and, therefore, to be scalable, we show how it can be implicitly constructed and resampled from, thus resulting in both an efficient and effective resampling scheme. Finally, through experimentations both on challenging synthetic and real video sequences, we also show that our resampling method outperforms all classical resampling methods both in terms of the quality of its results and in terms of computation times.

Keywords Particle filter · resampling · dynamic Bayesian networks · articulated object tracking

C. Gonzales
Sorbonne Universités, UPMC Univ Paris 06, CNRS, UMR 7606, LIP6, F-75005, Paris, France
E-mail: christophe.gonzales@lip6.fr

S. Dubuisson
Sorbonne Universités, UPMC Univ Paris 06, CNRS, UMR 7222, ISIR, F-75005, Paris, France
E-mail: severine.dubuisson@isir.upmc.fr

1 Introduction

In computer vision, the increase in the quality of data (e.g., the resolution of video sequences) as well as their diversity and the multiplicity of their sources (multiple cameras, multiple modalities, *etc.*), stimulated people to increase as well the quality of their object models, leading to more precise models but also to higher-dimensional state spaces. As a consequence, solving problems defined in high-dimensional observation and/or state spaces has become of crucial importance. In this article, we consider the problem of sequential estimation of non parametric and multimodal densities evolving with time, by using the recursive Bayesian filtering framework. In particular, our goal is to track articulated structures with accuracy and within a reasonable time. This is considered as a challenging problem due to its high complexity. Actually, the state space of such a problem is inevitably high-dimensional and the estimation of the state of an object thus requires that of many parameters.

When the dynamics of the objects are linear or linearizable and when the uncertainties about their position are Gaussian or mixtures of Gaussians, tracking can be performed analytically by Kalman-like Filters [15]. Unfortunately, in practice, such properties seldom hold and people often resort to sampling to approximate solutions of the tracking problem. The Particle Filter (PF) methodology [28] is popular among these approaches and, in this paper, we focus on

it. PF consists of estimating the density over the states of the tracked object using weighted samples whose elements, called *particles*, are possible realizations of the object state. PF and its variants all use a resampling step to avoid a degeneracy problem, i.e., the case where all but one of the particle's weights are close to zero [22]. In such cases, although the sample maintains a diversity, the particles are too widely distributed, most of them being located on the tails of the density over the object states, hence implying that the majority of the particles are not representative for the estimation of this density. Without a resampling step that essentially substitutes the low-weight particles by particles of higher weights, hence concentrating the particles on the peaks of the density, this problem would necessarily occur [23].

A few resampling algorithms are classically used, e.g., multinomial [28], residual [49], stratified and systematic resamplings [42]. However, these methods have not been designed specifically to deal with high-dimensional spaces, in particular for multiple or articulated object tracking and, as such, they do not exploit the features of such problems. In this paper, we propose to improve the articulated object tracking, both in terms of tracking accuracy and computation time, by improving the particle filter's resampling step and making its role central. For this purpose, we introduce a new resampling algorithm called *Combinatorial Resampling* that exploits dynamic Bayesian networks independence properties to swap conditionally independent subparts of the particles, thereby producing particles nearer to the modes of the density to estimate. In essence, such swapping is similar to a crossover operator but, unlike the latter, it is guaranteed mathematically to unalter the estimation of the density. The swappings considered by Combinatorial Resampling produce new samples of exponential size from which resampling is actually performed. To be scalable, those are only implicitly created and resampled from. To the best of our knowledge, no work has addressed the tracking problem from this point of view with a soundness guarantee.

The paper is organized as follows. Section 2 recalls the basics of PF, its use for articulated object tracking and the partitioned sampling framework. Section 3 gives an overview of the resampling methods. Section 4 is dedicated to our new resampling method, that relies on dynamic Bayesian networks, and to its mathematical correctness. Section 5 provides experimental results both on challenging synthetic and real video sequences. Those highlight the efficiency of our method in terms of both the quality of its results and computation times. Finally, we give concluding remarks and perspectives in Section 6. All proofs are given in an appendix.

2 Particle Filter for Object Tracking

In this paper, object tracking consists of estimating a state sequence $\{\mathbf{x}_t\}_{t=1,\dots,T}$ from observations $\{\mathbf{y}_t\}_{t=1,\dots,T}$. From a probabilistic point of view, this problem amounts to estimate, for any t, the posterior density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ where $\mathbf{y}_{1:t}$ denotes tuple $(\mathbf{y}_1,\dots,\mathbf{y}_t)$. This can be computed iteratively using Eq. (1) and (2), which are referred to as a *prediction step* and a *correction step* respectively.

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

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

where $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$ and $p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1})$ represent the transition and the likelihood functions respectively.

2.1 Particle Filter (PF)

PF [28] approximates Eq. (1) and (2) using weighted samples $\{\mathbf{x}_{t+1}^{(i)}, w_{t+1}^{(i)}\}$, $i = 1,\dots,N$, where each $\mathbf{x}_{t+1}^{(i)}$ is a possible realization of state $\mathbf{x}_{t+1}$ called a *particle*. In its prediction step (Eq. (1)), PF propagates the particle set $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$ using an importance function $q(\mathbf{x}_{t+1}|\mathbf{x}_t^{(i)}, \mathbf{y}_{t+1})$ which may differ from $p(\mathbf{x}_{t+1}|\mathbf{x}_t^{(i)})$ (but, for simplicity, we will assume they do not); in its correction step (Eq. (2)), PF weights the particles using a likelihood function, so that

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

with $\sum_{i=1}^N w_{t+1}^{(i)} = 1$. 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: $p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t+1}) \approx \sum_{i=1}^N w_{t+1}^{(i)} \delta_{\mathbf{x}_{t+1}^{(i)}}(\mathbf{x}_{t+1})$, where $\delta_{\mathbf{x}_{t+1}^{(i)}}$ are Dirac masses centered on particles $\mathbf{x}_{t+1}^{(i)}$.

As shown in [52], 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 for articulated object tracking. Theoretical frameworks have been proposed to solve the problem, using specific likelihood models [69], prior information [14] or optimization-based local search [57, 7].

2.2 The Problem of Articulated Object Tracking with Particle Filter

Articulated objects are defined as objects that can be decomposed as a set of subparts linked by articulation joints and articulated object tracking consists of tracking all the subparts. For instance, for behavior analysis, one may wish to track all the body subparts of the person represented in Fig. 1 which consists of a torso (subpart 1), a left arm (subparts 2 and 3), a right arm (subparts 4 and 5) and the head (subpart 6). As a consequence, state $\mathbf{x}_t$ is the tuple of the states of all

these subparts. Let $\mathbf{x}_t^j$ represent the state of the jth subpart at time t, then $\mathbf{x}_t = (\mathbf{x}_t^1, \ldots, \mathbf{x}_t^6)$. More formally, let $\mathcal{X}$ and $\mathcal{Y}$ represent the state and observation spaces respectively and assume they 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, i.e., the object is decomposed into P different subparts. Then, states $\mathbf{x}_t$ and observations $\mathbf{y}_t$ are tuples $(\mathbf{x}_t^1, \ldots, \mathbf{x}_t^P)$ and $(\mathbf{y}_t^1, \ldots, \mathbf{y}_t^P)$ respectively. By abuse of notation, in the rest of the paper, for any set $J = \{j_1, \ldots, j_k\} \subseteq \{1, \ldots, P\}$, $\mathbf{x}_t^J$ will denote the tuple $(\mathbf{x}_t^{j_1}, \ldots, \mathbf{x}_t^{j_k})$, which corresponds to the states of the subparts in J. Similarly, $\mathbf{x}_t^{(i),J}$ will denote the tuple of the parts in J of the ith particle.

Despite significant advances in the last years, the problem of articulated object tracking still remains unsolved in its full extent. Yet, this open problem has both a strong theoretical and practical interest. Particle filter-based approaches have addressed this problem in different ways. Here, we only consider works that model the object using a skeleton, i.e., using a model where the object is represented as a set of subparts linked by articulation joints. But there also exist model-free methods [46]. One can find a good and recent review of articulated object tracking as well as pose estimation approaches in the chapter 2 of de Campos' PhD thesis [20]. The skeleton-based approaches can be roughly divided into two classes. In the first one, the estimation process is performed in the whole-dimensional state and observation spaces $\mathcal{X}$ and $\mathcal{Y}$ but the algorithms try to restrict the search locally in the neighborhood of some locations, either by adapting the prediction step to the non linear and/or fast motions, or by improving the focusing capacity of the correction step. In the second class, the approaches decompose the state or observation spaces into smaller ones and apply on them propagations and corrections sequentially. As these spaces are smaller, these algorithms achieve better accuracy while being at the same time faster. The tracking algorithm we propose in this paper belongs to this second class.

2.2.1 Working in the whole state and observations spaces

A first way to solve efficiently a specific articulated object tracking problem is to add mathematical constraints to the particles' propagation model. The aim is to narrow the search for the tracking solution. In many papers, this is achieved by introducing into the tracking model the physical constraints of the articulated objects [11,56,70,48]. For example, for human tracking, in [71], the constraints are introduced during the simulation step, while in [10,35] they are included into the importance function. Other approaches include object priors [33,16,34], exploit knowledge about the object's behaviors [17], about how the objects should look [13] and about their interactions with the environment [43]. Recently, an original idea was proposed in [34]: unlike previous approaches, the motion prediction is expressed in terms of spatial positions rather than in terms of joint constraints. Priors are then expressed in spatial coordinates and projected into the space of joint angles using an inverse kinematics model. One can also learn with time and then refine the prediction step. For example, in [30] the pose is learnt and its estimation at each time step is used to propagate particles. In [17], activities are learnt and used to better search in the state space. All these approaches are very effective on specific articulated object tracking since they take advantage of the features of the object or of its motion to adapt its model. Unfortunately, they are dedicated to specific object tracking and require a remodeling to be adapted to other ones.

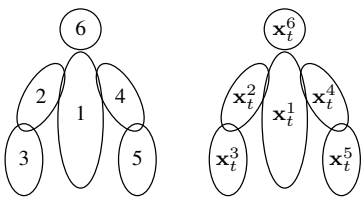


Fig. 1 An articulated object.

The stochastic nature of particle filtering and the combinatorial size of the articulated object state spaces never guarantee the filter to produce particle sets nearby the modes of the densities. Consequently, combining the filter with local search techniques can significantly improve it by better focusing the particles on those modes. This explains why optimization approaches are also popular among the community working on improving the correction step. Many other optimization methods were introduced into PF (e.g., scatter-search [58], stochastic gradient-based descents [36], new stochastic meta-descent approaches [8], leading to an efficient Smart Particle Filter [9]), but the most famous optimization approach that has been introduced into particle filter is simulated annealing, leading to *Annealed Particle Filter* (APF) [21]. APF consists of adding *pseudo* annealing layers to PF's correction step in order to better diffuse particles in the state space. APF is one of the best algorithms for tracking in high dimensional spaces. However, all these methods and their variations [60] 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 guarantee toward the modes of the densities. Note that, even if it is based on a model-free representation (*i.e.* not the articulated representation we consider here), Basin Hopin Monte Carlo [46] sampling seems to be a promising approach that avoids falling into local minima. 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 descrip-

tion of state space. The *Progressive Particle Filter* [12] is an example of such a strategy. Particle Swarm Optimization has also been used in conjunction with PF [39,72,62, 40,44]. 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 [19]). Most of these approaches rely on heuristics, whose convergence and technical correctness cannot be guaranteed. Moreover, they often require "tuning" using some parameters whose role is difficult to understand and their values hard to justify.

The aforementioned methods often suppose that all the needed observations are available at each time slice, which may not necessarily be the case in practice. Moreover, their main drawback is to require strong priors about the object to track, that prevent them to be extensible to general articulated object tracking problems.

The next subsection deals with approaches that decompose the state and observation spaces into a set of subspaces of reasonable sizes where PF can be applied.

2.2.2 Decomposing the state and observation spaces

Among the approaches that exploit the fact that, in many problems, both the system dynamics and the likelihood function are decomposable over small subspaces, Partitioned Sampling (PS) [51] is probably the most popular. The key idea, that will be detailed in Section 2.3, is 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. However, despite recent improvements [65,26,73], 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 [41] in the context of a general PF for Dynamic Bayesian Networks (DBN). Here, the importance functions 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 conditional probability of the node given its parents in the DBN. In [61], the sampling idea of [41] is combined with the resampling scheme proposed in [51] 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 [66,38]. It combines the PF framework with the well-known Loopy Belief Propagation algorithm [59,74] for speeding-up computations (but at the expense of approximations). It has been successfully applied on many problems of high dimensions [67,64,4,37,47]. The *Rao-Blackwellized Particle Filter for DBN* (RBPF) [24] uses a decomposition of state space $\mathcal{X}$ as the Cartesian product of the state spaces of two subparts $\mathcal{X}^1 \times \mathcal{X}^2$ that fulfill the following condition: the conditional posterior distribution of the second subpart given the first one $p(\mathbf{x}^2_{1:t}|\mathbf{y}_{1:t}, \mathbf{x}^1_{1:t})$ can be estimated using classical techniques such as Kalman filter. The joint posterior distribution $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$ can then be estimated by first estimating the marginal posterior distribution of the first subpart $p(\mathbf{x}^1_{1:t}|\mathbf{y}_{1:t})$ using PF and, then, the conditional posterior $p(\mathbf{x}^2_{1:t}|\mathbf{y}_{1:t}, \mathbf{x}^1_{1:t})$ using Kalman filter. As the dimension of the state space of the first subpart is smaller than that of the whole state space, the sampling step of particle filter for the first subpart needs fewer particles and the variance of the estimation can be reduced. Though RBPF is very effective 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 subparts fulfilling the condition. The framework introduced in [5] 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 importance densities. This approach offers a very flexible way of choosing the importance 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 [51,52] for its simplicity and generalization potential.

Among the algorithms that exploit the decomposition of the state and observations spaces into a set of subspaces, we shall also mention those that use crossover operators to improve the particle sets, see, e.g., [21,45,18]. The key idea is to select pairs of particles, say the ith and jth ones in the particle set, and to swap their values $\mathbf{x}_t^{(i),K}$ and $\mathbf{x}_t^{(j),K}$ on some set K of subparts. Using adequate rules for selecting triples (i, j, K), it has been shown empirically that the number of particles needed for tracking can be significantly reduced. However, it is important to note that such algorithms are no more mathematically sound, i.e., they are not proved to estimate correctly the posterior density. Actually, as Deutscher and Reid point out, the application of crossover operators are "done in the hope that when highly fit building blocks are brought together they will have a good chance of forming a very fit complete individual".

The literature exploiting decomposition to enhance tracking is large and we cannot be exhaustive. As a last example on MCMC-based methods, let us cite the work in [27], where conditional independences among the object's subparts are taken into account to parallelize sampling while guaranteeing unbiasedness. Decomposition of observation spaces has also received attention, notably through the fusion of multiple observations in principled ways [31].

2.2.3 A comparison with the proposed approach

In this paper, which is an extension of [25], we propose an efficient general articulated object tracking framework which differs from the above works in several ways. First, unlike those mentioned in Subsection 2.2.1, we rely neither on specific knowledge about the object nor on its behavior or on any motion prior. We can of course include such knowledge into our framework to enhance its accuracy, but this is not compulsory. Unlike optimization-based approaches, we propose a model whose unbiased convergence toward the posterior density is guaranteed. To achieve performances as good as optimization methods like APF, our method heavily relies on the decomposition of the state and observation spaces. In this class, we significantly outperform PS and its variants in terms of tracking accuracy, of computation times and of reduction of the noises induced by resamplings (we actually perform much fewer resamplings). There exist other methods that exploit dynamic Bayesian network's (DBN) independences (e.g., RBPF) but those either assume strong hypotheses on the distributions of the random variables in subspaces (e.g., in RBPF, $p(\mathbf{x}^2_{1:t}|\mathbf{y}_{1:t}\mathbf{x}^1_{1:t})$ can be estimated by a Kalman filter), or strong hypotheses on the independence structure of the DBN. In our model, we do not and, therefore, it can be used in more general situations. Our exploitation of decomposition relies on processing several parts of the articulated object simultaneously, which is close to parallelizing the tracking. But, unlike [27], our purpose here is not to speed-up computations but rather to increase the quality of tracking as this simultaneous processing opens the path to our new resampling scheme that significantly increases tracking accuracy. This one is close to crossover operators but, unlike the latter, it is *guaranteed* to never alter the estimation of the posterior density.

2.3 Partitioned Sampling (PS)

PS's key idea is to exploit a natural decomposition of the system dynamics w.r.t. subspaces $\mathcal{X}^1, \ldots, \mathcal{X}^P$ of state space $\mathcal{X}$ (see Fig. 1) in order to apply PF only on those subspaces. As subspaces $\mathcal{X}^j$ are usually much smaller than $\mathcal{X}$, the number of particles needed for tracking can be significantly reduced. PS uses a tailored sampling scheme, called "weighted resampling", which ensures that the particle sets resulting from the sequential applications of PF actually represent the joint distribution of the whole state space and are focused on its peaks [50]. So, assume that $\mathcal{X}$ and $\mathcal{Y}$ 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 and that the dynamics of the system follows this decomposition, i.e., $\mathbf{x}_t = f_t(\mathbf{x}_{t-1}, \mathbf{n}^{\mathbf{x}}_t) = f^P_t \circ f^{P-1}_t \circ \cdots \circ f^1_t(\mathbf{x}_{t-1}, \mathbf{n}^{\mathbf{x}}_t)$, where $\circ$ is the usual function composition operator, $\mathbf{n}^{\mathbf{x}}_t$ is a noise, and each function $f^i_t : \mathcal{X} \mapsto \mathcal{X}$ modifies the particles' states only on subspace $\mathcal{X}^i$ [1]. Then PS propagates iteratively each subpart $\mathbf{x}^i_t$ using importance function f^i_t, it applies its correction step followed by a "weighted resampling" that focuses particles on the peaks of the posterior density. When the likelihood function decomposes as well on subspaces $\mathcal{Y}^i$, i.e., when: $p(\mathbf{y}_t|\mathbf{x}_t) = \prod_{i=1}^P p^i(\mathbf{y}^i_t|\mathbf{x}^i_t)$, where $\mathbf{y}^i_t$ and $\mathbf{x}^i_t$ are the projections of $\mathbf{y}_t$ and $\mathbf{x}_t$ on $\mathcal{Y}^i$ and $\mathcal{X}^i$ respectively, weighted resampling can be achieved by first multiplying the particles' weights by $p^i(\mathbf{y}^i_t|\mathbf{x}^i_t)$ and, then, by performing a usual resampling. Note that such decomposition naturally arises when tracking articulated objects. This leads to the condensation diagram given in Fig. 2, where operations "$*f^i_t$" refer to propagations of particles using proposal functions f^i_t defined above, "$\times p^i_t$" refer to the correction steps where particle weights are multiplied by $p^i(\mathbf{y}^i_t|\mathbf{x}^i_t)$, and "$\sim$" refers to usual resamplings. MacCormick and Isard showed the mathematical soundness of this diagram [52].

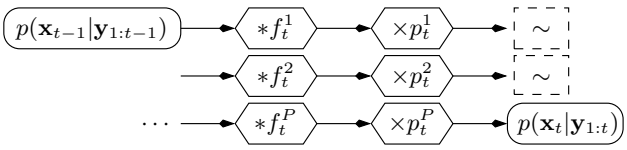


Fig. 2 Partitioned Sampling condensation diagram.

Although PS has been designed to process object subparts sequentially, it can be described as part of a more general class of particle filters that process several object subparts simultaneously. More formally, let K denote the number of steps of such particle filters. In the rest of the paper, for any step $j \in \{1, \ldots, K\}$ of the particle filter, let:

- P_j be the set of object subparts processed simultaneously by the particle filter at its jth step,
- $Q_j = \bigcup_{h=1}^{j} P_h$ be the set of object subparts processed up to (including) the jth step,
- $R_j = \bigcup_{h=j+1}^{P} P_h$ be the set of object subparts still to be processed by the particle filter.

In addition, for simplicity of notations, let $Q_0 = R_K = \emptyset$. Then, PS can be described as Algorithm 1, where K is set to P, the number of subparts in the tracked object, and each P_j is a singleton.

[1] Note that, in [50], functions f^i_t 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.

Input: A particle set $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$ at time t, an image $\mathcal{I}$
Output: A particle set $\{\mathbf{x}_{t+1}^{(i)}, w_{t+1}^{(i)}\}$ at time $t+1$
$Q \leftarrow \emptyset$; $R \leftarrow \{1, \ldots, P\}$
foreach P_j **in** $\{P_1, \ldots, P_K\}$ **do**
  $Q' \leftarrow Q$; $R' \leftarrow R$
  foreach k **in** P_j **do**
    $Q' \leftarrow Q \cup \{k\}$; $R' \leftarrow R' \backslash \{k\}$
    $\{(\mathbf{x}_{t+1}^{(i),Q'}, \mathbf{x}_t^{(i),R'})\} \leftarrow$ propagate subpart k of $(\{\mathbf{x}_{t+1}^{(i),Q}, \mathbf{x}_t^{(i),R}\})$
    $\{(w_{t+1}^{(i),Q'}, w_t^{(i),R'})\} \leftarrow$ correct $(\{(\mathbf{x}_{t+1}^{(i),Q'}, \mathbf{x}_t^{(i),R'}), (w_{t+1}^{(i),Q}, w_t^{(i),R})\}, \mathcal{I})$
  $Q \leftarrow Q'$; $R \leftarrow R'$
  $\{(\mathbf{x}_{t+1}^{(i),Q}, \mathbf{x}_t^{(i),R}), (w_{t+1}^{(i),Q}, w_t^{(i),R})\} \leftarrow$ resample $(\{(\mathbf{x}_{t+1}^{(i),Q}, \mathbf{x}_t^{(i),R}), (w_{t+1}^{(i),Q}, w_t^{(i),R})\})$
return $\{\mathbf{x}_{t+1}^{(i)}, w_{t+1}^{(i)}\}$

Algorithm 1: Partitioned Sampling `PS`.

3 Resampling Methods

PF and its variants all use a resampling step to avoid a problem of degeneracy of the particles, i.e., the case when all but one of the particle's weights are close to zero. Note that, without this step, this problem would necessarily occur. Several resampling schemes are classically used, that we shall review briefly now. Comparisons of their pros and cons can be found in [22]. Note that weighted resampling has already been presented in Subsection 2.3.

Multinomial resampling consists of selecting N numbers k_i, $i = 1, \ldots, N$, w.r.t. a uniform distribution $U((0,1])$ on $(0,1]$. Then, sample $\mathcal{S} = \{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ is substituted by a new sample $\mathcal{S}' = \{\mathbf{x}_t^{(D(k_i))}, \frac{1}{N}\}_{i=1}^N$ where $D(k_i)$ is the unique integer j such that $\sum_{h=1}^{j-1} w_t^{(h)} < k_i \leq \sum_{h=1}^{j} w_t^{(h)}$ [28]. In most implementations, the complexity of *multinomial resampling* is in $O(N \log N)$, where N refers to the sizes of samples $\mathcal{S}$ and $\mathcal{S}'$. However, in [53], it is shown how it can be implemented in an $O(N)$ algorithm.

Stratified resampling differs from multinomial resampling by selecting randomly the k_i's w.r.t. the uniform distribution $U((\frac{i-1}{N}, \frac{i}{N}])$ [42]. Its complexity is in $O(N)$.

In **systematic resampling**, a real number k is drawn w.r.t. $U((0, \frac{1}{N}])$ and, then, the k_i's are defined as $k_i = \frac{i-1}{N} + k$ [42]. Its complexity is also in $O(N)$.

Residual resampling [49] is performed in two steps. First, for every $i \in \{1, \ldots, N\}$, $n'_i = \lfloor N w_t^{(i)} \rfloor$ duplicates of particle $\mathbf{x}_t^{(i)}$ of $\mathcal{S}$ are inserted into $\mathcal{S}'$. The $R = N - \sum_{i=1}^N n'_i$ particles still needed to complete the N-sample $\mathcal{S}'$ are drawn randomly according to the multinomial distribution $\text{Mult}(R; Nw_t^{(1)} - n'_1, \ldots, Nw_t^{(N)} - n'_N)$. The complexity of this method is in $O(N + R \log N)$.

Weighted resampling also samples indices $k_1, \ldots, k_N$ to construct $\mathcal{S}'$ and its complexity follows that of the k_i's resampling. In our experiments, these were computed by *multinomial resampling*, hence our weighted resampling's implementation is in $O(N \log N)$.

Except weighted resampling, all the above resampling schemes assign weight $1/N$ to all the particles in $\mathcal{S}'$. Next, we propose a new resampling method that exploits the structure within articulated objects to improve drastically the efficiency of particle filtering while being competitive in terms of computation times.

4 Combinatorial Resampling Particle Filter

The particle filter we propose in this paper relies on Algo. 1 with carefully chosen sets of object subparts $P_1, \ldots, P_k$ (see page 5) and, most importantly, and this is the main contribution of the paper, a new resampling scheme called *Combinatorial Resampling* specifically designed for tracking objects modeled by several subparts. More precisely, Combinatorial Resampling exploits probabilistic independences among sets of object subparts $P_1, \ldots, P_K$ to permute conditionally independent particles' subparts, thereby producing better quality samples while guaranteeing that the estimation of the posterior density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is unaltered.

4.1 Identifying Parallelizable Sets $P_1, \ldots, P_K$

To be sound mathematically, sets $P_1, \ldots, P_K$ shall be such that all the object subparts in any set P_j are conditionally independent given the subparts in $\cup_{h=1}^{j-1} P_h$. Bayesian networks (BN) [59] and dynamic Bayesian networks (DBN) [54] provide an efficient way to determine these independences.

Definition 1 (Bayesian network (BN) [59]) A Bayes net is a pair $(G, \mathbf{P})$ where $G = (\mathbf{V}, \mathbf{A})$ is a directed acyclic graph, each node $X \in \mathbf{V}$ corresponds to a random variable[2] and arcs (X, Y) in $\mathbf{A}$ represent probabilistic dependences between variables X and Y. $\mathbf{P} = \{p(X|\mathbf{Pa}(X)) : X \in \mathbf{V}\}$ is a set of conditional probabilities, where $\mathbf{Pa}(X)$ is the set of parents of X in G. The joint probability $p(\mathbf{V})$ is then equal to $\prod_{X \in \mathbf{V}} p(X|\mathbf{Pa}(X))$.

BNs can model the uncertainties inherent to object tracking. Let $\mathbf{x}_t^j \in \mathcal{X}^j$ denote the state of the jth subpart of the tracked object. When $\mathbf{V}$ is the set of states $\{\mathbf{x}_t^j\}$ and observations $\{\mathbf{y}_t^j\}$, BNs [59] are an attractive alternative to Markov chains for representing the uncertainties in object tracking (see Fig. 3). Actually, by decomposing the joint distribution over $(\mathbf{x}_t, \mathbf{y}_t)$ into a product of (low-dimensional) conditional distributions, they enable the exploitation of independences among sets of random variables $\mathbf{x}_t^j, \mathbf{y}_t^j$, thereby

[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 $X \in \mathbf{V}$ to denote a node in the BN and its corresponding random variable.

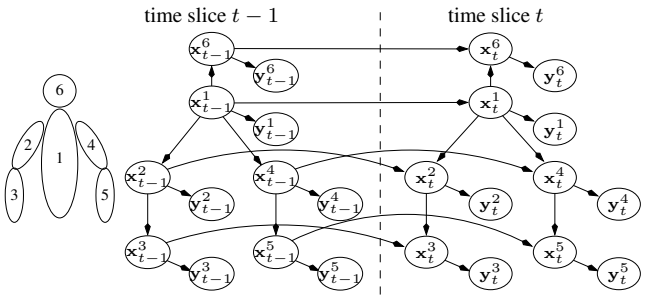


Fig. 3 An unrolled OTDBN.

reducing the complexity of tracking. For instance, Fig. 3 represents part of a BN tracking the person of Fig. 1. Essentially, the articulation joints correspond to arcs in the BN. But BNs do not involve any time dimension, so, for object tracking, DBNs, which are an extension of BNs including this feature, are preferable. In this paper, we will not present general DBNs [54] but rather focus on their definition adapted to tracking that we called *object tracking DBNs*:

Definition 2 (Object Tracking DBN – OTDBN) An OTDBN is a pair $(B_1, B_{\rightarrow})$ of BNs. B_1 is the BN defined over $\{\mathbf{x}_1^i, \mathbf{y}_1^i\}_{i=1}^P$ and represents the joint probability $p(\mathbf{x}_1, \mathbf{y}_1)$, where $\mathbf{x}_1^i$ and $\mathbf{y}_1^i$ represent the state and observation of object subpart i in time slice 1. BN $B_{\rightarrow}$ defines the transition between consecutive time slices $p(\mathbf{x}_t, \mathbf{y}_t | \mathbf{x}_{t-1}, \mathbf{y}_{t-1})$. The unrolled BN of the OTDBN over time slices 1 to T is obtained by appending to B_1 $(T-1)$ times the generic BN $B_{\rightarrow}$. In addition, OTDBNs satisfy the following properties:

1. inside every time slice t, each node $\mathbf{x}_t^i$ can have at most one parent, i.e., for every subpart i and every time slice t, there exists at most one arc $\mathbf{x}_t^j \rightarrow \mathbf{x}_t^i$;
2. for every time slice $t > 1$, node $\mathbf{x}_t^i$ has one and only one parent in time slice $t-1$, which is $\mathbf{x}_{t-1}^i$, i.e, for every subpart i and every time slice $t > 1$, there exists an arc $\mathbf{x}_{t-1}^j \rightarrow \mathbf{x}_t^i$ iff $j = i$;
3. for each state node $\mathbf{x}_t^i$, there exists an observation node $\mathbf{y}_t^i$ whose only parent is $\mathbf{x}_t^i$;
4. nodes $\mathbf{y}_t^i$ have no children.

Fig. 3 represents an unrolled OTDBN: B_1 and $B_{\rightarrow}$ are the BNs in the first and second time slices. The properties of Definition 2 can be easily justified: within a time slice, dependences among random variables should reflect those in the skeleton of the object; as in practice, most are tree-shaped, Property 1 follows. This property is not compulsory for our approach but it simplifies the proofs given in the appendix. As for Property 2, if the state of object subpart i depends on that of subpart j, this is probably due to an articulation joint between them. In this case, $\mathbf{x}_t^i$ should depend on $\mathbf{x}_t^j$ rather than on $\mathbf{x}_{t-1}^j$. Properties 3 and 4 assert that state $\mathbf{x}_t^i$ is observed through a dedicated random variable $\mathbf{y}_t^i$. As can be seen, the four above properties are rather mild for object tracking and they hold in most applications. They actually do in Fig. 3. One of the key properties of BNs and OTDBNs is their independence model, which is called d-separation:

Definition 3 (d-separation [59]) Two nodes $\mathbf{x}_t^i$ and $\mathbf{x}_s^j$ of an unrolled OTDBN are conditionally dependent given a set of nodes $\mathbf{Z}$ iff there exists a chain, i.e., an undirected path, $\{\mathbf{c}_1 = \mathbf{x}_t^i, \ldots, \mathbf{c}_n = \mathbf{x}_s^j\}$ linking $\mathbf{x}_t^i$ and $\mathbf{x}_s^j$ in the unrolled OTDBN such that the following two conditions hold:

1. for every node $\mathbf{c}_k$ such that the arcs in the path are of type $\mathbf{c}_{k-1} \rightarrow \mathbf{c}_k \leftarrow \mathbf{c}_{k+1}$, 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 *blocked*). If there exists an active chain linking two nodes, those are dependent given $\mathbf{Z}$ and are called d-connected, otherwise they are independent given $\mathbf{Z}$ and are called d-separated.

In our body tracking problem, given the position of the torso up to the current time, both arms are independent, which is precisely what the OTDBN of Fig. 3 encodes. More formally, this OTDBN encodes that $\mathbf{x}_t^4$ and $\mathbf{x}_t^5$ are conditionally independent of $\mathbf{x}_t^2$ and $\mathbf{x}_t^3$ given $\mathbf{x}_{1:t}^1$.[3]

Let us now explain how OTDBNs and their d-separation can be exploited to process simultaneously several objects subparts without introducing any bias in the estimation of the posterior density. 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. 3, $\mathbf{Pa}(\mathbf{x}_t^2) = \{\mathbf{x}_t^1, \mathbf{x}_{t-1}^2\}$ and $\mathbf{Pa}_t(\mathbf{x}_t^2) = \{\mathbf{x}_t^1\}$. Assume that the tracked object's 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 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 $j > 1$, $P_j = \{k \in \{1, \ldots, P\} \backslash \bigcup_{h<j} P_h : \mathbf{Pa}_t(\mathbf{x}_t^k) \subseteq \bigcup_{r \in P_{j-1}} \{\mathbf{x}_t^r\}\}$.

In other words, P_1 represents the set of nodes in time slice t that have no parent in time slice t. As such, they should be the roots of the object's tree skeleton. For instance, in Fig. 3, $P_1 = \{1\}$ corresponds only to the torso. P_2 is the set of subparts whose parents are in P_1, i.e., P_2 corresponds to the object subparts that have an articulation joint with the torso. In our example, $P_2 = \{2, 4, 6\}$ (head and upper arms). Finally, $P_3 = \{3, 5\}$ corresponds to the forearms, which are connected to the upper arms.

[3] Note however that $(\mathbf{x}_t^2, \mathbf{x}_t^3)$ and $(\mathbf{x}_t^4, \mathbf{x}_t^5)$ are not independent given $\mathbf{x}_t^1$ because, for instance, chain $\{\mathbf{x}_t^2, \mathbf{x}_{t-1}^2, \mathbf{x}_{t-1}^1, \mathbf{x}_{t-1}^4, \mathbf{x}_t^4\}$ is active. Considering that $(\mathbf{x}_t^2, \mathbf{x}_t^3)$ and $(\mathbf{x}_t^4, \mathbf{x}_t^5)$ are independent given $\mathbf{x}_t^1$ is a common mistake.

It is not hard to see that, by d-separation, all the nodes $\mathbf{x}_t^k$ of a given P_j are conditionally independent given their parents $\mathbf{Pa}(\mathbf{x}_t^k)$. Hence, PS can propagate/correct these nodes simultaneously and produce a correct estimation of the posterior density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. This justifies the condensation diagram of Fig. 4 where, for every $j \in \{1,\ldots,K\}$, $P_j = \{i^1_{P_j},\ldots,i^{k_j}_{P_j}\}$. Its correctness follows from Proposition 1.

Proposition 1 *The particle set resulting from Algorithm 1, with sets P_j defined as above, sets $Q_j = \bigcup_{h=1}^{j} P_h$ and $R_j = \bigcup_{h=j+1}^{K} P_h$, represents $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.*

4.2 Particle's Substate Permutations

There are two major differences between the diagrams of Fig. 2 and Fig. 4: the latter performs fewer resamplings, thus it introduces less noise in the particle set but, more importantly, it enables to produce better fitted particles by swapping their conditionally independent subparts. Actually, consider again our body tracking example and assume that we processed object subparts P_1 and P_2, i.e., all subparts except the forearms have been processed. Assume we generated the 3 particles $\mathbf{x}_t^{(i)} = (\mathbf{x}_t^{(i),1}, \mathbf{x}_t^{(i),2}, \mathbf{x}_{t-1}^{(i),3}, \mathbf{x}_t^{(i),4}, \mathbf{x}_{t-1}^{(i),5}, \mathbf{x}_t^{(i),6})$ shown in Fig. 5.a where the shaded areas represent the object's true state (ground truth). Remember that subpart 1 is the torso, subparts 2,4,6 are the left and right upper arms and the head respectively, and subparts 3 and 5 are the left and right forearms. As shown in the proof of Proposition 1, after processing the subparts of sets P_1 and P_2, (a.k.a. $Q_2 = \{1,2,4,6\}$) the particle set represents the conditional density $p(\mathbf{x}_t^{Q_2}, \mathbf{x}_{t-1}^{R_2}|\mathbf{y}_{1:t}^{Q_2}, \mathbf{y}_{1:t-1}^{R_2})$, with $R_2 = \{3,5\}$. Up to a normalizing constant, this is equal to density $p(\mathbf{x}_t^{Q_2}, \mathbf{x}_{t-1}^{R_2}, \mathbf{y}_{1:t}^{Q_2}, \mathbf{y}_{1:t-1}^{R_2})$. By the OTDBN of Fig. 3 and by d-separation, $(\mathbf{x}_t^2, \mathbf{x}_{t-1}^3, \mathbf{y}_{1:t}^2, \mathbf{y}_{1:t-1}^3)$ is conditionally independent of the rest of the OTDBN given $\{\mathbf{x}_{1:t}^1\}$. The same applies to $(\mathbf{x}_t^4, \mathbf{x}_{t-1}^5, \mathbf{y}_{1:t}^4, \mathbf{y}_{1:t-1}^5)$ and to $(\mathbf{x}_t^6, \mathbf{y}_{1:t}^6)$. Hence,

$$\begin{aligned} & p(\mathbf{x}_t^{Q_2}, \mathbf{x}_{t-1}^{R_2}, \mathbf{y}_{1:t}^{Q_2}, \mathbf{y}_{1:t-1}^{R_2}) \\ &= \int p(\mathbf{x}_{1:t-1}^1, \mathbf{x}_t^{Q_2}, \mathbf{x}_{t-1}^{R_2}, \mathbf{y}_{1:t}^{Q_2}, \mathbf{y}_{1:t-1}^{R_2})\, d\mathbf{x}_{1:t-1}^1 \\ &= \int p(\mathbf{x}_{1:t}^1, \mathbf{y}_{1:t}^1) p(\mathbf{x}_t^2, \mathbf{x}_{t-1}^3, \mathbf{y}_{1:t}^2, \mathbf{y}_{1:t-1}^3|\mathbf{x}_{1:t}^1) \\ &\qquad p(\mathbf{x}_t^4, \mathbf{x}_{t-1}^5, \mathbf{y}_{1:t}^4, \mathbf{y}_{1:t-1}^5|\mathbf{x}_{1:t}^1) p(\mathbf{x}_t^6, \mathbf{y}_{1:t}^6|\mathbf{x}_{1:t}^1)\, d\mathbf{x}_{1:t-1}^1. \end{aligned} \tag{3}$$

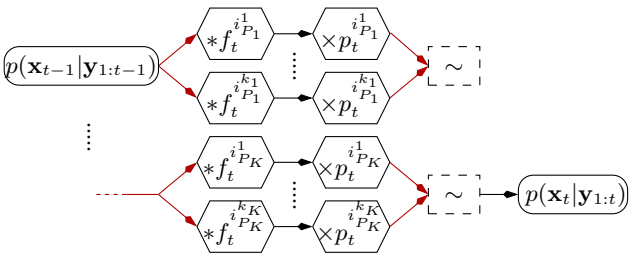


Fig. 4 Multiple-parts simultaneously processed PS condensation diagram. Differences from PS (Fig. 2) are highlighted in red.

Intuitively, subparts $2,3$ of the particles estimate density $p(\mathbf{x}_t^2, \mathbf{x}_{t-1}^3, \mathbf{y}_{1:t}^2, \mathbf{y}_{1:t-1}^3|\mathbf{x}_{1:t}^1)$, subparts $4,5$ estimate density $p(\mathbf{x}_t^4, \mathbf{x}_{t-1}^5, \mathbf{y}_{1:t}^4, \mathbf{y}_{1:t-1}^5|\mathbf{x}_{1:t}^1)$. So, for fixed values of $\mathbf{x}_{1:t}^1$, permuting the values on $\mathcal{X}^2 \times \mathcal{X}^3$ among the particles cannot alter the estimation of $p(\mathbf{x}_t^2, \mathbf{x}_{t-1}^3, \mathbf{y}_{1:t}^2, \mathbf{y}_{1:t-1}^3|\mathbf{x}_{1:t}^1)$ because density estimations are insensitive to permutations within samples. Similarly, any permutation of the values on $\mathcal{X}^4 \times \mathcal{X}^5$ of particles that have the same value of $\mathbf{x}_{1:t}^1$ cannot alter the estimation of $p(\mathbf{x}_t^4, \mathbf{x}_{t-1}^5, \mathbf{y}_{1:t}^4, \mathbf{y}_{1:t-1}^5|\mathbf{x}_{1:t}^1)$. A fortiori, these permutations cannot alter the estimation of the joint posterior density as defined in Eq. (3).

This can be exploited to improve the quality of the particle set. Indeed, consider the 3 particles of Fig. 5.a: those are represented by unfilled ellipses and the ground truth by gray areas. Assume that particles $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ have the same value $\mathbf{x}_{1:t}^1$. Then, by the preceding paragraph, their right (or left) arms can be permuted, hence resulting in the new particle set of Fig. 5.b. Remark that we just substituted 2 particles, $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$, which had low weights due to their bad estimation of the object's right or left arm states, by one particle $\mathbf{x}_t'^{(1)}$ with a high weight (due to a good estimation of all the object's subparts) and another one $\mathbf{x}_t'^{(2)}$ 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 density. Note however that not all permutations are allowed: for instance, none can involve particle $\mathbf{x}_t^{(3)}$ because its torso differs from that of the other particles. This leads to the improved filter described in the diagram of Fig. 6, where operations "$\rightleftharpoons^{P_j}$" refer to the particle subpart swappings briefly described. Remark that, after the resampling operation on subparts P_j, the particles with high weights will be duplicated, which will enable swapping when processing the next subparts P_{j+1} (which are their children).

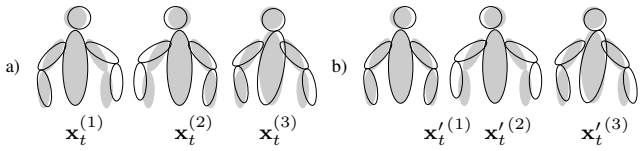


Fig. 5 The particle swapping scheme: a) before swapping; b) after swapping.

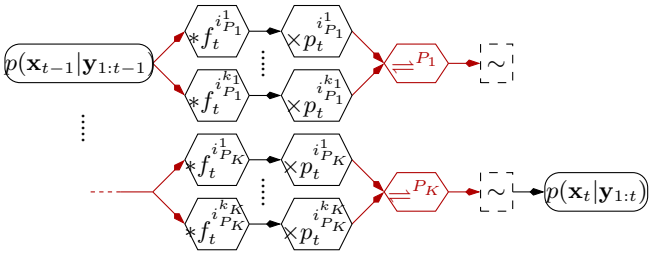


Fig. 6 A Partitioned Sampling scheme improved by simultaneous multiple-parts processing and swapping. Differences from PS (Fig. 2) are highlighted in red.

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 intuitively: if an upper arm is swapped between two particles, their forearms should be swapped as well. More formally, as shown below, when swapping part $\mathbf{x}_t^k$ among particles, it is also necessary and sufficient to permute all the nodes that are descendants of $\mathbf{x}_t^k$ in time slice t, as well as the weights of these subparts, to guarantee the unbiasedness of the estimations. In Algo. 1, for any set of subparts P_h, upon completion of P_h's processing, a resampling is performed that assigns equal weights $1/N$ to all the particles. Therefore, after the simultaneous propagation and correction steps on set of subparts P_j, up to scaling constant $1/N$, the weight of the ith particle is equal to the product of the weights assigned to each subpart $r \in P_j$, i.e., $p(\mathbf{y}_t^r|\mathbf{x}_t^{(i),r})$. Let us call such weights $w_t^{(i),r}$. Then, just before the resampling step on subparts P_j, the weighted particle set is equal to $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j}), \prod_{r\in P_j} w_t^{(i),r}\}_{i=1}^N$. The next proposition justifies which permutations can be performed on this set without altering the posterior density.

Proposition 2 *Let $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j}), \prod_{r\in P_j} w_t^{(i),r}\}_{i=1}^N$ be the particle set obtained after the propagation and correction step of subparts in P_j by Algo. 1. Let $k \in P_j$ and let $\mathbf{Desc}_{t-1}(\mathbf{x}_{t-1}^k)$ be the set of descendants of $\mathbf{x}_{t-1}^k$ in time slice $t-1$ (including both state and observation nodes). Let $\sigma : \{1,\dots,N\} \mapsto \{1,\dots,N\}$ be any permutation such that $\mathbf{x}_s^{(i),h} = \mathbf{x}_s^{(\sigma(i)),h}$ for all the nodes $\mathbf{x}_s^h \in \cup_{s=1}^t \mathbf{Pa}_s(\mathbf{x}_s^k)$. Then, the weighted particle set resulting from the application of σ on the subparts of $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}$ corresponding to $\{\mathbf{x}_t^k\} \cup \mathbf{Desc}_{t-1}(\mathbf{x}_{t-1}^k)$, as well as on their corresponding weights, still estimates $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}|\mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$.*

The above proposition considers permutations only over particles that have the same trajectories over all time slices for the parents of the kth object subpart, i.e., $\cup_{s=1}^t \mathbf{Pa}_s(\mathbf{x}_s^k)$. Such permutations are called *admissible*. Due to d-separation, this requirement is theoretically compulsory to ensure that the estimation of the posterior density is unaltered and any non-admissible permutation will necessarily alter this estimation because it will modify conditionally dependent subparts as if they were independent. However, in practice, state spaces $\mathcal{X}^k$ are continuous, e.g., they represent the object's position, its orientation, its scale, etc., that are all real-valued. As such, densities $p(\mathbf{x}_t^k|\mathbf{Pa}(\mathbf{x}_t^k))$ are also usually continuous functions (e.g., mixtures of Gaussians). Hence, it is highly unlikely that the propagation/correction step of the particle filter produces two particles with precisely the same state $\mathbf{x}_t^k$. As a consequence, two particles having the same state $\mathbf{x}_t^k$ are most likely duplicates of another particle resulting from a resampling step. As such, they thus have the same trajectory $\cup_{s=1}^t \mathbf{Pa}_s(\mathbf{x}_s^k)$. So, for tracking, it is only a very mild approximation to allow permutations taking into account only the values of parents at time t instead of over time slices 1 to t. This suggests tracking in practice using "almost admissible permutations":

Definition 4 (Almost admissibile permutations) A permutation σ is almost admissible if and only if it satisfies the hypotheses of Proposition 2 except that $\mathbf{x}_s^{(i),h}$ is required to be equal to $\mathbf{x}_s^{(\sigma(i)),h}$ only for $s = t$.

The advantage of using almost admissible permutations over admissible permutations is clearly that this enables the particle filter to work with a limited history (only that at time t), hence making it time and memory efficient. But it must be stressed that, in theory, almost admissible permutations cannot guarantee that the estimation of the posterior density remains unaltered. Nevertheless, by dealing with continuous spaces, those biases are very unlikely to occur.

We next show how these permutations can be exploited at the resampling level to improve samples.

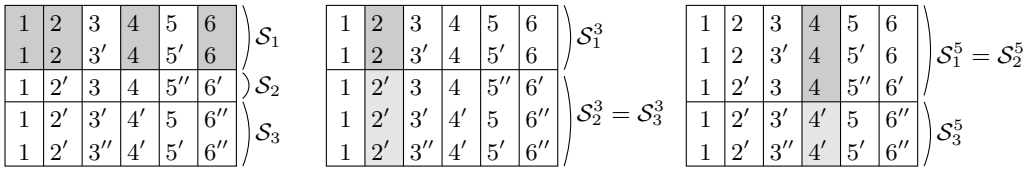
4.3 Our Resampling Approach

All the permutations satisfying the conditions of Proposition 2 can be applied to the particle set without altering the estimation of the posterior density. In addition, the latter is unaffected by duplications of all the particles within the particle set. This suggests Combinatorial Resampling:

Definition 5 (Combinatorial Resampling) Let $\mathcal{S}$ be the particle set at the jth step of Algo. 1. For any $k \in P_j$, let Σ^k be the set of permutations w.r.t. the kth subpart satisfying the conditions of Proposition 2. Let $\Sigma = \prod_{k\in P_j} \Sigma^k$, i.e., Σ is the Cartesian product of all the admissible permutations over all the object subparts of P_j. Let $\mathcal{S}' = \cup_{\sigma\in\Sigma}\{$particle set resulting from the application of σ to $\mathcal{S}\}$. Combinatorial resampling consists of applying any resampling algorithm over the combinatorial set $\mathcal{S}'$ instead of over $\mathcal{S}$.

To illustrate *combinatorial resampling*, consider three particles of the OTDBN of Fig. 3: let $\mathbf{x}_t^{(1)} = \langle 1,2,3,4,5,6\rangle$, $\mathbf{x}_t^{(2)} = \langle 1,2',3',4',5',6'\rangle$ and $\mathbf{x}_t^{(3)} = \langle 1'',2'',3'',4'',5'',6''\rangle$, where each number, j, j', j'', corresponds to a distinct value of state $\mathbf{x}_t^j$. Assume that $\mathcal{S} = \{\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, \mathbf{x}_t^{(3)}\}$ at the 2nd step of Algo. 1, i.e., the object subparts just processed are $P_2 = \{2,4,6\}$. Subparts $\{2,3\}$, $\{4,5\}$ and $\{6\}$ can be permuted in $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ because the value of their torso, i.e. 1, are identical, hence $\mathcal{S}'$ is the union of the result of all such permutations over $\mathcal{S}$ and is thus equal to:

$$
\begin{array}{rll}
\mathcal{S}' = \{ & \langle 1,2\ ,3\ ,4\ ,5\ ,6\ \rangle\ \langle 1,2',3',4',5',6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle & \text{(permut. } \sigma_1\text{)} \\
 & \langle 1,2\ ,3\ ,4\ ,5\ ,6'\rangle\ \langle 1,2',3',4',5',6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle & \text{(permut. } \sigma_2\text{)} \\
 & \langle 1,2\ ,3\ ,4',5',6\ \rangle\ \langle 1,2',3',4\ ,5\ ,6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle & \text{(permut. } \sigma_3\text{)} \\
 & \langle 1,2\ ,3\ ,4',5',6'\rangle\ \langle 1,2',3',4\ ,5\ ,6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle & \text{(permut. } \sigma_4\text{)} \\
 & \langle 1,2',3',4\ ,5\ ,6\ \rangle\ \langle 1,2\ ,3\ ,4',5',6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle & \text{(permut. } \sigma_5\text{)} \\
 & \langle 1,2',3',4\ ,5\ ,6'\rangle\ \langle 1,2\ ,3\ ,4',5',6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle & \text{(permut. } \sigma_6\text{)} \\
 & \langle 1,2',3',4',5',6\ \rangle\ \langle 1,2\ ,3\ ,4\ ,5\ ,6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle & \text{(permut. } \sigma_7\text{)} \\
 & \langle 1,2',3',4',5',6'\rangle\ \langle 1,2\ ,3\ ,4\ ,5\ ,6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle\ \} & \text{(permut. } \sigma_8\text{)}
\end{array}
$$



a) sets $\mathcal{S}_h$ b) sets $\mathcal{S}_h^3$. Gray areas $= \mathbf{Pa}_t(\mathbf{x}_t^3)$ c) sets $\mathcal{S}_h^5$. Gray areas $= \mathbf{Pa}_t(\mathbf{x}_t^5)$

Fig. 7 Sets $\mathcal{S}_h$ and $\mathcal{S}_h^k$. Each row represents a particle and each number a value of the subpart of the particle.

Note that there exist particles, like $\mathbf{x}_t^{(3)}$, that are duplicated several times within $\mathcal{S}'$. Constructing $\mathcal{S}'$ in extension is impossible in practice because $|\Sigma|$ tends to grow exponentially with N, the number of particles, and with $|P_j|$, the number of subparts processed in parallel.

Fortunately, we can sample over $\mathcal{S}'$ without actually constructing it. Consider the resampling at the jth step of Algo. 1, i.e., when subparts P_j have just been propagated and corrected. Our key idea is simply to use a two-step resampling scheme: first, randomly draw values for the subparts in Q_{j-1} and, then, conditioned on these values, choose those for the P_j's and their descendants. Actually, by the hypotheses of Proposition 2, once the values of the subparts in Q_{j-1} are known, all the parents of the subparts in P_j are known as well and, thus, for each $k \in P_j$, the set of particles, say $\mathcal{S}(k)$, that have the same value on $\mathbf{Pa}(\mathbf{x}_t^k)$ as that chosen at the 1st step for $\mathbf{x}_t^{Q_{j-1}}$, can be linearly determined. In addition, by d-separation, subpart k is independent of the other subparts in P_j and, thus, can be dealt with independently. Hence, to guarantee that the posterior density is unaltered, it is sufficient to draw randomly from $\mathcal{S}(k)$ w.r.t. the weights $w_t^{(i),k}$ of the particles in $\mathcal{S}(k)$. So, the main issue is to select the subparts in Q_{j-1} without altering the posterior density. But, as we will see, this just amounts to compute a weight for each value of $\mathbf{x}_t^{Q_{j-1}}$.

For this purpose, let $\mathbf{i}_1, \ldots, \mathbf{i}_R$ be a partition of $\{1, \ldots, N\}$ such that, for any $h, h' \in \{1, \ldots, R\}$, all the particles in $\{\mathbf{x}_t^{(i)}\}_{i \in \mathbf{i}_h}$ have the same value on $\mathbf{x}_t^{Q_{j-1}}$ and this one differs from those of any particle in $\{\mathbf{x}_t^{(i)}\}_{i \in \mathbf{i}_{h'}}$. Let $\mathcal{S}_1 \ldots, \mathcal{S}_R$ be the corresponding particle sets. For instance, Fig. 7.a displays 5 particles of the OTDBN of Fig. 3, where each row represents a particle and values r, r', r'' represent distinct states of object subpart r. Assume that our particle filter just propagated subparts $P_j = \{3, 5\}$. Then Q_{j-1}, the set of subparts propagated at previous steps, is equal to $\{1, 2, 4, 6\}$. Therefore, the 5-element particle set $\mathcal{S}$ can be partitioned into the three sets $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ of Fig. 7.a. Actually, as highlighted by the gray areas, the first two particles have the same values on subparts $1, 2, 4, 6$ and the values of the other particles differ on at least one of those subparts. Clearly, selecting a value for the object subparts in Q_{j-1}, as does the first step of our resampling method, is equivalent to selecting one of the sets $\mathcal{S}_h$ as defined above. Let $N_1, \ldots, N_R$ denote the respective sizes of sets $\mathcal{S}_1, \ldots, \mathcal{S}_R$.

For any particle set $\mathcal{S}_h \in \{\mathcal{S}_1, \ldots, \mathcal{S}_R\}$ and any subpart $k \in P_j$, we denote by $\mathcal{S}_h^k$ the subset of the particles in $\mathcal{S}$ that have the same value on subpart $\mathbf{Pa}_t(\mathbf{x}_t^k)$ as the particles in $\mathcal{S}_h$. For instance, Fig. 7.b shows subsets $\mathcal{S}_h^3$, related to object subpart $k = 3$ (remember that, by the OTDBN of Fig. 3, $\mathbf{Pa}_t(\mathbf{x}_t^3) = \{\mathbf{x}_t^2\}$ and, therefore, sets $\mathcal{S}_h^3$ are the sets of particles whose values in the gray column are identical). Similarly, Fig. 7.c shows subsets $\mathcal{S}_h^5$. Remark that $\mathcal{S}_h^k$ is not necessarily a subset of $\mathcal{S}_h$ (see, e.g., $\mathcal{S}_2^3$). Once a value for subparts Q_{j-1} has been selected by the first step of our resampling algorithm, i.e., once a set $\mathcal{S}_h$ has been chosen, any value of subpart k (and its descendants) of the particles in $\mathcal{S}_h^k$ can be concatenated to that chosen for subparts Q_{j-1} in order to produce a new particle. As shown below, the resulting particle set still estimates correctly the posterior density. To finish with notations, for each set $\mathcal{S}_h^k$, let $\mathbf{i}_h^k \subseteq \{1, \ldots, N\}$ denote the indices w.r.t. $\mathcal{S}$ of the particles of $\mathcal{S}_h^k$. For instance, on Fig. 7.b, $\mathbf{i}_2^3 = \{3, 4, 5\}$ because the third, fourth and fifth particles in $\mathcal{S}$ have value $2'$ on object subpart 2. Let N_h^k denote the cardinal of $\mathcal{S}_h^k$. Finally, for any $h \in \{1, \ldots, R\}$, let $W_h^k = \sum_{i \in \mathbf{i}_h^k} w^{(i),k}$, where $w^{(i),k}$ is the weight assigned to subpart k of the ith particle in $\mathcal{S}$. W_h^k is thus the sum of the weights assigned to the kth subpart of the particles in $\mathcal{S}_h^k$. Then, the following proposition holds:

Proposition 3 *Assign to each set $\mathcal{S}_h$ an (unnormalized) weight W_h defined by $W_h = \prod_{k \in P_j} \frac{N_h}{N_h^k} \times W_h^k$. Then, Algorithm 2 produces a particle set estimating the same posterior density as that given in input.*

Note that the complexity of Algo. 2 is not high: its first step is to determine sets $\mathcal{S}_1, \ldots, \mathcal{S}_R$. This should be done by sorting the particles in $\mathcal{S}$ w.r.t. their $\mathbf{x}_t^{Q_{j-1}}$ values. But, by the continuous nature of the state space, having identical values for $\mathbf{x}_t^{Q_{j-1}}$ is equivalent to having identical values for any $\mathbf{x}_t^k$, $k \in Q_{j-1}$. Therefore, we just need to sort the particles in $\mathcal{S}$ w.r.t. their values in one subpart of Q_{j-1}. Assuming the size of a state $\mathbf{x}_t^k$ is bounded by X, determining sets $\mathcal{S}_1, \ldots, \mathcal{S}_R$ can be done in $O(XN \log N)$. Computing the N_h can then be done linearly in $O(N)$. Determining sets $\mathcal{S}_h^k$ can also be done by sorting w.r.t. each subpart in P_j, hence in $O(|P_j|XN \log N)$. Computing all the N_h^k and W_h^k is then done in $O(N)$. Computing W_h can then be done in $O(|P_j|)$. All these quantities can thus be computed in $O(|P_j|XN \log N)$. Algo. 2 first samples the cen-

Input: A particle set $\mathcal{S} = \{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j}), w_t^{(i)}\}_{i=1}^N$
Output: A new particle set $\{(\mathbf{x}_t^{\prime\prime(i),Q_j}, \mathbf{x}_{t-1}^{\prime\prime(i),R_j}), w_t^{\prime\prime(i)}\}_{i=1}^N$
1 **for** $i = 1$ **to** N **do**
2 $h \leftarrow$ sample $\{1, \ldots, R\}$ w.r.t. unnormalized weights
3 $W_1, \ldots, W_R$
4 $\mathbf{x}_t^{\prime\prime(i),Q_{j-1}} \leftarrow \mathbf{x}_t^{(z),Q_{j-1}}$ where z is any element in $\mathbf{i}_h$
5 **foreach** k *in* P_j **do**
6 $i_h^k \leftarrow$ sample set $\mathbf{i}_h^k$ w.r.t. unnormalized weights
7 $\{w_t^{(r),k}\}_{r \in \mathbf{i}_h^k}$
8 $\mathbf{x}_t^{\prime\prime(i),k} \leftarrow \mathbf{x}_t^{(i_h^k),k}$
9 $\mathbf{x}_{t-1}^{\prime\prime(i),\mathbf{Desc}_{t-1}(\mathbf{x}_{t-1}^k)} \leftarrow \mathbf{x}_{t-1}^{(i_h^k),\mathbf{Desc}_{t-1}(\mathbf{x}_{t-1}^k)}$
10 **return** $\{\mathbf{x}_t^{\prime\prime(i)}, 1/N\}_{i=1}^N$

Algorithm 2: Efficient combinatorial resampling over $\mathcal{S}'$.

tral parts of the particles (lines 2–4), which can be done in $O(|Q_{j-1}|XN)$ using, for instance, systematic resampling. Then, all the subparts in P_j are selected, which can be done in $O(|P_j|XN)$, and their descendants are also added. Hence, overall, sampling complete particles is done in $O(PXN)$. Therefore, the overall complexity of combinatorial resampling over subparts in P_j is $O(|P_j|XN \log N + PXN)$ and, iterating over all subparts, is in $O(PXN \log N + KPXN)$. By comparison, the complexity of sampling with multinomial resampling iteratively over each subpart (as done by partitioned sampling) is $O(PN \log N + P^2XN)$.

We shall now provide experiments highlighting the effectiveness and the efficiency of our resampling scheme and its corresponding particle filter framework.

5 Experimentations

In this section, we test our resampling method and compare it to multinomial, systematic, stratified, residual and weighted resamplings in terms of estimation error and computing time. We also compare our tracker to APF, which is one of our best competitors for articulated object tracking. Here, it should be noted that all the compared methods are integrated into the classical PS framework, i.e., that which processes the object subparts one by one. We could have also integrated them into a framework that processes in parallel each set P_j like our method does. But the result of such trackers would be an absolute disaster: actually, assume that $P_j = \{1, \ldots, k\}$, i.e., k subparts are processed in parallel, then there is only a slight chance that some particles resulting from the processing of P_j have high weights on all these k subparts. Therefore, most particles will fail to track some subparts of the object and, after resampling, there is not much chance that the surviving particles correctly track all the subparts of the object. As a consequence, such tracker quickly fails to track the object. Our method avoids this problem by introducing the swapping operation. In our experiments, we first perform tests on synthetic video sequences to study the robustness of the different approaches with regards to several parameters. We then perform tests on real video sequences to show that our approach is also the most competitive to tackle complex real high-dimensional problems. All the results presented in the next sections are a mean over 30 runs performed on a MacBook Pro with a 2.66 GHz Intel Core i7.

5.1 Test Setup

Articulated objects are modeled by a set of P polygonal subparts (or regions): a first set P_1 contains the central subparts (at least one polygon) to which are linked $|P_j|$, $j > 1$, arms of length at most $K - 1$ (potentially, arm's lengths can differ). See Fig. 8.(d) for two examples: single and multiple object tracking. State vectors contain the parameters describing all the subparts and are defined by $\mathbf{x}_t = \{\mathbf{x}_t^1, \ldots, \mathbf{x}_t^P\}$, with $\mathbf{x}_t^k = \{x_t^k, y_t^k, \theta_t^k\}$ for $k \in P_1$ and $\mathbf{x}_t^k = \{\theta_t^k\}$ for $k \notin P_1$, where (x_t^k, y_t^k) is the center of subpart k, and θ_t^k is its orientation, $k = 1, \ldots, P$. This model results from the fact that subparts in P_j, $j \neq 1$, are linked to those of P_{j-1} by articulation joints. Overall, we have $|\mathcal{X}| = P + 2|P_1|$. 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 one or several articulated object(s). The polygons are manually positioned in the first frame, and the articulated object's joint distribution is then always estimated starting from its central subpart P_1. Particles are propagated using a random walk whose variances σ_x^2, σ_y^2 and σ_θ^2 have been empirically fixed. Of course, for the subparts in P_j, $j > 1$, only σ_θ is used. For all the synthetic tests, we fixed $\sigma_x = \sigma_y = 2$ pixels for the subparts of P_1 and $\sigma_\theta = 0.025$ rad for all the P subparts. For the real tests, we fixed $\sigma_x = \sigma_y = 3$ pixels for the subparts of P_1 and $\sigma_\theta = 0.08$ rad for all the P subparts. Particle weights are computed by measuring the similarity, using the Bhattacharyya distance [6] $\mathbf{d}$, between the distribution of pixels (8-bin histograms) in the regions covered by the estimated object and that given by reference histograms. 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, $\lambda = 50$ (empirically fixed).

We compare six different resampling approaches, plus APF with one layer of simulated annealing. Multinomial, systematic, stratified, residual and weighted resampling, as well as the APF scheme, are integrated into PS. PS propagates and corrects particles polygon after polygon to derive a global estimation of the object. For combinatorial resampling, the object's arms are considered conditionally independent given the central subpart and, thus, the $|P_j|$ subparts, $j > 1$, correspond to the jth joints of all the arms. For weighted resampling, function g is set empirically to $g(x) = e^{20x}$ to favor the selection of high-weighted par-

ticles over low-weighted ones (this was set to get the best estimation results).

Results are compared w.r.t. two criteria: resampling computation times and estimation errors. The latter are given by the sum of the Euclidean distances between each corner of the estimated polygonal subparts and its corresponding corner in the ground truth. Qualitative results are given by superimposing on the frames of the sequences a colored articulated object corresponding to the estimation derived from the weighted sum of the particles.

5.2 Tests on Synthetic Video Sequences

We first performed experiments on synthetic data because this enabled us to create sequences varying the criteria whose impact on our algorithm's efficiency are the most important, namely the number of subparts processed in parallel and the length of the object's arms. As such, this resulted in a fine picture of the different behaviors of our algorithm. We have generated our own synthetic video sequences composed of 300 frames of 800×640 pixels. Each video displays at least one articulated object randomly moving and deforming over time, subject to either weak or strong motions. Examples are given in Fig. 8.(a-c). With various numbers of subparts, the articulated objects are designed to test the ability of resampling to deal with high-dimensional state spaces.

Figure 9 shows, for various numbers N of particles, the plots of the estimation errors, frame by frame, resulting from the six tested resampling approaches for tracking an object defined with $|P_j| = 3$ and $K = 3$ (see the examples on Fig. 10). Average errors and standard deviations over the whole video sequence are reported in the table of Fig. 9. These first tests show different properties of our combinatorial resampling, that will be studied more deeply in the next sections. We briefly discuss them below.

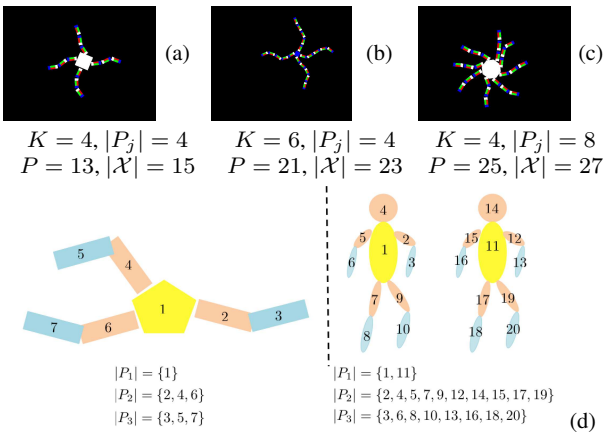
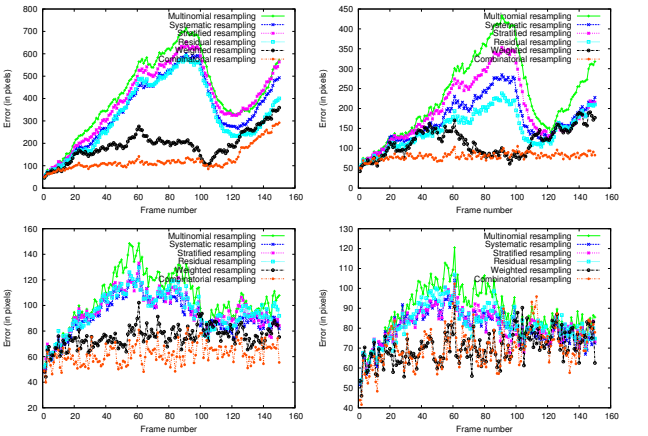


Fig. 8 (a-c) Excerpts of frames from our synthetic video sequences, and the features of the corresponding articulated objects (number of arms $|P_j|$, $j > 1$, length of arms $K - 1$, total number of subparts P, and dimension of state space $|\mathcal{X}|$). (d) Illustration of the notations for two cases. On the left, single object tracking ($P = 7$): subpart 1 is first processed, then subparts of set P_2 are computed in parallel, then subparts of P_3 (all the arms have the same length). On the right, multiple object tracking ($P = 20$): subparts of sets P_j, $j = 1, 2, 3$, are successively computed in parallel (lengths of the arms can vary).



N		Multi.	Syst.	Strat.	Resid.	Weight.	Combi.
10	e	417	348	389	327	190	**110**
	s	77	43	68	36	33	**17**
20	e	231	170	193	148	115	**81**
	s	48	40	42	29	15	**7**
50	e	104	92	95	95	73	**61**
	s	10	9	6	6	3	**1**
100	e	87	80	80	81	70	**60**
	s	4	3	3	3	2	**1**

Fig. 9 First two rows: estimation errors (in pixels) in function of the frame number resulting from the six resampling approaches on an object with $|P_j| = 3$ and $K = 3$; from left to right, top to bottom, $N = \{10, 20, 50, 100\}$. Third row: the corresponding average estimation errors (e, in pixels) and their standard deviations (s, in pixels), depending on N

Estimation errors. Multinomial resampling seems to be the worst approach, in particular in cases of divergence of the filter (see for instance the errors during time intervals $[60, 100]$ and $[120, 150]$ when $N = 20$). Systematic, stratified and residual resamplings provide quite similar results, but can also diverge, while weighted resampling performs better. However, frame by frame, the estimation errors of our approach are always lower than or equivalent to those of the other methods. In addition, as shown in the table at the bottom of Fig. 9, our approach provides the smallest means over all the video sequence of the estimation errors (for $N = 100$, all methods did converge).

Stability. On all the graphs given in Fig. 9, our resampling approach is the less diverging one. This shows the stability of the proposed approach, compared to others. Standard deviations (see the table of Fig. 9) confirm this observation: the lowest ones are always those of combinatorial resampling. Here again, one can note that multinomial resampling is the less stable approach.

Convergence. The table of Fig. 9 provides the average over

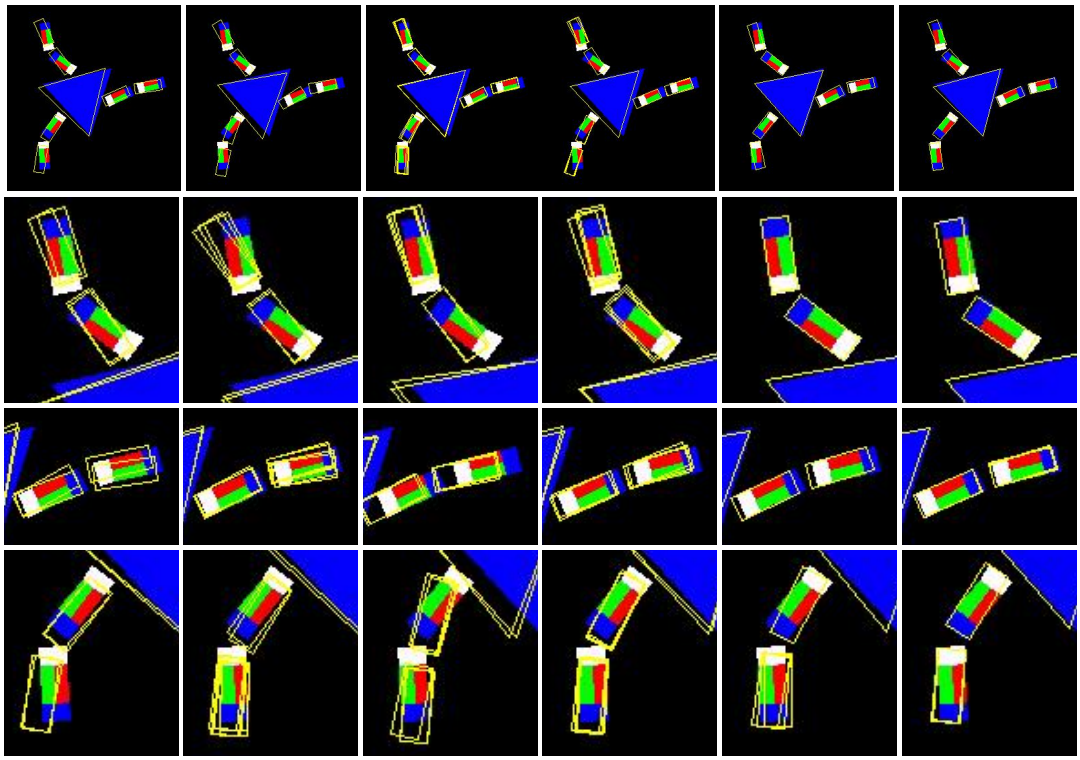


Fig. 10 Qualitative estimation results (zooms) on image 64 of a sequence showing an articulated object with $K = 3$, $|P_j| = 3$ and $N = 50$, for, from left to right: multinomial, systematic, stratified, residual, weighted and combinatorial resampling. Top row: the estimated object (particles' average) is superimposed in yellow. Rows 2, 3 and 4: 10 best particles (i.e., those with the highest weights) are superimposed on, respectively, the top, right and bottom arms.

all the video sequences of the estimation errors resulting from all the approaches. One can see that our combinatorial resampling with $N = 20$ particles induces estimation errors equivalent to those of multinomial, systematic, stratified and residual resamplings executed with $N = 100$ particles, and is also equivalent to weighted resampling with $N = 50$ particles. This shows that, on this example, our approach converges twice to 4 times faster than the other methods.

Accuracy. Fig. 10 shows zooms on qualitative estimation results in frame 64 ($N = 50$). On the top row, the average location of the particles (i.e., the estimated object) is superimposed in yellow on top of the ground truth. We can see that the best qualitative results are given by weighted and combinatorial resamplings. The bottom rows show zooms on the ten best particles (i.e., those with the highest weights), also displayed in yellow. Here combinatorial resampling clearly provides the best results because, unlike the other methods, its ten best particles do not present a large discrepancy (for instance, on the last row of this figure, all approaches but combinatorial resampling have the ten best particles scattered).

This first test shows some attractive features of our method, but those need to be confirmed by performing more tests varying different key parameters, such as the number N of particles, the number $|P_j|$ of arms, and their length $K - 1$. The next subsections address the comparisons of the estimation errors and resampling computation times depending on these parameters, on synthetic and real video sequences.

5.2.1 Role of N (number of particles)

Estimation errors. Fig. 11 (first two rows) shows a convergence study depending on the number N of particles for estimating/tracking the three objects of Fig. 8. For all these tests, combinatorial resampling outperforms all the other methods: i) it converges faster since about only $N = 100$ particles are necessary to do so when the other methods often require 300 particles to converge; ii) combinatorial resampling's error at convergence is much lower than that of the other methods. For instance, on the left column, combinatorial resampling reaches the convergence error of the other methods (about 230 pixels) with only $N = 20$ particles and, with 100 particles, its error decreases to 204 pixels. When the length of the arms increases (middle column), combinatorial resampling stays robust, whereas multinomial, systematic, stratified and residual resamplings tend to fail (estimation errors twice higher). Weighted resampling and APF seem more stable, but give estimation errors 15% (resp. 22%) higher than those of combinatorial resampling. Finally, when the number of subparts treated in parallel increases

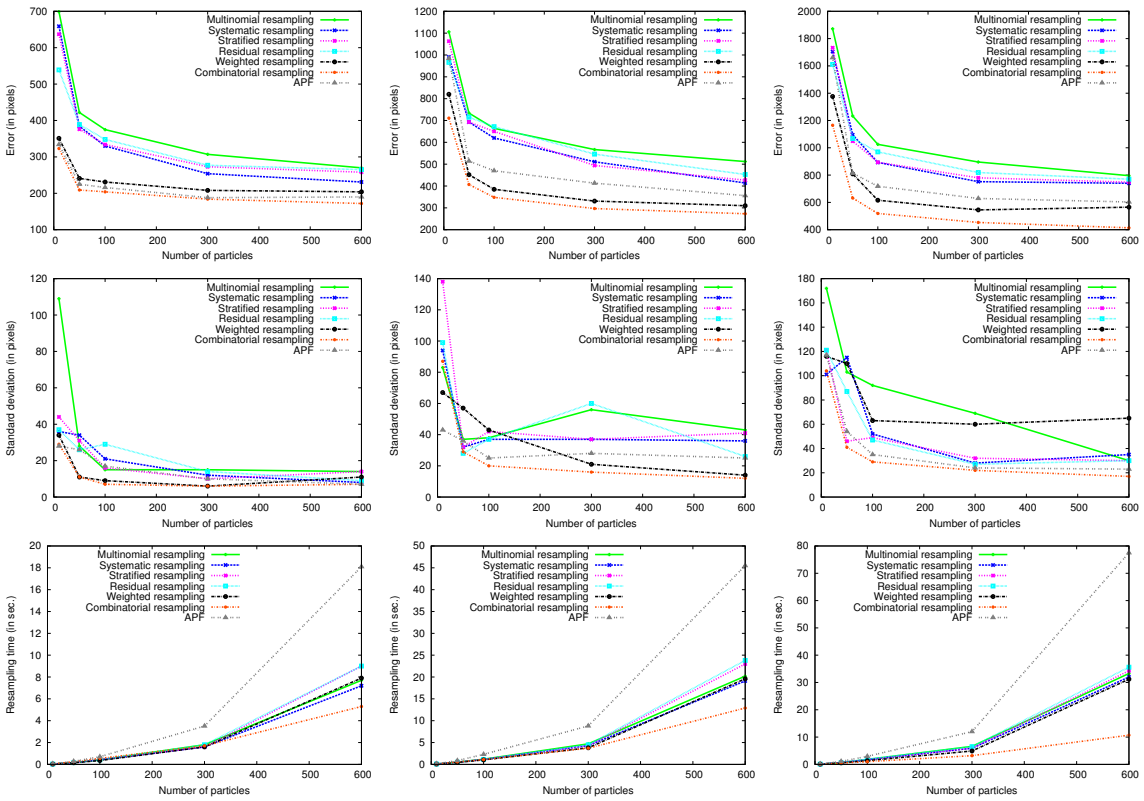


Fig. 11 Comparison of convergence w.r.t. N for different resampling approaches and APF: from top to bottom errors, standard deviation and resampling times for the estimation of density of various articulated objects. From left to right with $|P_i| = 4$, $K = 4$ (Fig. 8.(a)), (b), with $|P_i| = 4$, $K = 6$ (Fig. 8.(b)) and with $|P_i| = 8$, $K = 4$ (Fig. 8.(c)).

(right column), combinatorial resampling also stays stable: its estimation error is 35% lower than weighted resampling, 40% lower than APF, and more than 82% lower that the three other resampling approaches. Concerning standard deviations (middle row of Fig. 11), combinatorial resampling appears to be more stable, especially when the dimension of the state space increases (K or $|P_j|$ higher). It seems that weighted resampling is particularly perturbed when $|P_j|$ increases (see the right image of Fig. 11): this will be discussed in Section 5.2.3.

Resampling times. They are given in Fig. 11 (last row), depending on the number N of particles used for estimation. APF gives higher resampling times, due to its additional annealing layer involving more resampling steps. The five other compared resampling approaches have equivalent computation times. The best approach is combinatorial resampling, especially when the number of particles is high (600). When tracking the object of Fig. 8(a-c), the resampling times are considerably lower with combinatorial resampling than with the other methods: up to 2.8 times faster for the object of Fig. 8(a), up 3.2 to faster for the object of Fig. 8(b) and up to 6.9 faster for the object of Fig. 8(c). This is due to the fact that our tracker performs fewer resamplings than the other methods (K instead of P resamplings). Hence, even if performing combinatorial resampling once is longer than performing another method, overall, combinatorial resampling is globally faster. Note also that combinatorial resampling's computation times increase more slowly with N than the other methods.

5.2.2 Role of K (length of arms)

When K increases, PS's scheme makes the object subparts computed at the end of the algorithm (those whose indices are close to P) more subject to noise that those computed first. This impacts the estimation errors as well as resampling times.

Estimation errors. Comparative results are given in Fig. 12, depending on K ($|P_j| = 4$) and using $N = 100$ and $N = 300$ particles. Concerning estimation errors, we note that weighted, APF and combinatorial resamplings give the best results, while the other approaches give higher and equivalent errors. Lower estimations errors are always given by the proposed approach, using a small number ($N = 100$) or a high number ($N = 300$) of particles. In particular, the gap between the other approaches and ours increase with the dimension of the state space (i.e., with K): for $N = 100$ (resp. $N = 300$), for $K = 2$, combinatorial resampling's errors are

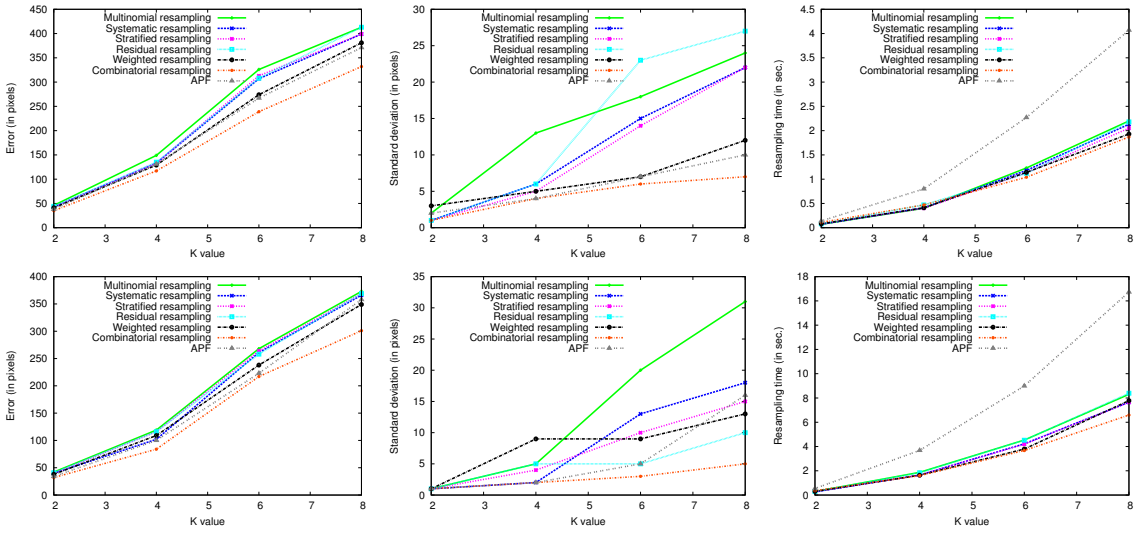


Fig. 12 Estimation errors (left), standard deviations (middle) and resampling times (right), depending on K (length of arms), for $|P_j| = 4$ for a tracking with $N = 100$ particles (top row) and $N = 300$ particles (bottom row).

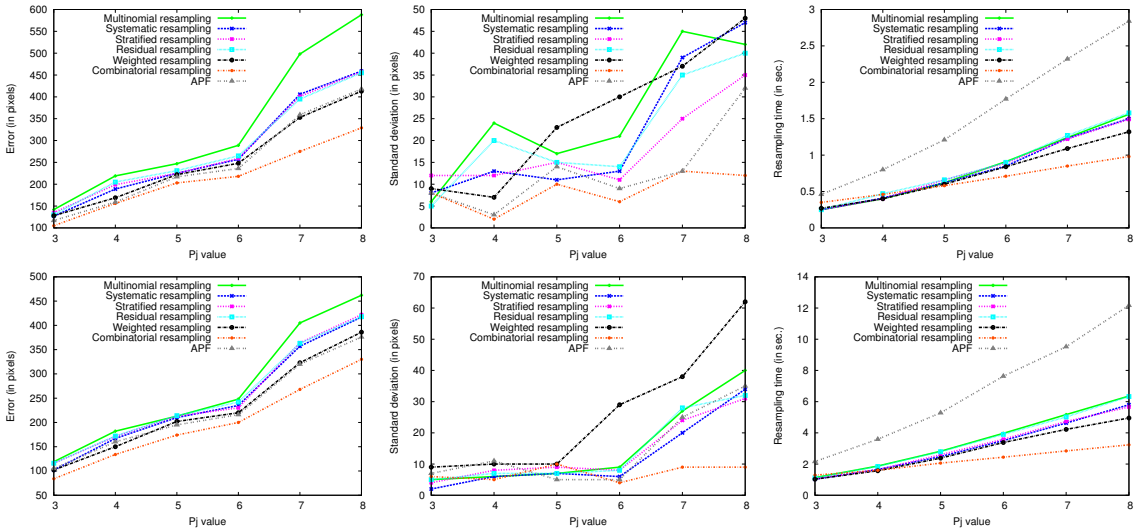


Fig. 13 Estimation errors (left), standard deviation (middle) and resampling computation times (right), depending on $|P_j|$ (number of arms), for $K = 4$ for a tracking with $N = 100$ particles (top row) and $N = 300$ particles (bottom row).

15% (resp. 12%) lower than those of weighted resampling (and APF), and for $K = 8$, combinatorial resampling's errors are 20% (resp. 18%) lower than those of weighted resampling (and APF). We also notice that the standard deviations are more stable with our approach (i.e., they are always lower and they increase slowly compared to the other approaches): this shows the stability of combinatorial resampling, even for high-dimensional state spaces.

Resampling times. Comparative results depending on K ($|P_j| = 4$) are given in the last column of Fig. 12. We can see that all approaches have resampling times that increase similarly with K, but our combinatorial resampling gives lower resampling times, especially when K increases. Note that APF's resampling times increase faster with K: increasing the arms' length also increase the number of resampling steps, because of the supplementary correction step of APF due to the simulated annealing layer. Actually, with one annealing layer and $|P_j| = 4$, each time K is increased by a constant a ($a \geq 1$), this adds $4 * a$ resampling steps to APF.

5.2.3 Role of $|P_j|$ (number of arms)

The number of arms is also a parameter that makes the state space dimension increase. As such, it can have an impact on resampling times and estimation errors. We recall that in PS' scheme, all the subparts in a set P_j are computed

sequentially while, with our approach, they are performed in parallel.

Estimation errors. Comparative results, depending on $|P_j|$ ($K = 4$) are given in Fig. 13: the graphs on the left show the average estimation errors and the graphs at the center show the average error's standard deviations. Here again, the lowest estimation errors result from our combinatorial resampling, then from APF and, finally, from weighted resampling. The other approaches give similar estimation errors. The gap between the errors resulting from the other approaches and ours also increases with the dimension of the state space (i.e., with $|P_j|$): for $N = 100$ particles (resp. $N = 300$), for $|P_j| = 3$, combinatorial resampling's errors are 20% (resp. 18%) lower than those of the second best approaches (APF and weighted resampling), and for $|P_j| = 8$, combinatorial resampling's errors are 25% (resp. 20%) lower than those of APF and weighted resampling. Combinatorial resampling's standard deviations are more stable and, as for previous tests, increase slowly, whereas those of weighted resampling and of the other approaches increase significantly with $|P_j|$. This shows that the latter are much more dependent on the random generations of particles than our approach.

Resampling times. Comparative results are given in the right column of Fig. 13. As for previous tests, and for the same reasons, we observe that APF is the slowest approach. Multinomial, stratified, systematic and residual resamplings are influenced equivalently by $|P_j|$ (i.e., resampling times increase linearly with $|P_j|$ as do the number of subparts of the object, here, and, thus, the number of resampling steps). For $N = 100$ and $N = 300$ particles, our approach is slower than the other methods with $|P_j| = 2$, and it has equivalent resampling times with $|P_j| = 4$. When $|P_j| > 4$, our approach becomes faster: for $N = 100$ (resp. $N = 300$) particles, it is from 1.7 to 2.6 (resp. from 2.3 to 3.3) times faster than APF, and is from 1.2 to 1.7 (resp. from 1.2 to 2) times faster than the other methods. These tests show the interest of using our combinatorial resampling when many object subparts can be processed simultaneously. In real-world applications, $|P_j|$ is often high, which makes the resampling times of our approach considerably low. For instance, tracking the two articulated objects (people) of Fig. 8.(d), $|P_j|$ can be as high as 10. Tests of multiple object tracking are given in Section 5.2.4 for synthetic data and in Section 5.3 for a real video sequence.

5.2.4 The case of 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: 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 dimension of the state space is multiplied by M. We can also use one filter per object, and each such filter just works in the reduced state space corresponding to the object it tracks: it should actually need fewer particles for an accurate tracking. The goal of this subsection is to show that, with combinatorial resampling, we can use one filter for all the objects, and be 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. In this test, we track $M = 2$ objects, each one being defined with $|P_j| = 4$ and $K = 3$. These objects are moving and deforming independently over time. When only one filter is used to estimate the two objects, we use $N = 300$ or $N = 600$ particles, and when one filter is used per object, we use $N = 150$ or $N = 300$ particles per filter. Table 1 provides comparative results concerning the estimation errors, the standard deviations and the resampling times, that are discussed below.

Estimation errors. First, note that the lowest estimation errors result from combinatorial resampling (using either one or two filters). Remark that, with the other approaches, using two filters is more interesting than using only one for all the objects. This holds for all the numbers N of particles tested and it results from the fact that the "lower" the dimension of the state space, the more robust PS is known to be. Conversely, for combinatorial resampling, the estimation errors are equivalent or lower when using only one filter instead of two. This follows from the fact that increasing state space dimensions also increase the efficiency of swapping (because the products of the best weights $w_t^{(i),k}$ also tend to increase). This is an interesting result, that highlights that our proposed resampling is particularly well-suited for high-dimensional problems.

Resampling times. We observe that, for combinatorial resampling, resampling times are equivalent when using one or two filters (see columns $N = 300$ and $N = 2 \times 300$ of Table 1) because the two filters perform twice the number of resamplings of the single filter but the latter are made in a space twice larger than that used by the two filters. On the contrary, for the other approaches, times are approximately divided by two when dealing with one filter per object. This follows from the fact that, although the number of resamplings is identical whether one or two filters are used, the dimension of the state space for the single filter case is twice that of the multiple filter case. Resampling times for combinatorial resampling are lower than those of the other approaches. This is due to the treatment of the object subparts in parallel (here $|P_j| = 8$) which compensates the overhead due to the computations of weights $W_1, \ldots, W_R$. Our filter converges with $N = 2 \times 150$ particles, when only weighted resampling converges with $N = 2 \times 300$ particles. Finally, APF is still the slowest approach because of its supplementary annealing layer.

Table 1 Comparison of estimation errors (e, in pixels), standard deviation (s, in pixels) and resampling times (t, in seconds) for the estimation of the density of two articulated objects ($K = 3$ and $|P_j| = 4$ for each object), using: one filter per object ($N = 2 \times 150$, $N = 2 \times 300$) or one filter for the two objects ($N = 300$, $N = 600$).

N		Multi.	Syst.	Strat.	Resid.	Weight.	Combin.	APF
2×150	$e(s)$	402(12)	373(10)	372(12)	382(14)	280(6)	**237(6)**	319(13)
	t	0.88	0.84	0.85	0.84	**0.8**	0.9	1.61
300	$e(s)$	439(22)	417(19)	413(21)	434(18)	293(19)	**212(6)**	396(11)
	t	3.94	3.64	3.75	3.81	3.11	**2.14**	7.27
2×300	$e(s)$	383(8)	347(12)	355(11)	359(8)	261(8)	**234(3)**	291(10)
	t	2.18	1.91	2.04	2.13	2.01	**1.94**	4.18
600	$e(s)$	423(13)	398(8)	404(10)	406(14)	280(9)	**210(7)**	367(14)
	t	15.8	14.8	14.8	15.5	15.05	**6.4**	34.21

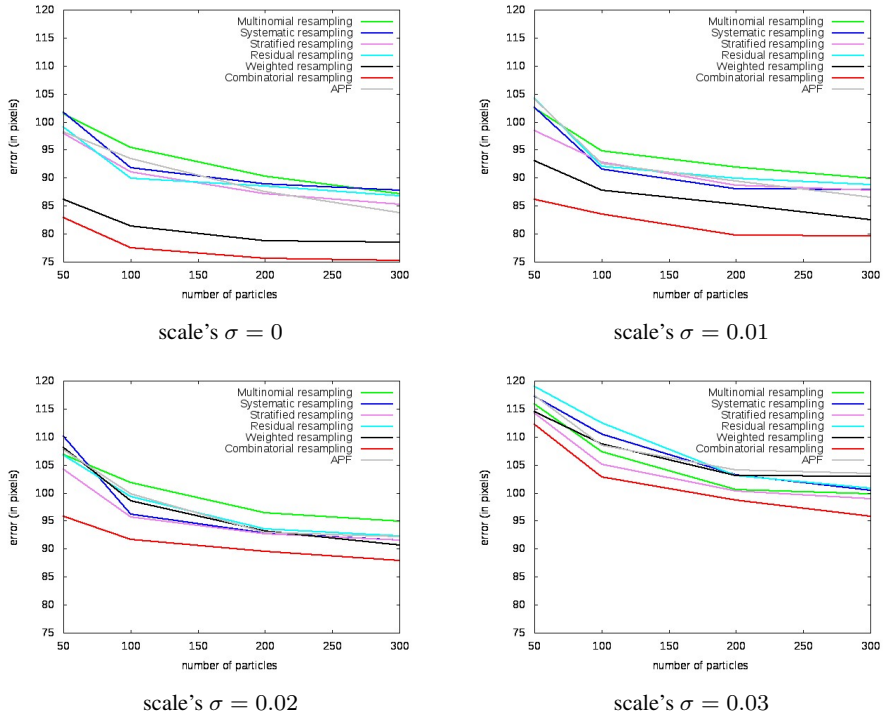
Fig. 14 Tracking results on `JumpRope` ($N = 500$, frames 10, 89, 121, 165) obtained with, row 1, weighted resampling, and rows 2, our combinatorial resampling.

Overall, our approach produces more accurate results than the other approaches and is also faster. In addition, from the accuracy point of view, using one single filter with combinatorial resampling for all the objects is better than using one per object, both in terms of resampling times and estimation errors. Moreover, it requires much fewer particles than the other approaches and can thus be significantly faster. This can prove to be particularly useful when dealing with large-scale state spaces.

5.2.5 Extending the objet's representation

So far in the experiments, the size of the state space we considered was $|\mathcal{X}| = P + 2|P_1|$, where $|P_1|$ is the number of central parts (i.e., it is related to the number of tracked objects), and P the total number of parts (including the central ones). Hence, the high dimension of the state space resulted only from the number of objects and the multiplicity of their subparts. But extending the object's representation by including into it scales or other features is another source of complexity that also results in high-dimensional state spaces and we shall see its impact in this subsection. As an illustration, we consider here adding scales, i.e., the central subparts $\mathbf{x}_t^k$, $k \in P_1$, are modeled as quadruples $\{x_t^k, y_t^k, \theta_t^k, \alpha_t^k\}$, where α_t^k represents a scaling factor, and the other subparts $\mathbf{x}_t^k$, $k \notin P_1$, are modeled as pairs $\{\theta_t^k, \alpha_t^k\}$. As a consequence, the new state space's size is equal to $|\mathcal{X}| = 2P + 2|P_1|$.

In this new setting, the efficiency of our method will certainly decrease but this decrease is not so much related to the size of the state space of each subpart as to the "quality" of the particles: if the scales are not too widespread around that of the ground truth, then, among the particle set, there will exist particles close to the ground truth and this is sufficient for our method to outperform the others. On the contrary, if all the particles are far from the ground truth, then our method will be equivalent to the others because it will just combine "bad" particles' subparts with other "bad" particles' subparts. As an illustration, we performed experiments on an object with $|P_j| = 4$ (4 arms) and $K = 3$ (1 central part and arms of length 2) and whose scale remains unchanged over time (scale = 1). We added in our particle filter a scale feature whose importance function is a Gaussian with mean 1 and standard deviation equal to 0, 0.1, 0.2 or 0.3 ($\sigma = 0$ corresponds to a model without scale changing).



scale's $\sigma = 0$ scale's $\sigma = 0.01$

scale's $\sigma = 0.02$ scale's $\sigma = 0.03$

Fig. 15 Average errors for particle filters including a scaling feature in function of the number of particles.

The average over 20 runs are given on Figure 15. As can be seen, the larger the σ value, the smaller the discrepancy between our method and the others: this is due to the fact that when σ is large, all the particle's subparts are far from the ground truth and, as a consequence, permutations almost never provide "good" particles (when $\sigma = 0.03$, some scales change by almost 10% from one frame to the next). On the contrary, if σ is small (e.g., 0.1), for each subpart there exist "good" subparticles and, after permutations, those produce good overall particles.

As a consequence, even if each subpart is modeled by a high dimensional space, whenever it is possible to guarantee that some particles' subparts will not be too far away from the ground truth (by controlling appropriately the parameters of the particle filter), our method will outperform the others. The farther all the particles will be from the ground truth, the smaller the discrepancy between combinatorial resampling and the other resampling methods. Probably, the only way to ensure accurate tracking when adding new features is to increase the number of particles: as a matter of fact, if the weights of the particles depend on the combination of all the features, it is useless to decompose these features as several nodes in the dynamic Bayesian network in order to increase the opportunities for permutations. But this drawback also holds for all the other methods.

Finally, as far as the computation times are concerned, compared to the other resampling algorithms, adding new features does not have a significant impact. Actually, the computation of the weights used for permutations is independent of the number of features, it depends only on the likelihood weights. In addition, with an efficient implementation, swapping particle's subparts requires only swapping pointers, which again does not depend on the number of features. The only impact on computation time is when new particle sets are created for the next time slice but this operation is exactly the same for all the resampling methods.

5.3 Tests on Real Video Sequences

5.3.1 `UCF50` dataset

We tested our approach on sequences from the UCF50 dataset[4], to demonstrate the efficiency of our combinatorial resampling 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. We manually annotated these sequences to get a ground truth in order to compute estimation errors.

Fig. 14 shows tracking results on the `JumpRope` sequence (containing 290 frames of 320×240 pixels) with $N = 500$ particles. In this sequence, a person quickly moves from left to right while jumping, and crossing/uncrossing his arms and legs. For this test, we defined an articulated object

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

with $P = 12$ subparts, hence $|\mathcal{X}| = 14$, and we compared the estimations resulting from PS with a weighted resampling (first row) with those resulting from our proposed resampling approach (second row). As can be observed, our approach produces better results: its estimations are more stable along the sequence. For example, on the first images, we can see that the estimation of the articulated object fails with weighted resampling but is correct with our combinatorial resampling. Estimation errors, error's standard deviations, as well as resampling times obtained for each of the six compared approaches are reported on Table 2, depending on N. As for synthetic sequences, our test shows that the higher the number of particles, the more our algorithm outperforms the others in terms of computation times and estimation errors. It is also always more accurate, and stable (smaller error's standard deviations).

Qualitative tracking results are given in Fig. 16 using weighted and combinatorial resampling and $N = 1000$ for the 242 320×240 frames of the `Fencing` sequence of the UFC50 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 $P = 27$ subparts, resulting in $|\mathcal{X}| = 29$. This sequence is well suited to highlight the efficiency of our approach that processes in parallel both the different objects and their independent subparts. We compared single filters (one for the two objects) and 2 independent filters (one per object). Here again, our combinatorial resampling improves the tracking results. Table 2 confirms this qualitative analysis, showing estimation errors, error's standard deviations and resampling times depending on the number of filters used (one or two) and the number N of particles per object (500 or 1000). As observed for synthetic sequences, our approach reduces both the estimation errors and resampling times. Our tests also show that results given by our approach (both in terms of total computation times and estimation errors) are equivalent whether we use one filter for the two objects or one filter per object. This shows its ability to correctly consider independent subparts and work in the appropriate subspaces. Concerning resampling times, the proposed approach is the less influenced by N. In particular, for $N = 2000$, our approach is 6.6 times faster than APF (whose resampling times drastically increase with N), and approximately 2.6 times faster than the four other approaches.

Fig. 17 presents zooms on the sequence on which the best 20 particles (i.e., those with the highest weights) are superimposed on the video images for the six tested resampling approaches (here $N = 500$). For the first four resampling approaches, there is a discrepancy among the best particles and we can see that some of them fail to correctly estimate parts of the objects (see the torso and head of the left fencer for multinomial resampling, or the left leg of the right fencer for systematic resampling). For weighted and combinatorial resamplings, we observe only one particle: this means that the 20 best particles have been duplicated by resampling from the same particle. But we can note that this best particle is a better estimation for combinatorial resampling (its global weight before resampling is 0.83, while its value is 0.75 for weighted resampling). This shows that combinatorial resampling better positions particles near the modes of the estimated density.

All these results confirm those obtained for synthetic video sequences. The only difference here comes from the computation of the likelihood, which is much longer for real video sequences (histograms are less empty). Note that, for these tests, we did not use sophisticated likelihood, that could take into account scale changes or occlusion, *etc.* This explains why estimation errors can be higher sometimes.

In the next subsections, we quantitatively and qualitatively test the capacity for monocular tracking of our approach on more complex video sequences that present cluttered backgrounds and self-occlusions. For all these sequences, we use a more sophisticated likelihood, namely the bi-directional silhouette-based likelihood function proposed by Sigal *et al.* in [63], which combines both silhouette and edge informations. Tests in [63] have actually shown this likelihood was achieving the best results. For our approach, the tracker was manually initialized in the first frame. For the other approaches, the results reported are those given by their authors.

5.3.2 `MoBo` dataset

`MoBo` dataset [29] is a publicly available dataset that is used to test human articulated body tracking [2,32], in particular for gait analysis. 25 subjects have been acquired under 8 color camera while walking on a treadmill at different speeds. A body silhouette is available for each image. As this dataset is not provided with any ground truth, we only give here qualitative results obtained in one of the sequences on two different views (front view: camera 3 and semi-lateral view: camera 5). The human is modeled using $P = 10$ subparts, resulting in $|\mathcal{X}| = 12$. Figure 18.(a) gives tracking results obtained with our approach, and Figure 18.(b) those obtained with the classical PS approach (with multinomial resampling). From our results, we can see that self-occlusions do not perturb the tracking. This can be observed in particular in the lateral view where an arm passes through the body, or a leg is hiding another one. Fig. 18.(b) shows that, unlike our approach, PS fails to correctly tracking in case of self occlusion: the combinatorial resampling is more efficient to find the modes of the likeli-

Table 2 Estimation errors (e, in pixels), standard deviations (s, in pixels) and resampling times (t in seconds), for objects of real sequences `JumpRope` and `Fencing`, depending on the number of filters (one or two) and the number N of particles per filter.

		JumpRope			Fencing					
	N	100	200	500	500	2×500	1000	2×1000	2000	2×2000
Multi.	e	320	305	257	475	424	436	392	401	370
	s	28	23	10	48	20	13	9	12	8
	t	**0.3**	0.9	3	4.2	2.7	44.9	34.1	115.2	61.4
Syst.	e	326	290	249	436	407	422	348	383	360
	s	27	23	11	16	15	15	12	13	10
	t	**0.3**	0.9	2.8	4	2.4	42.7	32.9	113.7	61.6
Strat.	e	304	281	249	440	410	418	349	385	357
	s	27	24	11	26	14	12	9	10	8
	t	**0.3**	**0.8**	2.9	3.8	2.6	42.3	32.7	114.2	61.9
Resid.	e	302	275	247	447	401	428	352	391	359
	s	26	21	19	17	16	18	13	12	11
	t	**0.3**	0.85	2.87	4.33	2.61	44.18	33.58	115.1	63.4
Weight.	e	265	242	232	402	386	397	329	303	289
	s	16	13	12	24	16	22	18	9	6
	t	**0.3**	0.9	2.6	4.1	2.4	43.5	32.7	121.5	68.4
Combin.	e	**240**	**216**	**192**	**365**	**352**	**321**	**308**	**264**	**258**
	s	**13**	**10**	**7**	**12**	**10**	**10**	**8**	**6**	**4**
	t	0.4	**0.8**	**2.3**	**2.4**	**2.5**	**15.8**	**15**	**41**	**42.6**
APF	e	264	244	228	429	402	398	339	301	281
	s	7	14	22	23	18	12	10	10	8
	t	0.6	1.7	5.5	5.1	8.7	55.7	102.9	282.1	138

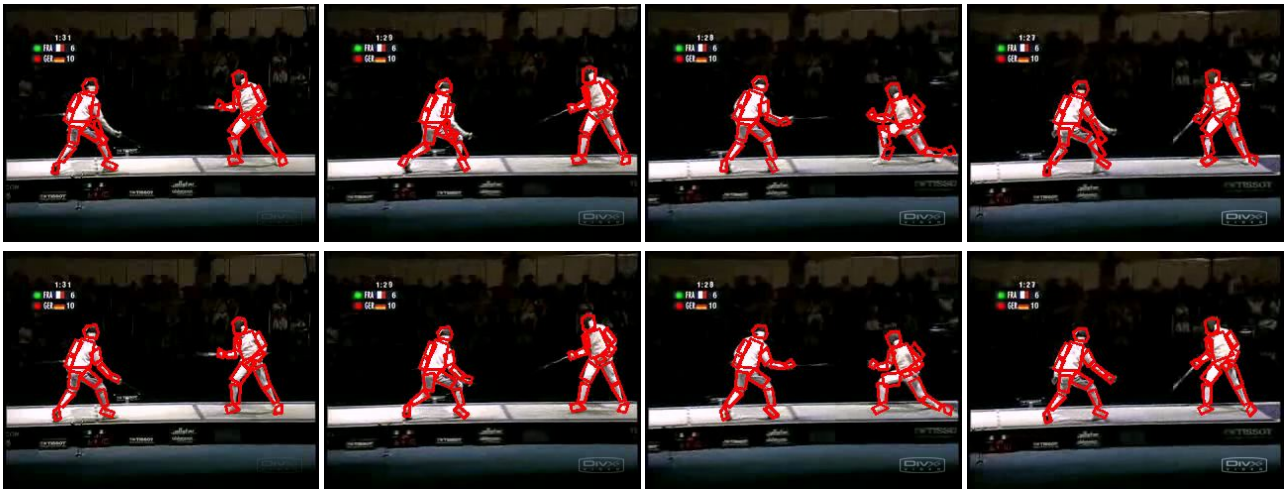


Fig. 16 Tracking results on `Fencing` sequence ($N = 2000$, frames 40, 80, 100 and 120) obtained with, weighted resampling in row 1, and our combinatorial resampling in row 2.

hood, and thus to better estimate the position of the different parts of the articulated model on the human body.

5.3.3 `MOCAP` dataset

The `MOCAP` dataset [3] includes three sequences of subjects walking under different views. We have tested our algorithm on the `Lee walk` sequence and we used Brown University evaluation software to provide monocular tracking errors. Tracking results obtained by our approach are given in Fig. 19: here again, despite the cluttered background and the self-occlusions, our tracker achieves good qualitative performances. A ground truth is provided with this dataset as well as a protocol to compute tracking errors (by measuring the difference of positions of the estimated joints of the articulated model with those of the ground truth). We can thus give quantitative results and compare those with other articulated tracking approaches. These results are reported in Table 3. We compared our approach with four others: a method that decomposes the state space (PS), another that adds an optimization step to perform a mode seeking (APF with 5 layers), a method relying on physical constraints on the human body (Physics [70]) and finally an approach that includes a hierarchical search strategy (HPSO [39]). Results show that our approach gives results similar to the constrained-based

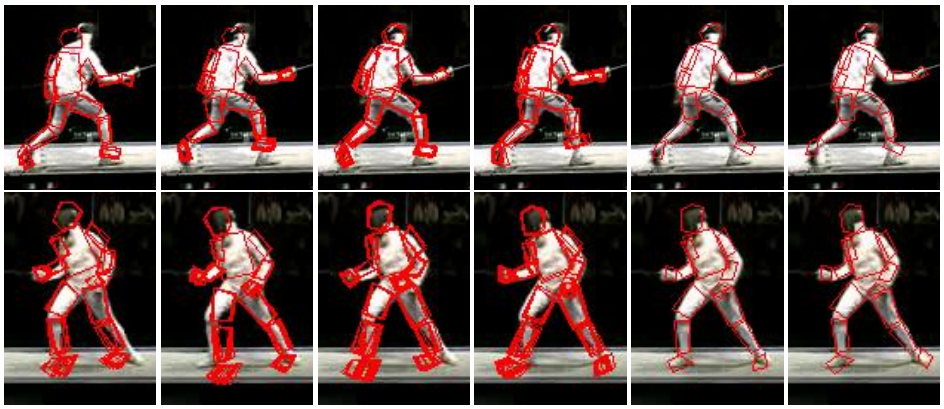
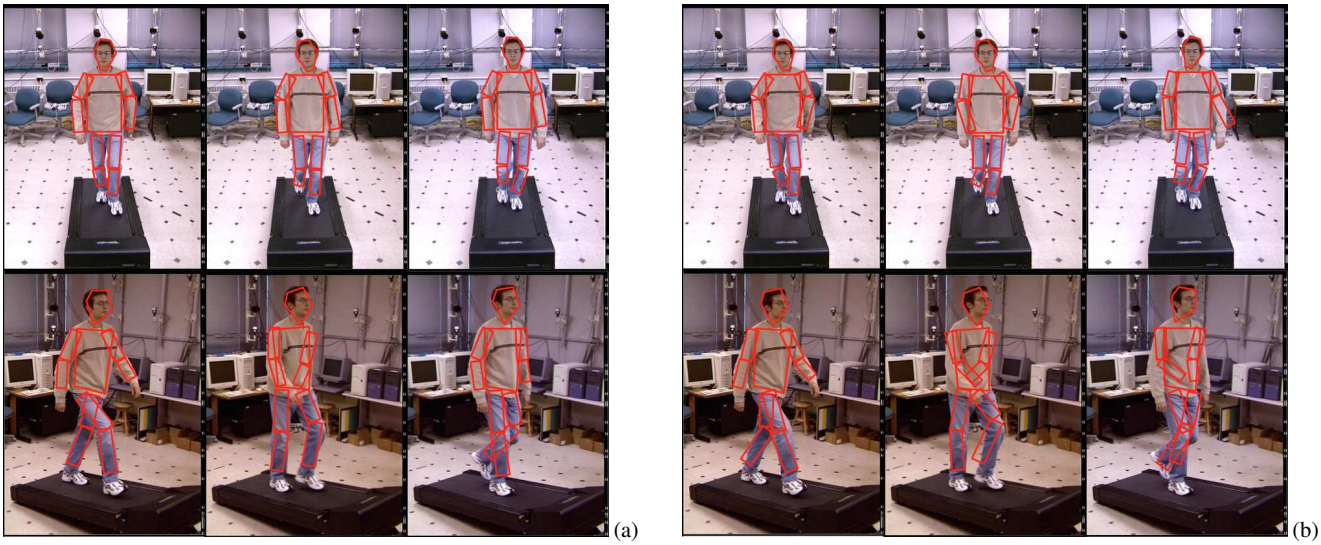


Fig. 17 Qualitative estimation results (zooms) on image 64 of the `Fencing` sequence showing two articulated objects tracked with $N = 500$ particles. From left to right: multinomial, systematic, stratified, residual, weighted and combinatorial resampling. From top to bottom rows: the 20 best particles (i.e., those with the highest weights) are superimposed on respectively the left and right fencers.



(a) (b)

Fig. 18 Qualitative results on `MoBo` dataset [29] (Subject 4, top: camera 3, bottom: camera 5). Tracking results obtained, (a) with combinatorial resampling, and (b) with PS (multinomial resampling). For both approaches, we used $N = 500$ particles.

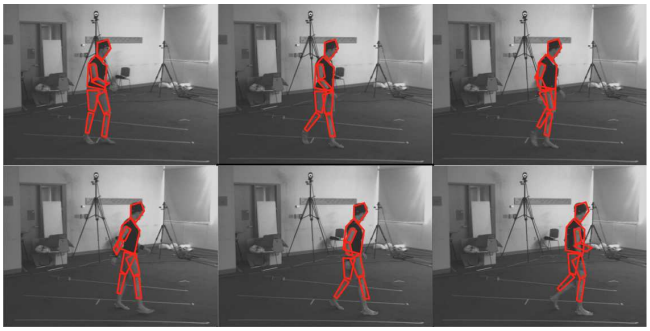


Fig. 19 Qualitative results on the `MOCAP` dataset [3] (`Lee walk` sequence) obtained with our approach with $N = 250$ particles.

one, while being more stable. This once again shows the effectiveness of our combinatorial resampling.

Table 3 Comparison of estimation errors (e, in mm) and standard deviation (s, in mm) of different articulated object tracking algorithms on the `Lee walk` sequence of the `MOCAP` dataset. Results correspond to an average over several runs, with $N = 250$ particles for the first 4 approaches, and $N = 10$ for HPSO (see details in [39]).

	Proposed	PS	APF	Physics [70]	HPSO [39]
e	37.8	62.3	48.2	36.3	52.5
s	6.9	45.7	43.2	9.0	11.5

5.3.4 `TUD-Campus` dataset

`TUD-Campus` dataset [1] shows people walking in a street. We ran our algorithm to track three persons of the sequence and qualitative results are given in Figure 20. Note that, because we do not handle occlusion (only self-occlusion),

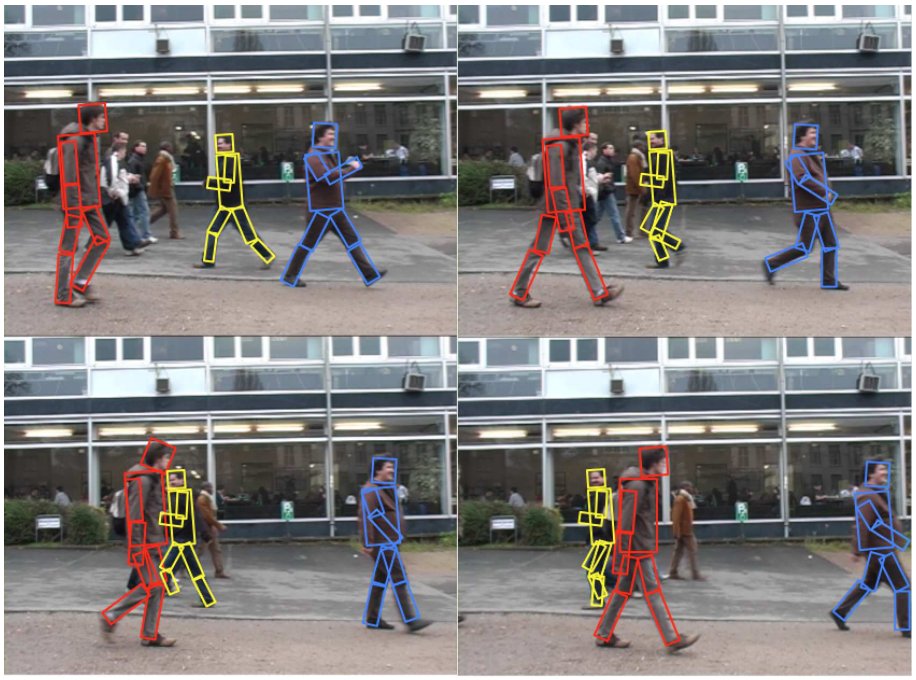


Fig. 20 Qualitative on three person tracking results on `TUD-Campus` dataset [1], obtained by our approach with $N = 800$.

we manually indicate to our algorithm at each time interval when one object (in yellow) is hidden by another (in red). As can be seen, our approach succeeded in tracking simultaneously the three people modeled as articulated objects, despite the high dimensional state space as well as the hard outdoor conditions for tracking.

5.3.5 `HumanEva` dataset

HumanEva datasets [63] have been intensively used to test 2D or 3D articulated object tracking. They include an evaluation software and a generic framework that allows comparison between different approaches. Because video sequences have been acquired using the motion capture technology, a ground truth is available to measure the quality of the estimated articulated object. In particular, an estimation error measure is proposed to evaluate tracking accuracy with available estimated joints. These datasets are divided into two sets: `HumanEva-I` and `HumanEva-II`, and each sequence shows a person performing a specific action (or a series of actions) from a point of view. In the first dataset, the scene was acquired using 3 color cameras whereas 4 color cameras were used for the second dataset.

In Fig. 21 we can see qualitative tracking results of combinatorial resampling on `Box` sequence of `HumanEva-I` dataset, viewed from color camera 1 (front view), in which the person is mimicking a boxing match. This sequence presents self-occlusions, and, as can be seen in this figure, our tracker is able to correctly estimate the articulated object.

Although it is difficult to compare our results with other approaches based on particle filtering (the number of particles used vary, parameters can change - in particular, motion prior -, the tracking error is sometimes "adapted" from the one proposed in this evaluation software), we report in Table 4 comparative results between our estimations and those of other approaches in different video sequences. In our case, we always used the parameters as described in Section 5.1, except that we increased the number of particles to $N = 400$ and only considered monocular tracking (for the `Walking` sequence, we performed tracking when the person is in quasi-frontal view). Depending on the tested sequence, we compare our approach with other that consider tracking in similar conditions as as ours (monocular tracking, tracking during interval times to get frontal views, *etc.*). We compared with two PSO-based approaches [39,55], two optimization based ones [60,21], two physical constraint-based ones [48,68] (prior models) and the classical PS. As can be seen, our approach achieves the best results for three of the four tested sequences, and its results are similar to the best ones for the fourth sequence. The standard deviation is always lower, as observed for synthetic video sequences.

6 Conclusion

In this paper, we have introduced a new resampling method called *Combinatorial Resampling*, whose mathematical correctness has been proven. From a given sample $\mathcal{S}$, this algorithm implicitly constructs a new sample $\mathcal{S}'$ exponentially

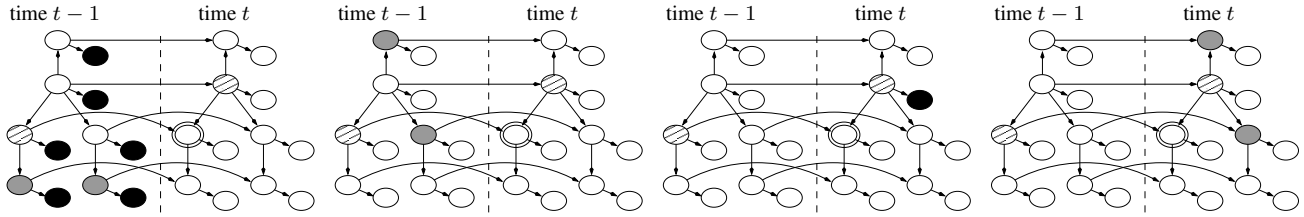
Fig. 21 Qualitative results on `HumanEva I` dataset [63],`Box` sequence (view from color camera 1), obtained by our approach with $N = 400$.

Table 4 Comparison of estimation errors (e, in mm) and standard deviations (s, in mm) for the estimation of the density of articulated objects for different approaches on different sequences (views from color camera 1, except for `Walking` sequence) of `HumanEva-I` dataset.

		Proposed	PS	APF	HAPSOPF [55]	HPSO [39]	GPAPF [60]	CRBM [68]	Li *et al.* [48]
`S1 Gesture`	e	81.1	99.4	105.2	95.1	101.2	-	-	-
	s	5.6	11.1	13.8	6.0	9.2	-	-	-
`S2 ThrowCatch1`	e	65.2	134.7	107.6	212.3	232.8	-	-	68.0
	s	11.9	29.7	47.7	10.2	12.7	-	-	22.1
`S2 Box`	e	74.2	123.1	107.6	-	-	-	75.3	70.0
	s	6.5	29.1	34.0	-	-	-	9.7	22.7
`S2 Walking`	e	68.3	100.2	95.4	-	-	86.3	70.5	68.6
	s	10.9	21.0	34.7	-	-	27.1	24.2	24.6

larger than $\mathcal{S}$. By construction, $\mathcal{S}'$ is more representative than $\mathcal{S}$ of the posterior density over the whole state space. It actually creates many particles near the modes of this density while, at the same time, guaranteeing some diversity among these particles. In addition, this resampling scheme is integrated into a new Particle Filter framework that can process several parts in parallel, thereby reducing the number of resampling steps and, consequently, the noise introduced by these steps. Such a framework can only be effective when it includes swapping operations focusing the particles near the modes of the distribution, as our approach does. Resampling from $\mathcal{S}'$ produces much better results in terms of estimation errors and computation times than resampling from $\mathcal{S}$. This is confirmed by our experiments. Those highlight the fact that our approach outperforms the others compared in terms of accuracy (mean estimation errors over all the sequences are always lower with our resampling approach), but also in terms of stability despite the stochastic nature of the filter (error's standard deviations are stable and lower) and of convergence (fewer particles are needed to get the same tracking errors).

We described combinatorial resampling in terms of *stationary* dynamic Bayesian networks, i.e., DBNs whose structures do not evolve over time. Of course, to take into account occlusions, including self-occlusions, it may be wise to use structure-evolving DBNs, which are known in the literature as non-stationary DBNs. Combinatorial resampling can also be exploited with such DBNs: it is sufficient to recompute at each time slice the sets $\{P_1, \ldots, P_K\}$ that apply for this time slice. By their definition, these sets can be computed in time and space linear w.r.t. the number of arcs in the DBN substructure at time slice t (by sweeping this substructure). Combinatorial Resampling is also compatible with models including physical constraints. For instance, there are some between arms and forearms. As combinatorial resampling only swaps forearm substates that have the same arm sub-



a) $\mathbf{x}_t^k \perp\!\!\!\perp \mathbf{x}_{t-1}^{R_j} \cup \mathbf{y}_{1:t-1} | \mathbf{Pa}(\mathbf{x}_t^k)$ b) $\mathbf{x}_t^k \perp\!\!\!\perp \mathbf{x}_{t-1}^{P_j \setminus \{k\}} | \mathbf{Pa}(\mathbf{x}_t^k)$ c) $\mathbf{x}_t^k \perp\!\!\!\perp \mathbf{x}_t^{Q_{j-1}} \setminus \mathbf{Pa}_t(\mathbf{x}_t^k) \cup \mathbf{y}_t^{Q_{j-1}} | \mathbf{Pa}(\mathbf{x}_t^k)$ d) $\mathbf{x}_t^k \perp\!\!\!\perp \mathbf{x}_t^{P_j \setminus \{k\}} | \mathbf{Pa}(\mathbf{x}_t^k)$

Fig. 22 d-separation analysis for the proof of Proposition 1.

state, if the physical constraints were satisfied before swapping, they necessarily also hold after swapping.

A Proofs

Proof of Proposition 1. Proof by induction on j. Assume that, before processing the object subparts in P_j, particles estimate density $p(\mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}})$. This is clearly the case for $j = 1$ since P_1 are the first subparts processed. Remember that P_j, Q_j, R_j are the set of object subparts processed at the jth loop step, those processed up to (including) the jth loop step and those still to be processed respectively. We will now examine sequentially the densities estimated by the particle set after applying the PF's prediction step in parallel over the subparts in P_j, then after applying PF's correction step and, finally, after resampling. Note that, for simplicity of notation, Algo. 1 is stated slightly differently since it propagates and corrects each $k \in P_j$ one subpart after the other. But propagations are independent of the weights of the particles, hence the result of Algo. 1 is equivalent to first propagating all the particles in P_j and, then, correcting them.

1. Let us show that after the parallel propagations of the subparts in P_j (prediction step), the particle set represents $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}})$. For instance, on Fig. 3, after propagating the subparts in P_2, the particle set estimates $p(\mathbf{x}_t^{\{1,2,4,6\}}, \mathbf{x}_{t-1}^{\{3,5\}} | \, \mathbf{y}_{1:t-1}, \mathbf{y}_t^1)$, i.e., only the positions of the forearms still refer to time $t-1$ and the only observation taken into account at time t is that of the torso (subpart P_1). As the transition function of each subpart k is equal to $p(\mathbf{x}_t^k | \mathbf{Pa}(\mathbf{x}_t^k))$, all these parallel prediction steps correspond to computing:

$$\int p(\mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \prod_{k \in P_j} p(\mathbf{x}_t^k | \mathbf{Pa}(\mathbf{x}_t^k)) d\mathbf{x}_{t-1}^{P_j}. \quad (4)$$

By d-separation, a node is conditionally independent of all of its non descendants given its parents[5]. Hence, for every $k \in P_j$, $\mathbf{x}_t^k$ and $\mathbf{x}_{t-1}^{R_j} \cup \mathbf{y}_{1:t-1}$ are conditionally independent given $\mathbf{Pa}(\mathbf{x}_t^k)$ since the nodes in $\mathbf{x}_{t-1}^{R_j} \cup \mathbf{y}_{1:t-1}$ belong to time slice $t-1$ and are therefore non descendants of $\mathbf{x}_t^k$. As an example, in Fig. 22.a, let $\mathbf{x}_t^k$ be the doubly-circled node, then $\mathbf{Pa}(\mathbf{x}_t^k)$ corresponds to the striped nodes and $\mathbf{x}_{t-1}^{R_j}$ and $\mathbf{y}_{1:t-1}$ are the gray and black nodes respectively. For the same reason, $\mathbf{x}_t^k$ and $\mathbf{x}_{t-1}^{P_j \setminus \{k\}}$ are conditionally independent given $\mathbf{Pa}(\mathbf{x}_t^k)$. In Fig. 22.b, $\mathbf{x}_{t-1}^{P_j \setminus \{k\}}$ are represented by gray nodes. Similarly, by definition of sets P_j and by Property 1 of Definition 2, for any h, the ancestors in time slice t of any node in set $\mathbf{x}_t^{P_h}$ belong to $\mathbf{x}_t^{P_1} \cup \cdots \cup \mathbf{x}_t^{P_{h-1}} = \mathbf{x}_t^{Q_{h-1}}$. Therefore, $\mathbf{x}_t^{Q_{j-1}} \setminus \mathbf{Pa}_t(\mathbf{x}_t^k)$ and $\mathbf{y}_t^{Q_{j-1}}$ cannot be descendants of $\mathbf{x}_t^k$ and are thus conditionally independent of $\mathbf{x}_t^k$ given $\mathbf{Pa}(\mathbf{x}_t^k)$. In Fig. 22.c, $\mathbf{y}_t^{Q_{j-1}}$ is represented by the black node and $\mathbf{x}_t^{Q_{j-1}}$ is its parent. For the same reason, $\mathbf{x}_t^{P_j \setminus \{k\}}$ are non descendants of $\mathbf{x}_t^k$ and are thus conditionally independent of $\mathbf{x}_t^k$ given $\mathbf{Pa}(\mathbf{x}_t^k)$. They are represented by gray nodes in Fig. 22.d. Overall, $\mathbf{x}_t^k$ and $(\mathbf{x}_t^{Q_{j-1}} \setminus \mathbf{Pa}_t(\mathbf{x}_t^k)) \cup \mathbf{x}_{t-1}^{P_j \setminus \{k\}} \cup \mathbf{x}_{t-1}^{R_j} \cup \mathbf{y}_{1:t-1} \cup \mathbf{y}_t^{Q_{j-1}} \cup \mathbf{x}_t^{P_j \setminus \{k\}}$ are conditionally independent given $\mathbf{Pa}(\mathbf{x}_t^k)$.

Denote by $\{k_1, \ldots, k_h\}$ the elements of P_j. Then, by the preceding paragraph, for any $r \in \{1, \ldots, h\}$,

$$p(\mathbf{x}_t^{k_r} | \mathbf{Pa}(\mathbf{x}_t^{k_r})) = p(\mathbf{x}_t^{k_r} | \mathbf{Pa}(\mathbf{x}_t^{k_r}), (\mathbf{x}_t^{Q_{j-1}} \setminus \mathbf{Pa}_t(\mathbf{x}_t^{k_r})), \mathbf{x}_{t-1}^{P_j \setminus \{k_r\}}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}, \mathbf{x}_t^{k_1}, \ldots, \mathbf{x}_t^{k_{r-1}}).$$

By Properties 1 and 2 of Definition 2, $\mathbf{Pa}(\mathbf{x}_t^{k_r}) = \mathbf{Pa}_t(\mathbf{x}_t^{k_r}) \cup \{\mathbf{x}_{t-1}^{k_r}\}$. So, the above equation is equal to:

$$p(\mathbf{x}_t^{k_r} | \mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{P_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}, \mathbf{x}_t^{k_1}, \ldots, \mathbf{x}_t^{k_{r-1}}).$$

By definition of R_j (i.e., R_j is the set of subparts to be processed after the jth loop step), we have $R_{j-1} = P_j \cup R_j$. Therefore, the above equation is equivalent to:

$$p(\mathbf{x}_t^{k_r} | \mathbf{Pa}(\mathbf{x}_t^{k_r})) = p(\mathbf{x}_t^{k_r} | \mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}, \mathbf{x}_t^{k_1}, ..., \mathbf{x}_t^{k_{r-1}}).$$

Consequently, we have:

$$\begin{aligned}
&\prod_{r=1}^{h} p(\mathbf{x}_t^{k_r} | \mathbf{Pa}(\mathbf{x}_t^{k_r})) \\
&= \prod_{r=1}^{h} p(\mathbf{x}_t^{k_r} | \mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}, \mathbf{x}_t^{k_1}, \ldots, \mathbf{x}_t^{k_{r-1}}) \\
&= p(\mathbf{x}_t^{k_1}, \ldots, \mathbf{x}_t^{k_h} | \mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \\
&= p(\mathbf{x}_t^{P_j} | \mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}})
\end{aligned}$$

Consequently, the integral of Eq. (4) is equivalent to:

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

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

2. Let us show that after applying the parallel correction steps on the P_j subparts, the particle set estimates $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. These operations correspond to computing density

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

and, then, normalizing it. By d-separation, nodes $\mathbf{y}_t^k$ are conditionally independent of the rest of the OTDBN given $\mathbf{x}_t^k$, so, if we denote by $\{k_1, \ldots, k_h\}$ the elements of P_j, then, for any $r \in \{1, \ldots, h\}$,

[5] This is the *local Markov Property* and is the core of BNs [59].

$p(\mathbf{y}_t^{k_r}|\mathbf{x}_t^{k_r}) = p(\mathbf{y}_t^{k_r}|\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}, \mathbf{y}_t^{k_1}, \ldots, \mathbf{y}_t^{k_{r-1}})$ since $\mathbf{x}_t^{k_r} \in \mathbf{x}_t^{Q_j}$. Therefore:

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

So, before normalization, the particle set estimates density

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

which, when normalized, is equal to density

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

Finally, as resamplings do not alter densities, at the end of the algorithm, 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 of Proposition 2. If $j = 1$, the proposition trivially holds since σ is applied to all the nodes of the connected component of $\mathbf{x}_t^k$. Assume now that $j \neq 1$. Let $\mathbf{Desc}_t^{\mathbf{x}}(\mathbf{x}_t^k)$ and $\mathbf{Desc}_t^{\mathbf{y}}(\mathbf{x}_t^k)$ denote the set of states and observation nodes respectively in $\mathbf{Desc}_t(\mathbf{x}_t^k)$. We shall now partition the object subparts as described on Fig. 23 to highlight which subparts shall be permuted, which ones shall be identical to enable permutations and which subparts are unconcerned by permutations: let $\mathbf{x}_{t-1}^D = \mathbf{Desc}_{t-1}^{\mathbf{x}}(\mathbf{x}_{t-1}^k)$, $\mathbf{x}_t^{k'} = \mathbf{Pa}_t(\mathbf{x}_t^k)$, $\mathbf{x}_t^V = \mathbf{x}_t^{Q_j} \backslash (\{\mathbf{x}_t^k, \mathbf{x}_t^{k'}\})$ and $\mathbf{x}_{t-1}^W = \mathbf{x}_{t-1}^{R_j} \backslash \mathbf{x}_{t-1}^D$. Thus, the permuted subparts are $\mathbf{x}_t^k \cup \mathbf{x}_{t-1}^D$ (see Fig. 23), the identical subparts are $\mathbf{x}_{1:t}^{k'}$, and the subparts unconcerned by permutations are $\mathbf{x}_t^V \cup \mathbf{x}_{t-1}^W$. Similarly, $\mathbf{y}_{t-1}^D, \mathbf{y}_t^V, \mathbf{y}_{t-1}^W$ denote their corresponding observation nodes. Before permutations, the particle set estimates:

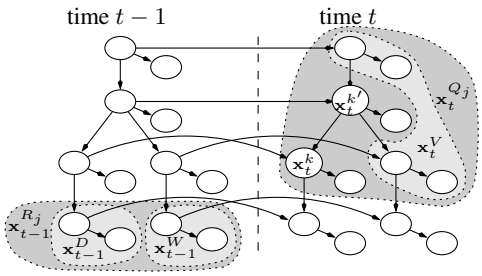


Fig. 23 d-separation analysis.

$$p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}|\mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$$
$$\propto p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$$
$$= p(\mathbf{x}_t^{\{k,k'\}\cup V}, \mathbf{x}_{t-1}^{D\cup W}, \mathbf{y}_{1:t}^{\{k,k'\}\cup V}, \mathbf{y}_{1:t-1}^{D\cup W})$$
$$= \int p(\mathbf{x}_t^{\{k\}\cup V}, \mathbf{x}_{1:t}^{k'}, \mathbf{x}_{t-1}^{D\cup W}, \mathbf{y}_{1:t}^{\{k,k'\}\cup V}, \mathbf{y}_{1:t-1}^{D\cup W}) d\mathbf{x}_{1:t-1}^{k'}$$

Given $\{\mathbf{x}_{1:t}^{k'}\}$, $S = \{\mathbf{x}_t^k\} \cup \mathbf{x}_{t-1}^D \cup \mathbf{y}_{1:t}^k \cup \mathbf{y}_{1:t-1}^D$ is conditionally independent of the rest of the OTDBN because, by Definition 3, no active chain can pass through an arc outgoing from a node in a conditioning set and, removing from the OTDBN the arcs outgoing from $\{\mathbf{x}_{1:t}^{k'}\}$, S is not connected anymore to the rest of the OTDBN. For the same reason, $\mathbf{x}_t^V \cup \mathbf{x}_{t-1}^W \cup \mathbf{y}_{1:t}^V \cup \mathbf{y}_{1:t-1}^W$ is conditionally independent of the rest of the OTDBN given $\{\mathbf{x}_{1:t}^{k'}\}$. Therefore, the above integral is equal to:

$$\int p(\mathbf{x}_{1:t}^{k'}, \mathbf{y}_{1:t}^{k'})\, p(\mathbf{x}_t^k, \mathbf{x}_{t-1}^D, \mathbf{y}_{1:t}^k, \mathbf{y}_{1:t-1}^D|\mathbf{x}_{1:t}^{k'})$$
$$p(\mathbf{x}_t^V, \mathbf{x}_{t-1}^W, \mathbf{y}_{1:t}^V, \mathbf{y}_{1:t-1}^W|\mathbf{x}_{1:t}^{k'})\, d\mathbf{x}_{1:t-1}^{k'}. \quad (5)$$

For fixed values of $\mathbf{x}_{1:t}^{k'}$, permuting particles over subparts $\{\mathbf{x}_t^k\} \cup \mathbf{x}_{t-1}^D$ as well as their weights induced by $\{\mathbf{y}_{1:t}^k\} \cup \mathbf{y}_{1:t-1}^D$ cannot change the estimation of density $p(\mathbf{x}_t^k, \mathbf{x}_{t-1}^D, \mathbf{y}_{1:t}^k, \mathbf{y}_{1:t-1}^D|\mathbf{x}_{1:t}^{k'})$ because estimations by samples are insensitive to the order of the elements in the samples. Moreover, it can neither affect the estimation of density $p(\mathbf{x}_t^V, \mathbf{x}_{t-1}^W, \mathbf{y}_{1:t}^V, \mathbf{y}_{1:t-1}^W|\mathbf{x}_{1:t}^{k'})$ since $\mathbf{x}_t^V \cup \mathbf{x}_{t-1}^W \cup \mathbf{y}_{1:t}^V \cup \mathbf{y}_{1:t-1}^W$ and $\{\mathbf{x}_t^k, \mathbf{y}_{1.t}^k\} \cup \mathbf{x}_{t-1}^D \cup \mathbf{y}_{1:t-1}^D$ are conditionally independent given $\{\mathbf{x}_{1:t}^{k'}\}$. Consequently, applying permutation σ on subparts $\{\mathbf{x}_t^k\} \cup \mathbf{Desc}_{t-1}^{\mathbf{x}}(\mathbf{x}_t^k) = \{\mathbf{x}_t^k\} \cup \mathbf{x}_{t-1}^D$ as well as on their weights cannot change the estimation of Eq. (5) and, therefore, of density $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}|\mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. □

Proof of Proposition 3. Denote $\mathcal{S} = \{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}_{i=1}^N$ and $\mathcal{S}'$ its combinatorial set (see Def. 5). In lines 2–3, Algo. 2 selects which central subpart Q_{j-1} particle $\mathbf{x}_t''$ should have. Subpart $\mathbf{x}_t^{(z),Q_{j-1}}$, $z \in \mathbf{i}_h$, should be selected w.r.t. the sum of the weights of the particles in $\mathcal{S}'$ having the same central subpart Q_{j-1}. Let us show that this is achieved using weights W_h.

In Definition 5, Σ^k is the set of all the possible permutations of the kth subpart of the particles in $\mathcal{S}$. Clearly, within each set $\mathcal{S}_h^k$, all the $N_h^k!$ permutations of the kth subpart of the particles of this set are admissible. They form the cycles within the permutations of Σ^k and, as such, a given permutation σ_h^k over subset $\mathcal{S}_h^k$ of $\mathcal{S}$ may appear many times within Σ^k. For instance, on the particles of Fig. 7.b, one permutation $\sigma \in \Sigma^3$ may swap subpart 3 of the first two particles (belonging to $\mathcal{S}_1^3$) and leave all the other particles of $\mathcal{S}$ untouched, and another permutation $\sigma' \in \Sigma^3$ may also swap subpart 3 of the first two particles and, additionally, swap subpart 3 of the last two particles (belonging to $\mathcal{S}_2^3$). As such, the permutation over $\mathcal{S}_1^3$, i.e., over the first two particles, appears in several permutations of Σ^3 (at least in σ and σ'). Let $\mathbb{S}^k$ denote the set of distinct sets $\mathcal{S}_h^k$ (for instance, in Fig. 7.b, $\mathbb{S}^3 = \{\mathcal{S}_1^3, \mathcal{S}_2^3\} = \{\mathcal{S}_1^3, \mathcal{S}_3^3\}$). Then $|\Sigma^k| = \prod_{\mathcal{S}_h^k \in \mathbb{S}^k} N_h^k!$ and, therefore, a given permutation σ_h^k over $\mathcal{S}_h^k$ appears $|\Sigma^k|/N_h^k! = \prod_{\mathcal{S}_r^k \in \mathbb{S}^k} N_r^k!/N_h^k!$ times in Σ^k.

For the moment, consider only the permutations over the kth subpart. Each time a permutation is applied on $\mathcal{S}_h^k$, this creates a new pack of $|\mathcal{S}_h| = N_h$ particles whose values on Q_{j-1} are equal to those of $\mathcal{S}_h$. By definition, $\mathcal{S}_h \subseteq \mathcal{S}_h^k$ and, usually, this is a strict inclusion. For instance, on Fig. 7, $\mathcal{S}_2 \subset \mathcal{S}_2^3$. Therefore, each aforementioned pack of N_h particles appears several times in $\mathcal{S}'$. For instance, on Fig. 7, $|\mathcal{S}_2^3| = 3$, hence this set induces 6 permutations, but $|\mathcal{S}_2| = 1$, hence, applying the 6 permutations over $\mathcal{S}_2^3$ necessarily implies that all packs in $\mathcal{S}_2'$ appear twice in this set (as a matter of fact, values 3, 3′ and 3′′ appear twice). Let us compute precisely how many times each pack appears. As $|\mathcal{S}_h| = N_h$ and $|\mathcal{S}_h^k| = N_h^k$, there are $A_{N_h^k}^{N_h}$ different possibilities to assign some k-part of $\mathcal{S}_h^k$ to the particles of $\mathcal{S}_h$, where $A_n^k = n!/(n-k)!$ stands for the number of k-permutations out of n elements. Hence, there are $A_{N_h^k}^{N_h}$ distinct packs. As there are $N_h^k!$ permutations within $\mathcal{S}_h^k$, packs are repeated $(N_h^k!/A_{N_h^k}^{N_h})$ times. In addition, by the preceding paragraph, permutations within $\mathcal{S}_h^k$ were already duplicated $\prod_{\mathcal{S}_r^k \in \mathbb{S}^k} N_r^k!/N_h^k!$ times in Σ^k, hence, overall, in $\mathcal{S}'$ (where only permutations over the kth subpart are performed), each pack shall appear $(\prod_{\mathcal{S}_r^k \in \mathbb{S}^k} N_r^k!/N_h^k!) \times (N_h^k!/A_{N_h^k}^{N_h}) = \prod_{\mathcal{S}_r^k \in \mathbb{S}^k} N_r^k!/A_{N_h^k}^{N_h}$ times.

Now, select one particle, say $\mathbf{x}_t^{(i)}$, in $\mathcal{S}_h$. By symmetry, the aforementioned permutations can assign any value of the kth subpart of the particles of $\mathcal{S}_h^k$ to $\mathbf{x}_t^{(i),k}$. So the sum of the weights of the resulting particles over all those permutations is equal to W_h^k times the number of times each value $\mathbf{x}_t^{(z),k}$, $z \in \mathbf{i}_h^k$, is repeated. There are N_h possibilities to choose the particle $\mathbf{x}_t^{(i)}$ of $\mathcal{S}_h$ to which $\mathbf{x}_t^{(z),k}$ is assigned. Once this is done, there remains $A_{N_h^k-1}^{N_h-1}$ possibilities to fill the other

particles. Overall, the weight induced by the permutations over the kth object subpart on the Q_{j-1}-central subpart of $\mathcal{S}_h$ is thus:

$$\frac{\prod_{\mathcal{S}_r^k \in \mathbb{S}^k} N_r^k!}{A_{N_h^k}^{N_h}} \times N_h \times A_{N_h^k - 1}^{N_h - 1} \times W_h^k = \left(\prod_{\mathcal{S}_r^k \in \mathbb{S}^k} N_r^k! \right) \times \frac{N_h}{N_h^k} \times W_h^k. \tag{6}$$

Let us now consider the permutations over all the subparts in P_j. Let $\mathcal{P}_N$ represent the set of all particle sets of size N. Similarly, let $2^{\mathcal{P}_N}$ denote the set of sets of particle sets of size N. Finally, let $f : P_j \times 2^{\mathcal{P}_N} \mapsto 2^{\mathcal{P}_N}$ be the function which i) given $k \in P_j$ and a single particle set $\mathcal{S}$ returns the union of the particle sets resulting from the application on $\mathcal{S}$ of all the admissible permutations w.r.t. subpart k; and ii) given $k \in P_j$ and a set $\{\mathcal{S}_1, \ldots, \mathcal{S}_r\}$ of particle sets, returns $\bigcup_{i=1}^r f(k, \{\mathcal{S}_i\})$. Let $P_j = \{k_1, \ldots, k_d\}$. Clearly, the combinatorial set $\mathcal{S}'$ can be obtained by a sequence of applications of f over $\mathcal{S}$:

$$\mathcal{S}' = f(k_d, f(k_{d-1}, \ldots, f(k_2, f(k_1, \{\mathcal{S}\})) \ldots)).$$

By the preceding paragraphs, the sum of the weights w.r.t. k_1 assigned to the particles of $f(k_1, \{\mathcal{S}\})$ whose Q_{j-1}-central subparts belong to $\mathcal{S}_h$ is given by Eq. (6). By definition of function f, if $f(k_1, \{\mathcal{S}\}) = \{\mathcal{S}_1, \ldots, \mathcal{S}_r\}$, then $f(k_2, f(k_1, \{\mathcal{S}\})) = \bigcup_{i=1}^r f(k_2, \{\mathcal{S}_i\})$ and, by the preceding paragraphs, the sum of the weights w.r.t. subpart k_2 assigned to the particles of $f(k_2, \{\mathcal{S}_i\})$ whose Q_{j-1}-central subparts belong to $\mathcal{S}_h$ is also given by Eq. (6). Therefore, the sum of the weights w.r.t. both k_1 and k_2 of the particles of $f(k_2, f(k_1, \{\mathcal{S}\}))$ whose Q_{j-1}-central subparts belong to $\mathcal{S}_h$ is the product over k_1, k_2 of the formula given in Eq. (6). By induction, after the application of the all the permutations over all the subparts of P_j, the weight assigned to the particles whose central subpart belongs $\mathcal{S}_h$ is thus that equal to:

$$\prod_{k \in P_j} \left(\left(\prod_{\mathcal{S}_r^k \in \mathbb{S}^k} N_r^k! \right) \times \frac{N_h}{N_h^k} \times W_h^k \right)$$
$$= \left(\prod_{k \in P_j} \prod_{\mathcal{S}_r^k \in \mathbb{S}^k} N_r^k! \right) \times \prod_{k \in P_j} \frac{N_h}{N_h^k} \times W_h^k.$$

Note that the first product, $\prod_{k \in P_j} \prod_{\mathcal{S}_r^k \in \mathbb{S}^k} N_r^k! = \prod_{k \in P_j} |\Sigma^k|$, is independent of h. Therefore, up to a proportional constant, the weight of the particles of $\mathcal{S}'$ whose Q_{j-1}-central subparts belong to $\mathcal{S}_h$ is equal to $\prod_{k \in P_j} \frac{N_h}{N_h^k} \times W_h^k$, which is precisely the quantity W_h given in the proposition.

Overall, lines 2–3 select correctly the Q_{j-1} subpart w.r.t. the weight they would have in $\mathcal{S}'$. Once this is done, by d-separation, all the subparts in P_j are independent and should be sampled w.r.t. $p(\mathbf{x}_t^k | \mathbf{Pa}_t(\mathbf{x}_t^k))$, which is done in lines 5–8 since $p(\mathbf{x}_t^k | \mathbf{Pa}_t(\mathbf{x}_t^k))$ is proportional to weight $w_t^{(i),k}$. Consequently, Algorithm 2 produces particle sets similar to those resulting from a resampling on $\mathcal{S}'$. Finally, as shown previously, $\mathcal{S}'$ estimates the same distribution as $\mathcal{S}$, hence Proposition 3 holds. □

References

1. Andriluka, M., Roth, S., Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)
2. Artner, N., Ion, A., Kropatsch, W.: Multi-scale 2d tracking of articulated objects using hierarchical spring systems. Pattern Recognition **44**(4), 800–810 (2011)
3. Balan, A., Sigal, L., Black, M.: A quantitative evaluation of video-based 3D person tracking. In: IEEE VS-PETS Workshop, pp. 349–356 (2005)
4. Bernier, O., Cheung-Mon-Chan, P., Bouguet, A.: Fast nonparametric belief propagation for real-time stereo articulated body tracking. Computer Vision and Image Understanding **113**(1), 29–47 (2009)
5. Besada-Portas, E., Plis, S., Cruz, J., Lane, T.: Parallel subspace sampling for particle filtering in dynamic Bayesian networks. In: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, pp. 131–146 (2009)
6. Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by their probability distributions. Bulletin of the Calcutta Mathematical Society **35**, 99–109 (1943)
7. Bray, M., Koller-Meier, E., Schraudolph, N., van Gool, L.: Stochastic meta-descent for tracking articulated structures. In: IEEE Workshop on Articulated and Nonrigid Motion, Conference on Computer Vision and Pattern Recognition, pp. 1–7 (2004)
8. Bray, M., Koller-Meier, E., Schraudolph, N., Van Gool, L.: Fast stochastic optimization for articulated structure tracking. Image and Visions Computing **25**(3), 352–364 (2007)
9. Bray, M., Kollermeier, E., Vangool, L.: Smart particle filtering for high-dimensional tracking. Computer Vision and Image Understanding **106**(1), 116–129 (2007)
10. Brubaker, M., Fleet, D., Hertzmann, A.: Physics-based person tracking using simplified lower-body dynamics. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007)
11. Brubaker, M., Fleet, D., Hertzmann, A.: Physics-based person tracking using the anthropomorphic walker. International Journal of Computer Vision **87**(1-2), 140–155 (2009)
12. Chang, I.C., Lin, S.Y.: 3D human motion tracking based on a progressive particle filter. Pattern Recognition **43**(10), 3621–3635 (2010)
13. Chang, W., Chen, C., Jian, Y.: Visual tracking in high-dimensional state space by appearance-guided particle filtering. IEEE Transactions on Image Processing **17**(7), 1154–1167 (2008)
14. de Chaumont, F., Dallongeville, S., Chenouard, N., Olivo-Marin, J.: Tracking multiple articulated objects using physics engines: Improvement using multi scale decomposition and quadtrees. In: IEEE International Conference on Image Processing, pp. 4637–4640 (2010)
15. Chen, Z.: Bayesian filtering: from Kalman filters to particle filters, and beyond. Tech. rep., McMaster University, Canada (2003)
16. Covell, M., Rahini, A., Harville, M., Darrell, T.: Articulated-pose estimation using brightness- and depth-constancy constraints. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. 438–445 (2000)
17. Darby, J., Li, B., Costen, N.: Behaviour based particle filtering for human articulated motion tracking. In: IEEE International Conference on Pattern Recognition, pp. 1–4 (2008)
18. Darby, J., Li, B., Costen, N., Fleet, D.J., Lawrence, N.D.: Backing off: Hierarchical decomposition of activity for 3d novel pose recovery. In: British Machine Vision Conference, pp. 1–11 (2009)
19. Das, S., Maity, S., Qu, B.Y., Suganthan, P.: Real-parameter evolutionary multimodal optimization: A survey of the state-of-the-art. Swarm and Evolutionary Computation **1**(2), 71–88 (2011)
20. De Campos, T.: 3D visual tracking of articulated objects and hands. Ph.D. thesis, Saint Anne's College (2006)
21. Deutscher, J., Reid, I.: Articulated body motion capture by stochastic search. International Journal of Computer Vision **61**(2), 185–205 (2005)
22. Douc, R., Cappé, O., Moulines, E.: Comparison of resampling schemes for particle filtering. In: International Symposium on Image and Signal Processing and Analysis, pp. 64–69 (2005)

23. Doucet, A., de Freitas, N., Gordon, N. (eds.): Sequential Monte Carlo methods in practice. Springer Verlag, New York (2001)
24. Doucet, A., de Freitas, N., Murphy, K., Russell, S.: Rao-Blackwellised particle filtering for dynamic Bayesian networks. In: Conference on Uncertainty in Artificial Intelligence, pp. 176–183 (2000)
25. Dubuisson, S., Gonzales, C., Nguyen, X.: DBN-based combinatorial resampling for articulated object tracking. In: Conference on Uncertainty in Artificial Intelligence, pp. 237–246 (2012)
26. Duffner, S., Odobez, J.M., Ricci, E.: Dynamic partitioned sampling for tracking with discriminative features. In: British Machine Vision Conference, pp. 1–11 (2009)
27. Gonzalez, J., Low, Y., Gretton, A., Guestrin, C.: Parallel Gibbs sampling: From colored fields to thin junction trees. In: Artificial Intelligence and Statistics (2011)
28. Gordon, N., Salmond, D.J., Smith, A.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proceedings of Radar and Signal Processing **140**(2), 107–113 (1993)
29. Gross, R., Shi, J.: The cmu motion of body (MoBo) database. Tech. Rep. CMU-RI-TR-01-18, Robotics Institute, Pittsburgh, PA (2001)
30. Guo, F., Qian, G.: Monocular 3D tracking of articulated human motion in silhouette and pose manifolds. EURASIP Journal on Image and Video Processing **2008**, 1–18 (2008)
31. Han, B., Joo, S.W., Davis, L.: Probabilistic fusion tracking using mixture kernel-based Bayesian filtering. In: International Conference on Computer Vision, pp. 1–8 (2007)
32. Han, T., Ning, H., Huang, T.S.: Efficient nonparametric belief propagation with application to articulated body tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 214–221 (2006)
33. Hauberg, S., Pedersen, K.S.: Stick it! articulated tracking using spatial rigid object priors. In: Asian Conference on Computer Vision, pp. 758–769 (2010)
34. Hauberg, S., Pedersen, K.S.: Predicting articulated human motion from spatial processes. International Journal of Computer Vision **94**(3), 317–334 (2011)
35. Hauberg, S., Sommer, S., Pedersen, K.: Gaussian-like spatial priors for articulated tracking. In: IEEE European Conference on Computer Vision, pp. 425–437 (2010)
36. Hofmann, M., Gavrila, D.: 3D human model adaptation by frame selection and shape-texture optimization. Computer Vision and Image Understanding **115**(11), 1559–1570 (2011)
37. Ihler, A., Fisher, J., Fisher, J., Willsky, A., Moses, R.: Nonparametric belief propagation for self-calibration in sensor networks. In: International Symposium on Information Processing in Sensor Networks, pp. 225–233 (2004)
38. Isard, M.: PAMPAS: real-valued graphical models for computer vision. In: IEEE International Conference on Computer Visions and Pattern Recognition, pp. 613–620 (2003)
39. John, V., Trucco, E., Ivekovic, S.: Markerless human articulated tracking using hierarchical particle swarm optimisation. Image and Vision Computing **28**(11), 15301547 (2010)
40. John, V., Trucco, E., Ivekovic, S.: Markerless human articulated tracking using hierarchical particle swarm optimization. Image and Vision Computing **28**(11), 1530–1547 (2010)
41. Kanazawa, K., Koller, D., Russell, S.: Stochastic simulation algorithms for dynamic probabilistic networks. In: Conference on Uncertainty in Artificial Intelligence, pp. 346–35 (1995)
42. Kitagawa, G.: Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. Journal of Computational and Graphical Statistics **5**(1), 1–25 (1996)
43. Kjellstrom, H., Kragic, D., Black, M.: Tracking people interacting with objects. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. 747–754 (2010)
44. Krzeszowski, T., Kwolek, B.: Articulated body motion tracking by combined particle swarm optimization and particle filtering. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. 147–154 (2010)
45. Kwok, N.M., Rad, A.B.: A modified particle filter for simultaneous localization and mapping. Journal of Intelligent and Robotic Systems **46**(4), 365–382 (2006)
46. Kwon, L., Lee, K.: Wang-Landau Monte Carlo-based tracking methods for abrupt motions. Pattern Analysis and Machine Intelligence **35**(4), 1011–1024 (2013)
47. Lanz, O.: Approximate Bayesian multibody tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(9), 1436–1449 (2006)
48. Li, R., Tian, T.P., Sclaroff, S., Yang, M.H.: 3D human motion tracking with a coordinated mixture of factor analyzers. In: Internation Journal on Computer Vision, vol. 87, pp. 170–190 (2010)
49. Liu, J., Chen, R.: Sequential Monte Carlo methods for dynamic systems. Journal of the American Statistical Association **93**, 1032–1044 (1998)
50. MacCormick, J.: Probabilistic modelling and stochastic algorithms for visual localisation and tracking. Ph.D. thesis, Oxford University (2000)
51. MacCormick, J., Blake, A.: A probabilistic exclusion principle for tracking multiple objects. In: IEEE International Conference on Computer Vision, pp. 572–587 (1999)
52. MacCormick, J., Isard, M.: Partitioned sampling, articulated objects, and interface-quality hand tracking. In: IEEE European Conference on Computer Vision, pp. 3–19 (2000)
53. Massey, B.: Fast perfect weighted resampling. In: International Conference on Acoustics, Speech, and Signal Processing, pp. 3457–3460 (2008)
54. Murphy, K.: Dynamic Bayesian Networks: Representation, inference and learning. Ph.D. thesis, UC Berkeley, Computer Science Division (2002)
55. Nguyen, X., Dubuisson, S., Gonzales, C.: Hierarchical annealed particle swarm optimization for articulated object tracking. In: Computer Analysis and Patterns, pp. 319–326 (2013)
56. Oikonomidis, I., Kyriazis, N.: Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In: IEEE International Conference on Computer Vision, pp. 2088–2095 (2011)
57. Oikonomidis, I., Kyriazis, N., Argyros, A.: Tracking the articulated motion of two strongly interacting hands. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1862–1869 (2012)
58. Pantrigo, J., Sanchez, A., Montemayor, A., Duarte, A.: Multidimensional visual tracking using scatter search particle filter. Pattern Recognition Letters **29**(8), 1160–1174 (2008)
59. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann, San Francisco, CA, USA (1988)
60. Raskin, L., Rudzsky, M., Rivlin, E.: Dimensionality reduction using a gaussian process annealed particle filter for tracking and classification of articulated body motions. Computer Vision and Image Understanding **115**(4), 503–519 (2011)
61. Rose, C., Saboune, J., Charpillet, F.: Reducing particle filtering complexity for 3D motion capture using dynamic Bayesian networks. In: International Conference on Artificial Intelligence, pp. 1396–1401 (2008)
62. Shutao, L., Mingkui, T., Tsang, I., Kwok, J.T.Y.: A hybrid PSO-BFGS strategy for global optimization of multimodal functions. IEEE Transactions on Systems, Man, and Cybernetics - Part B **41**(4), 1003–1014 (2011)
63. Sigal, L., Balan, A., Black, M.: HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. International Journal of Computer Vision **87**, 4–27 (2010)

64. Sigal, L., Isard, M., Sigelman, B., Black, M.: Attractive people: Assembling loose-limbed models using non-parametric belief propagation. In: Advances in Neural Information Processing Systems, pp. 1539–1546 (2003)
65. Smith, K., Gatica-perez, D.: Order matters: a distributed sampling method for multi-object tracking. In: British Machine Vision Conference, pp. 25–32 (2004)
66. Sudderth, E.B., Ihler, A., Isard, M., Freeman, W., Willsky, A.: Nonparametric belief propagation. Communications of ACM **53**, 95–103 (2010)
67. Sudderth, E.B., Mandel, M.I., Freeman, W.T., Willsky, A.S.: Visual hand tracking using nonparametric belief propagation. In: IEEE International Conference on Computer Visions and Pattern Recognition Workshops, pp. 189–197 (2004)
68. Taylor, G., Sigal, L., Fleet, D., G.E., H.: Dynamical binary latent variable models for 3D human pose tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 631–638 (2010)
69. Ukita, N.: Articulated pose estimation with parts connectivity using discriminative local oriented contours. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 3154–3161 (2012)
70. Vondrak, M., Sigal, L., Jenkins, O.: Physical simulation for probabilistic motion tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)
71. Vondrak, M., Sigal, L., Jenkins, O.: Physical simulation for probabilistic motion tracking. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)
72. Wang, Q., Xie, L., Liu, J., Xiang, Z.: Enhancing particle swarm optimization based particle filter tracker. In: International conference on Intelligent computing: Part II, pp. 1216–1221 (2006)
73. Widynski, N., Dubuisson, S., Bloch, I.: Fuzzy spatial constraints and ranked partitioned sampling approach for multiple object tracking. Computer Vision and Image Understanding **116**(10), 1076 – 1094 (2012)
74. Yedidia, J., Freeman, W., Weiss, Y.: Constructing free-energy approximations and generalized belief propagation algorithms. IEEE Transactions on Information Theory **51**(7), 2282–2312 (2005)