THÈSE DE DOCTORAT DE l'UNIVERSITÉ PIERRE ET MARIE CURIE

Pour l'obtention du titre de

Docteur en Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Exploitation de réseaux bayésiens dynamiques et du filtre particulaire pour le suivi d'objets articulés

présenté par Xuan Son NGUYEN

JURY

M. François Charpillet *Directeur de recherche, Loria*	Rapporteur
M. Patrick Pérez *Directeur de Recherche, Technicolor*	Rapporteur
Mme. Isabelle Bloch *Professeur à Télécom ParisTech*	Examinatrice
M. Patrick Bouthemy *Professeur à IRISA/INRIA Rennes*	Examninateur
M. Stéphane Doncieux *Professeur à l'Université Pierre et Marie Curie*	Examninateur
M. Jonathan Fabrizio *Maître de Conférence à l'EPITA*	Examninateur
Mme. Séverine Dubuisson *Maître de Conférence HDR à l'Université Pierre et Marie Curie*	Directrice de Thèse
M. Christophe Gonzales *Professeur à l'Université Pierre et Marie Curie*	Directeur de Thèse

Contents

Chapter 1

Introduction

Articulated object tracking has now become a very active research area in the field of computer vision. One of its applications, i.e. human tracking, is used in a variety of domains, such as security surveillance [11], human computer interface, gait analysis [103, 106],... The problem is also of interest from the theoretical point of view. Some of its challenges include, for example, the high dimensionality of state spaces, self-occlusions, kinematic ambiguities or singularities, making it hard to solve and hence, attractive for the tracking community.

Particle Filter (PF) [40, 29] has been shown to be an effective method for solving visual tracking problems. This is due to its ability to deal with non-linear, non-Gaussian and multimodal distributions encountered in such problems. The key idea of particle filter is to approximate the posterior distribution of the target object state by a set of weighted samples. These samples evolve using a proposal distribution and their weights are updated by involving new observations. Under some assumptions, it can be shown that the distribution estimated by particle filter converges in a statistical sense to the target distribution [27, 21]. Unfortunately, in high dimensional problems, such as articulated object tracking problems, the number of samples required for approximating the target distribution can be prohibitively large since it grows exponentially with the number of dimensions (e.g., the number of parts of the object), making the particle filter impractical. To reduce the complexity of tracking algorithms in such problems, various methods have been proposed [64, 108, 90, 44, 80, 107]. One family of approaches that has attracted many researchers is based on the decomposition of the state space into smaller dimensional subspaces where tracking can be achieved using classical methods [64, 80, 7, 18, 47]. This results in tracking algorithms that are linear instead of exponential in the number of parts of the object.

Bayesian Networks (BN) offer a very effective way to represent articulated objects and to express the relationship between their different parts since the object can be

naturally modeled by graphs where each part of the object is represented by a node and the physical link between two neighbor parts is represented by an edge. Most of conditional independence relationships induced by the structural constraints of the articulated objects can be easily encoded in BN. This kind of graphical model has been exploited for articulated object tracking in many works [72, 83, 108, 80, 18] and has been shown to be a powerful tool for modeling the tracking problem in decomposition approaches.

In this thesis, we focus on articulated object tracking and decomposition techniques to deal with the high dimensionality of the state space describing the problem. Our first approach is based on the state-of-the-art algorithm for articulated object tracking: Partition Sampling (PS) [64]. First, we develop an algorithm called Swapping-Based Partitioned Sampling (SBPS) [39, 31, 33]. In this algorithm, the prediction/correction step of PF is performed for a group of parts in parallel instead of part after part as in PS [39]. We also introduce an operation called *swapping* which produces better particles, i.e., particles that are nearer to the modes of the target distribution, after the correction step of the algorithm [31]. We provide a principled way to select the set of parts processed in parallel and to perform the swapping operation so that the posterior distribution is correctly estimated. This approach enables to reduce the number of resampling steps of PS and to increase the tracking accuracy due to the higher number of particles near the modes of the posterior distribution obtained by the swapping operation.

Because the swapping operation generates more particles near the modes of the posterior distribution, this might lead to a situation where the posterior distribution is represented by only a few distinct particles, that induces a loss of particle diversity, resulting in tracking failure in some cases, e.g. when there is a sudden change in movements of the object. To address this problem, we introduce an algorithm called DBN-Based Combinatorial Resampling for articulated object tracking [32]. Adding this resampling scheme into a particle filter produces a new algorithm called Particle Filter with Combinatorial Resampling (PFCR). Instead of aiming to find the best swapping, we create a particle set which contains particles generated from all possible permutations, implicitly constructed to avoid resampling over a particle set of exponential size. This approach allows increasing the number of particles near the modes of the posterior distribution but also the diversity of particles as compared to SBPS, thus improving the tracking accuracy and reducing tracking failure.

Our third approach for articulated object tracking, introduced in this thesis, is based on a hierarchical search and Particle Swarm Optimization (PSO) [48]. This approach called Hierarchical Annealed Particle Swarm Optimization Particle Filter

(HAPSOPF) aims to increase the tracking accuracy and reduce the computational cost of the tracking algorithm by integrating the benefits of these two methods. First, the searching efficiency is improved by performing PSO in subspaces whose dimension is much lower than that of the original space and therefore the tracking accuracy is increased. Second, the search is performed in the same manner as PS, leading to a reduced number of particles required for tracking. As a result, the computational cost of the tracking algorithm is reduced. Moreover, some important factors are introduced into the update equations of PSO to deal with the problem of noisy observation in articulated object tracking.

This thesis is organized as follows. In Chapter 2, we present the visual tracking problem and the PF's framework dedicated to this task. We also discuss some major challenges of articulated object tracking related to high dimensional state spaces. In Chapter 3, we give a non-exhaustive review of approaches for articulated object tracking, and we concentrate on those close to our approach. In Chapter 4, we introduce the theory of the swapping operation and Swapping-Based Partitioned Sampling. We provide a formal proof of correctness of our algorithm and an experimental comparison of our algorithm with PS-based algorithms. In Chapter 5, we introduce DBN-Based Combinatorial Resampling and its associated filter, Particle Filter with Combinatorial Resampling (PFCR). An experimental comparison of our algorithm with some classical resampling algorithms and with some filters are provided. In particular, we compare our two filters SBPS and PFCR. In Chapter 6, we explain Hierarchical Annealed Particle Swarm Optimization Particle Filter (HAPSOPF) and provide an experimental evaluation to compare it with some existing approaches for articulated object tracking. Finally, we draw some conclusions and directions for future research in Chapter 7.

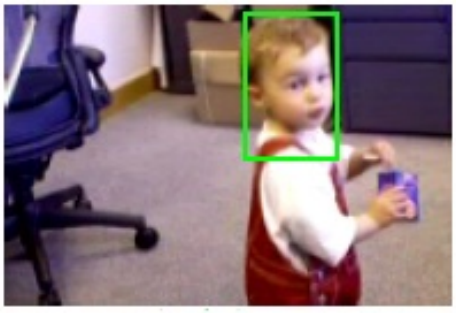
Chapter 2

Problem and Methodological Framework

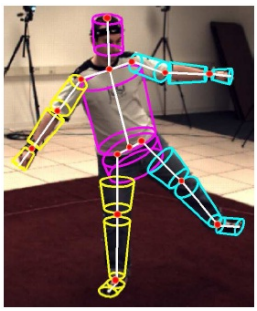
This chapter presents the articulated object tracking problem, the theoretical framework of particle filter and the challenges in applying it to this problem.

2.1 Articulated object tracking

The goal of tracking in video sequences is to infer the position of some moving objects by using the images obtained from one or several cameras. The objects can have a simple form, e.g., rigid objects, or a sophisticated form, e.g., articulated objects which consist of a number of rigid parts linked by joints. In the case of tracking an articulated object, the problem is known as the articulated object tracking. It consists of estimating the position and/or the orientation (2D or 3D) of all rigid parts



(**a**)



(**b**)

Figure 2-1: **Examples of visual tracking**. (**a**) Rigid object tracking. (**b**) Articulated object tracking (figures reproduced respectively from [75] and [87]).

of the object. An example of such a problem is given in Figure 2-1(**b**).

Articulated object tracking has a wide range of applications in many areas like human-machine interaction, medicine, security surveillance... In interactive game applications, the players, instead of pressing buttons, must use different gestures that are recognized by the application by measuring the hand configuration at each time. In human gait analysis, a human motion tracking system can provide the 3D position of several points of the human body in order to detect a tendency towards the fall of a senior, while observing his daily activities at home [82]. In sports science, a body part tracking system can help improving the performance of athletes during their competition [20]...

Articulated object tracking is one of the most challenging problems in visual tracking. In human tracking, some problems, such as different appearance of people, self-occlusion, complexity of human motion, complex background, various illumination conditions... make it hard to solve with satisfactory results. Without taking into account the characteristics for a particular application, the most challenging difficulty that has to be solved in this problem is the complexity incurred by the numerous degrees of freedom (DOFs) of the articulated object. Various approaches have been proposed to address this challenge [64, 108, 90, 44, 9, 80, 107, 24, 52, 47, 17]. The aim of this thesis is to propose some solutions to alleviate the high complexity of the articulated object tracking problem.

2.2 Particle filter

One of the original PF algorithms, also known as Sequential Importance Resampling (SIR), has been introduced in [40]. It then has been used and extended in different domains like signal and image processing, data analysis, forecasting, robotics, tracking, *etc.* PF for visual tracking, also known as Condensation, was first introduced by M. Isard [45]. It has been successfully applied and has been shown to be a very powerful method for visual tracking problem.

2.2.1 Filtering problem

From a Bayesian perspective, the tracking problem can be formulated as follows: Denote $\mathbf{x}_t$ the state of the target object at time t, and $\mathbf{y}_t$ an observation of $\mathbf{x}_t$ at time t, let $\mathbf{y}_{1:t} = \{\mathbf{y}_1, ..., \mathbf{y}_t\}$. The evolution of the state and the relation between the state and its observation can be described by the following equations:

$$\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{n}_t^{\mathbf{x}}) \tag{2.1}$$

$$\mathbf{y}_t = \mathbf{h}_t(\mathbf{x}_t, \mathbf{n}_t^{\mathbf{y}}) \tag{2.2}$$

where $\mathbf{f}_t$: $\mathbb{R}^{d_{\mathbf{x}}} \times \mathbb{R}^{d_{\mathbf{n}^{\mathbf{x}}}} \to \mathbb{R}^{d_{\mathbf{x}}}$, $\mathbf{h}_t$: $\mathbb{R}^{d_{\mathbf{x}}} \times \mathbb{R}^{d_{\mathbf{n}^{\mathbf{y}}}} \to \mathbb{R}^{d_{\mathbf{y}}}$ are possibly nonlinear functions of the state and of the observation, respectively; $d_{\mathbf{x}}$, $d_{\mathbf{y}}$, $d_{\mathbf{n}^{\mathbf{x}}}$, $d_{\mathbf{n}^{\mathbf{y}}}$ are the dimensions of the state space, the observation space, the process noise and the observation noise, respectively. In the context of visual tracking, $\mathbf{f}_t$ and $\mathbf{h}_t$ are usually represented in the probabilistic forms $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{y}_t|\mathbf{x}_t)$, which are respectively called *transition density* and *likelihood density*.

Example of head tracking with PF. Consider the problem of tracking the head of a person moving in a video sequence. In Figure 2-1(**a**), a rectangle specified by its center and size is used to model the head.

State space and its dimension. At any time t, the rectangle can be defined by 4 parameters u_t, v_t, w_t, h_t, which are the coordinates of its center, its width and its height, respectively. Then the state vector is given by $\mathbf{x}_t = [u_t, v_t, w_t, h_t]$. The state space $\mathcal{X}$ of the problem contains all state vectors $\mathbf{x}_t$, $t = 1, \ldots, T$, and its dimension is 4 in this example.

Usually, $\mathbf{f}_t$ and $\mathbf{h}_t$ are vector-valued and time-varying functions and $\mathbf{n}_t^{\mathbf{x}}$ and $\mathbf{n}_t^{\mathbf{y}}$ are independent and identically distributed noise sequences. The evolution of such system can be modeled in a probabilistic way by a Markov chain as in Figure 2-2.

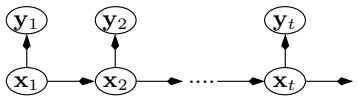


Figure 2-2: **Evolution of a dynamic system represented by a Markov chain**.

Given the two assumptions encoded by the above Markov chain, at any time slice, the state of the target object depends only on that at the previous time slice and the observation of the state depends only on that state. Inferring about such dynamic system involves estimating the *posterior distribution* $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ at any time $t = 1, ..., T$, with the assumption that the prior $p(\mathbf{x}_0)$ is available. In the context of visual tracking, this is equivalent to estimating the whole trajectory of the target object from the first time slice to time t, given all observations until time t. Typically, one only need to estimate the state of the target object at time t, given all observations until time t, which resorts to estimating $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. This is known as the *filtering distribution* and the corresponding problem is known as *filtering problem*.

The filtering problem is solved in two main steps by using the Bayes' theorem.

First, in a so-called *prediction step*, the density function $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ is computed:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1}, \tag{2.3}$$

with $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ the transition density related to dynamics function $\mathbf{f}_t$. Then, a *correction step* is applied to compute $p(\mathbf{x}_t|\mathbf{y}_{1:t})$:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1}), \tag{2.4}$$

When $\mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{n}_t^{\mathbf{x}})$ is linear in $\mathbf{x}_{t-1}$ and $\mathbf{n}_t^{\mathbf{x}}$, $\mathbf{h}_t(\mathbf{x}_t, \mathbf{n}_t^{\mathbf{y}})$ is linear in $\mathbf{x}_t$ and $\mathbf{n}_t^{\mathbf{y}}$, and when $\mathbf{n}_t^{\mathbf{x}}$ and $\mathbf{n}_t^{\mathbf{y}}$ follow Gaussian distributions, it can be shown [66] that $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$, $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ and $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ are all Gaussians and can be computed analytically using a Kalman filter. In this case, Equations 2.1 and 2.2 can be written as:

$$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{n}_t^{\mathbf{x}}$$

$$\mathbf{y}_t = \mathbf{H}\mathbf{x}_t + \mathbf{n}_t^{\mathbf{y}}$$

where $\mathbf{F}$ and $\mathbf{H}$ are known matrices, $\mathbf{n}_t^{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$, $\mathbf{n}_t^{\mathbf{y}} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$, $\mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$ and $\mathcal{N}(\mathbf{0}, \mathbf{R}_t)$ are Gaussian densities with mean $\mathbf{0}$ and covariances $\mathbf{Q}_t$ and $\mathbf{R}_t$, respectively. The initial state $\mathbf{x}_0$, the process noise $\mathbf{n}_t^{\mathbf{x}}$ and the observation noise $\mathbf{n}_t^{\mathbf{y}}$ are assumed to be mutually independent.

It can be shown [66] that:

$$p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_{t-1|t-1}, \Sigma_{t-1|t-1})$$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_{t|t-1}, \Sigma_{t|t-1})$$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{x}_{t|t}, \Sigma_{t|t})$$

where:

$$\mathbf{x}_{t|t-1} = \mathbf{F}_t\mathbf{x}_{t-1|t-1}$$

$$\Sigma_{t|t-1} = \mathbf{Q}_t + \mathbf{F}_t\Sigma_{t-1|t-1}\mathbf{F}_t^T$$

$$\mathbf{x}_{t|t} = \mathbf{x}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{H}_t\mathbf{x}_{t|t-1})$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - \mathbf{K}_t\mathbf{H}_t\Sigma_{t|t-1}$$

The problem then consists of estimating the means and covariances of densities. The choice of values for the initial state estimate $\mathbf{x}_{0|0}$ and the initial state covariance $\Sigma_{0|0}$ is based mainly on intuition or on expert's beliefs. In the above equations, $\mathbf{K}_t$ is

the Kalman gain which is given by:

$$\mathbf{K}_t = \Sigma_{t|t-1}\mathbf{H}_t^T(\mathbf{H}_t\Sigma_{t|t-1}\mathbf{H}_t^T + \mathbf{R}_t)$$

The Kalman gain plays a key role in the filter's behavior. A high gain causes the filter to give more importance to the observation while a low gain makes the filter trust more the model prediction and follow it more closely.

Unfortunately, most vision tracking problems involve non linear dynamics functions and non linear, non Gaussian and multimodal likelihood functions. In such cases, tracking methods based on PF [19, 40], also called Sequential Monte Carlo (SMC) methods, can be applied. PF is based on the *sequential importance sampling* technique that is explained in the next section.

2.2.2 Monte Carlo methods

Assume we need to approximate a probability density $p(\mathbf{x})$. If we can sample N independent random variables $\mathbf{x}^{(i)} \sim p(\mathbf{x})$, $i = 1, ..., N$, then an approximation of $p(\mathbf{x})$ can be given by:

$$\hat{p}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta_{\mathbf{x}^{(i)}}(\mathbf{x}),$$

where $\delta_{\mathbf{x}^{(i)}}(\mathbf{x})$ denotes the Dirac delta mass located at $\mathbf{x}^{(i)}$.

One particular application of the Monte Carlo method is for approximating the expectation of any function $f : \mathcal{X} \to \mathbb{R}$, given by:

$$E_p(f) = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x}$$

A Monte Carlo approximation of the above integral can be given by:

$$E_p^{MC}(f) = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}^{(i)})$$

When $p(\mathbf{x})$ is difficult to sample from, one can use *Importance Sampling* (IS) [8] to get a sample set approximating $p(\mathbf{x})$. This method is described next.

2.2.3 Importance Sampling (IS)

Assume we need to approximate a probability density $p(\mathbf{x})$ which is difficult to sample from but easy to evaluate using Monte Carlo simulations. IS consists of drawing

samples from a density $q(\mathbf{x}), \mathbf{x}^{(i)} \sim q(\mathbf{x}), i = 1, ..., N$, where $q(\mathbf{x})$ is called the *proposal density* or *importance density*, and then of estimating $p(\mathbf{x})$ as follows:

$$\hat{p}(\mathbf{x}) = \sum_{i=1}^{N} w^{(i)} \delta_{\mathbf{x}^{(i)}}(\mathbf{x})$$

where $w^{(i)} \propto \frac{p(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})}$ is the normalized importance weights of the ith sample.

It should be noted at this point that IS suffers from the curse of dimensionality. More precisely, in high dimensional problems one needs a large number of samples to achieve a good approximation of the target distribution.

2.2.4 Sequential Importance Sampling (SIS)

If we want to infer a dynamic state whose evolution is characterized by Equations 2.1 and 2.2 by estimating the posterior distribution $p(\mathbf{x}) = p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, then $p(\mathbf{x})$ could be a very complex and high dimensional distribution which cannot be well approximated by Monte Carlo methods like IS. To deal with this problem, one must use *Sequential Importance Sampling* (SIS). SIS approximates the posterior distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ by using the proposal density $q(\mathbf{x})$ of the form $q(\mathbf{x}) = q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. In order to avoid sampling in high-dimensional spaces, a common choice of $q(\mathbf{x})$'s form is the following factorized form:

$$q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}).$$

One then can sample $\mathbf{x}_{0:t}^{(i)}$ from $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ by sequentially sampling $\mathbf{x}_t^{(i)}$ from the proposal density $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})$ at each time t, which greatly reduces the dimensionality of the problem. An estimation of $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ can be given by:

$$\hat{p}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})$$

where $w_t^{(i)}, i = 1, \ldots, N$, are normalized importance weights which are computed using the principle of IS, $w_t^{(i)} \propto \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}, i = 1, ..., N$.

As a corollary, the filtering distribution can be approximated by:

$$\hat{p}(\mathbf{x}_t|\mathbf{y}_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{\mathbf{x}_t^{(i)}}(\mathbf{x}_t) \tag{2.5}$$

2.2.5 Particle filter for visual tracking

We can now establish a theoretical framework of particle filter (PF) for the visual tracking problem. As stated in Section 2.2.1, in tracking problems, one needs to estimate the filtering distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ at each time t (from this point on, we refer to this distribution as the posterior distribution, except when explicitly stated otherwise). In this case, the proposal density should be of the form [29]:

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t)$$

PF [27, 29] approximates the two probability densities in Equations 2.3 and 2.4 by a set of weighted samples, where each sample corresponds to an hypothesized state realization, also called *particle*. Particles with large weights indicate that they are near the modes of the target density while those with low weights indicate that they are near the tails of the target density (see Figure 2-3). Given a particle set $\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}$, $i = 1, \ldots, N$, at time $t-1$ which approximates the probability density $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$, the key idea of PF is to generate new particles $\{\mathbf{x}_t^{(i)}\}$ and to compute their weights $\{w_t^{(i)}\}$, $i = 1, \ldots, N$, by using Equations 2.3 and 2.4 so that the new particle set $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$, $i = 1, \ldots, N$, approximates the probability density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. PF consists of two main steps. First, a *prediction* of the object state (using previous observations) is computed. It consists of sampling a new particle set $\{\mathbf{x}_t^{(i)}\}$, $i = 1, \ldots, N$, according to a proposal density q:

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

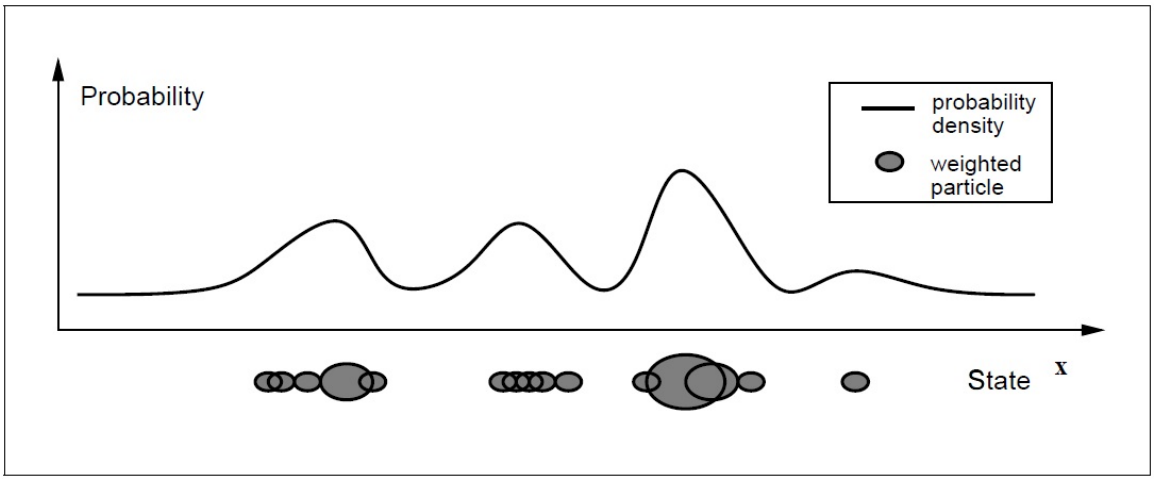


Figure 2-3: **Approximation of a distribution by a particle set.** Particle weights are shown at the bottom, where ellipses of large size indicate particles with high weight (figure reproduced from [62]).

The prediction step is followed by a *correction* of this prediction (using new available observations) by updating the weights of new particles using the principle of IS. It can be shown [5] that:

$$w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{x}_t^{(i)}) \frac{p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)}, \tag{2.6}$$

where $\sum_{i=1}^{N} w_t^{(i)} = 1$, $w_t^{(1)}, \ldots, w_t^{(N)}$ are normalized importance weights of particles, $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$ and $p(\mathbf{y}_t|\mathbf{x}_t^{(i)})$ are the transition density and the likelihood density as mentioned previously. The likelihood measures how well a hypothesized state $\mathbf{x}_t^{(i)}$ matches the true state after its transition. We will discuss more about the likelihood function in Section 2.2.5.3.

An estimation of the posterior distribution is then given by Equation 2.5. The best estimation of the state can be computed in different ways. For example, one can use the estimation of the expectation $E(\mathbf{x}_t|\mathbf{y}_{1:t})$, or the particle $\mathbf{x}_t^{(i)}$ with the highest weight $i = \arg\max_j \{w_t^{(j)}\}_{j=1}^{N}$.

Example of head tracking with PF (continued):

- Transition density: in the simplest setting, one can use a random walk

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}) = \mathcal{N}(\mathbf{x}_{t-1}^{(i)}, \Sigma)$$

where Σ is the maximum deviation of the person's head between two consecutive time slices. This transition density is often used when we have no assumption on the specific kind of movement of the objects.

- Proposal density: the general setting consists of choosing $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$. In such cases, the current weights become proportional to the likelihood.

- Likelihood density: histograms can be used to construct this density (see Section 2.2.5.3).

A common problem of PF (and SIS-based methods) is the *degeneracy* phenomenon [59] where, after a few time slices, all but one particle have weights close to zero. In this situation, only one particle actually contributes to the approximation of the posterior distribution and this often leads to poor approximations. A naive approach to reduce this phenomenon is to increase the number of particles over time to get a sufficient number of particles with substantial weights, but this approach is clearly impractical due to the tremendous increase of computation over time. We thus rely

on other methods to cope with this so-called degeneracy problem, that are discussed next.

2.2.5.1 Resampling and effective sample size

Resampling

The first method to reduce the degeneracy phenomenon is to add a *resampling* step in which particles with highest weights are duplicated and particles with lowest weights are discarded in such a way that the posterior distribution is not altered after resampling (this is known as the unbiasedness property). An estimation of the posterior distribution can then be given as follows:

$$\hat{p}(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta_{\bar{\mathbf{x}}_t^{(i)}}(\mathbf{x}_t), \tag{2.7}$$

where $\bar{\mathbf{x}}_t^{(i)}, i = 1, ..., N$, are the particles resulting from resampling.

One of the first resampling algorithms, *multinomial resampling*, was proposed in [34]. It 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 $\mathcal{S} = \{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$ is substituted by a new sample $\mathcal{S}' = \{\mathbf{x}_t^{(D(k_i))}, \frac{1}{N}\}$ where $D(k_i)$ is the unique integer j such that $\sum_{h=1}^{j-1} w_t^{(h)} < k_i \leq \sum_{h=1}^{j} w_t^{(h)}$. If $(n_1, \ldots, n_N)$ denotes the number of times each of the particles in $\mathcal{S}$ are duplicated, then $(n_1, \ldots, n_N)$ is distributed w.r.t. the multinomial distribution $\text{Mult}(N; w_t^{(1)}, \ldots, w_t^{(N)})$.

Although resampling reduces the effects of the degeneracy phenomenon, it does actually add an extra noise to the variance of the estimator of the posterior distribution [29, 21]. In other words, the variance of the estimator in Equation 2.7 is often higher than that of Equation 2.5. After the introduction of the multinomial resampling algorithm, many other resampling algorithms have been proposed that aim to reduce this extra noise. Next, we only detail the unbiased and most popular resampling algorithms found in the literature, namely *stratified resampling*, *systematic resampling* and *residual resampling*. The *weighted resampling* algorithm [63] will be discussed in Section 3.1.2.

Stratified resampling [50]. The method selects randomly the k_i's w.r.t. the uniform distribution $U((\frac{i-1}{N}, \frac{i}{N}])$. Then, sample $\mathcal{S} = \{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$ is substituted by a new sample $\mathcal{S}' = \{\mathbf{x}_t^{(D(k_i))}, \frac{1}{N}\}$ where $D(k_i)$ is determined as in multinomial resampling.

Systematic resampling [50]. Some number k is drawn w.r.t. $U((0, \frac{1}{N}])$ and the k_i's are defined as $k_i = \frac{i-1}{N} + k$. Then, sample $\mathcal{S} = \{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$ is substituted by a new sample $\mathcal{S}' = \{\mathbf{x}_t^{(D(k_i))}, \frac{1}{N}\}$ where $D(k_i)$ is determined as in multinomial resampling.

Residual resampling [57]. The method 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 $N - \sum_{i=1}^{n} n'_i$ particles still needed to complete the N-sample $\mathcal{S}'$ are drawn randomly using the multinomial distribution $\text{Mult}(N - \sum_{i=1}^{n} n'_i; N w_t^{(1)} - n'_1, \ldots, N w_t^{(N)} - n'_N)$. The weights assigned to all the particles in $\mathcal{S}'$ are $1/N$.

Another practical issue introduced by resampling algorithms, i.e. duplicating particles with highest weights and eliminating particles with lowest weights, is the *sample impoverishment* problem, a situation in which very few different particles have high weights. This causes a loss of diversity among the particles and can lead to tracking failures. That is why we introduce next the *effective sample size*.

Effective sample size

The degeneracy phenomenon of PF has been shown to be unavoidable [29], and the idea of reducing it by using resampling is to adaptively perform resampling whenever a significant degeneracy is observed. A measure named *effective sample size* has been proposed in [51, 58] for this purpose. It is defined as follows:

$$ESS_t = \frac{N}{1 + var_{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}(w_t^*(\mathbf{x}_{0:t}))}$$

where $w_t^*(\mathbf{x}_{0:t}) = \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}$ is referred to as true weight and $var_{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}(w_t^*(\mathbf{x}_{0:t}))$ is the variance of $w_t^*(\mathbf{x}_{0:t})$ that is computed w.r.t. $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$.

By comparing ESS_t with some threshold $ESS_{threshold}$, resampling is performed whenever $ESS_t < ESS_{threshold}$. However, it is impossible to evaluate ESS_t exactly in real problems. Instead, one can use another measure called *survival diagnostic* [64] (also named *estimated effective sample size* [26]) which is easy to compute and is a good estimation of ESS_t. This measure also provides a good way of assessing the quality of PF methods. The survival diagnostic is defined as follows:

$$D_t(N) = \frac{1}{\sum_{i=1}^{N} (w_t^{(i)})^2} \tag{2.8}$$

The intuition behind it is that it corresponds to the number of particle surviving the resampling, those that actually contribute to the discrete representation of the posterior distribution. If N_{min} is the minimum survival diagnostic to ensure an acceptable performance of PF methods, then one must have $D_t(N) > N_{min}$. In other words, if the survival diagnostic is low, the method is supposed to be inefficient.

2.2.5.2 Choice of the proposal density

The degeneracy problem can also be reduced by choosing a good proposal density. In fact, the choice of the proposal density is very crucial in PF methods. One needs to take into account two criteria to design it: the easiness to sample from it and the ability to compute the importance weights given in Equation 2.6. A general setting consists of taking $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$. The drawback of this proposal density is that the states are sampled from the prior distribution, without taking into account the current observation. In the cases where the observation density $p(\mathbf{y}_t|\mathbf{x}_t)$ is sharply peaked in the tails of $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$, this leads to a situation where only few particles will have high weights after the correction step of PF. Those will be duplicated by resampling, resulting in the sample impoverishment problem. It has been shown [29] that the optimal proposal density (in terms of minimizing the variance of importance weights conditioned upon $\mathbf{x}_{t-1}^{(i)}$ and $\mathbf{y}_t$) is $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)$. In this case, the formula of the importance weights in Equation 2.6 becomes:

$$w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{x}_{t-1}^{(i)}) = w_{t-1}^{(i)} \int p(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}) d\mathbf{x}_t \tag{2.9}$$

One thus needs to compute the integral $\int p(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}) d\mathbf{x}_t$ and to sample from $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)$, which is not an easy task, often impossible in visual tracking problems. A good proposal density thus must be close enough to the optimal one while remaining tractable. Many approaches have been proposed that aim to efficiently incorporate the current observations into the proposal density. One example of those is the auxiliary particle filter [76].

2.2.5.3 Likelihood function

In the context of visual tracking, the likelihood function is constructed using some features extracted from images, such as foreground silhouette [80, 7, 47, 112], edge [24, 47, 112], color [75, 61, 56, 65, 6], texture [111, 104, 3] or optical flow [14, 85, 100]. In this section, we briefly introduce some of the features commonly used in articulated object tracking, and chosen for some tests in this thesis. They include foreground silhouette, edge and color.

Foreground silhouette and edge

A popular method that uses foreground silhouette and edge to construct the likelihood function has been proposed in [24]. These features are widely used for visual tracking [80, 7, 47, 112]. Their advantage is that they can be used to track objects

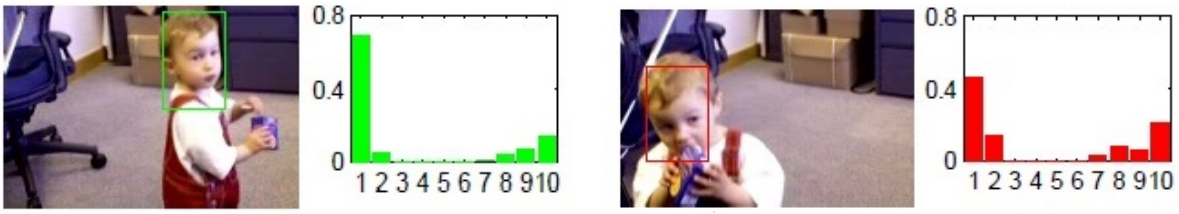
(**a**) (**b**) (**c**)

Figure 2-4: **Likelihood function constructed from foreground silhouette and edge features**. (**a**) Original image. (**b**) Foreground silhouette image. (**c**) Edge image.

whose shape can change with time. In human tracking, the foreground silhouette provides very important information for the 3D reconstruction of the body [2]. These features, however, are often costly to compute compared to color features. Their main limitation is that they do not provide depth information, which is very important when tracking under occlusion. An example is shown in Figure 2-4, where the observation provided by the foreground silhouette image is ambiguous. In this case, it is difficult to localize the correct position of the right leg by only using foreground silhouette.

Color

Color features have also been widely used for visual object tracking [61, 65, 6, 22]. In articulated object tracking, it is useful when dealing with self-occlusion since different body parts usually have different color distributions. Some advantages of this feature are its computational simplicity, robustness under rotations and changes in resolution. One simple method to build the likelihood function using color is to compute the histogram of colors of the region around the hypothesized state and then to compare this histogram with the reference one (which is computed at the first time slice, or updated while tracking). One example is shown in Figure 2-5. The dissimilarity between two histograms can be computed by a variety of distance measures, such as Bhattacharyya distance, Chi-squared distance, *etc.* Histograms can also be computed in different color spaces. Two commonly used color spaces are RGB [99] and HSV



(**a**) (**b**)

Figure 2-5: **Likelihood function constructed from color histogram**. (**a**) Reference histogram computed from the region around the initial position. (**b**) Histogram computed from the region around the hypothesized position (figure reproduced from [75]).

[35]. Each color space has its own limitations. For instance, the RGB color space is not suitable for tracking under lighting change conditions [30], while the HSV color space is more robust to these kinds of lighting changes but sensitive to noise [89].

Algorithm 2.1 gives an overview of PF, which combines all the aforementioned features.

Input: Particle set $\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N}$ at time $t-1$
Output: Particle set $\{\mathbf{x}_{t}^{(i)}, w_{t}^{(i)}\}_{i=1}^{N}$ at time t
Compute ESS_t using Equation 2.8
if $ESS_t < ESS_{threshold}$ **then**
  $\{\mathbf{x}_{t-1}^{(i)}, \frac{1}{N}\}_{i=1}^{N} \leftarrow \text{Resample}\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N}$
for $i = 1$ **to** N **do**
  $\{\mathbf{x}_{t}^{(i)}\}_{i=1}^{N} \leftarrow \text{Propagate}\{\mathbf{x}_{t-1}^{(i)}\}_{i=1}^{N}$ by $\mathbf{x}_{t}^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)$
  Compute the unnormalized particle weight $w_t^{(i)}$ using Equation 2.6
for $i = 1$ **to** N **do**
  Normalize the weight: $w_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}}$
return $\{\mathbf{x}_{t}^{(i)}, w_{t}^{(i)}\}_{i=1}^{N}$

Algorithm 2.1: Particle Filter (PF).

In the particular setting where $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$, all steps of PF can then be represented by the diagram in Figure 2-6. The "$\sim$" represents the resampling step, the "*" represents the application of the object's dynamics (prediction step), the "$\times$" represents the correction step using the observation density.

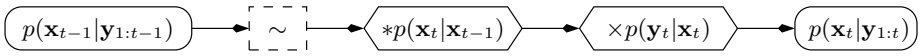


Figure 2-6: **Diagram of PF for visual tracking**.

2.2.6 Particle filter for articulated object tracking

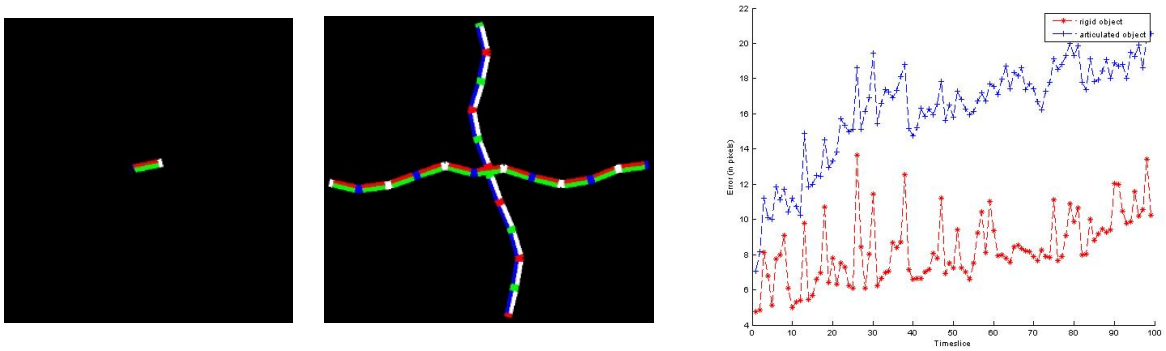
PF suffers from the curse of dimensionality and has usually been applied only for problems in low dimensional state spaces. This makes it difficult to use for articulated object tracking, especially in realistic real-world applications where a high level output is required. Intuitively, the prediction step of PF is based on IS and therefore it inherits its problem from IS. In fact, it can be shown [62] that the number of particles required for tracking grows exponentially with the number of parts of the target object. Let us introduce the *survival rate* [64], given by:

$$\alpha = \Big(\int \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})^2}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} d\mathbf{x}_{0:t} \Big)^{-1}$$

The relation between the survival diagnostic (see Equation 2.8) and the survival rate is $D(N) \approx N\alpha$ when $N \to \infty$.

Consider the problem of tracking an object defined in the state space $\mathcal{X}$. Denote N_{min} the minimum survival diagnostic to ensure an acceptable performance, If N is the number of particles used for this problem, then we must have $N\alpha > N_{min}$, or $N > N_{min}/\alpha$. Now, if we use the same tracker on the Cartesian product of d copies of the configuration space $\mathcal{X}$ (in other words, the same implementation is used to track d objects, each one with the state space $\mathcal{X}$), and N is the number of particles used for this problem so as to have the same performance as in the previous problem, then we have $\alpha = \alpha^d$ and one must have $N > N_{min}/\alpha^d$. In practice, it has been observed that $\alpha \ll 1$, then when d is high, $N \gg N_{min}$. The problem is that in PF-based visual tracking, the evaluation of the likelihood function which is usually time consuming must be done for each particle at each time slice. It is clear that one cannot use PF for high dimensional problems due to the prohibitively high computational cost of performing the likelihood computations for a very large number of particles.

Example. To illustrate the problem of PF in high dimensional state spaces, we conduct an experiment on a synthetic video sequence in which a rigid object and an articulated object moving over time are tracked using PF. The rigid object (see Figure 2-7(a)) is modeled by a rectangle whose state is represented by the coordinates of its center and its orientation. The articulated object consists of a set of 21 rectangles of the same size. For simplicity, we generated the video sequences



(**a**) (**b**) (**c**)

Figure 2-7: **Comparison of PF's errors when tracking in low and high dimensional spaces**. (**a**) and (**b**) respectively the rigid object and the articulated object to be tracked. (**c**) Tracking errors obtained by PF for these 2 objects (in blue: articulated object, in red: rigid object).

so that there is no self-occlusion between parts. We also suppose that the sizes of all rectangles are known in advance and do not change during tracking. This results in a 3-dimensional state space for the rigid object and a 23-dimensional state space for the articulated object (3 degrees of freedom (DOFs) for the center part, i.e. the location and orientation of the part, 1 DOF for each remaining part which just corresponds to the part's orientation). The likelihood functions are constructed from histograms (see the example in Section 2.2.5). Since we suppose there is no occlusion, the likelihood function for the articulated object can be factorized as the product of the likelihood functions for its subparts. In Figure 2-7, the tracking errors obtained by the PF tracker for the two objects are compared. For the rigid object, this error is the sum of the distances between 4 corners of its estimation by PF and those of the ground truth. For the articulated object, this error is the average of the previously described error over all of its parts. In both cases, the tracking error is averaged over 10 runs. The number of particles used for the rigid object and the articulated object is 50 and 2000, respectively. In this experiment, even though the number of particles used for tracking the articulated object is larger, the performance of the PF tracker is worse than that of the rigid object.

Note that in real-world problems, e.g., human body tracking, the state of the articulated object usually has at least 25 DOF [24] and, in the previous example,

all challenging difficulties of the articulated object tracking problem related to the presence of clutter, the occlusion problem, the lighting condition, *etc*, have been ignored for simplicity. We thus only focus on the performance of the sampling step of PF in high dimensional state spaces. This suggests that one needs a very large number of particles for PF to achieve a reasonable performance when tracking articulated objects, which makes PF impractical for real-world applications. A recent study [87] has also shown that the performance in terms of processing time of the state-of-the-art approaches for articulated object tracking (which uses the PF-based framework) is far from the real-time requirement in real-world applications.

Therefore, the need for seeking effective methods that deal with the high dimensional problems in visual tracking is clearly desirable, especially when tracking under the PF framework. As we discussed in this section, one way to reduce the complexity of PF in high dimensional spaces is to reduce the number of particles required for tracking. This is the key idea of many PF-based approaches for articulated object tracking. We will investigate these approaches in the following chapter.

Chapter 3

Algorithms for Articulated Object Tracking

This chapter gives a non-exhaustive review of the methods used for articulated object tracking. These methods can be roughly classified into two categories: generative methods and discriminative methods.

Generative methods (also known as model-based methods) define a model which consists of a set of parameters describing the positions and orientations of parts of the target object (in tree-based models, these parameters describe the position and angle of the root part and the joint angles of the other parts with respect to theirs parent parts in the tree). At each time step, tracking proceeds by fitting the model to features observed in images, using some optimization scheme (stochastic or deterministic optimization) to minimize some cost function which favors the matching of the model with the image features. Generative methods can be further classified into the following categories: decomposition and optimization methods. Decomposition methods exploit the local structure of the articulated object to reduce the complexity of tracking by modeling the articulated object tracking problem as one of inference in a graphical model and then by solving this problem using some inference methods. Optimization methods formulate the articulated object tracking problem as a high dimensional optimization problem and then use some optimization frameworks (for example, particle swarm optimization, gradient descent) in the whole state space of the target object to search for its best estimate. Methods which combine optimization with a hierarchical search to increase searching efficiency have also been proposed. The advantage of generative methods is their generalization: they can be used to track a wide range of motions. The drawbacks of these methods is that their computational cost is high (due to the evaluation of the cost function to be minimized), they often require manual initialization and often cannot recover from tracking failures.

Discriminative methods do not explicitly use a model but construct a mapping from vectors representing image features (e.g. silhouette) to vectors representing the configuration of the articulated object, by learning from a set of training data. This mapping is used to recover the configuration of the articulated object at each time step. In contrast to generative methods, discriminative methods can provide automatic initialization, they require less computational cost and can recover from tracking failures. Their disadvantage is that their application is limited to motions which are similar or close to motions used for training.

It should be noted that many aforementioned generative (or discriminative) methods can be seen as hybrid methods because they use some learning tasks to build some components required by inference or optimization algorithm in generative methods.

This chapter is organized as follows: decomposition and optimization methods are presented in Section 3.1 and Section 3.2, respectively. Discriminative methods are discussed in Section 3.3. Hybrid methods are mentioned in all of these sections, depending on their relations with methods described in the same section.

3.1 Decomposition approaches

A wide range of decomposition methods for articulated object tracking have been proposed in the literature [64, 108, 90, 44, 9, 80, 107]. Their key idea is to decompose the state space of the target object into a set of subspaces where the particle filter or a modified particle filter can be applied. Since the dimension of these subspaces is smaller than that of the original state space, sampling in these subspaces is more efficient than in the original space and, therefore, fewer particles are needed to achieve good performance. This technique, however, must take into account the constraints induced by the articulated structure of the target object in order to obtain a robust solution to the tracking problem. Many of these methods [9, 107] consist of a set of particle filters that interact with each other to obtain a collaborative solution.

Since graphical models allow to decompose a joint probability distribution as a product of conditional probability distributions, they can naturally be used for this kind of methods. For instance, Dynamic Bayesian Networks (DBNs) are used in [108, 80], while Markov Random Fields (MRFs) are used in [90, 44, 9, 107]. Moreover, in articulated object tracking, many structural constraints induced by the articulated structure can be modeled as probabilistic dependencies, which can then be easily incorporated into graphical models for inference. In Sections 3.1.1 to 3.1.4, we present DBNs and some DBNs-based decomposition approaches for articulated object tracking, while in Sections 3.1.5, 3.1.6 and 3.1.7, MRFs and some related decomposition

approaches are discussed.

3.1.1 Dynamic Bayesian Networks (DBNs)

Definition 3-1 (Bayesian network (BN)) *A BN is a pair* $(G, \mathbf{P})$ *where* $G = (\mathbf{V}, \mathbf{A})$ *is a directed acyclic graph (DAG),* $\mathbf{V}$ *and* $\mathbf{A}$ *are a set of nodes and a set of arcs, respectively. Each node* $X \in \mathbf{V}$ *corresponds to a random variable*[1]*.* $\mathbf{P}$ *is the set of the probability distributions of each node* $X \in \mathbf{V}$ *conditionally to its parents* $\mathbf{pa}(X)$ *in* G*, i.e.,* $p(X|\mathbf{pa}(X))$*. The joint probability over all the random variables in* $\mathbf{V}$ *is then the product of all these conditional distributions:*

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

BNs [73] can model the uncertainties inherent to object tracking problems: in this case, $\mathbf{V}$ is the set of state variables $\{\mathbf{x}_t\}$ and observation variables $\{\mathbf{y}_t\}$. The probabilistic dependencies between these variables are then encoded by arcs in the network. Figure 3-1 represents a generic BN designed for object tracking. For articulated objects, the state space $\mathcal{X}$ can be partitioned into $\mathcal{X}^1 \times \cdots \times \mathcal{X}^P$ where each $\mathcal{X}^i, i = 1, ..., P$, is the subspace corresponding to a part of the target object. In this case, instead of having only one node $\mathbf{x}_t$ per "time slice", the BN contains one node $\mathbf{x}_t^i$ per $\mathcal{X}^i$ and per time slice. The probabilistic relationships between these variables are encoded by arcs. DBNs [67], or more precisely 2TBNs, are an extension of BNs specifically designed to cope with time evolving systems:

Definition 3-2 (DBN: 2-slice Temporal Bayesian Network (2TBN)) *A 2TBN is a pair* $(B_1, B_{\rightarrow})$ *of BNs.* B_1 *represents the BN at time slice* $t = 1$ *and represents the joint probability* $p(\mathbf{x}_1, \mathbf{y}_1)$*. The BN* $B_{\rightarrow}$ *defines the transition between different time slices* $p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{x}_{t-1}, \mathbf{y}_{t-1})$*. It is assumed that the BN in each time slice* $t > 1$ *is identical to* $B_{\rightarrow}$*.*

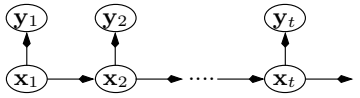


Figure 3-1: **A generic BN for object tracking**.

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

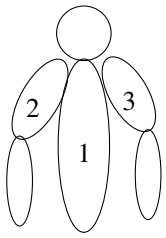


Figure 3-2: **Upper body tracking**. The body parts to be tracked are the torso, the upper left arm and the upper right arm.

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

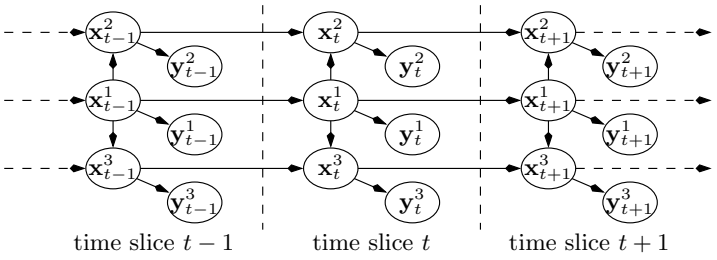


Figure 3-3: **A dynamic Bayesian Network for the body tracking problem of Figure 3-2**.

From a particle filtering perspective, the key feature of BNs and DBNs is their graphical representation of probabilistic independences, which is called the d-separation criterion [73].

Definition 3-3 (d-separation [73]) *Two nodes $\mathbf{x}_t^i$ and $\mathbf{x}_s^j$ of a DBN are dependent conditionally to a set of nodes $\mathbf{Z}$ if and only if there exists a chain $\{\mathbf{c}_1 = \mathbf{x}_t^i, \ldots, \mathbf{c}_n = \mathbf{x}_s^j\}$ linking $\mathbf{x}_t^i$ and $\mathbf{x}_s^j$ in the DBN such that the following two conditions hold:*

1. *for every node $\mathbf{c}_k$ such that the DBN's arcs are $\mathbf{c}_{k-1} \rightarrow \mathbf{c}_k \leftarrow \mathbf{c}_{k+1}$, either $\mathbf{c}_k$ or one of its descendants is in $\mathbf{Z}$;*

2. none of the other nodes $\mathbf{c}_k$ *belongs to* $\mathbf{Z}$.

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

In Figure 3-3, consider the DBN that consists of only 3 time slices $t-1$, t, $t+1$. Assume that $\mathbf{Z}$ is the set of observation nodes. The chain $\mathbf{x}_{t-1}^2 \leftarrow \mathbf{x}_{t-1}^1 \rightarrow \mathbf{x}_t^1 \rightarrow \mathbf{x}_t^3$ is active. Hence, $\mathbf{x}_{t-1}^2$ and $\mathbf{x}_t^3$ are dependent conditionally to $\mathbf{Z}$. If set $\mathbf{Z}$ now includes $\mathbf{x}_t^1$, then this chain is blocked (by $\mathbf{x}_t^1$). If $\{\mathbf{x}_{t-1}^1, \mathbf{x}_t^1, \mathbf{x}_{t+1}^1\} \in \mathbf{Z}$, then the two nodes $\mathbf{x}_{t-1}^2$ and $\mathbf{x}_{t+1}^3$ are d-separated since any chain between them is blocked by one of the three nodes $\mathbf{x}_{t-1}^1, \mathbf{x}_t^1, \mathbf{x}_{t+1}^1$.

Sections 3.1.2, 3.1.3 and 3.1.4 are dedicated to approaches using DBNs for articulated object tracking.

3.1.2 Partitioned Sampling (PS)

Partitioned Sampling (PS) has been introduced by MacCormick [64]. It is an effective PF designed for tracking complex objects with large state space dimensions using only a reduced number of particles. Its key idea is to partition the state space into an appropriate set of subspaces and to apply sequentially PF on each subspace. PS uses a tailored sampling scheme, called "weighted resampling", which ensures that the set of particles resulting from the sequential applications of PF actually represents the joint distribution of the whole state space and are focused on its peaks.

Definition 3-4 (weighted resampling [64]) *Let* $g : \mathcal{X} \mapsto \mathbb{R}$ *be any strictly positive continuous function, where* $\mathcal{X}$ *denotes the state space. Given a set of particles* $\mathcal{S}_t = \{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ *with weights* $w_t^{(i)}$*, weighted resampling proceeds as follows: let* ρ_t *be defined as* $\rho_t(i) = g(\mathbf{x}_t^{(i)}) / \sum_{j=1}^N g(\mathbf{x}_t^{(j)})$ *for* $i = 1, \ldots, N$*. Select independently indices* $k_1, \ldots, k_N$ *according to probability* ρ_t*. Finally, construct a new set of particles* $\mathcal{S}'_t = \{\mathbf{x}'^{(i)}_t, w'^{(i)}_t\}_{i=1}^N$ *defined by* $\mathbf{x}'^{(i)}_t = \mathbf{x}_t^{(k_i)}$ *and* $w'^{(i)}_t = w_t^{(k_i)} / \rho_t(k_i)$.

MacCormick [62] shows that $\mathcal{S}'_t$ represents the same probability distribution as $\mathcal{S}_t$ while focusing the particles on the peaks of g.

PS's key idea is to exploit some natural decomposition of the system dynamics w.r.t. subspaces of the state space in order to apply PF only on those subspaces. This leads to a significant reduction in the number of particles required for tracking. So, assume that state space $\mathcal{X}$ and observation space $\mathcal{Y}$ can be partitioned as $\mathcal{X} = \mathcal{X}^1 \times \cdots \times \mathcal{X}^P$ and $\mathcal{Y} = \mathcal{Y}^1 \times \cdots \times \mathcal{Y}^P$ respectively. For instance, a system representing

a hand could be defined as $\mathcal{X}^{\text{hand}} = \mathcal{X}^{\text{palm}} \times \mathcal{X}^{\text{thumb}} \times \mathcal{X}^{\text{index}} \times \mathcal{X}^{\text{middle}} \times \mathcal{X}^{\text{ring}} \times \mathcal{X}^{\text{little}}$. Assume in addition that the dynamics of the system follows this decomposition, i.e., that:

$$f_t(\mathbf{x}_{t-1}, n_t^{\mathbf{x}}) = f_t^P \circ f_t^{P-1} \circ \cdots \circ f_t^2 \circ f_t^1(\mathbf{x}_{t-1}), \tag{3.1}$$

where $\circ$ is the usual function composition operator and where each function $f_t^i : \mathcal{X} \mapsto \mathcal{X}$ modifies the particles' states only on subspace $\mathcal{X}^i$ [2].

The PF scheme consists of resampling particles, of propagating them using proposal function f_t and, finally, of updating their weights using the observations at hand. Here, the same result can be achieved by substituting the f_t propagation step by the sequence of applications of the f_t^i as given in Equation 3.1, each one followed by a weighted resampling that produces new particles sets focused on the peaks of a function g. To be effective, PS thus needs g to be peaked on the same region as the posterior distribution restricted to $\mathcal{X}^i$. When the likelihood function decomposes as well on subsets $\mathcal{Y}^i$, i.e., when:

$$p(\mathbf{y}_t|\mathbf{x}_t) = \prod_{i=1}^{P} p^i(\mathbf{y}_t^i|\mathbf{x}_t^i), \tag{3.2}$$

where $\mathbf{y}_t^i$ and $\mathbf{x}_t^i$ are the projections of $\mathbf{y}_t$ and $\mathbf{x}_t$ on $\mathcal{Y}^i$ and $\mathcal{X}^i$ respectively, weighted resampling focusing on the peaks of the posterior distribution on $\mathcal{X}^i$ can be achieved by first multiplying the particles' weights by $p^i(\mathbf{y}_t^i|\mathbf{x}_t^i)$ and, then, by performing a usual resampling. Note that Equation 3.2 naturally arises when tracking articulated objects. This leads to the condensation diagram given in Figure 3-4, where operations "$*f_t^i$" refer to propagations of particles using proposal function f_t^i as defined above, "$\times p_t^i$" refers to the correction steps where particle weights are multiplied by $p^i(\mathbf{y}_t^i|\mathbf{x}_t^i)$ (see Equation 3.2), and "$\sim$" refers to usual resamplings. MacCormick and Isard showed that this diagram produces mathematically correct results [64].

For instance, conditionally to states $\mathbf{x}_t^i$, observations $\mathbf{y}_t^i$ of Figure 3-3 are independent of the other random variables. Consequently, Equation 3.2 implicitly holds in this DBN. In addition, by Definition 3-3, $\mathbf{x}_t^1$ is independent of $\mathbf{x}_{t-1}^2$ and $\mathbf{x}_{t-1}^3$ conditionally to $\{\mathbf{x}_{t-1}^1\}$. Similarly, $\mathbf{x}_t^2$ is independent of $\mathbf{x}_{t-1}^3$ conditionally to $\{\mathbf{x}_t^1, \mathbf{x}_{t-1}^2\}$ and $\mathbf{x}_t^3$ is independent of $\mathbf{x}_t^2$ conditionally to $\{\mathbf{x}_t^1, \mathbf{x}_{t-1}^3\}$. As a consequence, the condensation diagram of Figure 3-4 can be exploited to track this object since the "probabilistic" propagations/corrections of each part of the object only depend on this part and its parents in the DBN. Therefore, by their d-separation property, DBNs provide a sound

[2] Note that, in [62], functions f_t^i are more general since they can modify states on $\mathcal{X}^i \times \cdots \times \mathcal{X}^P$. However, in practice, particles are often propagated only one $\mathcal{X}^j$ at a time.

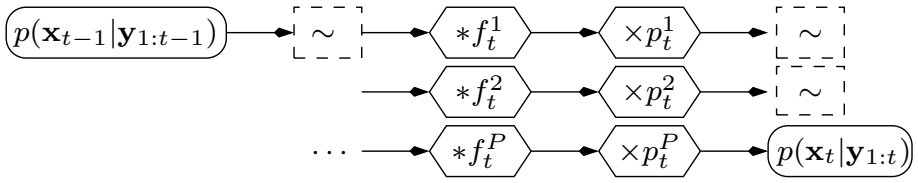


Figure 3-4: **Partitioned Sampling condensation diagram**: PS starts with a particle set estimating $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$. It first propagates particles using proposal function f_t^1 (applied on $\mathcal{X}^1$), then it corrects them using function p_t^1 and resamples them. Second, it propagates and corrects the second part of the resulting particle set (over $\mathcal{X}^2$) using f_t^2 and p_t^2 respectively, and resamples the result. And so on. After iterating over the P parts of the object, the particle set estimates $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

mathematical framework for proving the correctness of PS.

We now formalize PS in terms of operations over DBNs. For this purpose, for any set $J = \{j_1, \ldots, j_k\}$, let $\mathbf{x}_t^J$ denote the tuple $(\mathbf{x}_t^{j_1}, \ldots, \mathbf{x}_t^{j_k})$, i.e., the tuple of the states of the object parts in J. For instance, on Figure 3-5, if $J = \{2, 3\}$, then $\mathbf{x}_t^J$ represents the state of the whole left arm. Similarly, let $\mathbf{x}_t^{(i),J}$ denote the tuple of the parts in J of the ith particle. For instance, for $J = \{2, 3\}$, $\mathbf{x}_t^{(i),J}$ corresponds the state of left arm as represented by the ith particle. In the rest of this thesis, we will assume that the object is composed of precisely P parts (in Figure 3-5, $P = 6$). Now, we shall describe a slight generalization of PS where PF is iteratively applied on sets of object parts instead of just singletons like PS does. When PF is applied on a set, it is applied independently (in parallel) on all its elements. We need to distinguish at each step of such tracking algorithm the parts that were already processed by PF from those that are not yet. Thus, for any step j,

- let P_j denote the set of object parts being processed at the jth step (in the case of PS, $P_j = \{j\}$);
- let $Q_j = \sum_{h=1}^{j} P_h$ denote the set of all the object parts processed up to (including) the jth step;

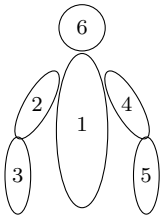


Figure 3-5: **Example of an articulated object modeling the upper part of the human body.**

- let $R_j = \sum_{h=j+1}^{P} P_h$ denote the set of the object parts yet to process after the jth step is completed.

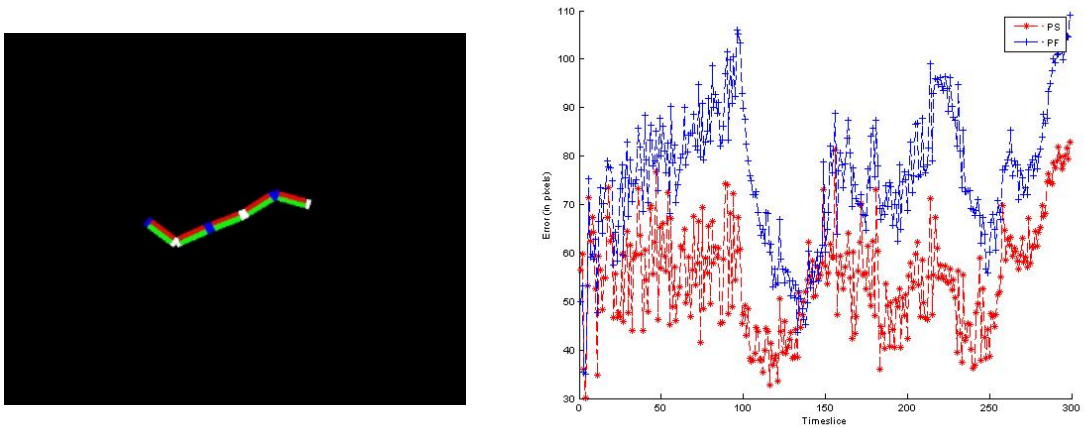
Figure 3-5 illustrates these notations: here, $P_1 = \{1\}$, i.e., PF is first applied only on the torso; $P_2 = \{2, 4, 6\}$, i.e., at its second step, the tracking algorithm applies PF in parallel on parts 2, 4 and 6 (which are supposed to be independent given the position of the torso). Therefore, at the second step, parts $Q_2 = \{1, 2, 4, 6\}$ have been processed and there remains to process parts $R_2 = \{3, 5\}$. Thus, if PF has propagated all the parts in Q_2 from time $t-1$ to t, in the particles, the parts in R_2 still refer to time $t-1$. Let K denote the number of steps of the tracking algorithm, i.e., the number of sets P_j (for PS, $K = P$). PS now can be described in Algorithm 3.1.

Input: Particle set $\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}$ at time $t-1$, image $\mathcal{I}$
Output: Particle set $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$ at time t

$Q \leftarrow \emptyset$; $R \leftarrow \{1, \ldots, P\}$
for $j = 1$ **to** K **do**
  foreach k **in** P_j **do**
    $Q' \leftarrow Q \cup \{k\}$; $R' \leftarrow R \backslash \{k\}$
    $\{(\mathbf{x}_t^{(i),Q'}, \mathbf{x}_{t-1}^{(i),R'})\} \leftarrow$ propagate the kth part in $\{(\mathbf{x}_t^{(i),Q}, \mathbf{x}_{t-1}^{(i),R})\}$
    $\{(w_t^{(i),Q'}, w_{t-1}^{(i),R'})\} \leftarrow$
      correct the kth part in $(\{(\mathbf{x}_t^{(i),Q'}, \mathbf{x}_{t-1}^{(i),R'}), (w_t^{(i),Q}, w_{t-1}^{(i),R})\}, \mathcal{I})$
    $Q \leftarrow Q'$; $R \leftarrow R'$
  $\{(\mathbf{x}_t^{(i),Q}, \mathbf{x}_{t-1}^{(i),R}), (w_t^{(i),Q}, w_{t-1}^{(i),R})\} \leftarrow$
    resample $(\{(\mathbf{x}_t^{(i),Q}, \mathbf{x}_{t-1}^{(i),R}), (w_t^{(i),Q}, w_{t-1}^{(i),R})\})$
return $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$

Algorithm 3.1: Partitioned Sampling PS.

Example: Figure 3-6 shows a comparison of PS's and PF's performance when tracking an articulated object in a synthetic video sequence. PS starts tracking the object from its center part, then the two neighbor parts of this part and so on. In both cases, the number of particles used for tracking is 50. As can be observed, by processing parts iteratively, i.e., by tracking iteratively in small subspaces, PS is much more accurate than PF.



(**a**) (**b**)

Figure 3-6: **PS versus PF performances**: (**a**) Tracked articulated object. (**b**) Tracking errors (mean in pixels over 10 runs) obtained by PS (in red) and PF (in blue). Note that PS significantly outperforms PF.

3.1.3 Partitioned Sampling plus Interval Particle Filter (PSIPF)

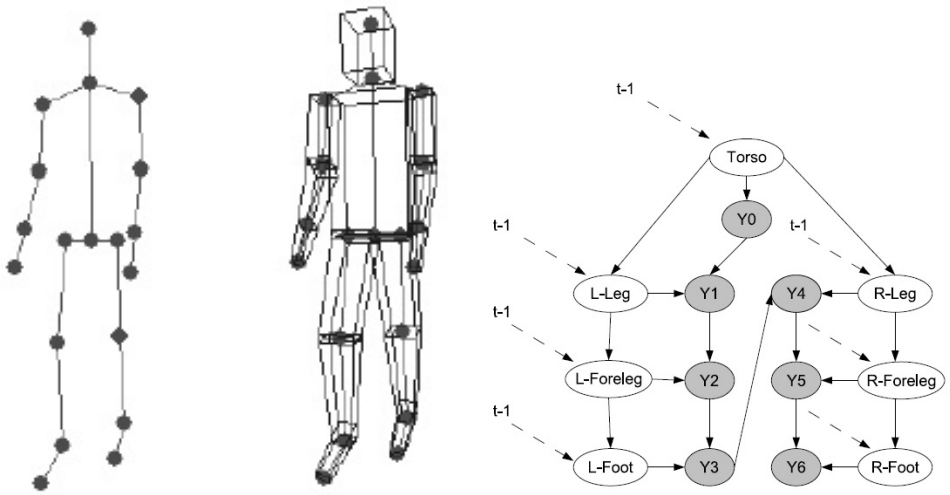
An approach similar to PS has been introduced in [80]. It uses a factorized representation of the model and the likelihood function to reduce the number of particles required for tracking. To improve the tracking accuracy and deal with the problem of occlusion, a deterministic sampling technique has been used which generates more particles around regions of high likelihood and ensures that the tracking does not diverge in case of ambiguous observations.

A BN is used to represent the articulated object which consists of a set of joints and segments (see Figure 3-7(a)). The segments of the model are estimated in a hierarchical way: the first segment is projected into the current image and then compared to the silhouette. The common pixel between the silhouette and the projection of this segment is marked as mask and is used as reference for the estimation of the next segment following the hierarchical order. In this way, the likelihood function can be decomposed as follows:

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

where $pred(\mathbf{y}_t^i)$ is the observation that precedes $\mathbf{y}_t^i$ in the evaluation process (see Figure 3-7(b)).

In the prediction step, the proposed approach uses the deterministic sampling technique introduced in [82]. Assume that the state space $\mathcal{X}$ of the target object can



(**a**) (**b**)

Figure 3-7: **Human tracking with PSIPF:** (**a**) The body model. (**b**) The DBN for the lower parts of the body, that shows how the model and the likelihood function can be factorized (figure reproduced from [80]).

be divided into two subspaces $\mathcal{X} = \mathcal{X}^L \times \mathcal{X}^R$, where $\mathcal{X}^L$ is the subspace of interest. Denote d_L the dimension of $\mathcal{X}^L$. We attempt to derive a weighted particle set at time slice t from the one at time slice $t-1$. First, an interval is defined for each subspace of $\mathcal{X}^L$ which depends on the values of the particles on that subspace at time slice $t-1$ and the maximum angular velocity of joints in human motion. This interval is then discretized into q values. The particle set for time slice t is generated by sampling the subspaces $\mathcal{X}^L$ and $\mathcal{X}^R$ in two different ways: the subspace $\mathcal{X}^L$ is sampled by taking all possible combinations of values from d_L intervals (resulting in q^{d_L} vectors), the subspace $\mathcal{X}^R$ is sampled from a Gaussian distribution (whose standard deviation also depends on the joint speed limits of human motion) centered on the particles on $\mathcal{X}^R$ at time slice $t-1$. In the resampling step, only a fixed number of distinct particles with highest weights is kept for the next time slice. The proposed approach was shown to be effective in reducing the tracking error and in recovering from ambiguous situations (e.g., occlusion between limbs).

3.1.4 Rao-Blackwellized Particle Filter (RBPF)

RBPF was introduced in [28] as an approximate inference method in DBNs. Considering the problem of estimating the posterior distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. The method is

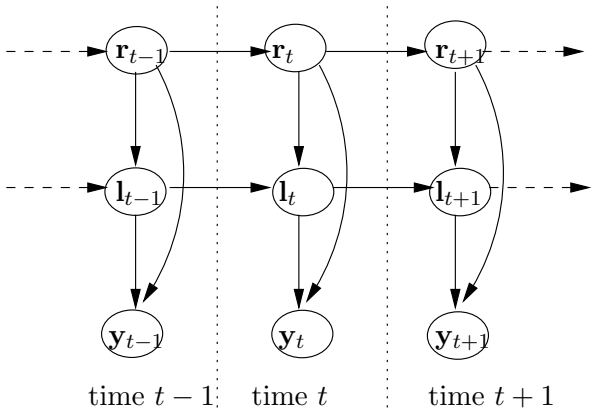


Figure 3-8: **RBPF for DBN**: $\mathbf{r}_t$, $\mathbf{l}_t$ and $\mathbf{y}_t$ represent the root variables, leaf variables and observable variables at time t, respectively. Given the value of $\mathbf{r}_{0:t}$, the distribution $p(\mathbf{l}_{0:t}|\mathbf{r}_{0:t}, \mathbf{y}_{1:t})$ can be computed exactly using a Kalman filter.

motivated by the following decomposition:

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = p(\mathbf{l}_{0:t}|\mathbf{r}_{0:t}, \mathbf{y}_{1:t})p(\mathbf{r}_{0:t}|\mathbf{y}_{1:t})$$

where $\mathbf{x}_t$ is partitioned into two sets $\mathbf{r}_t$ (root variables) and $\mathbf{l}_t$ (leaf variables) (see Figure 3-8).

Assume that the distribution $p(\mathbf{l}_{0:t}|\mathbf{r}_{0:t}, \mathbf{y}_{1:t})$ can be calculated exactly. Then, an approximation of the distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ can be obtained via an approximation of the distribution $p(\mathbf{r}_{0:t}|\mathbf{y}_{1:t})$, which can be done using Monte Carlo methods. The filtering distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t}) = p(\mathbf{r}_t, \mathbf{l}_t|\mathbf{y}_{1:t})$ can be obtained as a corollary. Since the dimension of $p(\mathbf{r}_{0:t}|\mathbf{y}_{1:t})$ is smaller than that of $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, the sampling step for the former needs fewer particles and the variance of the estimation can be reduced [21]. This method, however, cannot be applied for inference in general DBNs as the required assumptions do not hold in this case.

In [108], RBPF has been exploited to track human body in 3D environments. The 25-dimensional state $\mathbf{x}_t$ of the target object is partitioned into two substates: the substate $\mathbf{r}_t$ consists of the global position and orientation, the head orientation and the right-side poses (right hand, right leg), the substate $\mathbf{l}_t$ consists of the left-side poses (left hand, left leg). Using Partial Least Square (PLS) regression [81], a model (linear function) is learnt to describe the correlation between the left-side poses and the right-side poses. The basic assumption is that there exists a linear/Gaussian relationship between the left-side pose and the right-side pose conditional on the learnt model. Once the model has been learnt, the following steps are performed at each time slice. First, the right-side pose is propagated using a dynamic model. Then,

the left-side pose is predicted from the previously propagated right-side pose and the learnt model, using a Kalman filter. Next, a correction step and a resampling step for the particle set in which each particle represents the entire body pose are performed as in PF. Finally, the left-side pose is corrected using its auxiliary observation, which is approximated by its estimate at the previous time slice $\mathbf{l}_{t-1}$. This approach has some limitations, however. First, it is based on the assumption that the left-side pose and the right-side pose are highly correlated during tracking, which restricts its application in many scenarios, since this assumption only holds in cyclic motion activities like human walking or jogging. Second, the model is learnt to capture the linear relationships between the left-side pose and the right-side pose, which is not good enough to model the non-linear relationship in motion data.

In the next subsections, we discuss the Markov Random Fields (MRFs), an undirected graphical model which is also widely used for articulated object tracking. We restrict our attention to a particular class of MRFs, known as pairwise MRFs. This class of MRFs is useful for applications in computer vision like human tracking since inference is computationally efficient.

3.1.5 Markov Random Fields (MRFs)

Definition 3-5 (Pairwise Markov Random Field (MRF)) *A pairwise MRF is defined by a pair* $(G, \mathbf{P})$ *where* $G = (\mathbf{V}, \mathbf{E})$ *is an undirected graph,* $\mathbf{V}$ *and* $\mathbf{E}$ *are a set of nodes and a set of edges, respectively. Each node corresponds to a random variable*[3]*. The set of nodes* $\mathbf{V}$ *is partitioned into two sets* $\mathbf{V_X} = \{\mathbf{x}_t^i\}$ *and* $\mathbf{V_Y} = \{\mathbf{y}_t^i\}$*, corresponding to the set of hidden and observable variables respectively. The set of edges* $\mathbf{E}$ *is also partitioned into two sets* $\mathbf{E_X} = \{(\mathbf{x}_s^i, \mathbf{x}_t^j)\}$ *and* $\mathbf{E_Y} = \{(\mathbf{x}_t^i, \mathbf{y}_t^i)\}$*, which correspond to the edges linking two hidden variables and linking one hidden node and its observable node respectively. Finally,* $\mathbf{P}$ *is a set of nonnegative functions defined as* $\mathbf{P} = \{\psi_{st}^{ij}(\mathbf{x}_s^i, \mathbf{x}_t^j) : (\mathbf{x}_s^i, \mathbf{x}_t^j) \in \mathbf{E_X}\} \cup \{\phi_t^i(\mathbf{x}_t^i, \mathbf{y}_t^i) \equiv \phi_t^i(\mathbf{x}_t^i) : \mathbf{x}_t^i \in \mathbf{V_X}\}$*. Pairwise functions* ψ_{st}^{ij} *are called compatibility functions or potential functions whereas unary functions* ϕ_t^i *are called likelihood functions.*

MRFs encode the joint probability over all the random variables in $\mathbf{V}$ *as follows:*

$$p(\mathbf{V}) = \frac{1}{Z} \prod_{(\mathbf{x}_s^i, \mathbf{x}_t^j) \in \mathbf{E_X}} \psi_{st}^{ij}(\mathbf{x}_s^i, \mathbf{x}_t^j) \prod_{\mathbf{x}_t^i \in \mathbf{V_X}} \phi_t^i(\mathbf{x}_t^i), \tag{3.3}$$

where Z *is a normalizing constant that ensures that* $p(\mathbf{V})$ *integrates to 1.*

[3]As for BNs, by abuse of notation, we will use interchangeably $X \in \mathbf{V}$ (resp. $\mathbf{V}$) to denote a node in the network (resp. all the nodes) and its corresponding random variable (resp. all the random variables).

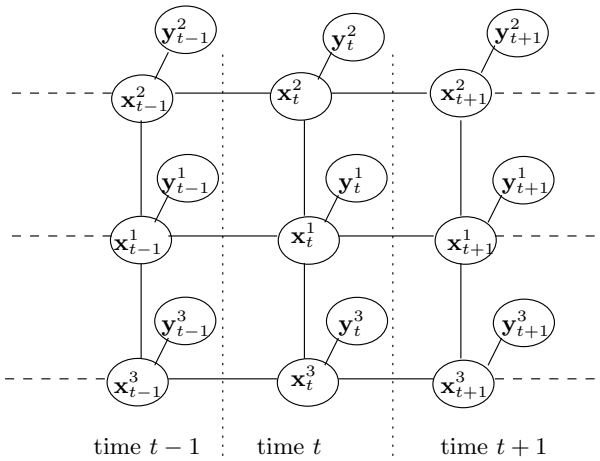


Figure 3-9: **A pairwise Markov Random Field for the body tracking problem of Figure 3-2** .

Like BNs, pairwise MRFs are often used to model the uncertainties in object tracking. In this case, $\mathbf{V_X}$ is the set of state variables $\{\mathbf{x}_t\}$ and $\mathbf{V_Y}$ is that of the observation variables $\{\mathbf{y}_t\}$. The MRF thus encodes the probabilistic dependencies/independencies between all the random variables in $\mathbf{V}$. When dealing with articulated object tracking, each node of the MRF corresponds to a part of the target object. Figure 3-9 shows an example of the MRF used to model the upper body of Figure 3-2, where $\mathbf{x}^1_t, \mathbf{x}^2_t, \mathbf{x}^3_t$ represent the torso, the left arm the right arm respectively and $\mathbf{y}^1_t, \mathbf{y}^2_t, \mathbf{y}^3_t$ represent the observation on these nodes. Intuitively, the graph in one time slice of Figure 3-9 expresses the fact that conditionally to the configuration of the torso, the position and orientation of arm are statistically independent. The edges between nodes in two consecutive time slices, e.g. $\mathbf{x}^1_{t-1}$ and $\mathbf{x}^1_t$ express the temporal coherences in the movements of body parts to be tracked. Pairwise potential functions capture the constraints between different pairs of parts of the articulated object, such as spatial constraints [46], temporal constraints [88], appearance symmetry (i.e. similar appearance between arms or legs) [79, 97], kinematic constraints [97], occlusion [46]. Note that higher order potential functions can also be used, which can express more sophisticated constraints between different groups of parts, but in this case, the inference algorithm in the MRF is more computationally expensive. Note also that, unlike BNs, the MRF's potential functions need not be conditional probabilities: they just need to be nonnegative functions.

Approaches that use pairwise MRFs for articulated object tracking can be roughly categorized into two classes: those which compute the marginal distributions (node's belief in the context of Belief propagation or Loopy Belief propagation) for all the nodes in the MRF and those which compute the maximum a posteriori (MAP) solu-

tion of the MRF. In MAP problems, the aim is to estimate the instantiation $\mathbf{x}$ of the nodes of $\mathbf{V_X}$ that maximizes the posterior distribution $p(\mathbf{x}|\mathbf{y})$ given some observations $\mathbf{y}$ over the variables in $\mathbf{V_Y}$. In the next section, we investigate some approaches in the first class. We first present the main ideas of the popular method for inference in MRFs, known as Belief Propagation (BP) [73], which has been widely used for articulated object tracking. We then discuss some approaches that use BP or a similar principle for articulated object tracking. In Section 3.1.7, we review some approaches in the second class that use MAP inference for articulated object tracking.

3.1.6 Loopy Belief Propagation (LBP)

Belief Propagation (BP) and its approximation scheme, Loopy Belief Propagation (LBP), introduce auxiliary variables $m_{st}^{ij}(\mathbf{x}_t^j)$ that can be intuitively understood as messages from some hidden node $\mathbf{x}_s^i$ to one of its neighbor nodes $\mathbf{x}_t^j$ in $\mathbf{E_X}$. These messages indicate which distribution state node $\mathbf{x}_t^j$ should follow. The key idea of BP and LBP is to send such messages along the MRF in such a way that the marginal posterior distributions of each hidden variable $\mathbf{x}_t^i$ can be computed locally (i.e., by considering only its neighborhood in the network). The difference between BP and LBP is that BP can only be used in acyclic networks and, in this case, it provides an exact computation of the posterior distributions. LBP, on the other hand, is an approximation scheme used in cyclic networks. Basically, it consists of iterating BP until some convergence criterion is met.

Let $\mathbf{N}(\mathbf{x}_s^i)$ denote the set of hidden neighbors of $\mathbf{x}_s^i$, i.e., $\mathbf{N}(\mathbf{x}_s^i) = \{\mathbf{x}_t^j : (\mathbf{x}_s^i, \mathbf{x}_t^j) \in \mathbf{E_X}\}$. Then, in BP and LBP, messages are computed iteratively using the equation below:

$$m_{st}^{ij}(\mathbf{x}_t^j) \propto \sum_{\mathbf{x}_s^i} \psi_{st}^{ij}(\mathbf{x}_s^i, \mathbf{x}_t^j)\ \phi_s^i(\mathbf{x}_s^i) \prod_{\mathbf{x}_r^k \in \mathbf{N}(\mathbf{x}_s^i)\setminus\{\mathbf{x}_t^j\}} m_{rs}^{ki}(\mathbf{x}_s^i). \tag{3.4}$$

The marginal distribution of $\mathbf{x}_s^i$ (the node's *belief*) is estimated by:

$$p(\mathbf{x}_s^i) \propto \phi_s^i(\mathbf{x}_s^i) \prod_{\mathbf{x}_t^j \in \mathbf{N}(\mathbf{x}_s^i)} m_{ts}^{ji}(\mathbf{x}_s^i). \tag{3.5}$$

Messages m_{st}^{ij} are first initialized as vectors filled with ones. Then, they are updated iteratively until convergence.

Example. Consider the MRF of Figure 3-10. Using Equation 3.5, the belief at

node $\mathbf{x}_t^1$ is given by:

$$p(\mathbf{x}_t^1) \propto \phi_t^1(\mathbf{x}_t^1) m_{tt}^{21}(\mathbf{x}_t^1)$$

Using Equation 3.4 for $m_{tt}^{21}(\mathbf{x}_t^1)$, we have:

$$p(\mathbf{x}_t^1) \propto \phi_t^1(\mathbf{x}_t^1) \sum_{\mathbf{x}_t^2} \psi_{tt}^{21}(\mathbf{x}_t^2, \mathbf{x}_t^1) \phi_t^2(\mathbf{x}_t^2) m_{tt}^{32}(\mathbf{x}_t^2)$$

Using again Equation 3.4 for $m_{tt}^{32}(\mathbf{x}_t^2)$, we have:

$$\begin{aligned} p(\mathbf{x}_t^1) &\propto \phi_t^1(\mathbf{x}_t^1) \sum_{\mathbf{x}_t^2} \psi_{tt}^{21}(\mathbf{x}_t^2, \mathbf{x}_t^1) \phi_t^2(\mathbf{x}_t^2) \sum_{\mathbf{x}_t^3} \psi_{tt}^{32}(\mathbf{x}_t^3, \mathbf{x}_t^2) \phi_t^3(\mathbf{x}_t^3) \\ &\propto \sum_{\mathbf{x}_t^2, \mathbf{x}_t^3} \phi_t^1(\mathbf{x}_t^1) \phi_t^2(\mathbf{x}_t^2) \phi_t^3(\mathbf{x}_t^3) \psi_{tt}^{21}(\mathbf{x}_t^2, \mathbf{x}_t^1) \psi_{tt}^{32}(\mathbf{x}_t^3, \mathbf{x}_t^2) \\ &\propto \sum_{\mathbf{x}_t^2, \mathbf{x}_t^3} p(\mathbf{x}_t^1, \mathbf{x}_t^2, \mathbf{x}_t^3) = p(\mathbf{x}_t^1) \end{aligned}$$

The term on the right-hand side of the final equation is, up to a proportional constant, exactly the marginal distribution of $\mathbf{x}_t^1$ which needs to be computed.

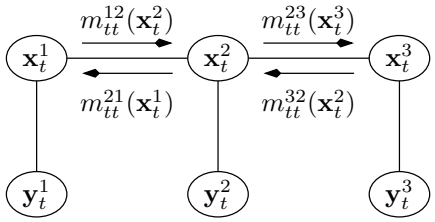


Figure 3-10: **Message passing in LBP**.

LBP is described in Algorithm 3.2. In acyclic graphs, LBP is guaranteed to converge to a fix point in one pass (i.e., each message m_{st}^{ij} is computed only once). In this case, it is precisely equal to BP and it computes exactly the marginal distributions of each node $\mathbf{x}_t^i$. In graphs with cycles, there is no guarantee that the algorithm converges, and, even if it does converge, there is no guarantee that the probabilities obtained are correct [110]. The stopping criterion is usually either a maximum number of iterations or a threshold on the change of marginal distributions (or both).

BP and LBP have been successfully applied for articulated object tracking problems. Ramanan [77] have learnt an appearance model for each body part and incorporated this model into potential functions of the MRF modeling the human body. Andriluka et al. [4] used a similar approach, where a kinematic prior and an ap-

Input: MRF $(G, \mathbf{P})$
Output: the marginal distributions of each hidden node
foreach *edge* $(\mathbf{x}_s^i, \mathbf{x}_t^j) \in \mathbf{E_X}$ **do**
    create messages m_{st}^{ij} and m_{ts}^{ji} filled with ones
Let $\mathbf{M}$ be the set of all messages m_{st}^{ij} and m_{ts}^{ji}
repeat
    foreach *message* $m_{st}^{ij} \in \mathbf{M}$ **do**
        Mark m_{st}^{ij} as not updated yet
    foreach *unmarked message* $m_{st}^{ij} \in \mathbf{M}$ **do**
        update $m_{st}^{ij}(\mathbf{x}_t^j)$ using Equation 3.4
        mark m_{st}^{ij} as updated
    foreach *node* $\mathbf{x}_s^i \in \mathbf{V_X}$ **do**
        compute $p(\mathbf{x}_s^i)$ using Equation 3.5
until *Stopping criterion is satisfied* ;
return $\{p(\mathbf{x}_s^i) : \mathbf{x}_s^i \in \mathbf{V_X}\}$

Algorithm 3.2: Loopy Belief Propagation (LBP).

pearance models for body parts were learnt and then used as potential functions for inference with BP. Shen et al. [83] used probabilistic variational analysis for BP's equations (3.4 and 3.5). The discretization of state vectors allows them to perform BP and mean field (MF) algorithm on DBNs. In [71], a combination of BP and mean shift (MS) was exploited for tracking multiple objects and articulated objects. At each step, the proposed algorithm starts with a set of predicted states for each node. Then a grid of samples is generated, centered on these predicted states. Using BP, the weights of these samples are computed and then used in a MS procedure to move them to new locations near the modes, considered as the new predicted state for each node. The proposed approach considerably reduces the computational cost.

One limitation of BP and LBP is that they cannot be applied to computer vision problems where the nodes of the MRFs are continuous. Thus, a variant of LBP for continuous MRFs has been introduced in [44, 90], which is known as Particle Message Passing (PAMPAS) or Nonparametric Belief Propagation (NBP). PAMPAS alleviates the above problem by combining the ideas of LPB and PF, in which each message and node belief are approximated by a particle set. At each update of a message between two nodes, a new particle set for this message is sampled from some proposal densities which is constructed by taking into account all messages associated with the neighbor nodes. The assumptions required to make the computation feasible are such that each message is represented by a mixture of Gaussians with a reasonably

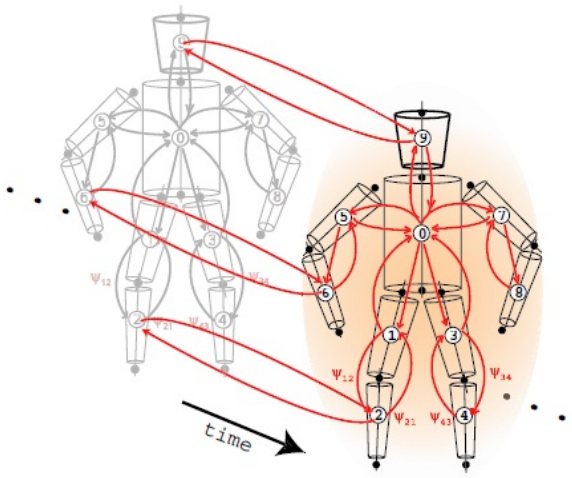


Figure 3-11: **Graphical model for human tracking by PAMPAS** (figure reproduced from [88]).

small number of mixture components. The potential functions are also assumed to be mixtures of Gaussians. The message updating process is thus equivalent to updating the parameters of these mixtures until some stopping conditions are met. The major disadvantage of this algorithm is the high complexity of the sampling algorithm for a product of mixture of Gaussians: if we have D mixtures of T components, then the complexity of the sampling algorithm is $O(T^D)$. One could get around this problem by using a Gibbs sampler [90], in this case the complexity of the sampling algorithm is $O(MTD)$, where M is the number of iterations of the sampler.

PAMPAS has been exploited in [88] for human tracking. Figure 3-11 shows the human body model used by this approach, where each body part has an associated state vector defining its position and orientation. For each joint connecting two neighbor parts, two conditional densities are learnt independently from human motion data. These densities are then used as potential functions in PAMPAS. To provide automatic initialization and failure recovery, a proportion of particle sets generated during the message updating process are drawn from proposal densities obtained from body part detectors. At each time slice, the prior information about the location and pose of body parts provided by body part detectors is given to the tracker. These information are encoded as local priors at nodes of the MRF and, then, are used in the message updating process to assemble the full body. The tracking result from some iterations of PAMPAS is illustrated in Figure 3-12.

A similar approach for articulated body tracking has been introduced in [42]. To reduce the computational cost of the Gibbs sampler in PAMPAS, the proposed approach uses mode propagation and kernel fitting. This results in an algorithm whose

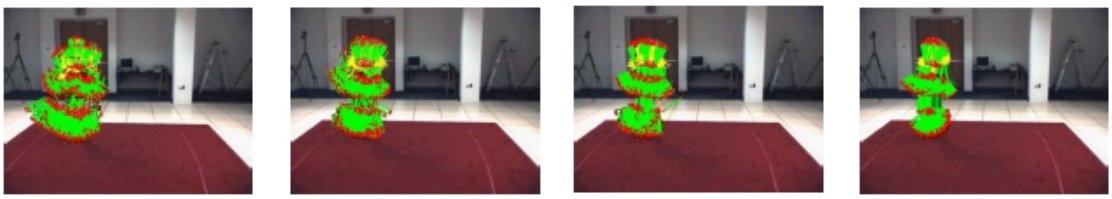


Figure 3-12: **Human tracking by PAMPAS**. Samples for each body part are drawn from the marginal distributions computed by PAMPAS for the first four iterations (figure reproduced from [86]).

complexity is $O(T^4(D-1))$. In [91], Sudderth et al. used PAMPAS to implement a hand tracker. However, differently from the work in [88] which learns potential functions from human motion data, the proposed approach encodes all local constraints into potential functions. In [92], PAMPAS was adapted to handle occlusions for hand tracking. By augmenting the configuration of each node of the MRF with a set of binary hidden variables and introducing potential functions which encode all of the occlusion relationships between related nodes, the modified algorithm can track hand configuration under occlusions robustly. The work of Wu et al. [107] has the same spirit as PAMPAS, but use probabilistic variational analysis instead of BP to derive a closed-form analytic solution for the tracking problem and then use mean field (MF) iterations to obtain an approximate estimation. One limitation of PAMPAS is that at each update of a message, a new particle set must be generated. This makes it computationally expensive due to the evaluations of the likelihood function for new particles. A fast nonparametric belief propagation has been introduced in [9] to reduce the need for these repeated evaluations. Drawing from the same ideas as PAMPAS, the proposed approach has some major differences. First, it uses a particle set for directly representing the node's belief. Second, only one particle set is generated for each node's belief. Once the particles of this set have been sampled, their weights are recomputed through several iterations to better approximate the node's belief by taking into account links between neighbor nodes. Applied to upper body tracking with stereo and color images, the resulting algorithm achieves quasi real-time performance.

3.1.7 Maximum a posteriori (MAP) inference based approaches

Some approaches have formulated the articulated object tracking problem as one of MAP inference on MRF and then solved this problem using exact or approximate inference methods. Branch-and-Bound (BB) algorithms have been proposed to find the MAP solution in pairwise MRFs for human tracking [93, 97]. In [93] the MRF

model is relaxed into a mixture of star-models, where each star-model consists of a node and its neighbors in the original MRF. The original MAP inference problem is then solved by using a BB algorithm on each star-model to find the MAP solutions of these models. The branching strategy of the proposed algorithm was based on the evaluation of the likelihood of hypothesis space. Spaces which are unlikely to contain the MAP solution are ignored to increase searching efficiency. Tian et al. [97] also proposed an efficient BB algorithm which only takes constant time to evaluate the bounds to search for the MAP solution in a tree model (the structure of the MRF is a tree).

3.2 Optimization-based approaches

Optimization-based methods have also been shown to be effective for articulated object tracking. Such methods can be classified into two classes: local optimization methods and global optimization methods. Typically, these methods are based on the optimization of an objective function which is the matching function between the model and the image features observed. Since the objective function to be optimized in such problems is usually multi-modal (and non-linear), the optimization process tends to be trapped into local optima. To cope with this problem, many of these methods use a particle-based stochastic framework to provide them with the ability to handle multiple hypotheses.

In Sections 3.2.1, 3.2.2 and 3.2.3, we present some optimization methods which are successfully applied for articulated object tracking with good results reported in the literature, especially those that use a particle-based framework [24, 47] as mentioned above.

It is worth mentioning that many optimization methods can be combined with a hierarchical search to achieve a better performance in terms of speed and accuracy. By using a hierarchical search in optimization steps and performing optimization for only one subpart (or group of subparts) at each step, one can obtain a better estimation with fewer particles. For instance, a combination of Annealed Particle Filter [24] and PS [64] has been proposed in [7] and a combination of Particle Swarm Optimization [48] and PS has been proposed in [47]. We will discuss these methods in Sections 3.2.4, 3.2.5 and 3.2.6.

3.2.1 Annealed Particle Filter

Simulated Annealing (SA) [49] is a stochastic method to optimize a multi-modal objective function. Its major advantage over other methods is its ability to avoid being trapped into local optima. We first give the SA basics, then the way it was introduced into PF's framework.

Assume that we need to find solutions to a maximization problem with the objective function $f(\mathbf{x})$. We turn this problem into the problem of sampling $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}$ from the distribution:

$$p(\mathbf{x}) \propto e^{-f(\mathbf{x})}$$

The solution of the maximization problem is obtained by taking:

$$\mathbf{x}^* = \arg\max\{f(\mathbf{x}^{(1)}), ..., f(\mathbf{x}^{(N)})\}.$$

$\mathbf{x}^*$ is thus generated with a probability proportional to $e^{-f(\mathbf{x}^*)}$. In order to increase the chance of getting $\mathbf{x}^*$ during the sampling process, SA uses the following distribution:

$$p_\lambda(\mathbf{x}) \propto e^{-\lambda f(\mathbf{x})}$$

As $\lambda \to +\infty$, the probability of getting $\mathbf{x}^*$ approaches 1. Then the sampling process gives a sample which contains only $\mathbf{x}^*$ or points very close to $\mathbf{x}^*$. In the case where $f(\mathbf{x})$ has many isolated maxima, however, simply using a large value for λ and sampling from $p_\lambda(\mathbf{x})$ yields poor results since the sample can easily become trapped into a local mode of $p_\lambda(\mathbf{x})$. To alleviate this problem, SA proceeds with several layers. At layer $m, m = M, \ldots, 0$, it takes a sample from the distribution:

$$p_{\lambda_m}(\mathbf{x}) \propto e^{-\lambda_m f(\mathbf{x})}$$

where $\lambda_0 > \lambda_1 > ... > \lambda_M$. This set of values is known as the annealing schedule.

The value of λ_M is set to be small (in physical language, the temperature, which is inversely proportional to λ, is initially high). The sample of the layer M is generated from a flat distribution, that therefore results in a coarse exploration of the search space. At the next layers, the value of λ is gradually increased, which increases the peakiness of the distribution used for sampling. This helps moving the sample to the regions that contain much of the probability mass of the distribution. When the sampling process is completed at layer 0, the sample of this layer should be concentrated around the peaks of good maxima.

SA was introduced into PF's framework giving Annealed Particle Filter (APF) as

described below.

APF has been introduced by Deustcher in [24] for articulated object tracking in high dimensional state spaces. Unlike PF which aims to approximate the posterior distribution of the state of the target object at each time slice, APF formulates the tracking problem as one of maximizing the likelihood function $p(\mathbf{y}_t|\mathbf{x}_t)$. It uses the same idea as simulated annealing to deal with multi-modal likelihood functions. Moreover, it uses a particle-based stochastic framework to improve its ability of handling multi-modal densities.

Recall that $p(\mathbf{y}_t|\mathbf{x}_t)$ is the likelihood function and the goal is to estimate the state of the target object at each time slice t by a set of N weighted particles. At each layer m, $m = M, \ldots, 0$, APF generates a particle set whose elements approach the global optimum $\max_{\mathbf{x}_t}\{p(\mathbf{y}_t|\mathbf{x}_t)\}$ during the annealing process by sampling from:

$$p_m(\mathbf{y}_t|\mathbf{x}_t) \propto p(\mathbf{y}_t|\mathbf{x}_t)^{\beta_m}$$

where $\beta_0 > \beta_1 > ... > \beta_M$.

The values of $\beta_M, ..., \beta_0$ are chosen similarly to SA: they gradually increase the peakiness of the distribution used for sampling to avoid getting stuck into poor local maxima. The principle is illustrated in Figure 3-13. On the left, the particles get stuck in poor local maxima. On the right, the annealing process gradually increases the peakiness of the distribution, which helps the particles to escape from local maxima.

In SA, the parameter λ at each layer plays a role of limiting the number of accepted "bad" candidate points. If λ is small, many "bad" candidate points are accepted, and a large part of the solution space is accessed. If λ is high, we get the opposite effect, fewer "bad" candidates are accepted and "good" candidates are more focused. In APF, at each layer, the resampling step has an effect of multiplying particles with high weights and eliminating particles with low weights. The particles which survive this resampling step can be considered as accepted candidates for the next layer as in SA. As discussed in Section 2.2, the number of such particles can be approximated by using survival rate, which is defined as follows:

$$\alpha = \frac{1}{N\sum_{i=1}^{N}(w_{t,(m)}^{(i)})^2}$$

where $w_{t,(m)}^{(i)}, i = 1, \ldots, N$, are normalized weights of the particle set at time t and layer m, $w_{t,(m)}^{(i)} \propto p(\mathbf{y}_t|\mathbf{x}_{t,(m)}^{(i)})^{\beta_m}$, $\mathbf{x}_{t,(m)}^{(i)}, i = 1, \ldots, N$, are particles at time t and layer m.

One can see that α is a function of β_m, $\alpha = \alpha(\beta_m)$ which is a monotonic decreasing

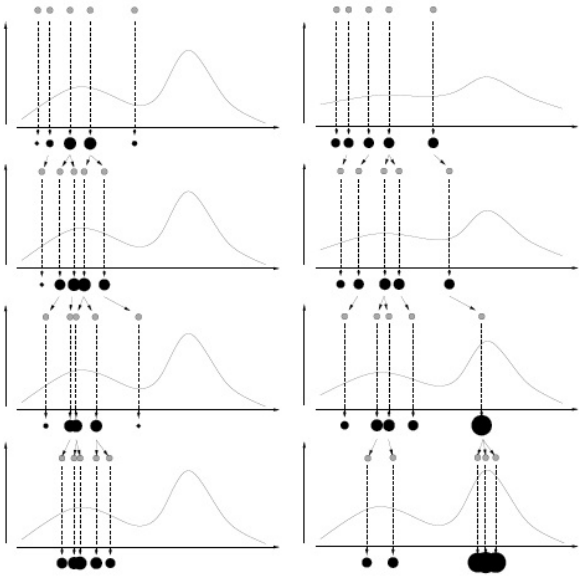


Figure 3-13: **Principle of SA applied into PF's framework**. Each dot represents a particle. The larger the dot, the higher the weight of the particle. On the left, SA is not used and on the right, SA is used which gradually increases the peakiness of the distribution. (figure reproduced from [36]).

function. Then by varying the value of β_m, one can figure out a value so that $\alpha(\beta_m)$ approaches a desired value $\alpha_{desired}$ (this can be done via gradient descent methods), which is called rate of annealing. The values of $\beta_0, ..., \beta_M$ can thus be computed for each time slice.

At time slice t, denote $\mathcal{S}^w_{t,(m)}$ the set of N weighted particles at layer m, $m = M, \ldots, 0$.

$$\mathcal{S}^w_{t,(m)} = \{\mathbf{x}^{(i)}_{t,(m)}, w^{(i)}_{t,(m)}\}^N_{i=1}$$

APF is described in Algorithm 3.3.

Input: Particle set $\{\mathbf{x}_{t-1,(0)}^{(i)}, w_{t-1,(0)}^{(i)}\}_{i=1}^{N}$

Output: Particle set $\{\mathbf{x}_{t,(0)}^{(i)}, w_{t,(0)}^{(i)}\}_{i=1}^{N}$

for $i = 1$ **to** N **do**

  $\{\mathbf{x}_{t,(M)}^{(i)}\}_{i=1}^{N} \leftarrow$ Propagate$\{\mathbf{x}_{t-1,(0)}^{(i)}\}_{i=1}^{N}$ by $\mathbf{x}_{t,(M)}^{(i)} = f(\mathbf{x}_{t-1,(0)}^{(i)}) + \mathbf{B}_0$

  Compute the normalized particle weight $w_{t,(M)}^{(i)}$: $w_{t,(M)}^{(i)} \propto p(\mathbf{y}_t|\mathbf{x}_{t,(M)}^{(i)})$

for $m = M$ **downto** 1 **do**

  Compute β_m so that $\alpha(\beta_m) = \frac{1}{N\sum_{i=1}^{N}((w_{t,(m)}^{(i)})^{\beta_m})^2} = \alpha_{desired}$

  for $i = 1$ **to** N **do**

    Update the annealed particle weight: $w_{t,(m)}^{(i)} = (w_{t,(m)}^{(i)})^{\beta_m}$

  for $i = 1$ **to** N **do**

    Normalize the weight: $w_{t,(m)}^{(i)} = \frac{w_{t,(m)}^{(i)}}{\sum_{j=1}^{N} w_{t,(m)}^{(j)}}$

  $\{\bar{\mathbf{x}}_{t,(m)}^{(i)}, \frac{1}{N}\}_{i=1}^{N} \leftarrow$ Resample$\{\mathbf{x}_{t,(m)}^{(i)}, w_{t,(m)}^{(i)}\}_{i=1}^{N}$.

  $\{\mathbf{x}_{t,(m-1)}^{(i)}\}_{i=1}^{N} \leftarrow$ Propagate$\{\bar{\mathbf{x}}_{t,(m)}^{(i)}\}_{i=1}^{N}$ using: $\mathbf{x}_{t,(m-1)}^{(i)} = \bar{\mathbf{x}}_{t,(m)}^{(i)} + \mathbf{B}_m$, where $\mathbf{B}_m \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_{(m)})$.

  for $i = 1$ **to** N **do**

    Compute the normalized particle weight $w_{t,(m-1)}^{(i)}$ of $\mathbf{x}_{t,(m-1)}^{(i)}$: $w_{t,(m-1)}^{(i)} \propto p(\mathbf{y}_t|\mathbf{x}_{t,(m-1)}^{(i)})$

return $\{\mathbf{x}_{t,(0)}^{(i)}, w_{t,(0)}^{(i)}\}_{i=1}^{N}$

Algorithm 3.3: Annealed Particle Filter.

In Algorithm 3.3, f is the dynamic function and $\mathbf{B}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_{(0)})$. When no dynamic function is available, we set $f(\mathbf{x}_{t-1,(0)}^{(i)}) = \mathbf{x}_{t-1,(0)}^{(i)}$. At time slice t, the optimal configuration of the target object is estimated by using $\mathcal{S}_{t,0}^{w}$ as follows:

$$\hat{\mathbf{x}_t} = \sum_{i=1}^{N} w_{t,(0)}^{(i)} \mathbf{x}_{t,(0)}^{(i)}$$

APF also needs other parameters to work, including the number of layers M, the covariance matrices $\mathbf{P}_{(m)}, m = M, \ldots, 0$, and the rates of annealing $\alpha_M, ...\alpha_1$. The number of layers and the rates of annealing are empirically determined. The covariance matrix $\mathbf{P}_{(m)}, m = M, \ldots, 1$, can be computed by using a deterministic method [24] as follows:

$$\mathbf{P}_{(m)} = \alpha_M \times ... \times \alpha_m \times \mathbf{P}_{(0)}$$

where $\mathbf{P}_{(0)}$ is a covariance matrix whose diagonal elements are fixed to the half the maximum expected movement of the corresponding model parameter over one time slice. In [25], $\mathbf{P}_{(m)}$ is computed differently, so that:

$$\mathbf{P}_{(m)} \propto \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_{t,(m)}^{(i)} - \mathbf{x}_{t,(m)}^{mean}).(\mathbf{x}_{t,(m)}^{(i)} - \mathbf{x}_{t,(m)}^{mean})^T$$

where $\mathbf{x}_{t,(m)}^{mean}$ is the sample mean of the particle set $\mathcal{S}_{t,(m)}^{w}$ and $(\mathbf{x}_{t,(m)}^{(i)} - \mathbf{x}_{t,(m)}^{mean})^T$ is the transpose of $(\mathbf{x}_{t,(m)}^{(i)} - \mathbf{x}_{t,(m)}^{mean})$.

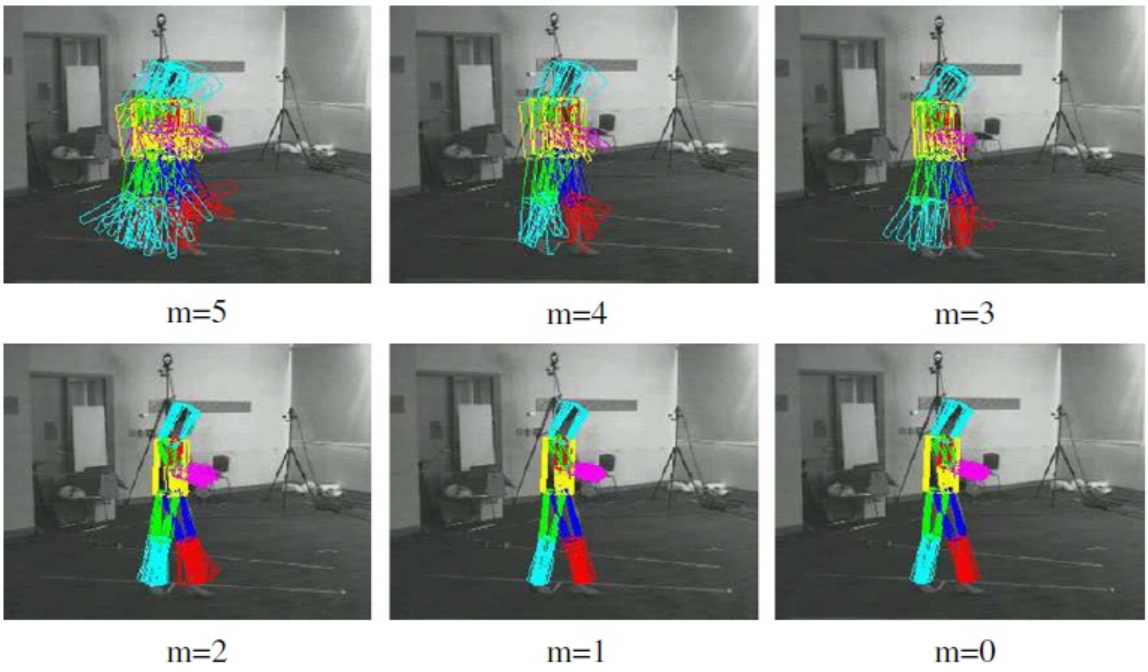


Figure 3-14: **The effectiveness of the annealing process in APF for human tracking**. (figure reproduced from [78]).

Example. Figure 3-14 shows an example of human tracking by APF during one time slice. In this example, six layers were used to search for the global optimum poses. At each layer, the particle set where each particle represents a hypothesized pose of the tracked body is drawn to visualize the effectiveness of the annealing process in APF. At the first layer ($m = 5$), the particle set spreads over the search space and contains many incorrect poses. At the next layers, the particle set is gradually focused on correct poses thanks to the annealing process. At the final layer ($m = 0$), the particle set contains only particles which are close to the correct pose, giving a good estimation of the tracked human body.

Initialization:
$\{\mathbf{x}_{(0)}^{(i)}\}_{i=1}^{N}$ are randomly drawn from $\mathcal{X}$, $\{\mathbf{v}_{(0)}^{(i)}\}_{i=1}^{N}$ are randomly drawn from $[0,1]$
$\mathbf{s}^{(i)} = \mathbf{x}_{(0)}^{(i)}, i = 1, \ldots, N$
$\mathbf{s}^{g} = \arg\max\{f(\mathbf{s}^{(i)})\}_{i=1}^{N}$
$\mathrm{m} = 0$
repeat
  Update $\{\mathbf{v}_{(m+1)}^{(i)}\}_{i=1}^{N}$ using Equation 3.6
  Update $\{\mathbf{x}_{(m+1)}^{(i)}\}_{i=1}^{N}$ using Equation 3.7
  Update $\mathbf{s}^{(i)}, i = 1, \ldots, N$
  Update $\mathbf{s}^{g}$
  $m = m + 1$
until *Stopping criterion is satisfied* ;

Algorithm 3.4: Particle Swarm Optimization.

3.2.2 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a population based optimization technique introduced in [48] which was first used for simulating social behavior. It was later modified by several researchers to improve its search capabilities and convergence properties. PSO has recently been applied for human tracking [52, 47] with good empirical results.

We search for $\mathbf{x} \in \mathcal{X}$ that maximizes a *fitness function* (*cost function*) $f : \mathcal{X} \to \mathbb{R}$, subject to the constraint $\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}$. A swarm consists of N particles, each particle representing a candidate solution to the search problem. Denote $\mathbf{x}_{(m)}^{(i)}$ the ith particle, $\mathbf{x}_{(m)}^{(i)} = (\mathbf{x}_{(m)}^{(i),1}, \ldots, \mathbf{x}_{(m)}^{(i),P}) \in \mathcal{X}$. Unlike evolutionary algorithms, each particle in PSO is given a velocity $\mathbf{v}_{(m)}^{(i)} = (\mathbf{v}_{(m)}^{(i),1}, \ldots, \mathbf{v}_{(m)}^{(i),P}) \in \mathcal{X}$ and has the ability of memorizing its best state $\mathbf{s}^{(i)}$ computed so far, $\mathbf{s}^{(i)} = (\mathbf{s}^{(i),1}, ..., \mathbf{s}^{(i),P}) \in \mathcal{X}$. Let $\mathbf{s}^{g}$ be the current global best state, i.e. $\mathbf{s}^{g} = \arg\max\{f(\mathbf{s}^{(i)})\}_{i=1}^{N}$.

The evolution of particles in PSO is described by the 2 following equations:

$$\mathbf{v}_{(m+1)}^{(i)} = \mathbf{v}_{(m)}^{(i)} + \beta_1 r_1(\mathbf{s}^{(i)} - \mathbf{x}_{(m)}^{(i)}) + \beta_2 r_2(\mathbf{s}^{g} - \mathbf{x}_{(m)}^{(i)}) \tag{3.6}$$

$$\mathbf{x}_{(m+1)}^{(i)} = \mathbf{x}_{(m)}^{(i)} + \mathbf{v}_{(m+1)}^{(i)} \tag{3.7}$$

where β_1, β_2 are constants, r_1, r_2 are random numbers drawn from [0,1].

The right-hand side of Equation 3.6 consists of three terms which can be intuitively interpreted as follows: the first term means that the velocity of a particle at each time slice depends on that at the previous time slice; the second term is a *cognitive* part

which represents the strategy of the particle to get close to the optimal position, based on its own experience (local search); the third term is a *social* part which represents the strategy of the particle to get close to the optimal position, based on the experience of the global population (global search).

All stages of PSO are given in Algorithm 3.4, where the stopping criterion is usually either a maximum number of iterations or a threshold on the improvement of $\mathbf{s}^g$.

PSO has the ability of balancing between the local and global search strategies of the particles by setting the appropriate values for the constants β_1, β_2. PSO with inertia weight [84] was introduced to add a new parameter in Equation 3.6 as follows:

$$\mathbf{v}_{(m+1)}^{(i)} = w\mathbf{v}_{(m)}^{(i)} + \beta_1 r_1(\mathbf{s}^{(i)} - \mathbf{x}_{(m)}^{(i)}) + \beta_2 r_2(\mathbf{s}^g - \mathbf{x}_{(m)}^{(i)}) \tag{3.8}$$

where w is the inertia weight and $w\mathbf{v}_{(m)}^{(i)}$ is the inertial velocity.

A large inertia weight results in an exploration of the search space (global search) while a small inertia weight limits the search around the globally best particle (local search). The value of w can be fixed or adaptively changed throughout the search.

PSO has been integrated into PF's framework in [52] for human tracking. The tracker is indeed a particle filter with an additional step processed via PSO after the prediction step to shift the particles toward more promising regions in the search space. PSO has also been successfully applied for hand tracking [68] where the problem is formulated as an optimization problem.

The idea of APF was also incorporated into PSO to strengthen the searching capabilities of PSO in human tracking, [112]. For that, some modifications of Equation 3.8 were proposed. First, because the use of inertial velocity given in Equation 3.8 often results in impossible human poses, it is replaced for each iteration m with a covariance matrix $\mathbf{P}_{(m)}$, whose terms gradually decrease with PSO iterations (APF layers) as follows:

$$\mathbf{P}_{(m)} = \alpha_0 * \mathbf{P}_{(m-1)}$$

where α_0 is a constant, The covariance matrix is initialized such that diagonal elements have values equal to the maximum expected movement of the corresponding model configuration parameters over one time step. Second, it was also observed that the local and global optima are no longer trustable when the particle swarm is near the global optima (at the end of PSO's iteration) due to the presence of noise in images: it should then have less impact on guiding the swarm. That is why Equation 3.8

is now modified as follows:

$$\mathbf{v}_{(m+1)}^{(i)} = r_0 \mathbf{P}_{(m)} + \beta_0 exp(1 - \frac{m}{M}) r_1 (\mathbf{s}^{(i)} - \mathbf{x}_{(m)}^{(i)}) + \beta_0 exp(1 - \frac{m}{M}) r_2 (\mathbf{s}^{g} - \mathbf{x}_{(m)}^{(i)}) \quad (3.9)$$

where r_0 is a random number drawn from [0,1].

PSO is easy to implement and is a powerful method for multi-dimensional non-linear optimization. Its convergence, however, is difficult to analyze since it depends on the choice of the fitness function. It suffers from the same problem as any other search algorithm in high dimensional search space: it becomes computationally expensive. In Section 3.2.5, we briefly describe a method which uses a hierarchical search with PSO to reduce the number of particles required for tracking.

3.2.3 Other optimization-based approaches

Gradient descent has also been successfully applied for articulated object tracking. In [12], a stochastic meta descent (SMD) has been introduced for 3D hand tracking. Tracking proceeds by optimizing the difference between the hand model and the depth maps generated by a structured light sensor. Hand constraints are incorporated into the optimization process through constrained gradients. To avoid getting stuck into local minima, the set of points used to evaluate the objective function are selected randomly at each iteration. In [13], the work is extended by adding surface orientation terms to the objective function to increase the robustness of the tracker. In [43], SMD is combined with PF to implement a multiple hypotheses tracker, called smart particle filter. Gall et al. [37] introduced a multi-layer framework for human tracking. At the first layer, a global optimization is used to obtain an estimate of human body, that is then refined by smoothing, followed by a local optimization step. Pantrigo et al. [70] combined PF and population-based metaheuristics for articulated object tracking, consisting of two stages. At the first stage, a particle set is propagated and corrected via PF. At the second stage, a fixed number of particles are selected from the previous particle set and are combined to obtain higher weighted ones using Path Relinking [38] or Scatter Search [53] approaches.

Jiang et al. [46] formulated the human tracking problem as a linear integer programming one. Various types of constraints, such as spatial constraints, appearance symmetry constraints, position symmetry constraints, are expressed as constraints of the integer linear program, and then relaxed into a mixed integer linear program and solved using a Branch-and-Bound algorithm.

Tran et al. [98] combined dynamic programming with exhaustive search to solve the problem of upper body tracking. They constructed an energy function correspond-

ing to a weighted linear combination of unary and pairwise functions (appearance and spatial relations between all pairs of body parts to be tracked). The weights of different functions were pretrained using structure learning [94]. Upper body configuration is found by first optimizing the energy function to search for the best candidates of the triple (right arm, torso, left arm) and then by performing local search to improve the estimates and search for the remaining upper body parts.

In Sections 3.2.4, 3.2.5 and 3.2.6, we discuss some approaches which combine optimization with a hierarchical search for articulated object tracking.

3.2.4 Annealed Particle Filter plus Partitioned Sampling

A combination of APF and PS has been proposed in [7]. By partitioning the state space of the target object into subspaces and performing APF for each subspace in a hierarchical order as in PS, the proposed method aims to integrate the benefits of both methods. One common problem of PS is that its performance highly depends on the hierarchical order in which the subparts of the target object are estimated. This hierarchical order thus must be adapted to a specific problem and should not be fixed during the tracking in order to achieve the best performance. The proposed method then uses different hierarchical orders for PS, each order generates a subset of particles which are then combined and resampled according to their weights to form the new particle set for the next search of PS. This ensures that the new particle set will be collected from the best particles generated by each hierarchical order. This makes the method more stable and insensitive to a specific order in which the subparts of the target object are estimated.

3.2.5 Hierarchical Particle Swarm Optimization (HPSO)

Hierarchical PSO for human tracking combines PSO and a hierarchical search and has been introduced in [47]. In this approach, the human body is modeled by a kinematic tree containing 13 nodes. Each node corresponds to a specific body joint. The root node has 6 DOF which describe the global translation and orientation of the person. All other nodes can have up to 3 DOF (orientation of body joints), resulting in a 31-dimensional state space model.

The tracking with HPSO is automatically initialized. At the first frame, a set of particles and their velocities are initialized as in PSO. HPSO then starts searching the optimal configuration of the person for this frame. The search is processed in a hierarchical manner: starting with the torso and then proceeding towards the limbs. The search space is split into 12 different subspaces which are hierarchically optimized.

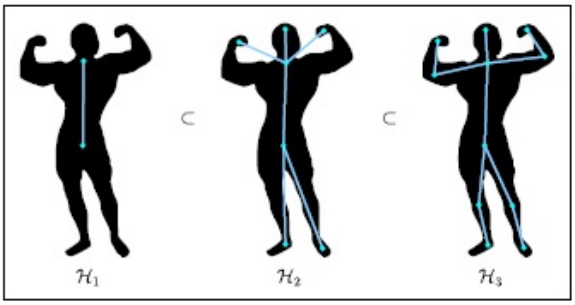


Figure 3-15: **Scalable human body model**. (figure reproduced from [17]).

At each optimization step for a subspace, the previous subspace in the hierarchical order of the same time slice is used to constrain the search. To reinforce the spatial constraint when optimizing in a subspace, the next subspace in the hierarchical order in the previous time slice is also used as an additional constraint. In this way, the hierarchical search is more efficient since the spatial constraints in the kinematic structure are propagated in two directions. Once the initial configuration at the first frame has been estimated, the initial particle set for the next frame results from a sampling from a Gaussian distribution centered in the current best estimate and the search for the optimal configuration of this frame is repeated. By the use of hierarchical search, the proposed method greatly reduces the number of particles used for tracking and performs well with a reasonable number of particles for the 31-dimensional state space model. With a limited number of particles, it can also recover from wrong estimates thanks to its ability to explore the search space.

3.2.6 Hierarchical Structure based Annealed Particle Filter (HSAPF)

An annealing strategy for human tracking has also been exploited in [17]. Instead of using a set of smoothed versions of the likelihood function to concentrate particles around its peaks, the proposed method uses a set of progressively refined human body models leading to the concept of *structural annealing*. In this approach, a human body model is composed of a root segment (torso) and a set of open kinematic chains modeling the head, arms and legs. Each kinematic chain can contain a different number of body parts. A scalable human body model (SHBM) is defined by: $\mathcal{M} = \{\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_M\}$, where M is the number of human body models in $\mathcal{M}$. An example of SHBM is given in Figure 3-15, where the human body model $\mathcal{H}_1$ contains only parameters describing the torso and more details of the human body are added into the human body models $\mathcal{H}_2$ and $\mathcal{H}_3$ for the full body tracking.

The key idea is to refine the human body models in $\mathcal{M}$ in a coarse-to-fine manner. For this purpose, $\mathcal{M}$ is defined such that $\mathcal{H}_1$ contains only parameters for the torso, $\mathcal{H}_i, i \geq 2$, contains more details about the entire human body than $\mathcal{H}_{i-1}$, and $\mathcal{H}_M$ contains all parameters describing the whole human body. Denote $\mathcal{X}_{\mathcal{H}_i}$, $i = 1, \ldots, M$, the state space of the human body model $\mathcal{H}_i$. Then the following conditions hold: $\mathcal{X}_{\mathcal{H}_1} \subset \mathcal{X}_{\mathcal{H}_2} \subset \ldots \subset \mathcal{X}_{\mathcal{H}_M}$.

A particle set $\mathcal{S}_{\mathcal{H}_i}$ is maintained for each human body model $\mathcal{H}_i$ in $\mathcal{M}$. At each time slice, the proposed algorithm consists of two stages. During the *forward stage*, the human body model is refined from model $\mathcal{H}_1$ until reaching model $\mathcal{H}_M$. Starting from model $\mathcal{H}_1$, the particle set for this model is propagated and corrected and then combined with those for the model $\mathcal{H}_2$ to generate an improved initial particle set for $\mathcal{H}_2$. This procedure is repeated until reaching the model $\mathcal{H}_M$. At any step, the particle sets for model $\mathcal{H}_i$ and for model $\mathcal{H}_{i+1}$, $i = 1, \ldots, M-1$, are combined using an operator $G_{forward}$. During the *backward stage*, the particle set obtained for the last human body model $\mathcal{H}_M$ is back propagated to previous human body models in $\mathcal{M}$ using an operator that first sorts $\mathcal{S}_{\mathcal{H}_M}$ and then simply generates a resampled set from $\mathcal{S}_{\mathcal{H}_M}$ for each human body model $\mathcal{H}_i, i = 1, ..., M-1$. An improved particle set for $\mathcal{H}_i$ can then be obtained by keeping only variables in each particle that belong to $\mathcal{H}_i$.

The proposed approach has been shown to be effective in tracking human with missing data. For example, when the observations of limbs are missing, it can track the torso thanks to model $\mathcal{H}_1$.

3.3 Discriminative approaches

A variety of regression approaches have been proposed for human estimation [2, 69]. In [1], images from a set of training images are encoded into 100-dimensional histograms. Using a non linear regression, the authors learn a mapping function between these histograms and a 55-dimensional pose vector which is then used to recover the 3D human pose. A more general approach is also introduced for dealing with cluttered backgrounds. Each image from a set of training images without cluttered backgrounds is divided into a grid of points. Overlapping image patches from this grid of points are then obtained and encoded into a 128-dimensional vector containing local gradient orientation histograms [60]. Non-negative matrix factorizations [55] are then used to obtain feature vectors which are more compact representations of these vectors (with only 30-40 dimensions). Finally, a mapping function from feature vectors to the pose vector is learnt by linear regression and used to recover the 3D human pose. In [69],

similar poses from a set of training poses are first clustered using k-means algorithm. A Support Vector Machine [15] classifier is then used for discriminating these pose clusters. The 3D human pose is recovered by a linear regressor which is learnt for each pose cluster.

In human pose estimation and human tracking, if we have prior knowledge about the performed actions, this can be exploited to reduce the high dimensionality of the state space. Many techniques, known as *dimensionality reduction* techniques, have been widely applied in human pose estimation and human tracking [54, 105]. The basic assumption is that body part movements are mutually dependent and therefore lie in a low-dimensional manifold. The original state space can thus be represented using only a smaller number of degrees of freedom than for the original articulated object. This can by achieved by learning the low-dimensional manifold (*latent space*) from the original state space (*data space*) using training data for a specific human activity.

In addition to the latent space, some learning methods, such as Gaussian Process Latent Variable Models (GPLVM) [54], Gaussian Process Dynamical Models (GPDM) [105] also provide a mapping from the latent space to the data space. This is particularly interesting for the PF's framework since we can reduce the number of particles required for tracking by sampling into the low-dimensional manifold and, then, by projecting back into the original space to evaluate the likelihood of particles. In [96], the proposed approach reduces the dimensionality of the problem by using GPLVM to learn a mapping from a 2-dimensional latent space to the data space and then directly applying PF to this latent space. The sampling step and resampling step is performed as in PF. In the correction step, the likelihood of a point in the latent space is evaluated by projecting this point to a point in the data space using the mapping learnt from GPLVM, and then evaluating the likelihood of the obtained point in the data space. To obtain an estimation of the target person at each time slice, the mean of the particle set in the latent space is calculated and then projected to the data space using the mapping. By learning from a training dataset of feasible poses, the efficiency of the sampling step is improved since the sampled poses are focused in regions of interest. This method, however, is based on the key assumption that human body poses that are similar in the data space tend to be mapped close to each other in the latent space, which is not always true, as pointed out in [41, 78]. A similar approach has been introduced in [109]. To reduce the computational cost and avoid getting trapped into local minima while learning with GPLVM, the authors developed an incremental learning algorithm for the GPLVM using stochastic gradient descent. Instead of applying PF into the latent space learnt by GPLVM, Raskin et

al. [78] proposed to perform APF in this latent space. In this approach, the state of the human body is divided into two parts. The first part consists of the global position and orientation. The second part consists of the joint angles describing the human pose. The latent space and the mapping between it and the original joint angles is learnt using only the second part. The state of the human body can thus be represented by a reduced vector consisting of the global position and orientation and the latent vector. At each time step, APF is performed in this vector space. To evaluate the weight for each pose, the joint angle vector is obtained by projecting the latent vector using the learnt mapping, the full state is then a concatenation of this joint angle vector with the vector describing the global position and orientation, and the evaluation of the weights is processed as in PF.

Urtasun et al. [102, 101] formulate the tracking problem as a nonlinear least-squares optimization problem whose objective function is constructed using GPLVM. Tian et al. [95] use the same idea to solve the problem of estimating the 2D pose of a person from silhouettes.

The main limitation of learning-based approaches is their lack of generality. The learnt model can only be used for tracking some action-specific motions. This is due to the fact that the amount of training data must be limited and therefore cannot account for the variability in appearances, lighting changes, clothes and limbs deformation, ... Moreover, in order to obtain a good model, the size of the training set is often large and the computational cost of the learning algorithm can be expensive.

As mentioned in Chapter 1, the main goal of this thesis is to develop new decomposition approaches to deal with the problem of high dimensional state spaces in articulated object tracking. In the next chapters, we propose three decomposition-based approaches. In Chapter 4, we introduce a novel approach that exploits the conditional independences encoded by the DBN modeling the articulated tracking to improve the tracking performance and computation time of PS, notably by introducing a new operator "swapping" particles. In Chapter 5, we develop a novel resampling approach that takes advantage of this swapping operator to resample over an implicitly created sample of an exponential size better representing the density to estimate. Finally, in Chapter 6, we propose a novel approach, based on PSO and hierarchical search, to reduce the computational cost of PSO and tackle the problem of noisy observations in articulated object tracking.

Chapter 4

Swapping-Based Partitioned Sampling

In this chapter, we present a new approach based on PS for articulated object tracking. Recall that an articulated object consists of P parts and its state at time t, $\mathbf{x}_t$, is decomposed as $\{\mathbf{x}_t^1, \ldots, \mathbf{x}_t^P\}$. The first step of our approach consists of parallelizing PS's propagations/corrections among sets of object parts $\mathbf{x}_t^i$ in such a way that the new propagations/corrections scheme does not alter the posterior density. These sets of object parts are selected using d-separation analysis in the DBN modeling the tracking problem. Next, we introduce a new operation called *swapping* that permutes some subsamples of the object parts processed in parallel so that the best subsamples are combined into new particles while still guaranteeing that the posterior density is correctly estimated. There are two advantages to this approach as compared to PS. First, it reduces the number of necessary resamplings since a set of parts instead of one part is processed at each propagation/correction, hence reducing the noise produced by resampling. Second, the swapping operation creates some particles which are more focused near the modes of the posterior density. Consequently, our approach is better than PS both in terms of tracking accuracy and of computation time.

The chapter is organized as follows. Section 4.1 presents our idea of parallelizing PS's propagations/corrections. In section 4.2, we explain how permutations over some subsamples of the object parts processed in parallel can improve the estimation of the posterior density. Section 4.3 provides an efficient way of implementing our algorithm. Section 4.4 contains an extensive experimental comparison, which demonstrates the interest of the swapping operation as well as the robust tracking achieved by our algorithm. Finally, Section 4.5 offers some conclusions.

4.1 A d-separation-based parallelization of PS

Our approach is based on the assumptions that the proposal transition function of a given part $\mathbf{x}_t^i$ of the object at time t depends only on that part in time $t-1$ (and possibly on other parts in time t) and the observations depend only on their corresponding state. Note that the first assumption is rather mild for object tracking and it holds in most applications. The second assumption holds when there is no occlusion between parts of the target object. For instance, both assumptions hold in the example in Figure 4-1. More formally, this leads to the following definition:

Definition 4-1 (Object Tracking DBN – OTDBN) *An OTDBN is a 2TBN ([67], Chapter 3) defined over random variables $\{\mathbf{x}_t^i, \mathbf{y}_t^i\}$, where $\mathbf{x}_t^i$ and $\mathbf{y}_t^i$ represent the state and observation of part i in time slice t. OTDBNs satisfy the following four conditions:*

1. *there does not exist any arc $\mathbf{x}_s^i \to \mathbf{x}_t^j$ with $s < t-1$ or $s > t$;*
2. *for every i and $t > 1$, there exists an arc $\mathbf{x}_{t-1}^j \to \mathbf{x}_t^i$ if and only if $j = i$;*
3. *for each node $\mathbf{x}_t^i$, there exists a node $\mathbf{y}_t^i$ whose only parent is $\mathbf{x}_t^i$;*
4. *nodes $\mathbf{y}_t^i$ have no children.*

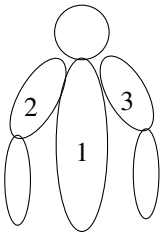


Figure 4-1: **Upper body tracking**. The body parts to be tracked are the torso, the upper left arm and the upper right arm.

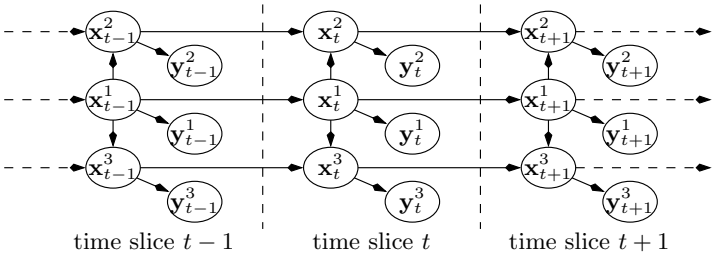


Figure 4-2: **A dynamic Bayesian network for body tracking in Fig 4-1**.

In Figure 4-2, by d-separation ([74], Chapter 3), it can be proved that $\mathbf{x}_t^3$ is independent of $\mathbf{x}_{t-1}^2$ conditionally to $\{\mathbf{x}_t^1, \mathbf{x}_{t-1}^3\}$ and observations $\mathbf{y}_t^i$ are independent of the other random variables conditionally to states $\mathbf{x}_t^i$. This implies that the propagations/corrections of $\mathbf{x}_t^2$ and $\mathbf{x}_t^3$ can be performed in parallel since none of them has any impact on the other one. Thus, this suggests the new condensation diagram of Figure 4-3 where object parts $\mathbf{x}_t^2$ and $\mathbf{x}_t^3$ are processed in parallel. The advantage of this diagram is that it performs fewer resamplings compared to the diagram of PS where one resampling step is performed after the correction of each part. Thus the new diagram introduces less noise in the estimation caused by these resampling steps.

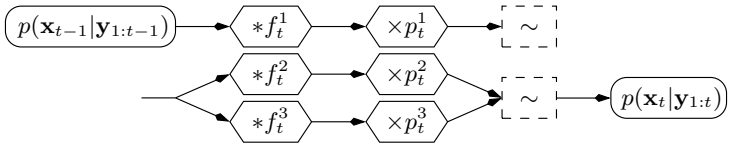


Figure 4-3: **Condensation diagram exploiting conditional independences.**

Now, let us extend the condensation diagram of Figure 4-3 for arbitrary OTDBNs. Let X_t denote a generic node of an OTDBN $\mathcal{G}$ in time slice t (so either $X_t = \mathbf{x}_t^i$ or $X_t = \mathbf{y}_t^i$ for some i). Let $\mathbf{pa}(X_t)$ and $\mathbf{pa}_t(X_t)$ denote the set of parents of X_t in $\mathcal{G}$ in all time slices and in time slice t only respectively. For instance, in Figure 4-2, $\mathbf{pa}(\mathbf{x}_t^2) = \{\mathbf{x}_t^1, \mathbf{x}_{t-1}^2\}$ and $\mathbf{pa}_t(\mathbf{x}_t^2) = \{\mathbf{x}_t^1\}$. Similarly, let $\mathbf{an}(X_t)$ and $\mathbf{an}_t(X_t)$ be the set of ancestors of X_t in all time slices and in time slice t only respectively. Assume that the probabilistic dependences between all random variables $\mathbf{x}_t^i$ and $\mathbf{y}_t^i$, $i = 1, \ldots, P$, are represented by an OTDBN $\mathcal{G}$. Let $\{P_1, \ldots, P_K\}$ be a partition of $\{1, \ldots, P\}$ defined by:

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

Intuitively, P_1 is the set of indices k of the object parts $\mathbf{x}_t^k$ that have no parent in time slice t; P_2 is the set of indices of the object parts whose parents in time slice t all belong to P_1, and so on. According to this definition, $P_1, \ldots, P_K$ follow the kinematic chain of the articulated object to be tracked. Consider again the human body model in Figure 4-1. In this example, P_1 contains the root of the kinematic chain, $P_1 = \{1\}$; P_2 contains the parts which follow the root of the kinematic chain, $P_2 = \{2, 3\}$. In the sequel, we will also use the following notations that were introduced in Section 3.1.2: for any $j = 1, \ldots, K-1$, let $Q_j = \cup_{h=1}^{j} P_h$ and $R_j = \cup_{h=j+1}^{K} P_h$, i.e., Q_j and R_j represent the set of parts processed up to the processing of parts P_j and those still to be processed respectively. Let $Q_0 = R_K = \emptyset$.

It is not hard to see that, by d-separation, all the nodes $\mathbf{x}_t^k$ of a given P_j are independent conditionally to their parents $\mathbf{pa}(\mathbf{x}_t^k)$. Consequently, PS can propagate/correct all these nodes independently (in parallel) and produce a correct estimation of the posterior joint density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. This suggests the condensation diagram of Figure 4-4 where, for every $j \in \{1,\ldots,K\}$, $P_j = \{i_{P_j}^1,\ldots,i_{P_j}^{k_j}\}$ and each object part $\mathbf{x}^{i_{P_j}^h}$ has its own proposal function $f_t^{i_{P_j}^h}$ and its own correction function $p_t^{i_{P_j}^h}$ (as described in Equation 3.2). In this diagram, all the propagations and corrections of the object parts in a given set P_j are thus performed in parallel and, subsequently, a resampling is performed over all these parts. The correctness of the diagram follows from Proposition 4-1.

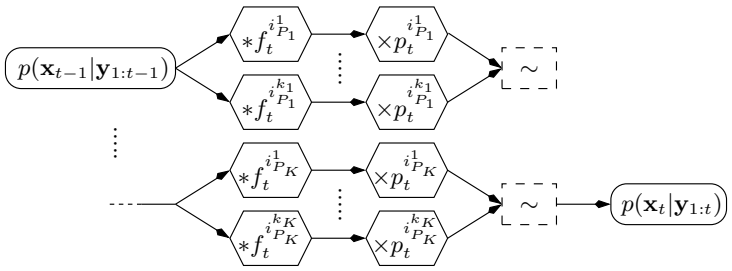


Figure 4-4: **Parallelized Partitioned Sampling condensation diagram.**

Proposition 4-1 *The set of particles resulting from the diagram of Figure 4-4 represents probability distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.*

Proof of Proposition 4-1: By abuse of notation, for any set $H \subseteq \{1,\ldots,P\}$, let $\mathbf{x}_t^H$ denote the set of nodes $\{\mathbf{x}_t^k : k \in H\}$.

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

$$\mathbf{x}_t^k \perp\!\!\!\perp \bigcup_{h \in P_j \setminus \{k\}} \{\mathbf{x}_t^h\} \cup \mathbf{x}_t^{Q_{j-1}} \cup \mathbf{x}_{t-1}^{R_{j-1}} \cup \mathbf{y}_{1:t-1} \cup \mathbf{y}_t^{Q_{j-1}} \mid \mathbf{pa}(\mathbf{x}_t^k). \tag{4.1}$$

If there existed in the OTDBN an active chain $\{\mathbf{c}_1 = \mathbf{x}_t^k,\ldots,\mathbf{c}_n\}$ between $\mathbf{x}_t^k$ and one of the nodes in the independent part of Equation 4.1, its first arc would necessarily be $\mathbf{c}_1 \rightarrow \mathbf{c}_2$ since $\mathbf{c}_1$'s parents are the conditioning set. Let $\mathbf{c}_V$ denote the last node such that, for all $2 \leq h \leq V$, the arcs of the chain are $\mathbf{c}_{h-1} \rightarrow \mathbf{c}_h$. By definition of sets P_j's, none of the nodes $\mathbf{x}_t^h \in \mathbf{x}_t^{P_j} \setminus \{\mathbf{x}_t^k\}$ is a descendant of $\mathbf{x}_t^k$. In addition, by definition of OTDBNs, nodes in time slice $t-1$ cannot be some descendant of $\mathbf{x}_t^k$. Consequently, $\mathbf{c}_V \neq \mathbf{c}_n$ and, thus, the active chain contains arcs $\mathbf{c}_{V-1} \rightarrow \mathbf{c}_V \leftarrow \mathbf{c}_{V+1}$. As $\mathbf{c}_V$ is a descendant of $\mathbf{x}_t^k$, neither it nor its descendants are in the conditioning set and, by d-separation, the chain cannot be active and Equation 4.1 holds.

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

$$\int p(\mathbf{x}_t^{Q_{j-1}}, \mathbf{x}_{t-1}^{R_{j-1}} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \prod_{k \in P_j} p(\mathbf{x}_t^k | \mathbf{pa}(\mathbf{x}_t^k)) \; d\mathbf{x}_{t-1}^{P_j} \tag{4.2}$$

because Equation 4.1 implies that no operation "$*f_t^k$" depends on $\mathbf{x}_t^{P_j \setminus \{k\}}$. Now, since for all $k \in P_j$, $\mathbf{pa}(\mathbf{x}_t^k) \subseteq \mathbf{x}_t^{Q_{j-1}} \cup \mathbf{x}_{t-1}^{R_{j-1}}$, Equation 4.1 implies that:

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

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

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

Therefore, Equation 4.2 is equivalent to:

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

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

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

$$p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}) \times \prod_{k \in P_j} p(\mathbf{y}_t^k | \mathbf{x}_t^k). \tag{4.3}$$

By definition of OTDBNs, nodes $\mathbf{y}_t^k$ have no children and only one parent: $\mathbf{x}_t^k$. Hence,

by d-separation, they are independent of the rest of the network conditionally to this parent. Therefore, as in the preceding paragraph, we can easily prove that:

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

Therefore, Equation 4.3 is equivalent to $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_t^{P_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}})$, which, when normalized, is equal to $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_{j-1}}, \mathbf{y}_t^{P_j}) = p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. The last step of the processing of P_j is a resampling that does not alter this distribution. Finally, note that this probability is precisely that assumed at the beginning of the proof. Hence, after processing all the P_j's, the particle set estimates $p(\mathbf{x}_t^{Q_K}, \mathbf{x}_{t-1}^{R_K} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_K}) = p(\mathbf{x}_t | \mathbf{y}_{1:t})$. □

4.2 Swapping-Based Partitioned Sampling (SBPS)

There are two major differences between the diagrams of Figures 3-4 and 4-4: the latter performs fewer resamplings, thus it introduces less noise in the particle set and, more importantly, it enables to produce better fitted particles by swapping their subparts. Actually, consider again our body tracking example and assume that we generated the 3 particles $\mathbf{x}_t^{(i)}$ of Figure 4-5(a) where $\mathcal{X}^1$ represents the torso and $\mathcal{X}^2$ and $\mathcal{X}^3$ represent the left and right hand respectively, and where the shaded areas represent the body's true state. According to the OTDBN of Figure 4-2, for fixed values of $\mathbf{x}_{1:t}^1$, the sets of left and right parts of the particles represent $p(\mathbf{x}_t^2, \mathbf{y}_{1:t}^2 | \mathbf{x}_{1:t}^1)$ and $p(\mathbf{x}_t^3, \mathbf{y}_{1:t}^3 | \mathbf{x}_{1:t}^1)$ respectively (summing out variables $\mathbf{x}_{1:t-1}^2, \mathbf{x}_{1:t-1}^3$ from the OTDBN). Hence, after permuting the values of the particles on $\mathcal{X}^2$ (resp. $\mathcal{X}^3$) for a fixed value of $\mathbf{x}_{1:t}^1$, distribution $p(\mathbf{x}_t^2, \mathbf{y}_{1:t}^2 | \mathbf{x}_{1:t}^1)$ (resp. $p(\mathbf{x}_t^3, \mathbf{y}_{1:t}^3 | \mathbf{x}_{1:t}^1)$) remains unchanged. A fortiori, this does not affect the representation of the joint posterior distribution:

$$\int p(\mathbf{x}_{1:t}^1, \mathbf{y}_{1:t}^1) p(\mathbf{x}_t^2, \mathbf{y}_{1:t}^2 | \mathbf{x}_{1:t}^1) p(\mathbf{x}_t^3, \mathbf{y}_{1:t}^3 | \mathbf{x}_{1:t}^1) d\mathbf{x}_{1:t-1}^1 = p(\mathbf{x}_t, \mathbf{y}_{1:t})$$

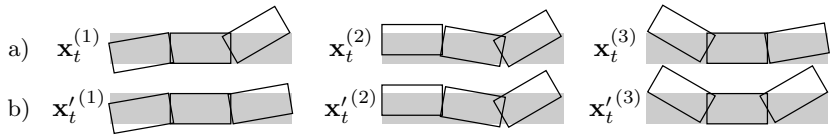


Figure 4-5: **The particle swapping scheme**: (a) before swapping; (b) after swapping

On Figure 4-5(a), particles $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(3)}$ have the same state on $\mathcal{X}^1$. Thus their

right parts can be permuted, resulting in the new particle set of Figure 4-5(b). Remark that we substituted 2 particles, $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(3)}$, which had low weights due to their bad estimation of the object's right or left part states, by one particle $\mathbf{x}_t'^{(1)}$ with a high weight (due to a good estimation of all the object's parts) and another one $\mathbf{x}_t'^{(3)}$ with a very low weight. After resampling, the latter will most probably be discarded and, therefore, swapping will have focused particles on the peaks of the posterior distribution. Note however that not all permutations are allowed: for instance, none can involve particle $\mathbf{x}_t^{(2)}$ because its center part differs from that of the other particles.

The SBPS algorithm introduces these swappings into the diagram of Figure 4-4, which leads to that of Figure 4-6, where operations "$\rightleftharpoons^{P_j}$" refer to the particle subpart swappings briefly described above. Remark that, after the resampling operation of part P_j, the particles with high weights are duplicated, enabling swapping when processing next part P_{j+1}.

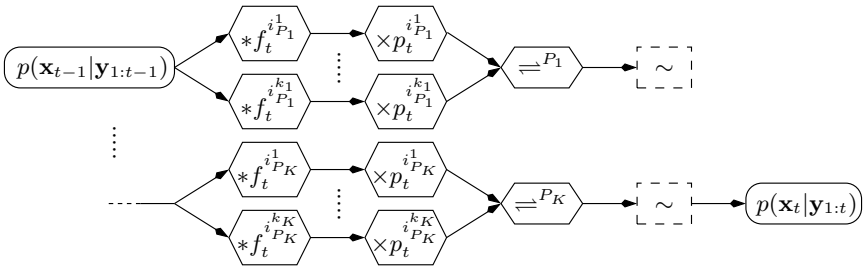


Figure 4-6: **Swapping-based Partitioned Sampling (SBPS) condensation diagram.**

Swappings need however to be further formalized. As $Q_j = \cup_{h=1}^{j} P_h$ and $R_j = \cup_{h=j+1}^{K} P_h$, particle $(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})$ now represents the state of the ith particle after j steps of SBPS and, as shown in the proof of Proposition 4-1, $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j}), w^{(i)}\}$ estimates $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}|\mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. Similarly, when needed, for any set $H \subseteq \{1, \ldots, P\}$, $\mathbf{x}_{1:t}^{(i),H}$ will refer to the whole trajectory of the parts in H of the ith particle, i.e., their values from time slice 1 to time slice t. Now, to guarantee that swapping operations $\rightleftharpoons^{P_j}$ do not alter the estimated distributions, it is not sufficient to permute only the subparts in P_j. The reason why can be easily understood using the OTDBN: the network encodes the joint distribution using conditional densities of the type $p(\mathbf{x}_t^h|\mathbf{pa}(\mathbf{x}_t^h))$, hence permuting only the elements of some subpart $\mathbf{x}_t^k$ in sample $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}$ will change the parents' values of any child $\mathbf{x}_t^r$ of $\mathbf{x}_t^k$, and thus the sample will no longer estimate correctly $p(\mathbf{x}_t^r|\mathbf{pa}(\mathbf{x}_t^r))$ nor the joint probability $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}|\mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. To avoid this, we need to permute $\{\mathbf{x}_t^{(i),r}\}$ similarly to $\{\mathbf{x}_t^{(i),k}\}$. More generally, it is compulsory to permute all the nodes that are linked to $\mathbf{x}_t^k$ by chains that do not pass through $\mathbf{x}_t^k$'s ancestors. More formally, to guarantee

that the estimation of the densities is unaltered by swappings, the set of subparts to permute similarly to $\mathbf{x}_t^k$ is called a "*swapping set*":

Definition 4-2 (Swapping set) *Let $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}$ be the particle set at the jth step of SBPS. Let $k \in P_j$ and let $Link_t(\mathbf{x}_t^k)$ denote the set $\{\mathbf{x}_t^k\} \cup \{\mathbf{x}_t^r$: there exists a chain between $\mathbf{x}_t^k$ and $\mathbf{x}_t^r$ passing only by nodes in time slice t and by no node in the ancestor set $\mathbf{an}_t(\mathbf{x}_t^k)\}$. In addition, let $Link_t^{Q_j}(\mathbf{x}_t^k) = \{\mathbf{x}_t^r \in Link_t(\mathbf{x}_t^k) : r \in Q_j\}$ and $Link_{t-1}^{R_j}(\mathbf{x}_t^k) = \{\mathbf{x}_{t-1}^r : \mathbf{x}_t^r \in Link_t(\mathbf{x}_t^k)$ and $r \in R_j\}$. Then, the set $Link_t^{Q_j}(\mathbf{x}_t^k) \cup Link_{t-1}^{R_j}(\mathbf{x}_t^k)$ is called a swapping set.*

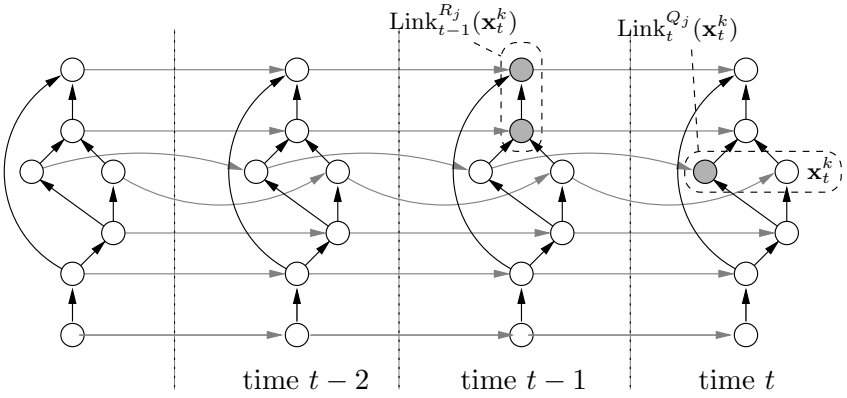


Figure 4-7: **Swapping sets.**

For instance, in the OTDBN of Figure 4-7 (where observation nodes have not been displayed to simplify the figure), the gray nodes represent the swapping set of node $\mathbf{x}_t^k$. We introduced swappings for fixed values of some nodes. As shown below, the d-separation criterion imposes that they correspond to the values of some nodes over *all time slices*. In other words, only *admissible permutations* guarantee that densities are correctly estimated:

Definition 4-3 (admissible permutation)
Let $k \in P_j$ and $A_t = \mathbf{an}_t(\mathbf{x}_t^k) \cap (\cup_{\mathbf{x}_t^h \in Link_t(\mathbf{x}_t^k)} \mathbf{pa}_t(\mathbf{x}_t^h))$. Finally, let $A = \{x_s^h : x_t^h \in A_t, 1 \leq s \leq t\}$. A permutation $\sigma : \{1, \ldots, N\} \mapsto \{1, \ldots, N\}$ is said to be admissible if and only if $\mathbf{x}_t^{(i),h} = \mathbf{x}_t^{(\sigma(i)),h}$ for all $h \in A$ and all $i \in \{1, \ldots, N\}$.

In Figure 4-8, set A_t corresponds to the two black nodes. In Figure 4-9, set A corresponds to the two lines of black nodes in the dotted rectangle at the bottom of the figure. Intuitively, the set A is chosen so that it d-separate Link_t and the rest of the graph. In Figure 4-9, by d-separation it is easy to see that Link_t are independent of the rest of the graph D conditionally to A.

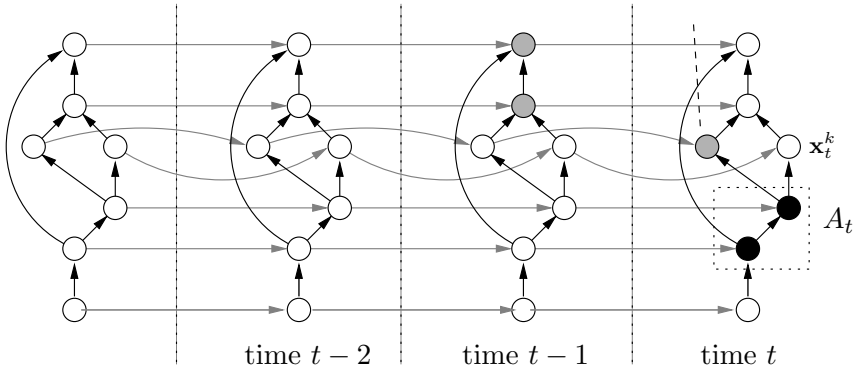


Figure 4-8: **Admissible permutations: set A_t.**

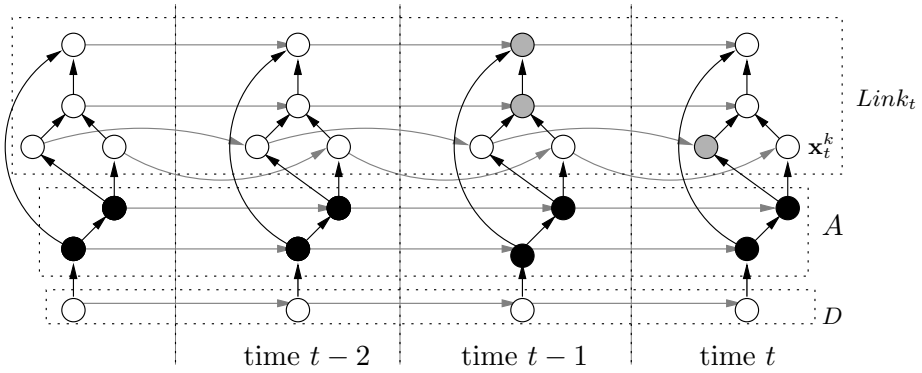


Figure 4-9: **Admissible permutations: whole set A.**

Proposition 4-2 *For any $j \in \{1, \ldots, K\}$, let $\rightleftharpoons^{P_j}$ be a set of admissible permutations σ_k of subparts $k \in P_j$ applied to the swapping set of $\mathbf{x}_t^k$. Then the set of particles resulting from SBPS represents $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.*

Proof of Proposition 4-2: Let $k \in P_j$ and let σ_k be an admissible permutation for $\mathbf{x}_t^k$ as described above. To prove the proposition, it is sufficient to show that, after applying σ_k on all the nodes in the swapping set of $\mathbf{x}_t^k$, the particle set still estimates $p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}|\mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})$. Let us partition $(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j})$ into the following sets:

- $L = \mathrm{Link}_t^{Q_j}(\mathbf{x}_t^k) \cup \mathrm{Link}_{t-1}^{R_j}(\mathbf{x}_t^k)$, i.e., the set of nodes to permute;
- A_t as described in Definition 4-3, i.e., the set of nodes that separate L from the rest of the OTDBN in time slice $t-1:t$;
- $B_t^1 = \mathbf{x}_t^{Q_j} \backslash (\mathrm{Link}_t^{Q_j}(\mathbf{x}_t^k) \cup A_t^{Q_j})$ and $B_{t-1}^2 = \mathbf{x}_{t-1}^{R_j} \backslash \mathrm{Link}_{t-1}^{R_j}(\mathbf{x}_t^k)$.

Duplicate these nodes over all times slices:

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

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

$$\begin{aligned}
p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j}, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j}) &\propto p(\mathbf{x}_t^{Q_j}, \mathbf{x}_{t-1}^{R_j} | \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})\\
&= p(L, A_t, B_t^1, B_{t-1}^2, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})\\
&= \int p(L, A, B_t^1, B_{t-1}^2, \mathbf{y}_{1:t-1}, \mathbf{y}_t^{Q_j})\ d(A\backslash A_t)\\
&= \int p(L, A, B_t^1, B_{t-1}^2, \mathbf{y}^{\text{Link}}, \mathbf{y}^A, \mathbf{y}^B)\ d(A\backslash A_t)\\
&= \int p(A, \mathbf{y}^A) p(L, \mathbf{y}^{\text{Link}} | A) p(B_t^1, B_{t-1}^2, \mathbf{y}^B | L, \mathbf{y}^{\text{Link}}, A) d(A\backslash A_t) \qquad (4.4)
\end{aligned}$$

because, in OTDBNs, conditioning by A is equivalent to conditioning by $A, \mathbf{y}^A$ due to conditions 3 and 4 of Definition 4-1.

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

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

first case, both conditions imply a contradiction. So $\mathbf{c}_{h-1} \notin \text{Link}$ and $\mathbf{c}_{h-1} \notin B$. Therefore, either $\mathbf{c}_{h-1} \in \text{Link}_t^{R_j}(\mathbf{x}_t^k)$ or $\mathbf{c}_{h-1} \in A$. We saw above that the first case leads to a contradiction, so $\mathbf{c}_{h-1} \in A$.

So, for the chain to be active, since we condition on the nodes in A, necessarily the chain has the following arcs: $\mathbf{c}_{h-2} \rightarrow \mathbf{c}_{h-1} \leftarrow \mathbf{c}_h$. Let $\mathbf{x}_z^u = \mathbf{c}_{h-2}$. Again, by definition of OTDBNs, either $(r = z) \wedge (u \neq w)$ or $(r \neq z) \wedge (u = w)$. In the second case, $\mathbf{c}_{h-2}$ is a duplicate of $\mathbf{c}_{h-1}$ in another time slice, hence $\mathbf{c}_{h-2} \in A$ and, by d-separation, the arc $\mathbf{c}_{h-2} \rightarrow \mathbf{c}_{h-1}$ blocks the chain. In the first case, by definition of A, $\mathbf{c}_{h-1} = \mathbf{x}_r^w \in \mathbf{an}_r(\mathbf{x}_r^k)$ and, since $\mathbf{c}_{h-2}$ is a parent of $\mathbf{c}_{h-1}$, $\mathbf{c}_{h-2} \in \mathbf{an}_r(\mathbf{x}_r^k)$. This implies that $\mathbf{c}_{h-2} \notin \text{Link}$. But, by assumption, $\mathbf{c}_h$ is the first node in the chain belonging to B. Hence, necessarily, $\mathbf{c}_{h-2} \in A$ and arc $\mathbf{c}_{h-2} \rightarrow \mathbf{c}_{h-1}$ blocks the chain. Overall, $\{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$ cannot be an active chain. Hence $(B_t^1 \cup B_{t-1}^2 \cup \mathbf{y}^B) \perp\!\!\!\perp L \cup y^{\text{Link}} | A$.

Now, exploiting this independence, Equation 4.4 becomes:

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

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

4.3 Swapping Based Partition Sampling in practice

It is important to note that, to be admissible, a permutation can only swap particles having identical values for *every* object part in A. Actually, in theory, only considering the current time slice may lead to incorrect results. For instance, in the DBN of Figure 4-2, $P_2 = \{2, 3\}$, i.e., object parts $\mathbf{x}_t^2$ and $\mathbf{x}_t^3$ are propagated/corrected in parallel and, therefore, are good candidates for swapping. Now, $\text{Link}_t(\mathbf{x}_t^2) = \{\mathbf{x}_t^2\}$, $A_t = \{\mathbf{x}_t^1\}$. By d-separation, $\text{Link}_t(\mathbf{x}_t^2)$ is not independent from $\mathbf{x}_t^3$ conditionally to A_t because the chain $\{\mathbf{x}_t^2, \mathbf{x}_{t-1}^2, \mathbf{x}_{t-1}^1, \mathbf{x}_{t-1}^3, \mathbf{x}_t^3\}$ is active. Hence, swapping over $\text{Link}_t(\mathbf{x}_t^2)$'s parts, some particles that have the same value on A_t but not on the rest of A, can modify the estimation of the joint posterior density $p(\mathbf{x}_t | \mathbf{y}_{1:t})$ since the $\mathbf{x}_t^3$ part is not swapped accordingly.

However, in practice, whenever two particles have the same value on A_t, the

continuous nature of the state space (location, angle,...) make it highly probable that one particle is a copy of the other due to resampling. Hence, their values on the whole of A should also be identical. In other words, due to the continuous nature of the state space, whenever two particles have precisely the same values on A_t, they should also have the same values on the whole of A. This leads to the following kind of permutations:

Definition 4-4 (almost admissible permutation) *Let $k \in P_j$. A permutation $\sigma : \{1,\ldots,N\} \mapsto \{1,\ldots,N\}$ is said to be almost admissible if and only if $\mathbf{x}_t^{(i),h} = \mathbf{x}_t^{(\sigma(i)),h}$ for all $h \in A_t$ and all $i \in \{1,\ldots,N\}$.*

Our implementation approximates the posterior distributions by performing swappings $\rightleftharpoons^{P_j}$ using almost admissible permutations. The advantage is that we do not need to keep track of the trajectories of the particles from time slices 1 to t. Another improvement can be made in the case where, considering only a single time slice, the OTDBN is a tree or a forest. This corresponds to the case where the structure of the articulated object (a group of articulated objects, in the case of forest) to be tracked is a tree. This is precisely the case in Figure 4-2 and it is often the case in articulated object tracking. In such setting, according to Definition 4-2, $\mathrm{Link}_t(\mathbf{x}_t^k) = \{\mathbf{x}_t^k\} \cup \{$ descendants of $\mathbf{x}_t^k$ in time slice $t\}$. Since $\mathbf{x}_t^{Q_j} \cap \{$ descendants of $\mathbf{x}_t^k$ in time slice $t\} = \emptyset$, we have $\mathrm{Link}_t^{Q_j}(\mathbf{x}_t^k) = \{\mathbf{x}_t^r \in \mathrm{Link}_t(\mathbf{x}_t^k) : r \in Q_j\} = \{\mathbf{x}_t^k\}$. In addition, $\mathrm{Link}_{t-1}^{R_j}(\mathbf{x}_t^k) = \{\mathbf{x}_{t-1}^r : \mathbf{x}_t^r \in \mathrm{Link}_t(\mathbf{x}_t^k)$ and $r \in R_j\} = \{$descendants of $\mathbf{x}_{t-1}^k$ in time slice $t-1\}$. According to Definition 4-3, $A_t = \mathbf{an}_t(\mathbf{x}_t^k) \cap (\cup_{\mathbf{x}_t^h \in \mathrm{Link}_t(\mathbf{x}_t^k)} \mathbf{pa}_t(\mathbf{x}_t^h)) = \mathbf{pa}_t(\mathbf{x}_t^k)$. So we shall permute only particles with the same value of $\mathbf{pa}_t(\mathbf{x}_t^k)$. Note that, when the single time slice subnetworks of OTDBNs are not trees or forest, there may exist some nodes $\mathbf{x}_t^k$ and $\mathbf{x}_t^h$ such that $k, h \in P_j$ and $\mathrm{Link}_t^{Q_j}(\mathbf{x}_t^k) \cap \mathrm{Link}_t^{Q_j}(\mathbf{x}_t^h) \neq \emptyset$. This is for instance the case of $\mathbf{x}_t^k$ and the nodes at the left of $\mathbf{x}_t^k$ in Figure 4-7. In this case, it is useless to consider permuting particles first w.r.t. $\mathbf{x}_t^k$ and, then, w.r.t. $\mathbf{x}_t^h$, the second permutation canceling the first one. In the case where single time slice subnetworks are trees or forests, this problem cannot arise since, as shown above, $\mathrm{Link}_t^{Q_j}(\mathbf{x}_t^k) = \{\mathbf{x}_t^k\}$ and $\mathrm{Link}_t^{Q_j}(\mathbf{x}_t^h) = \{\mathbf{x}_t^h\}$ and, therefore, $\mathrm{Link}_t^{Q_j}(\mathbf{x}_t^k) \cap \mathrm{Link}_t^{Q_j}(\mathbf{x}_t^h) = \emptyset$. Thus, in this case, $\mathrm{Link}_t^{Q_j}(\mathbf{x}_t^h)$ can be safely swapped without affecting any previous or subsequent swappings.

Finally, let us show how $\rightleftharpoons^{P_j}$ can determine attractive swappings, i.e., how high-peaked regions can be discovered. From the preceding paragraph, two particles with the same value on some $\mathbf{pa}_t(\mathbf{x}_t^r)$ have most probably been generated from the duplication of the same particle during a resampling step. In this case, for all $k \in Q_{j-1}$, they also have the same value of $\mathbf{x}_t^k$. We can then partition sample $\{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}$, $i = 1,\ldots,N$, into subsets $N_1,\ldots,N_R$ such that the particles of a set N_r, $r = 1,\ldots,R$,

have the same value of $\mathbf{x}_t^k$ for all $k \in Q_{j-1}$. It should be noted that in practice, we do not need to check for all values of substates in Q_{j-1} to construct the subsets $N_1, \ldots, N_R$: it is sufficient to check for the value of only one substate k where $\mathbf{x}_t^k = \mathbf{pa}_t(\mathbf{x}_t^r)$ and $r \in P_j$. More precisely, whenever two particles $\mathbf{x}_t^{(i)}$ and $\mathbf{x}_t^{(j)}$ are such that $\mathbf{x}_t^{(i),k} = \mathbf{x}_t^{(j),k}$ where $\mathbf{x}_t^k = \mathbf{pa}_t(\mathbf{x}_t^r)$ and $r \in P_j$, then they are in the same partition. Among each set N_r, all possible permutations are eligible. Let $\{r_1, \ldots, r_s\}$ be the elements of N_r. For each $k \in P_j$, let σ_k be the permutation that sorts weights $w_t^{(r_h),k}$, $h = 1, \ldots, s$, in decreasing order. By applying σ_k for all nodes in $\text{Link}_t^{Q_j}(\mathbf{x}_t^k) \cup \text{Link}_{t-1}^{R_j}(\mathbf{x}_t^k)$ and all $k \in P_j$, we get a permutation operation $\rightleftharpoons^{P_j}$ that assigns to the first particle the set of the highest weights, to the second one, the set of second best weights, *etc.* Thus, the first particles have the highest weights, and the last ones the lowest (they will thus be discarded at the next resampling step). Consider the example in Figure 4-10, which corresponds to the human body model in Figure 3-5, in which we omitted the head part for clarity reasons. The left table and right table represent the particle set before and after swapping, respectively. Here, we have a particle set of size $N = 5$. Assume that parts $P_j = \{3, 5\}$, i.e., the forearms, have just been processed and swappings are now performed to obtain an improved particle set. The initial particle set can be partitioned into $R = 2$ particle sets, where $N_1 = \{1, 2, 3\}$ and $N_2 = \{4, 5\}$, i.e., the first set contains the top 3 particles (rows) and the second set the 2 particles at the bottom. Actually, the first 3 particles (resp. the last 2) have the same values on parts 1, 2 and 4, which correspond to the values of $\mathbf{x}_t^{Q_{j-1}}$. For $k = 3$, the swapping operation sorts the first 3 values in the column corresponding to part 3 in decreasing order and also sorts the *4*th and *5*th values in this column in decreasing order. The same procedure is performed for $k = 5$, i.e., in the column corresponding to part 5. This results in the particle set in the right table of Figure 4-10, where the first particle in each partition is assigned the set of the highest weights, the second particle in each partition is assigned the set of the second highest weights, *etc.*

parts:	3	2	1	4	5
	6	1	0	3	9
	5	1	0	3	12
	7	1	0	3	8
	11	2	0	4	13
	10	2	0	4	14

3	2	1	4	5
7	1	0	3	12
6	1	0	3	9
5	1	0	3	8
11	2	0	4	14
10	2	0	4	13

Figure 4-10: **Example before (left) and after swapping (right).** Each row represents a particle $\mathbf{x}_t^{(i)}$ and each number a value $\mathbf{x}_t^{(i),j}$ of part j of the particle.

The following proposition shows that this process results in the permutation that increases the more the sum of the particles (when function f is linear) and that this tends to increase more the weights of particles with high weights (when function f is strictly convex). This justifies that our swappings tend to move the particles toward the modes of the distributions.

Proposition 4-3 *Let $f : \mathbb{R} \mapsto \mathbb{R}$ be a strictly increasing convex function. Let $P_j = \{i_1, \ldots, i_{|P_j|}\}$ be a set of conditional independent subparts and let $\{w^{(i),k}\}$, $i \in 1, \ldots, N$, $k \in P_j$, be the set of weights of N particles on subpart k. Let $\{v^{(i),k}\}$, $i \in 1, \ldots, N$, $k \in P_j$, be the weights resulting from the application of $\rightleftharpoons^{P_j}$ on $\{w^{(i),k}\}$. Finally, let Σ_{i_r}, $r \in \{1, \ldots, |P_j|\}$, denote the set of almost admissible permutations over $\mathbf{x}_t^{i_r}$ and let $\Sigma = \Sigma_{i_1} \times \cdots \times \Sigma_{i_{|P_j|}}$. Then $\rightleftharpoons^{P_j}$ is the unique set of permutations such that:*

$$\sum_{i=1}^{N} f\left(\prod_{k \in P_j} v^{(i),k}\right) = \max_{(\sigma_{i_1}, \ldots, \sigma_{i_{|P_j|}}) \in \Sigma} \sum_{i=1}^{N} f\left(\prod_{k \in P_j} w^{(\sigma_k(i)),k}\right) \tag{4.5}$$

Proof of Proposition 4-3: First, consider 2 particles $a, b \in 1, \ldots, N$, whose weights, say $t^{(a)}$ and $t^{(b)}$, are such that $\prod_{k \in P_j} t^{(a),k} \geq \prod_{k \in P_j} t^{(b),k}$ and such that there exists $h \in P_j$ such that $t^{(b),h} > t^{(a),h}$. Let $\rho = t^{(b),h} - t^{(a),h}$ and denote:

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

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

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

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

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

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

Let σ be a permutation maximizing $\sum_{i=1}^{N} f(\prod_{k\in P_j} w^{s_k(i),k})$ over all permutations $(s_1,\ldots,s_{|P_j|}) \in \mathcal{S}$. Denote by $v^{(i),k}$ the weights $w^{\sigma_k(i),k}$ resulting from the application of σ on the original particle weights $\{w^{(i),k}\}$. Without loss of generality, assume that σ sorts the "total" particle weights in decreasing order, i.e., that $\prod_{k\in P_j} v^{(a),k} \geq \prod_{k\in P_j} v^{(b),k} \iff a \leq b$. By the preceding paragraph, there cannot exist $h \in P_j$ such that there exists $i \geq 2$ such that $v^{(i),h} > v^{(1),h}$, else by permuting $v^{(i),h}$ and $v^{(1),h}$, we would get a permutation strictly increasing the sum of f(weights). For the same reason, there cannot exist $h \in P_j$ such that there exists $i \geq 3$ such that $v^{(i),h} > v^{(2),h}$. By induction, we have that, for any r, there cannot exist $h \in P_j$ such that there exists $i > r$ such that $v^{(i),h} > v^{(r),h}$. This precisely correspond to the permutation operation $\rightleftharpoons^{P_j}$. Therefore, $\rightleftharpoons^{P_j}$ is the unique permutation satisfying Equation 4.5. □

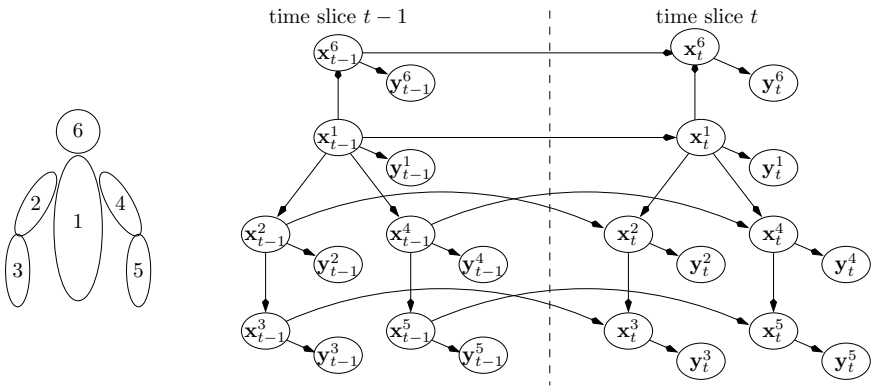


Figure 4-11: **The human body model used for tracking.**

Example. Considering the example of human body tracking in Fig 3-5. Fig 4-11 shows the DBN used for modeling this problem.

SBPS starts by processing part 1 (which includes the propagation, correction and resampling of part 1). At the next step, SBPS processes the children parts of part 1 in parallel, i.e. parts $\{2,4,6\}$. Figure 4-12 (first two rows) shows all possible permutations after the propagations of $\{2,4,6\}$. At the final step, SBPS

processes the children part of $\{2, 4, 6\}$ in parallel, i.e. parts $\{3, 5\}$. Figure 4-12 (last row) shows all possible permutations after the propagations of $\{3, 5\}$.

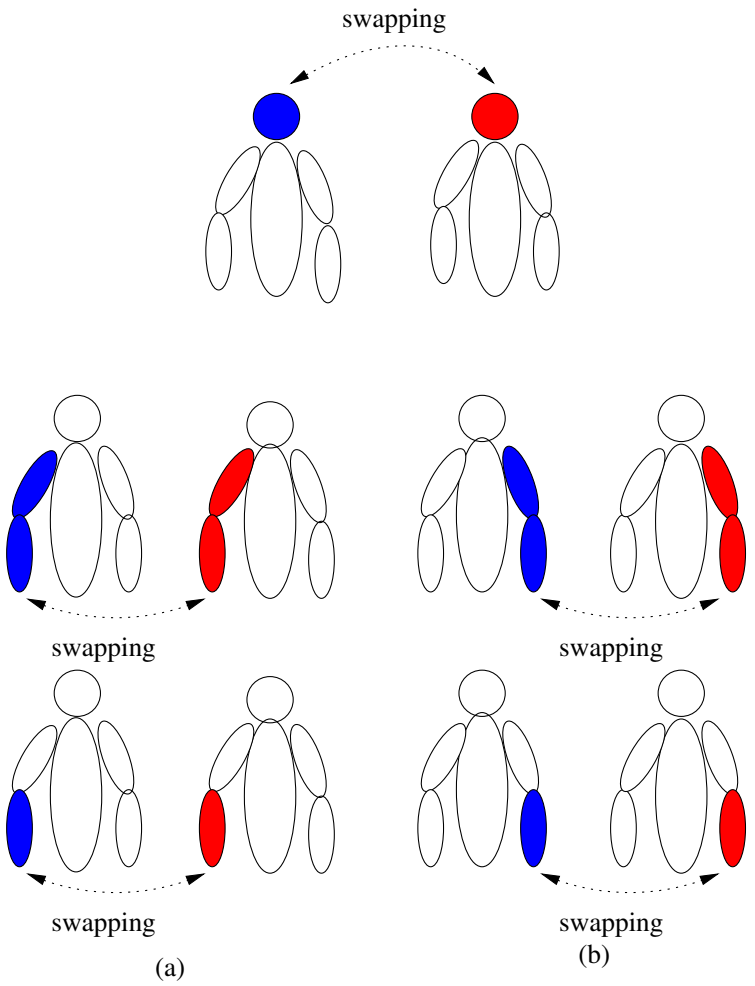


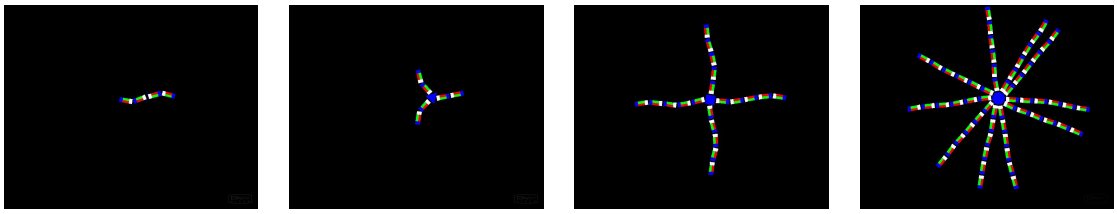
Figure 4-12: **Illustration of the swapping principle between two configurations (blue and red).** Top and middle lines: head, right arms and left arms swappings with identical torso. Last line, left: right lower arm swapping with the same value for the right forearms. Last line, right: left lower arm swapping with the same value for the left forearms.

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

4.4 Experimental results

4.4.1 Video sequences

Our experiments are first performed on synthetic video sequences so that we do not have to take into account specific properties of images (noise, *etc.*) and, additionally, we can simulate specific motions. This thus allows to compare particle filters on the sole basis of their estimation accuracy and computation times and, moreover, to focus comparisons on different features. Therefore, we have generated our own synthetic video sequences composed of 300 frames of 800×640 pixels. Each video displays an articulated object randomly moving and deforming over time, subject to weak or strong motions. Some examples are given in Figure 4-13. With various numbers of parts, articulated objects are designed to test the capacity of particle filters to deal with high-dimensional state spaces.



$K = 3$, $|P_j| = 2$
$P = 5$, $|\mathcal{X}| = 15$

$K = 3$, $|P_j| = 3$
$P = 7$, $|\mathcal{X}| = 21$

$K = 6$, $|P_j| = 4$
$P = 21$, $|\mathcal{X}| = 63$

$K = 7$, $|P_j| = 10$
$P = 61$, $|\mathcal{X}| = 183$

Figure 4-13: **Examples of frames extracted from our synthetic video sequences (300 frames of** 800×640 **pixels).** The features of the corresponding articulated objects (number of arms $|P_j|$, $j > 1$, length of arms $K - 1$, total number of parts P, and dimension of state space $\mathcal{X}$) are reported below.

4.4.2 Experimental setup

The tracked objects are modeled by a set of P polygonal parts (or regions): a central one P_1 (containing only one polygon) to which are linked $|P_j|$, $j > 1$, arms of length $K - 1$ (see Section 4.1, and Figure 4-13 for some examples). State vectors contain the parameters describing all the parts and are defined by $\mathbf{x}_t = \{\mathbf{x}_t^1, \ldots, \mathbf{x}_t^P\}$, with $\mathbf{x}_t^k = \{x_t^k, y_t^k, \theta_t^k\}$, where (x_t^k, y_t^k) is the center of part k, and θ_t^k is its orientation, $k = 1, \ldots, P$. We thus have $|\mathcal{X}| = 3P$. A particle $\mathbf{x}_t^{(i)} = \{\mathbf{x}_t^{(i),1}, \ldots, \mathbf{x}_t^{(i),P}\}$, $i = 1, \ldots, N$, is a possible spatial configuration, i.e., a realization, of the articulated object. Particles are propagated using a random walk whose variance has been empirically fixed for all tests ($\sigma_x = 1$, $\sigma_y = 1$ and $\sigma_\theta = 0.025$ rad).

A classical approach to compute particle weights consists of integrating the color distributions given by histograms into particle filtering [75]. In such cases, we measure the similarity between the distribution of pixels in the region of the estimated part of the articulated object and that of the corresponding reference region. The particle weights are then computed by $w_{t+1}^{(i)} = w_t^{(i)} p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}^{(i)}) \propto w_t^{(i)} e^{-\lambda \mathbf{d}^2}$, with, in our tests, $\lambda = 50$ and $\mathbf{d}$ the Bhattacharyya distance [10] between the target (prior) and the reference (previously estimated) 8-bin histograms. For all the compared algorithms, the tracking is manually initialized in the first frame of the tested sequence.

In order to show the effectiveness of SBPS and of its swapping operation, we compare SBPS, PS, APF with 1 layer of annealing and a variant of APF which we called SBAPF, which consists of adding a swapping after a layer of annealing. For all the compared filters, the articulated object's joint distribution is estimated starting from its central part P_1. Then, PS propagates and corrects particles polygon after polygon to derive a global estimation of the object (see the condensation diagram of Figure 3-4). Similarly, APF adds an optimization step (annealing) to estimate the parts of the articulated object sequentially. Quite differently, SBPS and SBAPF considers the object's arms as independent conditionally to the central part, and thus propagate, correct and swap simultaneously the P_j's parts, the jth joints of all the arms, for all $j = 2, \ldots, K$ (see Figure 4-6). SBPS, PS, SBAPF and APF are compared w.r.t. two criteria: computation times and estimation errors. The latter are given by the sum of the Euclidean distances between each corner of the estimated parts and its corresponding corner in the ground truth we have generated with the video sequences. All the results presented here are a mean over 250 runs performed on a PC with a 3.07 GHz Intel Core i7.

4.4.3 Qualitative tracking results

This subsection is dedicated to visual tracking evaluation. Figure 4-14 and 4-15 shows tracking results obtained with SBPS, PS, SBAPF and APF on two different video sequences: estimations correspond to the white articulated object superposed to the real images, and are the average (weighted sum) of all the particles. Only frames 50, 150 and 250 are displayed and, moreover, zooms focusing on the objects are made to better distinguish the discrepancy between the estimated and the real objects. As can be visually seen, SBPS (SBAPF) seems to be more robust than PS (APF) for tracking these objects. In Figure 4-16, the five best particles (i.e. those with the highest weights) are drawn in white. Here again, SBPS (SBAPF) concentrates more the best particles around the object (i.e., around the modes of the distribution to estimate) than PS (APF).

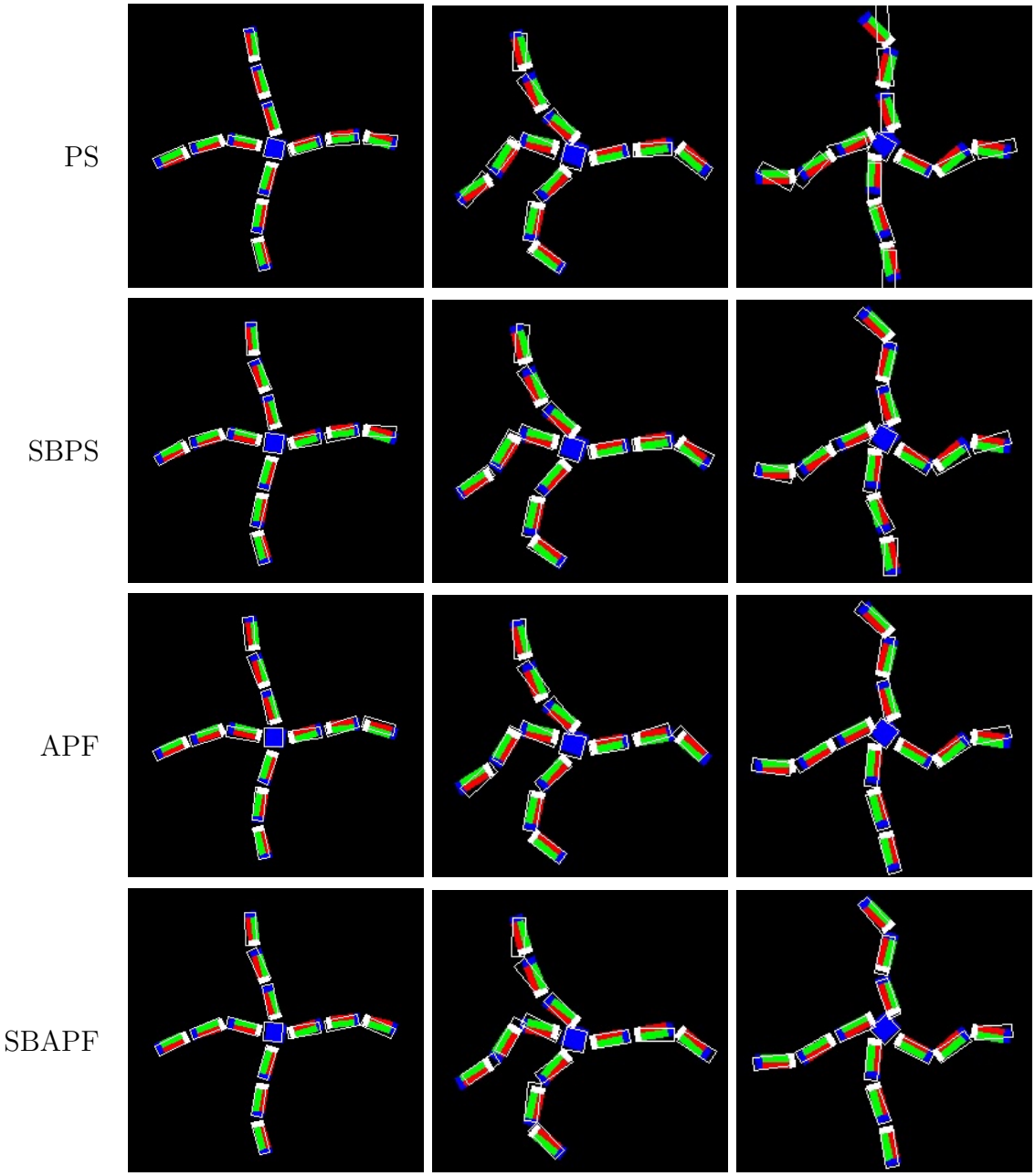


Figure 4-14: **Qualitative tracking results.** Sequence with object with $K = 3$, $|P_j| = 4$ ($|\mathcal{X}| = 15$), $N = 50$ particles. Estimated object (particles' average) is superposed in white on frames 50, 150 and 250.

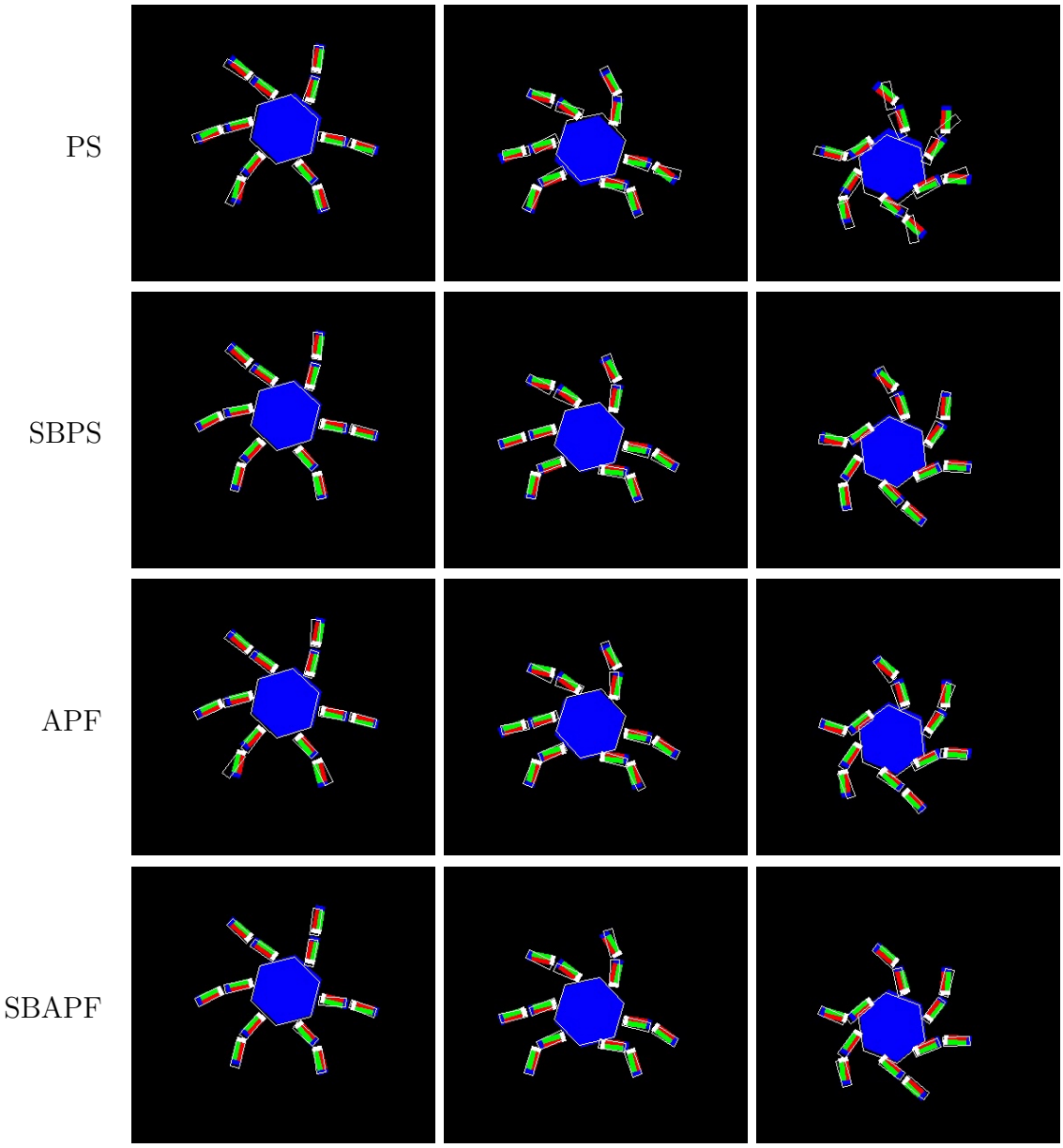


Figure 4-15: **Qualitative tracking results.** Sequence with object with $K = 2$, $|P_j| = 6$ ($|\mathcal{X}| = 15$), $N = 100$ particles. Estimated object (particles' average) is superposed in white on frames 50, 150 and 250.

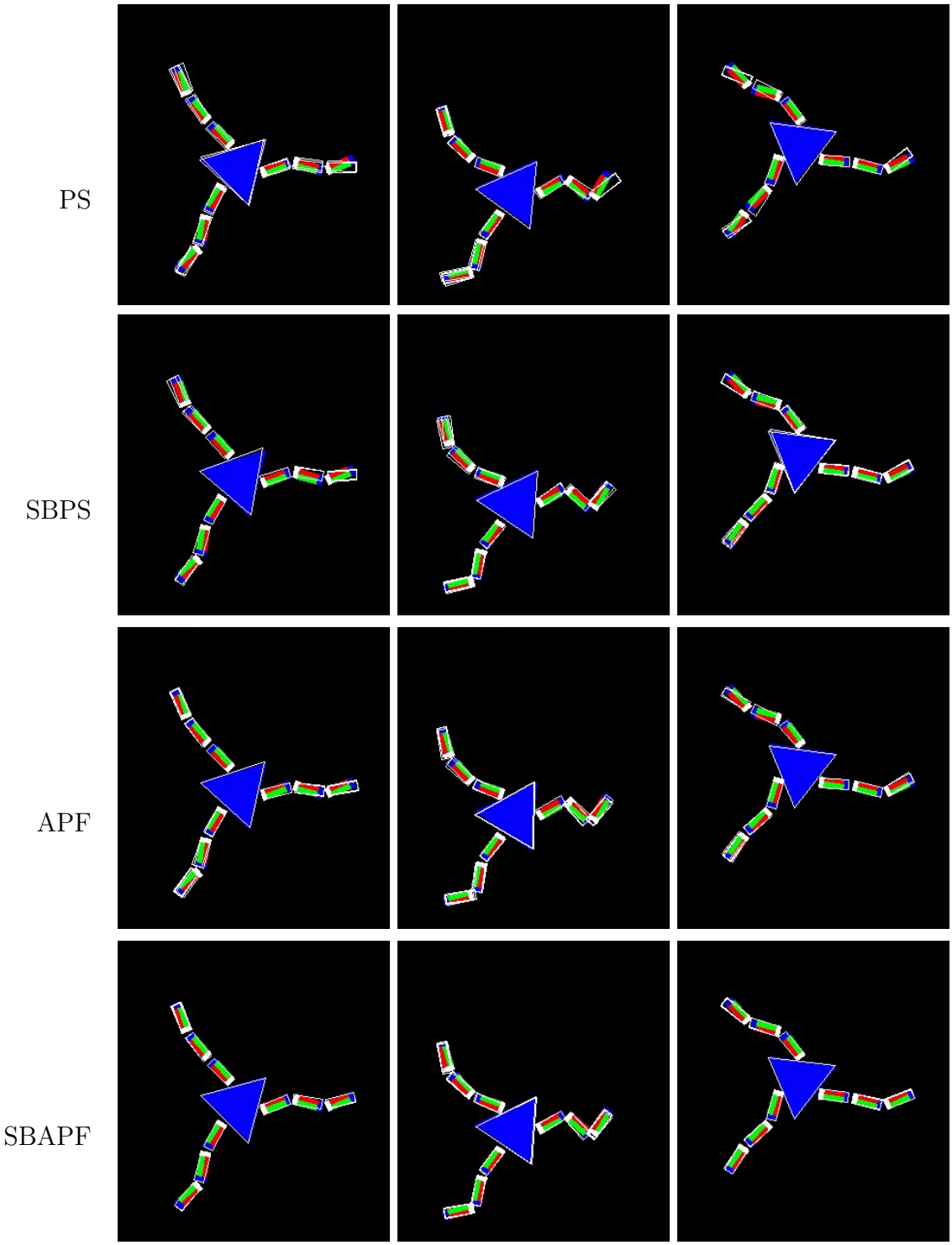
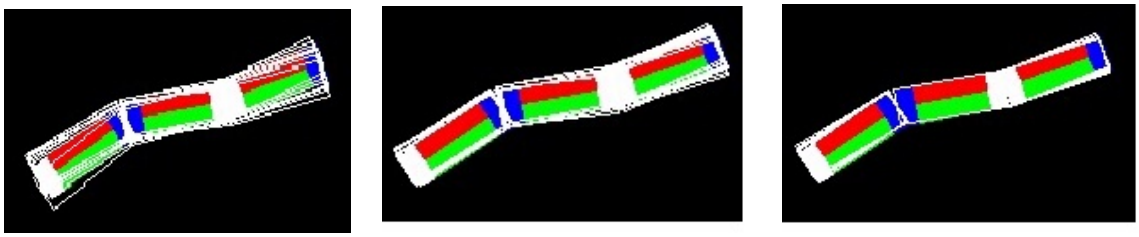


Figure 4-16: **Qualitative tracking results.** Sequence containing one object with $K = 3$, $|P_j| = 3$ ($|\mathcal{X}| = 12$). The five best particles (i.e. those with highest weights) are superposed on the frame 50, 150 and 250 (zooms).

All these qualitative results highlight the fact that, by embedding a swapping step, SBPS (SBAPF), seems to be more robust than PS (APF). Finally, we show in Figure 4-17 the effect of swapping: particles are more concentrated on the object, i.e. on the modes of the density to estimate.



(**a**) (**b**) (**c**)

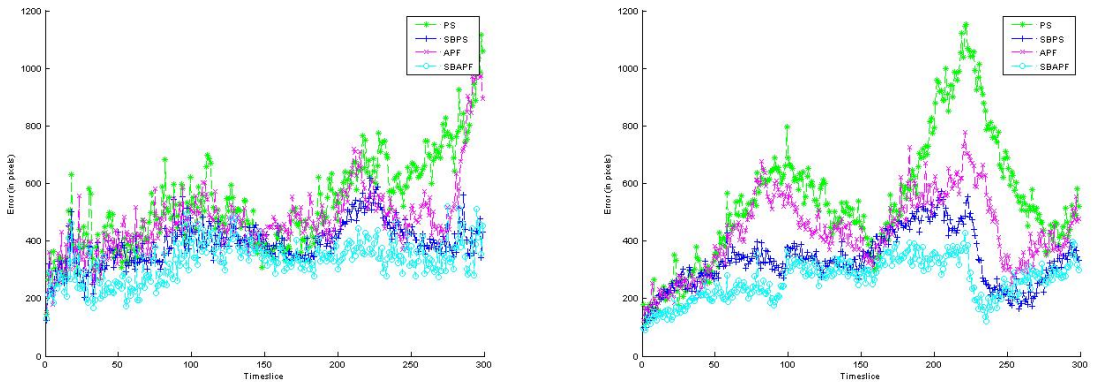
Figure 4-17: **The effectiveness of the swapping operation.** Figure (**a**) shows the particle set obtained by PS after processing the center part. Starting from this particle set, Figure (**b**) shows the particle set obtained by PS after processing the two extremities sequentially and Figure (**c**) shows the particle set obtained by processing the two extremities in parallel and then applying the swapping operation. The particle set in (**c**) are clearly more focused on the mode of the posterior distribution of the target object than that in (**b**).

The next subsections are dedicated to quantitative results and will confirm these first observations.

4.4.4 Quantitative tracking results

Specific synthetic sequences have been generated in which the object is more strongly deforming and moving during some time intervals ($[80-120]$ or $[200-250]$). Figure 4-18 shows the tracking errors over time for two objects defined in state spaces whose dimensions are 28 and 35 respectively. As can be seen, SBPS (SBAPF) is less affected than PS (APF) by strong motions of the object. This demonstrates the robustness of the proposed approach, in particular of the swapping operation.

One of the main features of SBPS and SBAPF is their ability to simultaneously deal with independent parts, making them effective for tracking in high-dimensional spaces. We then have performed experiments varying the two factors that contribute to increasing the state space dimensions: the number of parts that can be simultaneously treated (given by parameter $|P_j|$), and the length of the object's arms (given by parameter K). For a fair comparison, the number of particles used by methods are determined so that they evaluate the same number of times the likelihood function at each time step. For instance, if for PS and SBPS we have $N = 100$, then the number



(**a**) (**b**)

Figure 4-18: **Estimation errors for SBPS, PS, SBAPF and APF.** Some interval times are subject to strong motions and deformations of the object ($[80-120]$ and $[200-250]$). (**a**) Object with $K=5$, $|P_j|=5$ ($P=26$, $|\mathcal{X}|=28$), $N=80$. (**a**) Object with $K=4$, $|P_j|=8$ ($P=33$, $|\mathcal{X}|=35$), $N=100$.

of particles used by SBAPF and APF with one layer is $N=50$. Indeed, if SBAPF and APF use L layers of annealing, the number of particles used by APF and SBAPF should be divided by $L+1$, since the likelihood function is re-evaluated at each layer. In the rest of this section, N will always correspond to the number of particles used by PS and SBPS, and then APF and SBAPF use $N/(L+1)$ particles. By default, both filters use $L=1$ layer, and then $N/2$ particles.

4.4.4.1 Performances depending on K

In this set of experiments, $|P_j|$ is fixed to 4 (i.e., objects have 4 arms), and K varies from 2 to 8. Experiments were performed with various numbers of particles ($N=50$, $N=100$, $N=200$ and $N=300$). Figures 4-19 and 4-20 show the estimation errors and the standard deviations obtained for four methods. We can observe that SBPS (SBAPF) always outperforms PS (APF) in terms of estimation errors. In addition, the difference of estimation errors between SBPS and PS increases when the length of the object's arms K increases. SBAPF also outperforms APF. This is logical, because SBAPF adds a swapping step after the annealing. Note that SBPS gives higher errors than APF: when K increases, the swapping seems to be less robes than annealing for large values of N.

The standard deviations of SBPS (SBAPF) tend to be lower than those of PS (APF). When N increases, APF and SBAPF are more stable than other approaches, and PS becomes totally destabilized. For any value of N, SBAPF has the lowest and

most stable standard deviation. This shows the interest of the simultaneous independent treatments since those both reduce the estimation errors and their standard deviations. We will show in the next section that SBPS (SBAPF) also outperforms PS (APF) in terms of computation times, so that the gain in accuracy is not made at the expense of response times.

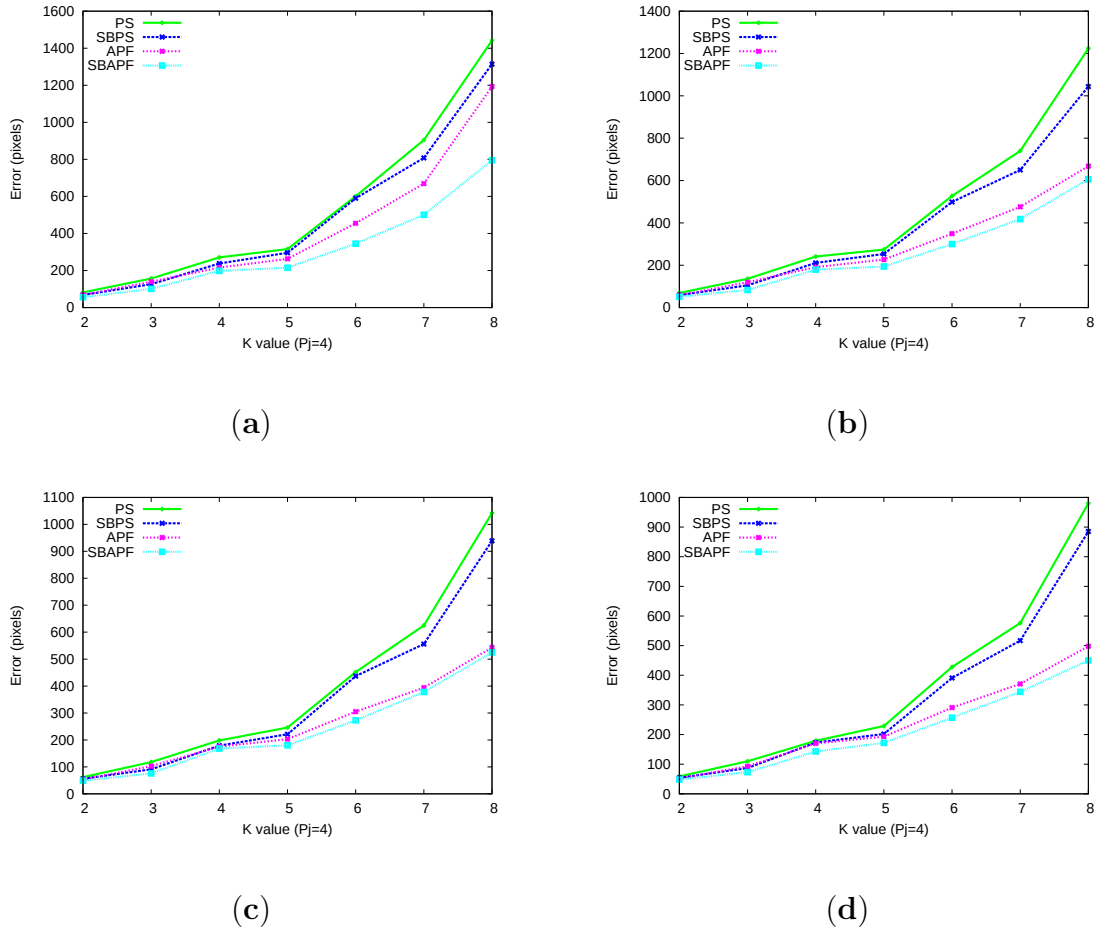


Figure 4-19: **Estimation errors of SBPS, PS, SBAPF and APF for tracking an object with $|P_j| = 4$, depending on K.** (**a**) $N = 50$. (**b**) $N = 100$. (**c**) $N = 200$. (**d**) $N = 300$.

4.4.4.2 Performances depending on $|P_j|$

In this set of experiments, K is fixed to 4 (i.e. the length of arms is 3) , and $|P_j|$ varies from 3 to 8. Again, experiments were performed with various numbers of particles ($N = 50$, $N = 100$, $N = 200$ and $N = 300$). Figures 4-21 and 4-22 show the estimation errors and the standard deviations obtained by the four methods. We can

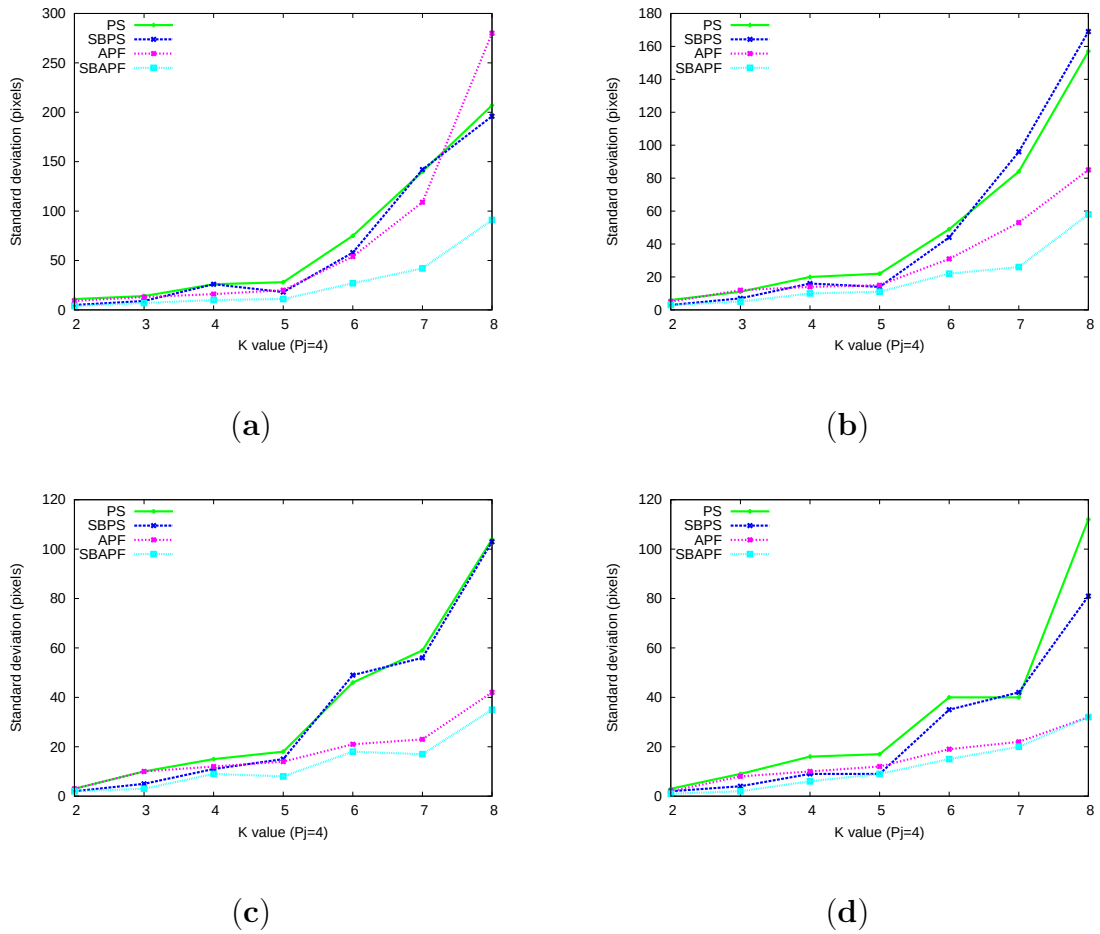


Figure 4-20: **Standard deviations of SBPS, PS, SBAPF and APF for tracking an object with $|P_j| = 4$, depending on K.** (**a**) $N = 50$. (**b**) $N = 100$. (**c**) $N = 200$. (**d**) $N = 300$.

observe some differences with the previous experiments. First of all, for any value of N, the two best filters are SBAPF and SBPS. This means that PS as well as APF are very perturbed by the increase of $|P_j|$.

Concerning standard deviations (Figure 4-22), here again, more stable results are obtained by SBPS and SBAPF. One can note that increasing $|P_j|$ perturbs less all methods than increasing K: standard deviations are divided by 3 or 4, and estimation errors are also lower. This can be explained by the fact that the sequential scheme adapted by all the approaches (inspired from PS's scheme) add noise iteratively and, thus, first parts in the process are better estimated than last parts: the longer the arms, the higher the estimation errors and standard deviations.

The fact that SBPS outperforms APF in terms of both estimation errors and

standard deviations shows that our swapping step is particularly effective when a lot of parts can be computed in parallel, compared to an optimization approach.

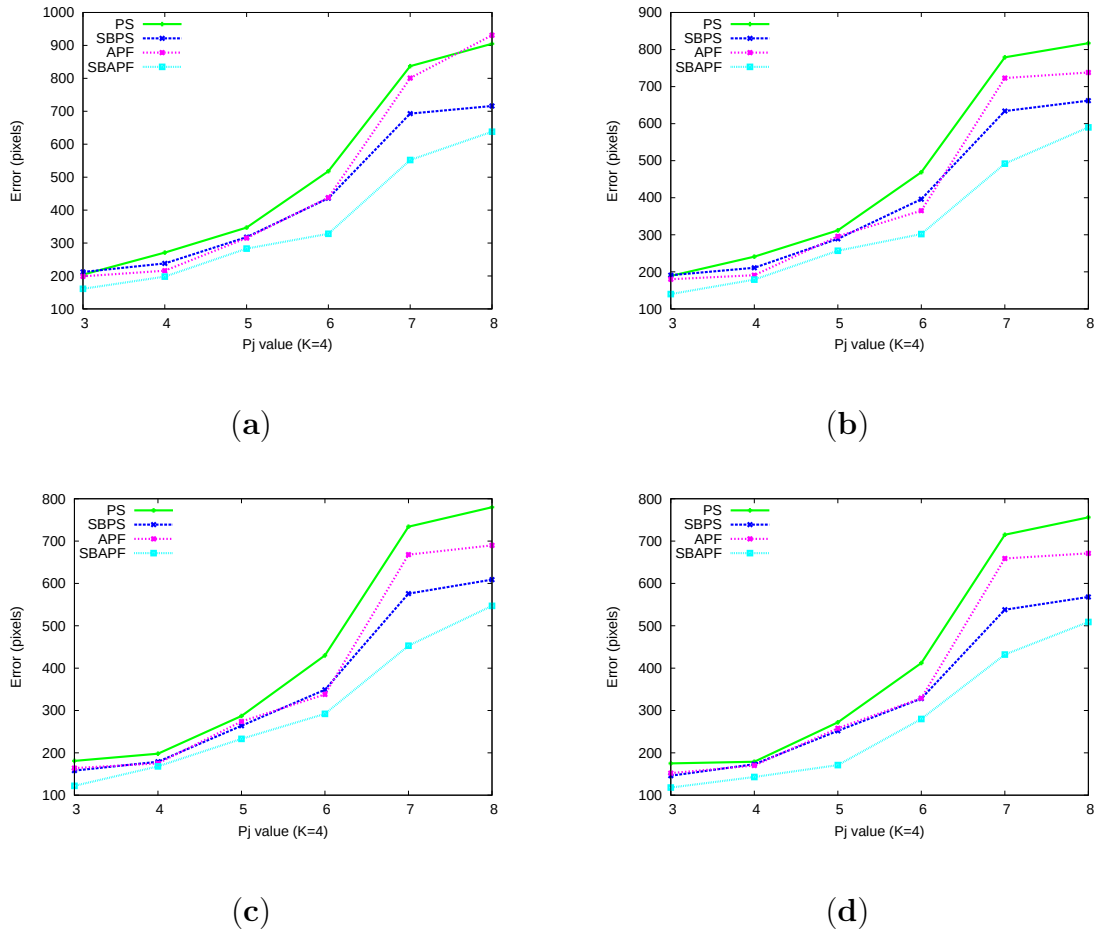


Figure 4-21: **Estimation errors of SBPS, PS, SBAPF and APF for tracking an object with $K = 4$, depending on $|P_j|$.** (**a**) $N = 50$. (**b**) $N = 100$. (**c**) $N = 200$. (**d**) $N = 300$.

4.4.4.3 Performances depending on N (convergence study)

We studied the convergence of SBPS, PS, SBAPF and APF in terms of the evolution of the tracking errors and the standard deviations w.r.t. the number N of particles. Figure 4-23 shows the estimation errors obtained for the four methods for two different tracked objects: one with long arms ($K = 8$ and $|P_j| = 4$), the other one with a lot a arms ($K = 4$ and $|P_j| = 6$). For both objects, we can observe that SBPS (SBAPF) converges faster than PS (APF): we can see on both examples the interest of the swapping step on PS and APF. As in the previous subsection, we observe that when

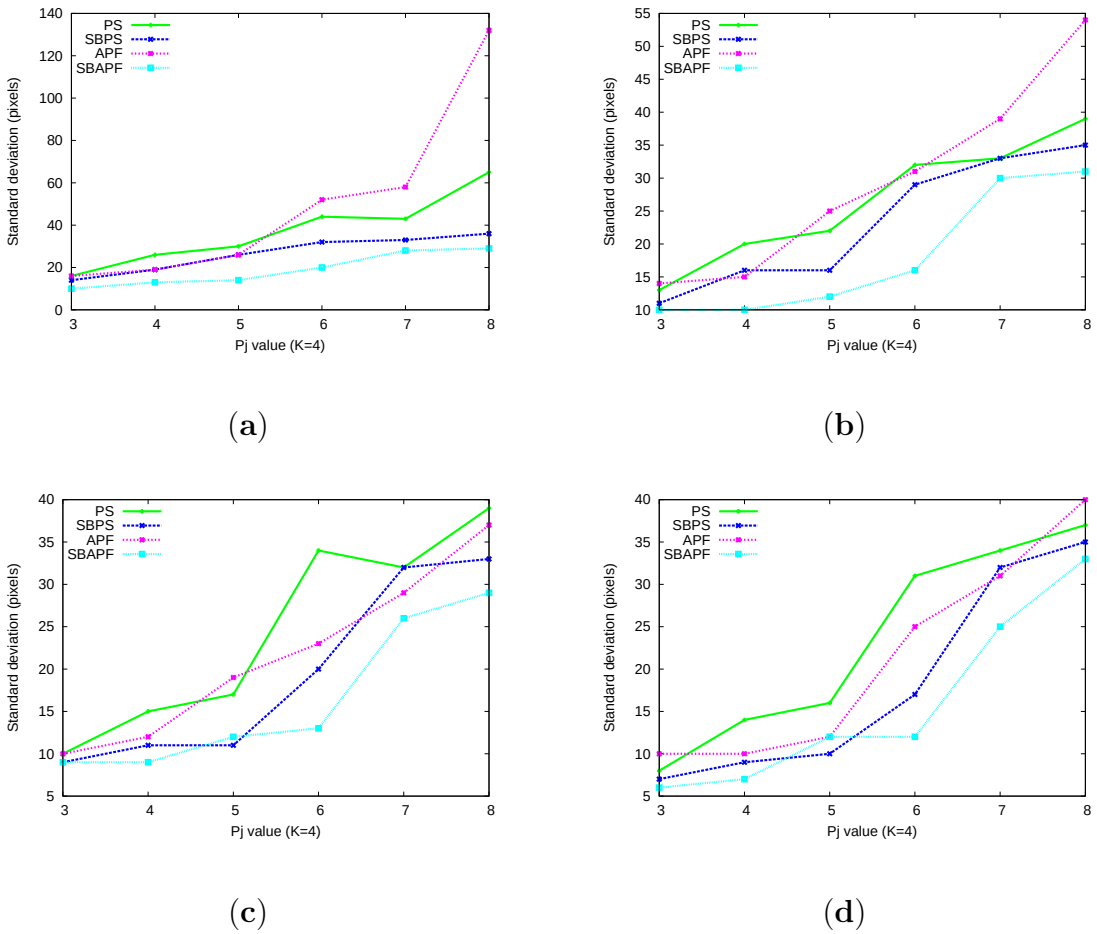


Figure 4-22: **Standard deviations of SBPS, PS, SBAPF and APF for tracking an object with $K = 4$, depending on $|P_j|$.** (**a**) $N = 50$. (**b**) $N = 100$. (**c**) $N = 200$. (**d**) $N = 300$.

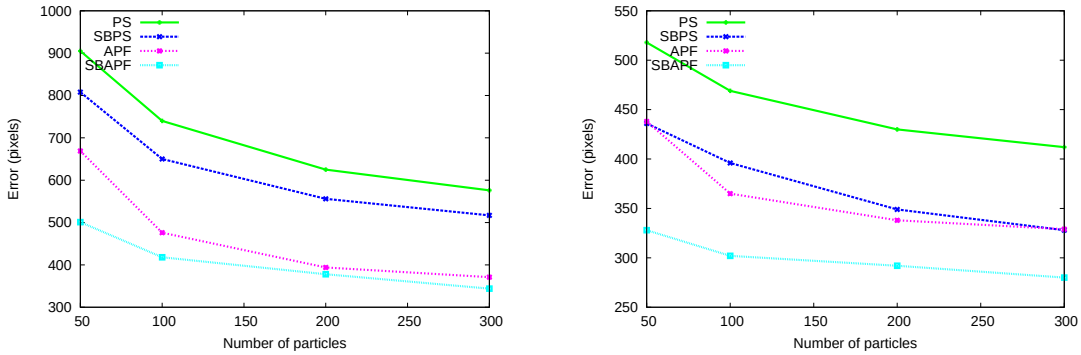


Figure 4-23: **Convergence study (estimation errors) for SBPS, PS, SBAPF and APF.** From left to right, an object with $K = 8$, $|P_j| = 4$ ($P = 33$, $|\mathcal{X}| = 35$), and an object with $K = 4$, $|P_j| = 6$ ($P = 25$, $|\mathcal{X}| = 27$).

dealing with a lot of arms (i.e. a lot of parts are processed in parallel), SBPS becomes equivalent to APF. SBAPF converges faster and better than all the other approaches: we need less particles to achieve lower estimation errors.

The standard deviations of four methods are given in Figure 4-24. SBPS's standard deviations converge at the same rate as that of PS when tracking object $K = 8$ and $|P_j| = 4$, and faster than PS when tracking object $K = 4$ and $|P_j| = 6$. SBAPF's standard deviation is always lower and more stable than the other approaches' one. Note here again the fact that increasing $|P_j|$ increases the interest of using the swap-

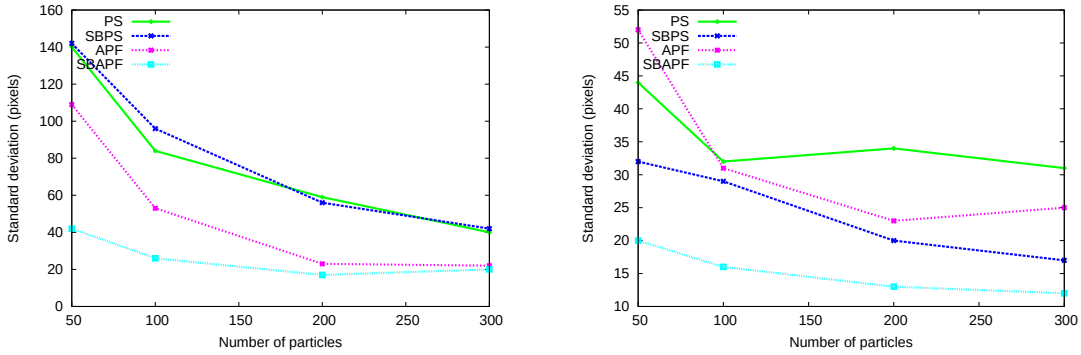


Figure 4-24: **Convergence study (standard deviations) for SBPS, PS, SBAPF and APF.** Two objects in Figure 4-23 were used. (left) object with $K = 8$, $|P_j| = 4$ and (right) object with $K = 4$, $|P_j| = 6$.

ping step (the standard deviation for such cases is always lower for SBPS and SBAPF).

To conclude, we have shown that our approach, SBPS, consisting in swapping independent subparts, decreases significantly the estimation errors as well as their standard deviation compared to PS. We have also shown the interest of the swapping operation when incorporating it into the APF framework, since in all experiments, SBAPF outperforms APF in terms of both estimation errors and standard deviations. The next section is dedicated to a comparison between an optimization process (i.e. the annealing) and our swapping operation in terms of estimation errors.

4.4.4.4 Swapping versus Annealing

One important factor that impacts on the performances of APF is the number of layers used for annealing. As shown previously, SBAPF, which incorporates a swapping step into the APF framework, improves significantly the performance of APF. This is however logical to think that adding swapping as a supplementary step after annealing could improve the efficiency of APF. That is why we investigate more deeply in the next paragraph the performances of APF in terms of number of layers and compare its estimation errors with those of SBAPF.

We thus have made the number L of layers of APF vary and have computed the estimation errors obtained for the two objects of the previous section (i.e. with $K = 8$, $|P_j| = 4$ and with $K = 4$, $|P_j| = 6$). We compared the results with those obtained for SBAPF with one layer. The number of particles N is fixed to 300 for SBAPF, and $N/(L+1)$ for APF. Figure 4-25 shows the corresponding estimation errors and their standard deviations obtained by APF (depending on L) in comparison with those by SBAPF (here $L = 1$). As can be observed, APF achieves its best performance when the number of layers is two (when tracking the object with $K = 8$, $|P_j| = 4$) or three (when tracking the object with $K = 4$, $|P_j| = 6$). In all cases, SBAPF always outperforms APF in terms of both estimation errors and standard deviations. These results show the interest of the swapping operation: adding one swapping step after annealing is sufficient to achieve performances that APF will never reach. Moreover, this figure also shows a strong disadvantage of APF: after two layers (when tracking the object with $K = 8$, $|P_j| = 4$) and three layers (when tracking the object with $K = 4$, $|P_j| = 6$), the set of particles of APF get stuck into local optima and thus cannot improve anymore the tracking performance (indeed, above 3 layers, the performance tends to decrease). On the contrary, by permuting some subsamples of the object parts so that the best subsamples are combined into new particles, the swapping operation can shift the particle set toward the modes of the posterior distribution and thus improves significantly the performances.

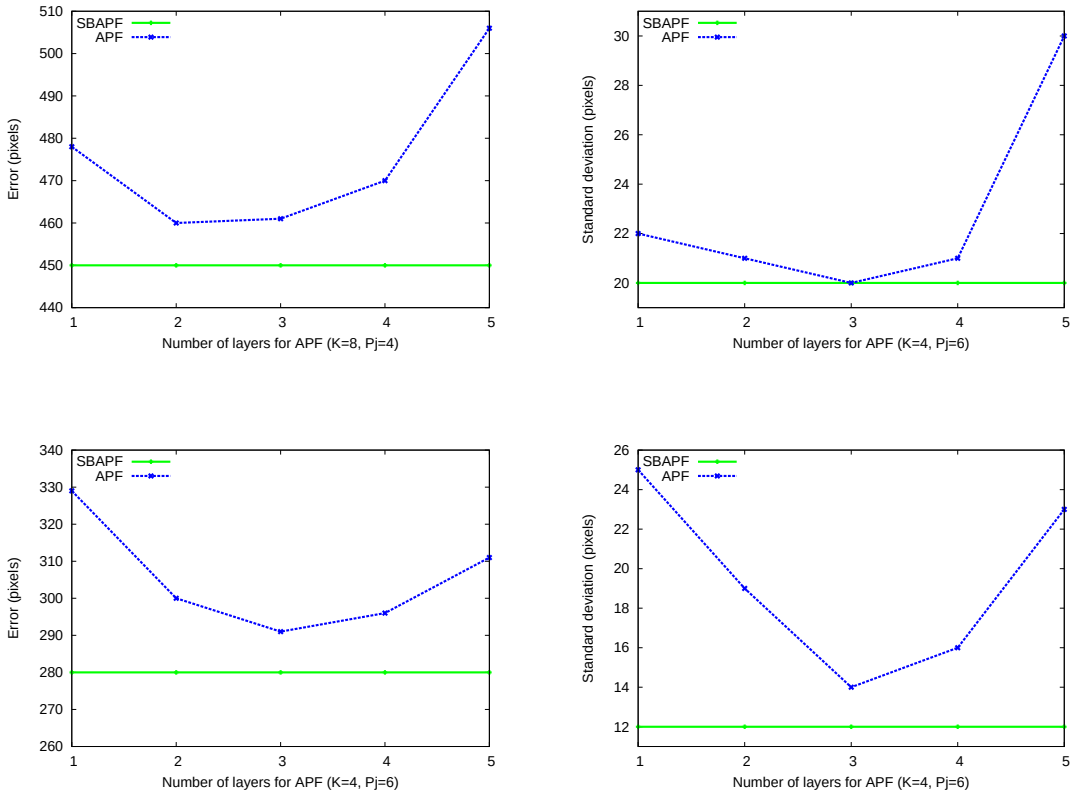


Figure 4-25: **Role of the number of layers.** Comparison between APF's and SBAPF's performances. First row, from left to right: estimation errors and standard deviations when tracking an object with $K = 8$, $|P_j| = 4$. Second row, from left to right: estimation errors and standard deviations when tracking an object with $K = 4$, $|P_j| = 6$.

In the next section, we show that all these improvements are not achieved at the expense of computation times.

4.4.5 Computation times

The global computation time of the tracking is supposed to be measured by considering all the different steps of the algorithms: propagations, corrections and resampling for classical approaches (PS and APF), plus our swapping step for SBPS and SBAPF. For all of these methods, propagations and corrections (likelihood computations) take the same time. Thus, we only need to consider resampling and swapping to compare

computation times. In Section 4, we argued that, by simultaneously processing independent parts of the objects, SBPS reduces the number of resamplings compared to PS (this is also true for SBAPF when comparing to APF). However, it also introduces a new swapping step that may potentially increase the global computation times. To show that this is not the case, as in the previous subsection, we studied the behaviors of SBPS, PS, SBAPF and APF in terms of resampling computation times and swapping computation times depending on K, $|P_j|$ and N.

4.4.5.1 Computation times depending on K

Figures 4-26 reports the resampling times for all approaches depending on K, i.e., the length of the $|P_j| = 4$ arms of an object tracked with $N = 50$, $N = 100$, $N = 200$ and $N = 300$ particles. Note that for SBPS and SBAPF, swapping times are included in these resampling times (see Figure 4-27 for more details). For any value of K, and for the 4 values of N tested, we note that SBPS and SBAPF require less computation times for resampling and swapping step than PS and APF need only for resampling. Moreover, this difference increases with K. For example, with $N = 300$ and $K = 9$, SBAPF is 1.2 second faster than APF (21% faster), and SBPS 0.6 second faster (10% faster).

Figure 4-27 shows separately the resampling and swapping times for SBPS in comparison with the resampling times for PS (those of SBAPF and APF are not shown since the comparison gives the same conclusions). The most significant observation is that when K increases, PS resampling times drastically increase compared to those of SBPS. This is due to the swapping step, that enables to construct better particles (i.e. with highest weights), located near the modes of the density to estimate, and then to reduce the resampling times. We can conclude that adding the swapping step does not increase the total computation times when K increases.

4.4.5.2 Computation times depending on $|P_j|$

Figures 4-28 and 4-29 show the curves for similar tests while making $|P_j|$, the number of arms of the tracked object, vary. In these experiments, we can observe that the computation times of SBPS (SBAPF) are significantly lower than those of PS (APF). While adding parts to compute in parallel to the articulated object drastically increases the resampling times of PS and APF, those for SBPS and SBAPF increase very slowly. Note that SBPS and SBAPF give similar times. For example, with $N = 300$ particles, and $|P_j| = 8$, SBPS and SBAPF are 1.5 second faster (65% faster). In the same curve (Figure 4-29(d)), we can see that when $|P_j|$ changes from

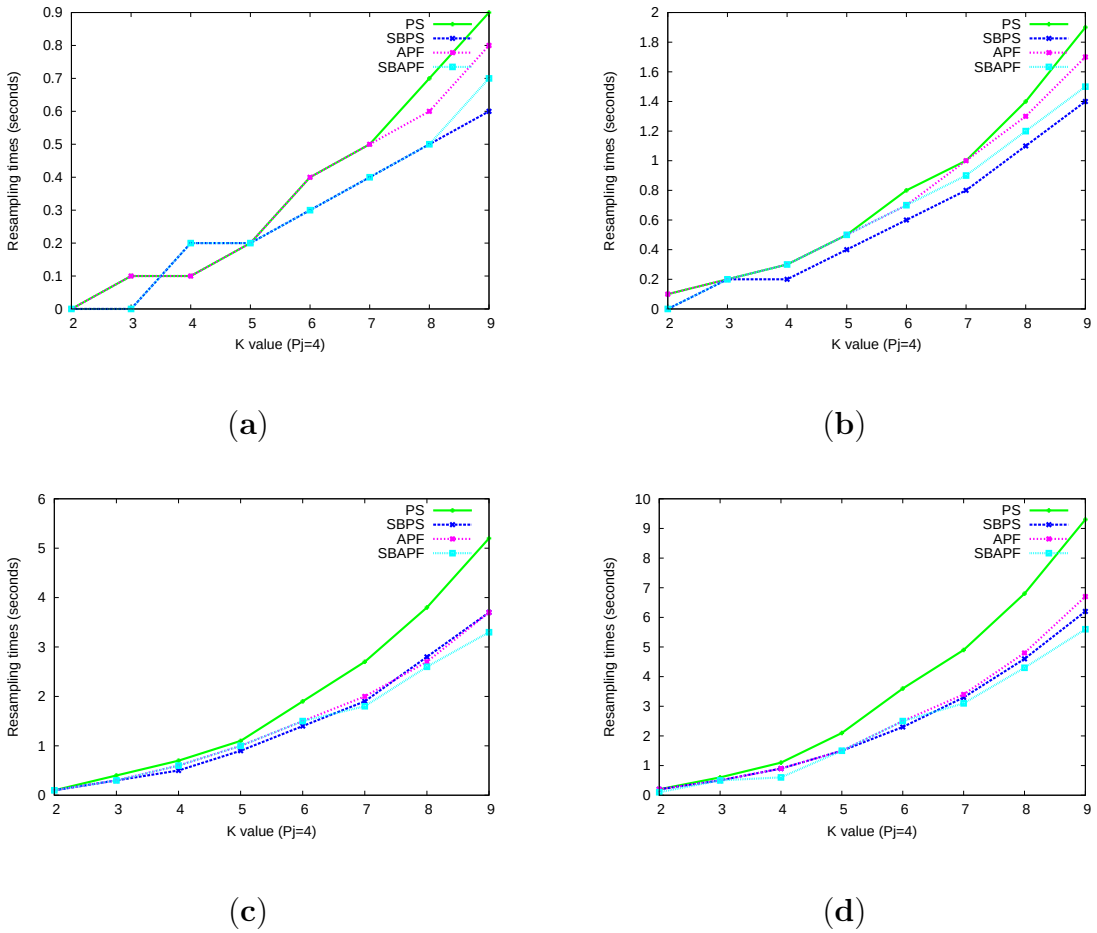


Figure 4-26: **Resampling times in seconds (resampling plus swapping steps).** Comparison between SBPS, PS, SBAPF and APF for tracking an object with $|P_j| = 4$ arms, depending on K. (**a**) $N = 50$. (**b**) $N = 100$. (**c**) $N = 200$. (**d**) $N = 300$.

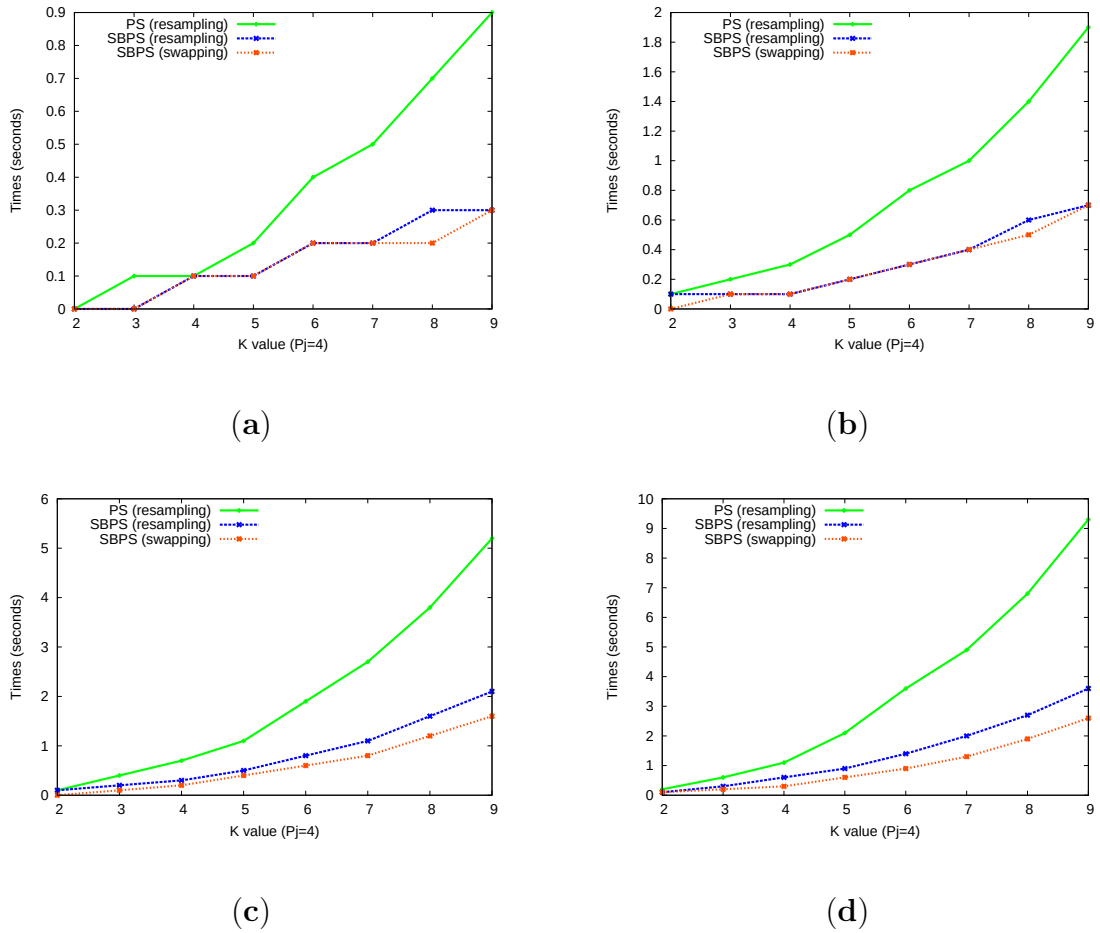


Figure 4-27: **Resampling and swapping times (in seconds).** Comparison between SBPS and PS for tracking an object with $|P_j| = 4$ arms, depending on K. (**a**) $N = 50$. (**b**) $N = 100$. (**c**) $N = 200$. (**d**) $N = 300$.

3 to 8, times are multiplied by 7.2 for PS, 5.5 for APF and 4 for SBPS and SBAPF. Although, compared to PS and APF, SBPS and SBAPF have additional swapping steps, the latter are significantly faster than the former in terms of resampling's and swapping's computation times. This is due to the fact that, by processing many different parts of the object simultaneously, SBPS and SBAPF reduce significantly the number of resamplings performed. As swapping is a fast operation, the latter is not sufficient to make SBPS's (SBAPF's) computations longer than those of PS (APF).

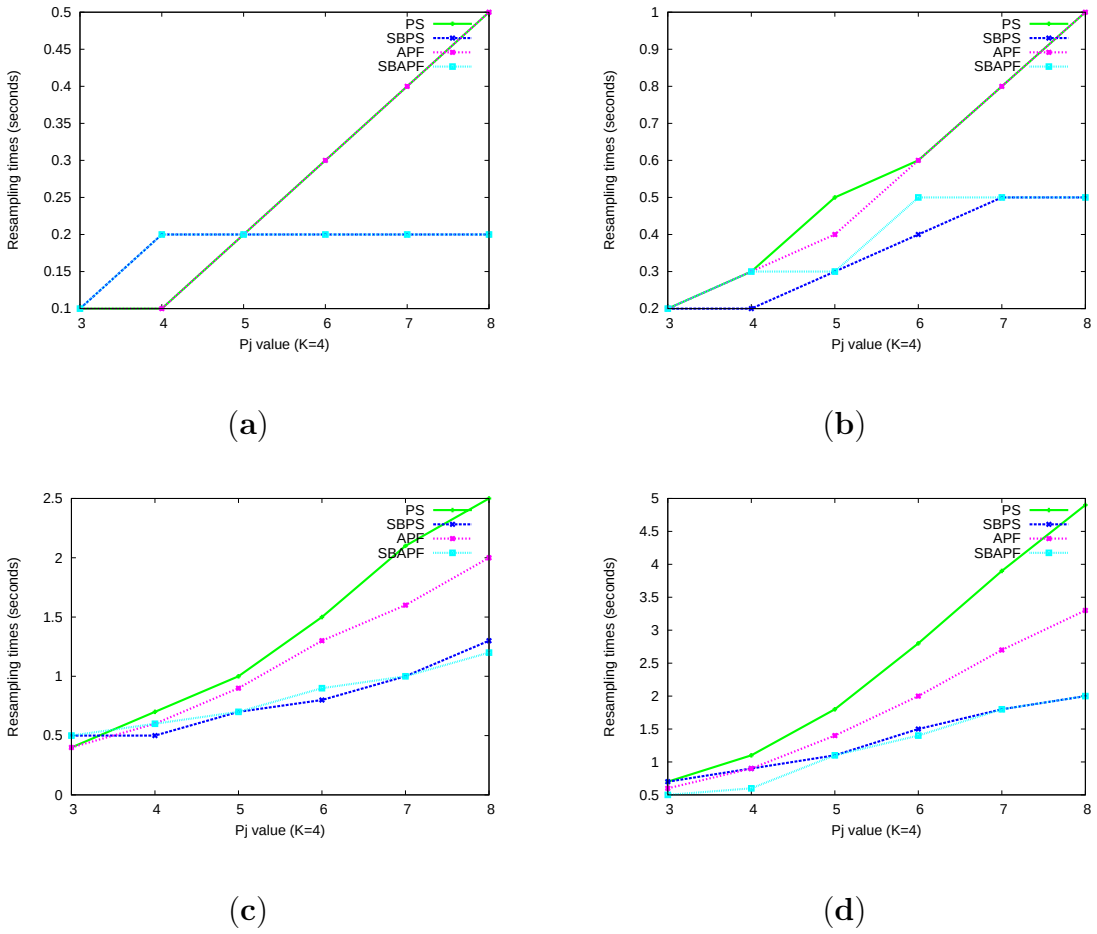


Figure 4-28: **Resampling times in seconds (resampling plus swapping steps).** Comparison between SBPS, PS, SBAPF and APF for tracking an object with $K = 4$ arms, depending on $|P_j|$. (**a**) $N = 50$. (**b**) $N = 100$. (**c**) $N = 200$. (**d**) $N = 300$.

The efficiency of the swapping operation in terms of computation time when $|P_j|$ increases is clearly shown in Figure 4-29. For example, when $|P_j|$ changes from 3 to 8, resampling times are multiplied by 7.2 for PS and 4 for SBPS (swapping times are also multiplied by 3).

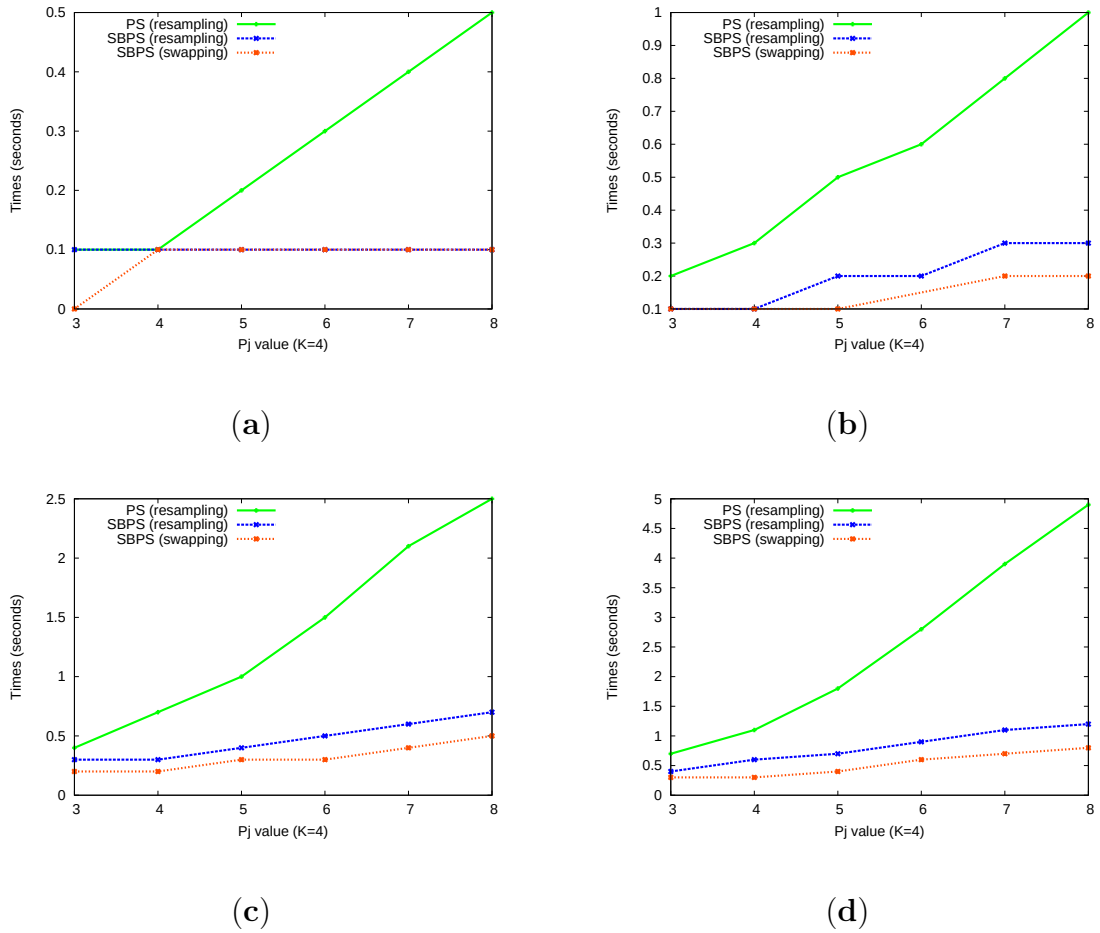
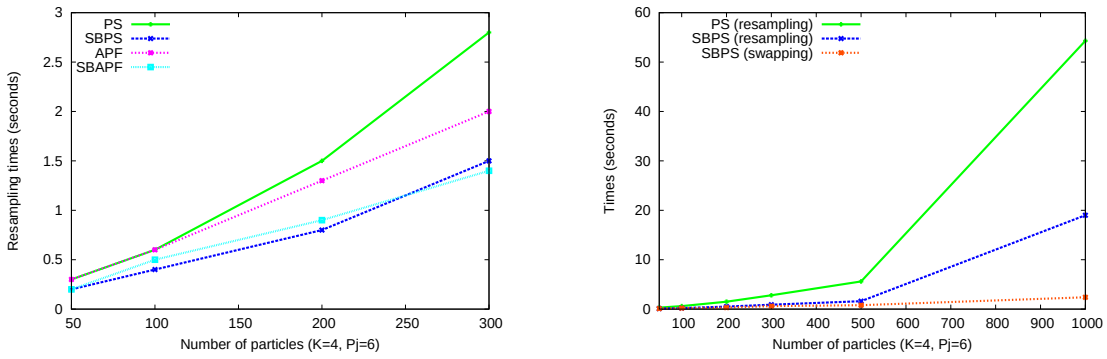


Figure 4-29: **Resampling and swapping times (in seconds).** Comparison between SBPS and PS for tracking an object with $K = 4$ arms, depending on $|P_j|$. (**a**) $N = 50$. (**b**) $N = 100$. (**c**) $N = 200$. (**d**) $N = 300$.

4.4.5.3 Computation times depending on N

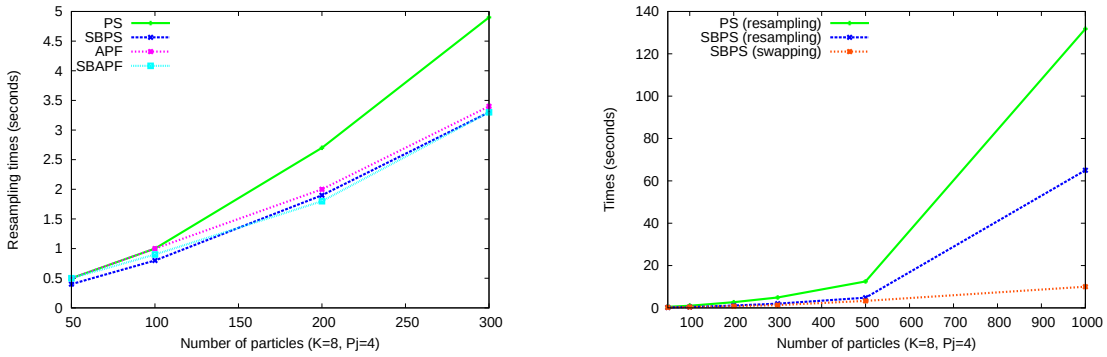
Finally, Figures 4-30 and 4-31 highlight how the number of particles used to track an object affects the computation times. In this experiment, two objects were tracked: one with $|P_j| = 6$ and $K = 4$ ($|\mathcal{X}| = 27$), the other with $|P_j| = 4$ and $K = 8$ ($|\mathcal{X}| = 35$). Remark that swapping times are linearly related to N, and that the differences of resampling times between PS and SBPS increase with N, as well as, of course, total computation times. We can observe that SBPS and SBAPF are the least drastically influenced by the increase of N. In Figure 4-30(b), we can see that PS's resampling times are multiplied by 60 when N increases from 50 to 1000, when SBPS's ones are multiplied by only 20.



(**a**) (**b**)

Figure 4-30: **Computation times of SBPS, PS, SBAPF and APF (in seconds) for tracking an object with $K = 4$ and $|P_j| = 6$.** (**a**) Total computation times of resampling and swapping step. (**b**) Computation times of resampling and swapping step of SBPS and PS.

In this section, we provided deep qualitative and quantitative tests on the case of single articulated object tracking to show that our algorithm outperforms the other ones compared (PS and APF) both in terms of computation times and estimation errors. The next section is dedicated to the specific case of multiple articulated object tracking. We will see that, in such cases, our swapping is very efficient, because lots of parts can be processed in parallel, increasing both the estimation accuracy and the computation times.



(**a**) (**b**)

Figure 4-31: **Computation times of SBPS, PS, SBAPF and APF (in seconds) for tracking an object with $K = 8$ and $|P_j| = 4$.** (**a**) Total computation times of resampling and swapping step. (**b**) Computation times of resampling and swapping step of SBPS and PS.

4.4.6 Extension to multiple articulated object tracking

In this section we address the multiple object tracking problem. There are two general ways to deal with such a task: either one filter can be used for all the tracked objects but, then, this filter has to deal with very high-dimensional state spaces, e.g., for M objects, the number of dimensions of the state space is multiplied by M; or one filter can be used per object, and each such filter just works in the reduced state space corresponding to the object it tracks. The goal of this subsection is to show that using one SBPS filter for all the objects is at least as efficient as using one PS filter per object: this will demonstrate the capacity of our approach to deal with high-dimensional subspaces by taking into account all the independences in the tracking problem.

Table 4.1 provides comparative results concerning the estimation errors and the computation times for both PS and SBPS. Here, each tracked object is defined by $|P_j| = 4$ and $K = 3$, and all objects are moving and deforming independently over time. We tested cases with $M = 2$ or $M = 3$ objects. When only one filter is used to estimate M objects, this one always uses $N = 100 \times M$ particles. When M filters are used (one per object), two configurations are tested: the case where each filter uses $N = 100 \times M$ particles and that where $N = 100$ particles are used per filter. As each such filter tracks only one object, it should actually need fewer particles for an accurate tracking. First, note that the lowest estimation errors result from SBPS

Table 4.1: Estimation errors (e, in pixels) and computation times (in seconds, t total computation time, r resampling time and s swapping time), for M objects with $K = 3$ and $|P_j| = 4$ (N is the number of particles used per filter).

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

(either one filter or M filters). Remark that using M PS filters is more interesting than using one for all the objects. This holds for all the numbers N of particles tested and it follows from the fact that the "lower" the dimension of the state space, the more robust PS is known to be. Conversely, for SBPS, the estimation errors are lower when using only one filter instead of M. This follows from the fact that increasing state space dimensions also increase the efficiency of swappings (because the products of the best weights $w^{(i),k}$ also tend to increase).

As for the response times, for a fixed number of particles, PS's resampling times are divided by M when dealing with one filter per object. This follows from the fact that, although the number of resamplings is identical whether one or M filters are used, the dimension of the state space for the single filter case is M times that of the multiple filter case. Conversely, SBPS's resampling times are equivalent when using one or M filters because the M filters perform M times the number of resamplings of the single filter but the latter are made in a space M times larger than those used by the M filters. For the same reason, swapping times are equivalent for one and M filters. Finally, when N decreases, all the computation times decrease, but the estimation errors increase, so that this induces a trade-off between response time and accuracy.

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

4.5 Conclusion

In this chapter, we have presented a new approach for sequential estimation of densities that has two main advantages. First, it exploits the probabilistic independences encoded into DBNs to apply particle filter computations on smaller subspaces. Second, it swaps some subsets of particles so that they concentrate around modes of the densities. We proposed a sound theoretical framework that guarantees that distributions are correctly estimated. In addition, we provided the time complexity of our approach. Experiments showed that our permutation operation is not time-consuming. The combination of this swapping step with the parallel processing of conditionally independent parts significantly reduces the number of required resampling steps, inducing overall computation times that are often smaller than PS. Moreover, we have shown that this gain of computation time increases with the state space dimension (number of parts of articulated objects, number of objects), as well as with the number of particles.

One of the limits of the proposed approach concerns the fact that swapping constructs very good particles (with high weights) as well as very bad ones (with low weights): this causes an increase of the variance of the particle set after swapping, that is observable for larges values of K or N for example. Another limit of the approach is that, by swapping only w.r.t. one permutation (see Proposition 4-3), not all the possible swappings are taken into account by our method, which could result in some situations in some sample impoverishment. This is the reason why we introduce in the next chapter another approach, called combinatorial resampling, that considers all the possible permutations for swapping the particles. As their number tends to increase exponentially, we show how to construct them and sample from them implicitly, thus making the algorithm tractable even for large particle sets.

Chapter 5

DBN-Based Combinatorial Resampling for Articulated Object Tracking

In this chapter, we introduce a new resampling method for articulated object tracking called *Combinatorial Resampling*. Our method is motivated by the swapping operation that has been introduced in Section 4.2. Each admissible permutation creates a new particle set of the same size as the original one, in which some particles are more focused near the modes of the posterior density. By considering all admissible permutations and aggregating all the particle sets resulting from their applications, we get a particle set of exponential size that actually contains the best particles from each permutation. Resampling from this set should thus not only allow to produce particles with high weights but also to promote diversity among particles. Now, there remains the question of how to resample over this particle set. Constructing it explicitly and resampling over it using some classical resampling algorithms is clearly impractical due to the exponential size of this set. Consequently, we propose an efficient resampling algorithm that: 1) avoids the need to construct the particle set explicitly from all admissible permutations; 2) guarantees that the posterior distribution is correctly estimated. In the rest of the chapter, we will however only consider OTDBNs for which the structure in each time slice is a tree, i.e., contain no cycle. As mentioned in the previous chapter, this hypothesis is rather mild for articulated object tracking. The results presented here can certainly be extended to situations where single time slice structures contain cycles, but, in this case, the formulas and their evaluation shall certainly be much more complicated.

The chapter is organized as follows. Section 5.1 presents the idea of Combinatorial Resampling and gives its theoretical soundness. Section 5.2 is dedicated to

experimental results. We first give in Section 5.2.1 an experimental comparison on synthetic video sequences of our combinatorial resampling with classical resampling algorithms, namely, multinomial resampling, residual resampling, stratified resampling, systematic resampling and weighted resampling. Then, in Section 5.2.2, we compare Particle Filter with Combinatorial Resampling (PFCR) with other classical filters, also on synthetic video sequences. Finally, in Section 5.2.3, we extend our tests on real video sequences, showing the interest of our approach for real tracking applications. Concluding remarks are given in Section 5.3.

5.1 Combinatorial Resampling (CR)

5.1.1 Definition

In this section we use the same notations as in Chapter 4. In this previous chapter, we argued that all the permutations satisfying Proposition 4-2 could be applied to the particle set without altering the estimation of the posterior density. For instance, let $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ be two particles whose torso positions are identical, then swapping their left arm and forearm positions cannot alter the density estimation. Similarly, the latter is unaffected by duplications of all the particles within a particle set. Combining these two features leads to Combinatorial Resampling, whose definition is given below.

Definition 5-1 (Combinatorial Resampling) *Let $\mathcal{S}$ be the particle set at the jth step of Algorithm 3.1. For any $k \in P_j$, let Σ_k be the set of permutations satisfying Proposition 4-2. Let $\Sigma = \prod_{k \in P_j} \Sigma_k$. 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 $\mathcal{S}$.*

To introduce the principle of CR, we now present an illustrative example.

5.1.2 Principle described in an illustrative example

Let us consider the example of Figure 5-1, and 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 $\mathbf{x}_t^{(3)} = \langle 1'', 2'', 3'', 4'', 5'', 6'' \rangle$ be three particles, where each number, $1, 1'', 2, 2', 2''$, etc., corresponds to the state of a part in the human body model in Figure 5-1. Assume that $\mathcal{S} = \{\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, \mathbf{x}_t^{(3)}\}$ at the 2nd step of Algorithm 3.1, i.e., the object parts just processed are $P_2 = \{2, 4, 6\}$. Parts $\{2, 3\}$, $\{4, 5\}$ and $\{6\}$ can be permuted in $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ because their torso, i.e. 1, are identical. Figure 5-2 shows an example of swapping right arms between particles with the same value for the torso.

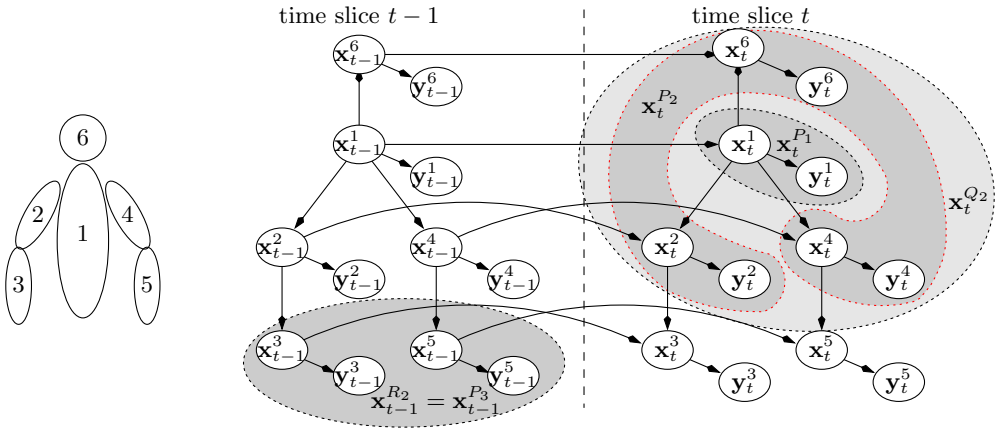


Figure 5-1: **A dynamic Bayesian network for human body tracking.**

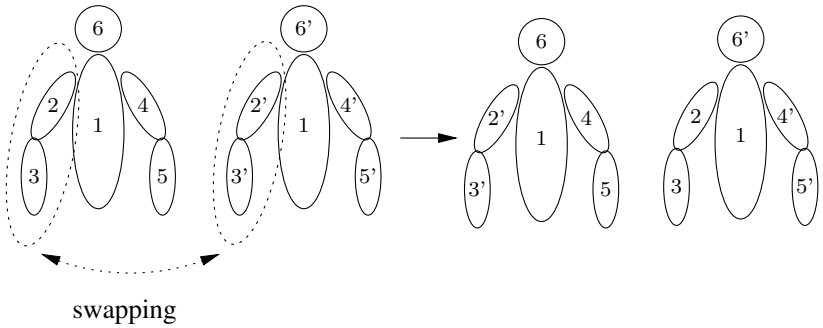


Figure 5-2: **Swapping right arms between two configurations with identical torso.**

Hence $\mathcal{S}'$, the combinatorial set defined in Definition 5-1, is the union of the result of all such possible permutations over $\mathcal{S}$ and is thus equal to:

$$\mathcal{S}' = \begin{array}{l} \langle 1,2\ ,3\ ,4\ ,5\ ,6\ \rangle\ \langle 1,2',3',4',5',6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle \\ \langle 1,2\ ,3\ ,4\ ,5\ ,6'\rangle\ \langle 1,2',3',4',5',6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle \\ \langle 1,2\ ,3\ ,4',5',6\ \rangle\ \langle 1,2',3',4\ ,5\ ,6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle \\ \langle 1,2\ ,3\ ,4',5',6'\rangle\ \langle 1,2',3',4\ ,5\ ,6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle \\ \langle 1,2',3',4\ ,5\ ,6\ \rangle\ \langle 1,2\ ,3\ ,4',5',6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle \\ \langle 1,2',3',4\ ,5\ ,6'\rangle\ \langle 1,2\ ,3\ ,4',5',6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle \\ \langle 1,2',3',4',5',6\ \rangle\ \langle 1,2\ ,3\ ,4\ ,5\ ,6'\rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle \\ \langle 1,2',3',4',5',6'\rangle\ \langle 1,2\ ,3\ ,4\ ,5\ ,6\ \rangle\ \langle 1'',2'',3'',4'',5'',6''\rangle \end{array}$$

Constructing $\mathcal{S}'$ in extension is impossible in practice because $|\Sigma|$ (see Definition 5-1) tends to grow exponentially with N, the number of particles. Fortunately, we can sample over $\mathcal{S}'$ without actually constructing it.

We shall explain the idea on the 5-sample particle set $\mathcal{S}$ illustrated on Figure 5-3, which corresponds to the object of Figure 5-1 in which we omitted the head part for

	parts:	3	2	1	4	5
		5	1	0	3	8
		6	1	0	3	9
jth particle	→	7	1	0	4	12
		10	2	0	4	13
		11	2	0	4	14

Figure 5-3: **Example of configuration.** Each row represents a particle $\mathbf{x}_t^{(i)}$ and each number a value $\mathbf{x}_t^{(i),j}$ of part j of the particle.

clarity reasons. Assume that parts $P_j = \{3, 5\}$, i.e., the forearms, have just been processed and we wish to sample over combinatorial sample $\mathcal{S}'$ induced by $\mathcal{S}$. To construct a new particle, the idea is to first select a value for the parts in Q_{j-1}, i.e., those processed at previous steps by PF and in which no permutation will occur. Here, $Q_{j-1} = \{1, 2, 4\}$. We thus first determine the different values of $\mathbf{x}_t^{Q_{j-1}}$ in $\mathcal{S}$ and partition $\mathcal{S}$ into sets $\mathcal{S}_1, \ldots, \mathcal{S}_R$ such that all the particles in each set $\mathcal{S}_h$ have the same value for $\mathbf{x}_t^{Q_{j-1}}$ (see Figure 5-4). In this figure, $\mathcal{S}_1$ thus contains the first two particles since their values on $\mathbf{x}_t^{Q_{j-1}} = \mathbf{x}_t^{\{1,2,4\}}$, are both $\langle 1, 0, 3 \rangle$.

parts:	3	2	1	4	5	
	5	1	0	3	8	$\mathcal{S}_1$
	6	1	0	3	9	
	7	1	0	4	12	$\mathcal{S}_2$
	10	2	0	4	13	$\mathcal{S}_3$
	11	2	0	4	14	

$P_j = \{\text{parts } 3,5\}$
$Q_{j-1} = \{\text{parts } 1,2,4\}$
$\mathbf{pa}_t(\mathbf{x}_t^3) = \{\mathbf{x}_t^2\}$
$\mathbf{pa}_t(\mathbf{x}_t^5) = \{\mathbf{x}_t^4\}$

Figure 5-4: **Sets $\mathcal{S}_h$.**

To each such set $\mathcal{S}_h$ is assigned a weight W_h defined below so that picking the value of $\mathbf{x}_t^{Q_{j-1}}$ in $\mathcal{S}_h$ w.r.t. weight W_h results in a particle set estimating the same distribution as that of $\mathcal{S}$. As, by hypothesis, the structure of the OTDBN in each time slice is a tree, once the value of $\mathbf{x}_t^{Q_{j-1}}$ has been chosen, there just remains to pick independently values for each part $\mathbf{x}_t^k$, $k \in P_j$, and its descendants, that are compatible with that chosen for $\mathbf{x}_t^{Q_{j-1}}$. Thus, for any $h \in \{1, \ldots, R\}$, let $\mathcal{S}_h^k$ denote the set of particles in $\mathcal{S}$ whose kth part value is compatible with the value of $\mathbf{x}_t^{Q_{j-1}}$ in $\mathcal{S}_h$. By Proposition 4-2, $\mathcal{S}_h^k$ is the set of particles in $\mathcal{S}$ that have the same value of $\mathbf{pa}_t(\mathbf{x}_t^k)$ as those in $\mathcal{S}_h$. For instance, in Figure 5-5, $\mathcal{S}_1^3$ is the set of the first 3 particles because all of them have value 1 on part 2.

parts: 3 2 1 4 5

$\mathcal{S}_1^3 = S_2^3$

5	1	0	3	8
6	1	0	3	9
7	1	0	4	12

$\mathcal{S}_3^3$

10	2	0	4	13
11	2	0	4	14

$P_j = \{\text{parts } 3,5\}$
$Q_{j-1} = \{\text{parts } 1,2,4\}$
$\mathbf{pa}_t(\mathbf{x}_t^3) = \{\mathbf{x}_t^2\}$
$\mathbf{pa}_t(\mathbf{x}_t^5) = \{\mathbf{x}_t^4\}$

Figure 5-5: **Sets $\mathcal{S}_h^k$.**

In the next section, we explain how the weights W_h are computed.

5.1.3 Computation of the sets' weights

To determine the aforementioned weights W_h, there just needs to count how many times the combinatorial set has duplicated $\mathcal{S}_h$. So, let $N_1, \ldots, N_R$ and $N_1^k, \ldots, N_R^k$ denote the sizes of $\mathcal{S}_1, \ldots, \mathcal{S}_R$ and $\mathcal{S}_1^k, \ldots, \mathcal{S}_R^k$ respectively. Let $N^k = \max\{N_1^k, \ldots, N_R^k\}$ and, for any $h \in \{1, \ldots, R\}$, let W_h^k denote the sum of the weights assigned to the kth part of the particles in $\mathcal{S}_h^k$, i.e., $W_h^k = \sum_{\mathbf{x}_t^{(i)} \in S_h^k} w^{(i),k}$. Then, as we shall prove below, for any h,

$$W_h = N_h \times \prod_{k \in P_j} \frac{N^k!}{A_{N_h}^{N_h}} \times A_{N_h^k - 1}^{N_h - 1} \times W_h^k, \tag{5.1}$$

where $A_n^k = n!/(n-k)!$ stands for the number of k-permutations out of n elements. Resampling over $\mathcal{S}'$ can thus be performed efficiently as in Algorithm 5.1. To scale-up to large particle sets, $\log(W_h)$ should be computed instead of W_h and the weights used in line 2 of Algorithm 5.1 should be $\exp(\log W_h - \log W)$, where $W = \max\{W_1, \ldots, W_R\}$.

The global algorithm of CR and its proof of correctness are finally given in the next section.

5.1.4 Algorithm and theoretical soundness

The global algorithm of the proposed resampling is given in Algorithm 5.1.

Proposition 5-1 *Algorithm 5.1 produces a particle set estimating the same density as that given in input.*

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 $\quad h \leftarrow$ sample $\{1, \ldots, R\}$ w.r.t. weights $W_1, \ldots, W_R$
3 $\quad \mathbf{x}_t^{\prime\prime(i),Q_{j-1}} \leftarrow \mathbf{x}_t^{(z),Q_{j-1}}$ where $\mathbf{x}_t^{(z)}$ is any element in $\mathcal{S}_h$
4 $\quad w_t^{\prime\prime(i)} \leftarrow 1$
5 $\quad$ **foreach** k **in** P_j **do**
6 $\quad\quad \mathbf{x}_t^{(r)} \leftarrow$ sample from $\mathcal{S}_h^k$ w.r.t. weights $\{w_t^{(r),k}\}_{\mathbf{x}_t^r \in \mathcal{S}_h^k}$
7 $\quad\quad \mathbf{x}_t^{\prime\prime(i),k} \leftarrow \mathbf{x}_t^{(r),k}$; $\quad w_t^{\prime\prime(i)} \leftarrow w_t^{\prime\prime(i)} \times w_t^{(r),k}$
8 $\quad\quad \mathbf{x}_{t-1}^{\prime\prime(i),\mathbf{Desc}_{t-1}(\mathbf{x}_{t-1}^k)} \leftarrow \mathbf{x}_{t-1}^{(r),\mathbf{Desc}_{t-1}(\mathbf{x}_{t-1}^k)}$
9 **return** $\{\mathbf{x}_t^{\prime\prime(i)}, w^{\prime\prime(i)}\}_{i=1}^N$

Algorithm 5.1: Efficient resampling over $\mathcal{S}'$.

Proof of Proposition 5-1: Let $\mathcal{S} = \{(\mathbf{x}_t^{(i),Q_j}, \mathbf{x}_{t-1}^{(i),R_j})\}_{i=1}^N$ and $\mathcal{S}'$ its combinatorial set (see Definition 5-1). In lines 2–3, Algorithm 5.1 selects which central part Q_{j-1} particle $\mathbf{x}_t''$ should have. By definition, this amounts to selecting one set $\mathcal{S}_h$ w.r.t. the sum of the weights of the particles in $\mathcal{S}'$ having the same central part as those in $\mathcal{S}_h$. Let us show that this is achieved using the weights described in Equation 5.1.

In Definition 5-1, Σ_k is the set of all the possible permutations of the kth part of the particles in $\mathcal{S}$. Clearly, within each set $\mathcal{S}_h^k$, all the $N_h^k!$ permutations of the kth part 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 over $\mathcal{S}_h^k$ shall appear many times within Σ_k. There is no need to count precisely how many times σ is repeated, what is important is that the repeated sets of particles estimate the same density as $\mathcal{S}$. To do so, remark that $N^k = \max\{N_1^k, \ldots, N_R^k\}$ is the size of the biggest set $\mathcal{S}_1^k, \ldots, \mathcal{S}_R^k$. Applying all the permutations over this set multiplies its size by $N^k!$, so the size of all the other sets should be multiplied by the same amount. Duplicating $N^k!/N_h^k!$ permutation σ guarantees that all the Q_{j-1}-central parts of the particles in $\mathcal{S}$ are duplicated the same number of times. Now, the particles in $\mathcal{S}_h$ also belong to $\mathcal{S}_h^k$. As $|\mathcal{S}_h| = N_h$, there are $A_{N_h^k}^{N_h}$ different possibilities to assign some k-part of $\mathcal{S}_h^k$ to the particles of $\mathcal{S}_h$. The number of times these permutations are repeated within those over $\mathcal{S}_h^k$ is thus $N_h^k!/A_{N_h^k}^{N_h}$. Hence, duplicating $(N^k!/N_h^k!) \times (N_h^k!/A_{N_h^k}^{N_h}) = N^k!/A_{N_h^k}^{N_h}$ times the permutations over $\mathcal{S}_h$ ensures that the particle set estimates the same density as $\mathcal{S}$. The same applies to all the other parts in P_j, hence the product in Equation 5.1.

Now, let us compute the sum of the weights of the particles resulting from all the permutations over $\mathcal{S}_h$. Each such permutation generates a new set of N_h particles.

By symmetry, if W is the sum of the weights of the first particle in each set, call it $\mathbf{x}_t^{(i)}$, then the overall sum we look for is $N_h \times W$. As permutations over the parts in P_j are independent, W is equal to the product over parts $k \in P_j$ of the sum W^k of the weights induced by all the permutations over the kth part, i.e., the permutations over $\mathcal{S}_h^k$. By symmetry, any weight in $\mathcal{S}_h^k$ can be assigned to $\mathbf{x}_t^{(i)}$, hence W^k is equal to the sum of all these weights, W_h^k, times the number $\mathcal{O}$ of occurrences of each weight induced by all the permutations over $\mathcal{S}_h^k$. For instance, if there are 3 weights 1,2,3, then there are $\mathcal{O} = 2$ permutations where the first particle has a weight of 1: $\langle 1, 2, 3 \rangle$ and $\langle 1, 3, 2 \rangle$. Once particle $\mathbf{x}_t^{(i)}$ has been assigned a weight, there remains $N_h - 1$ weights to assign to the other particles from a set of $N_h^k - 1$ weights, hence there are $\mathcal{O} = A_{N_h^k - 1}^{N_h - 1}$ possibilities. Overall, W^k is thus equal to $W_h^k \times A_{N_h^k - 1}^{N_h - 1}$ and we get Equation 5.1.

So, lines 2–3 select correctly the Q_{j-1} part. Once this is done, by d-separation, all the parts 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 $p(\mathbf{x}_t^k | \mathbf{pa}_t(\mathbf{x}_t^k)) \propto w_t^{(i),k}$. □

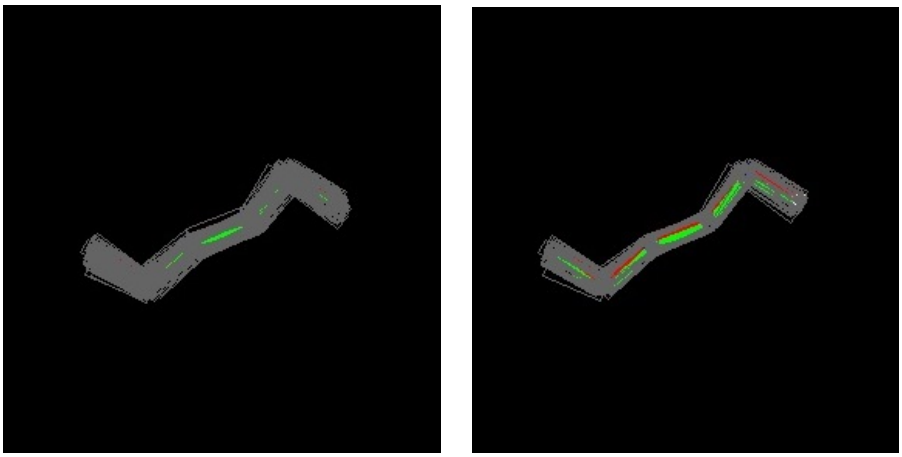
The next section is dedicated to the experimental results. We first compare CR with other resampling approaches, then compare different particle filters. Tests are made on synthetic and real video sequences, and the role of different parameters is studied.

5.2 Experimental results

We use the same video sequences as in Section 4.4 for these experiments. Recall that articulated objects are modeled by a set of P polygonal parts (or regions): a central one P_1 (containing only one polygon) to which are linked $|P_j|$, $j > 1$, arms of length $K - 1$. Particles are propagated using a random walk with $\sigma_x = 1$, $\sigma_y = 1$ and $\sigma_\theta = 0.025$. The particle weights are computed by $w_{t+1}^{(i)} = w_t^{(i)} p(\mathbf{y}_{t+1} | \mathbf{x}_{t+1}^{(i)}) \propto w_t^{(i)} e^{-\lambda \mathbf{d}^2}$, where the same parameters as in Section 4.4 were used, $\lambda = 50$ and $\mathbf{d}$ is the Bhattacharyya distance between the target (prior) and the reference (previously estimated) 8-bin 3-channel histograms. The articulated object's distribution is estimated starting from its central part P_1. All the algorithms are manually initialized. Results are compared w.r.t. three criteria: estimation errors, defined as the sum of the Euclidean distances between each corner of the estimated parts and its sibling in the ground truth, standard deviations and computation times. All the results presented here are a mean over 250 runs performed on a PC with a 3.07 GHz Intel Core i7.

Let us first consider the articulated object of Figure 5-6, defined with $K = 3$ and $|P_j| = 2$ ($|\mathcal{X}| = 15$). We have drawn in gray the particle set before (Figure 5-6(a))

and after (Figure 5-6(b)) the combinatorial resampling. One can see that, after the combinatorial resampling, the particle set is more concentrated on the object (i.e. the variance of the set was reduced by the process). The goal of this section is to show the interest of our resampling approach compared to other ones, and how its introduction into the particle filter framework can reduce estimation errors, while keeping reasonable computation times.



(**a**) (**b**)

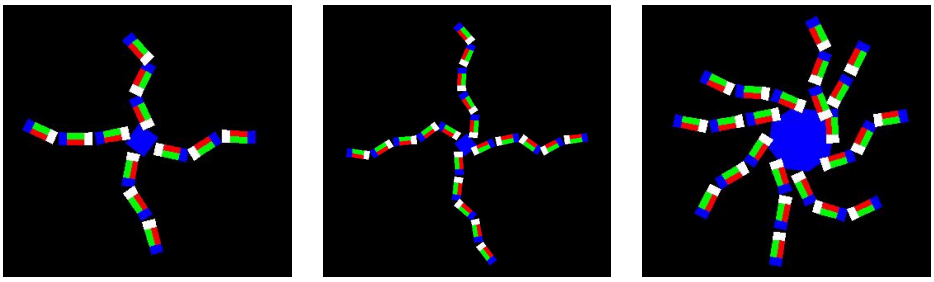
Figure 5-6: **The effectiveness of combinatorial resampling**. Figures (**a**) and (**b**) show a particle set before and after applying combinatorial resampling, respectively.

In Section 5.2.1, we first present an extended experimental comparison with different resampling methods. In Section 5.2.2, we provide a comparison of combinatorial resampling introduced into a particle filter framework (i.e. PFCR) with different filters, namely PS and APF, but also SBPS and SBAPF that were presented in Chapter 4. Finally, we test our approach and compare it with others on real video sequences.

5.2.1 Comparison with other resampling approaches

In this section, we work on the three sequences that are described in Figure 5-7.

We compare six different resampling approaches. The first five (multinomial, systematic, stratified, residual and weighted resampling) 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 independent conditionally to the central part as the P_j parts, $j > 1$, correspond to the jth joints of all the arms. All the arms are thus processed in parallel as described in



$K = 4$, $|P_j| = 4$
$P = 13$, $|\mathcal{X}| = 15$
(**a**)

$K = 6$, $|P_j| = 4$
$P = 21$, $|\mathcal{X}| = 23$
(**b**)

$K = 4$, $|P_j| = 8$
$P = 25$, $|\mathcal{X}| = 27$
(**c**)

Figure 5-7: **Tested synthetic video sequences (excerpts of frames).** Features of the corresponding articulated objects are given below (number of arms $|P_j|$, $j > 1$, length of arms $K - 1$, total number of parts P, and dimension of state space $\mathcal{X}$).

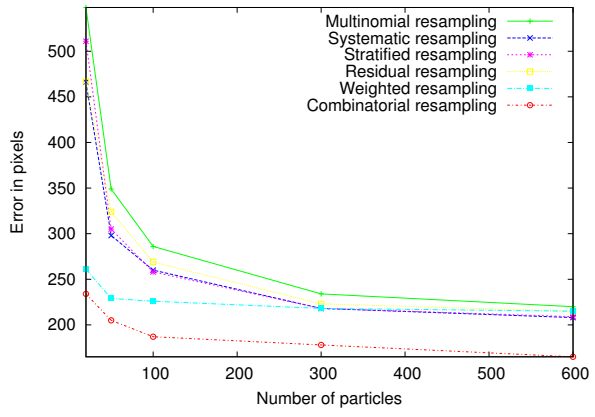
SBPS. For weighted resampling, function g is set empirically to $g(w) = e^{20w}$ to favor the selection of high-weighted particles over low-weighted ones.

5.2.1.1 Estimation errors

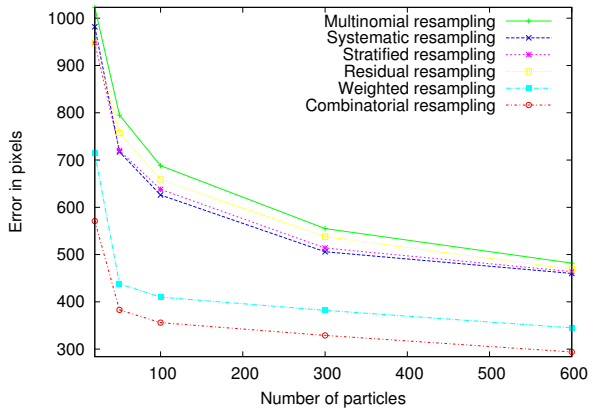
We first compared the estimation errors. Figures 5-8.(**a-c**) show a convergence study of the resampling methods depending on the number N of particles for the 3 objects of Figure 5-7. For all these tests, combinatorial resampling (CR) outperforms all the other methods: i) it converges faster (about only $N = 100$ particles are necessary to do so) when the other methods often require 300 particles to converge; ii) CR's error at convergence is much lower than that of the other methods. For instance, in Figure5-8(**a**), CR reaches the convergence error of the other methods (about 230 pixels) with only $N = 20$ particles and, with 100 particles, its error decreases to 112 pixels. When the length of the arms (given by $K - 1$) increases (Figure 5-8(**b**)), CR stays robust, whereas multinomial, systematic, stratified and residual resampling tend to fail (estimation errors twice higher). Weighted resampling seems more stable, but gives estimation errors 25% higher than those of CR. Finally, when the number of parts treated in parallel increases (Fig 5-8(**c**)), CR stays stable: with only $N = 20$ particles, its estimation error is 2.5 to 3 times lower than the one of the other resampling approaches.

5.2.1.2 Computation times

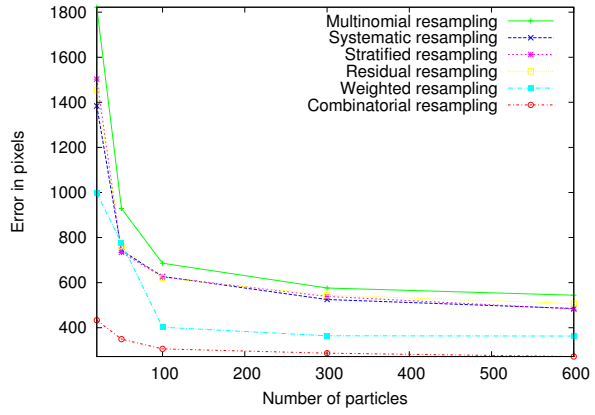
Resampling times (in seconds) over the whole sequences, are reported in Table 5.1 for the estimation of the densities of the objects of Figures 5-7(**a-c**) with $N = \{100, 600\}$



(**a**)



(**b**)



(**c**)

Figure 5-8: **Comparison of convergence for different resampling approaches.** estimation of the density of objects depending on N: **a** with $|P_i| = 4$, $K = 4$ (object of Fig. 5-7.(a)), (**b**) with $|P_i| = 4$, $K = 6$ (object of Fig. 5-7.(b)) and (**c**) with $|P_i| = 8$, $K = 4$ (object of Fig. 5-7.(c)).

particles. The first four resampling approaches have similar behaviors. Due to its additional step ensuring that weights are equal to $1/N$, which is required by Algorithm 3.1, weighted resampling is longer. The best approach is CR when the number of particles is high (600) and when the size of the P_j's is high. For instance, when tracking the object of Figure 5-7(**c**) (8 parts processed simultaneously), the resampling times are considerably lower with CR than with the other methods. This is due to the fact that, by processing several object parts simultaneously, the number of resamplings performed is significantly reduced. Hence, even if performing CR once is longer than performing another method, overall, CR is globally faster. Note also that CR's response times increase more slowly with N than the other methods. Finally, when K increases (Figure 5-7(**b**)), our approach also provides significantly smaller resampling times when N becomes high.

Table 5.1: **Resampling times (in seconds) for the estimation of the density of different objects, with $N = \{100, 600\}$.**

	Figure 5-7(**a**)		Figure 5-7(**b**)		Fig. 5-7(**c**)	
	100	600	100	600	100	600
Multinomial	0.5	17.1	1.3	46.9	1.8	79.6
Systematic	0.5	19.8	1.2	53.6	1.7	80.5
Stratified	0.5	16.9	1.3	44.8	1.7	74.9
Residual	0.5	20.3	1.3	55.7	1.8	83.4
Weighted	1.0	33.0	2.5	90.1	3.5	157.8
Combinatorial	0.7	10.6	1.5	26.3	1.5	22.3

In the next section, we now compare different kinds of filters, in particular we show that embedding our CR into a particle filter framework provides good results.

5.2.2 Comparison with other filters

In this section, we introduce our combinatorial resampling into SBPS to give the Particle Filter with Combinatorial Resampling (PFCR). More precisely, we substitute the permutation and resampling steps of SBPS by one Combinatorial Resampling operation. As we did in the previous chapter, we perform extended experimental tests to show the superiority of PFCR as compared with APF (because PS always gave the worst performances in Chapter 4 in terms on both computation times and estimation errors, we have chosen not to include this filter into our error estimation comparative tests in this chapter, but only for computation time comparisons). We also compare PFCR with SBPS and SBAPF, also presented in Chapter 4. As previously, we compare the behavior of the tested filters with respect to K, the length of arms, $|P_j|$,

the number of arms, and N the number of particles. Considering this experimental setup, we compare the performance of PFCR in terms of estimation errors, standard deviations and computation times with those of PS, SBPS, APF and SBAPF. Here again, the number of particles used by the different filters is determined so that all of them evaluate the same number of times the likelihood function (see Section 4.4.4). When not specified, when PFCR and SBPS use N particles for tracking, SBAPF (with 1 layer) uses $N/2$ particles, and APF uses $N/(L+1)$, with L the number of layers of annealing (by default, $L = 1$, so APF uses $N/2$ particles). For the same reasons discussed in Section 4.4, to compare the global computation times of PFCR with the other methods, we only study resampling times.

5.2.2.1 Performances depending on K

In this set of experiments, $|P_j|$ is fixed to 4, and K varies from 2 to 8. Experiments were performed with various numbers of particles ($N = 50$, $N = 100$, $N = 200$ and $N = 300$).

Estimation errors. Figure 5-9 shows the estimation errors obtained by SBPS, SBAPF, APF and PFCR. Lower estimation errors are provided by PFCR. Moreover, when K increases, the differences of estimation errors between PFCR and other filters increases, especially when the number N of particles is small. For example, for $N = 50$, when K varies from 2 to 8, PFCR's estimation error is multiplied by 100, and APF's one is multiplied by 240. Here we observe the effect of the combinatorial resampling, by just comparing SBPS and PFCR: the implicit set generated by CR and the resampling over it allows to construct a better particle set, considerably decreasing the estimation error. For example, for $N = \{50, 100, 200, 300\}$, and $K = 8$, SBPS gives estimation errors approximately 2.6 times higher than PFCR. Finally, if SBAPF improves APF by adding a swapping step after annealing, nevertheless, PFCR outperforms SBAPF. This shows the interest of CR for high-dimensional state spaces and its ability to concentrate particles around the modes of the density to estimate.

Errors' standard deviations. Standard deviations obtained by PFCR, SBPS, APF and SBAPF are given in Figure 5-10. As can be seen, PFCR also achieves lower standard deviations. But the most interesting observation is that standard deviations provided by PFCR are very stable and only slowly increase with K. For example, for $N = 50$, when K varies from 2 to 8, PFCR's standard deviation is multiplied by 20, and that of APF is multiplied by 250, and for $N = 300$, when K varies from 2 to 8, PFCR's standard deviation is multiplied by 9 while that of APF is multiplied by

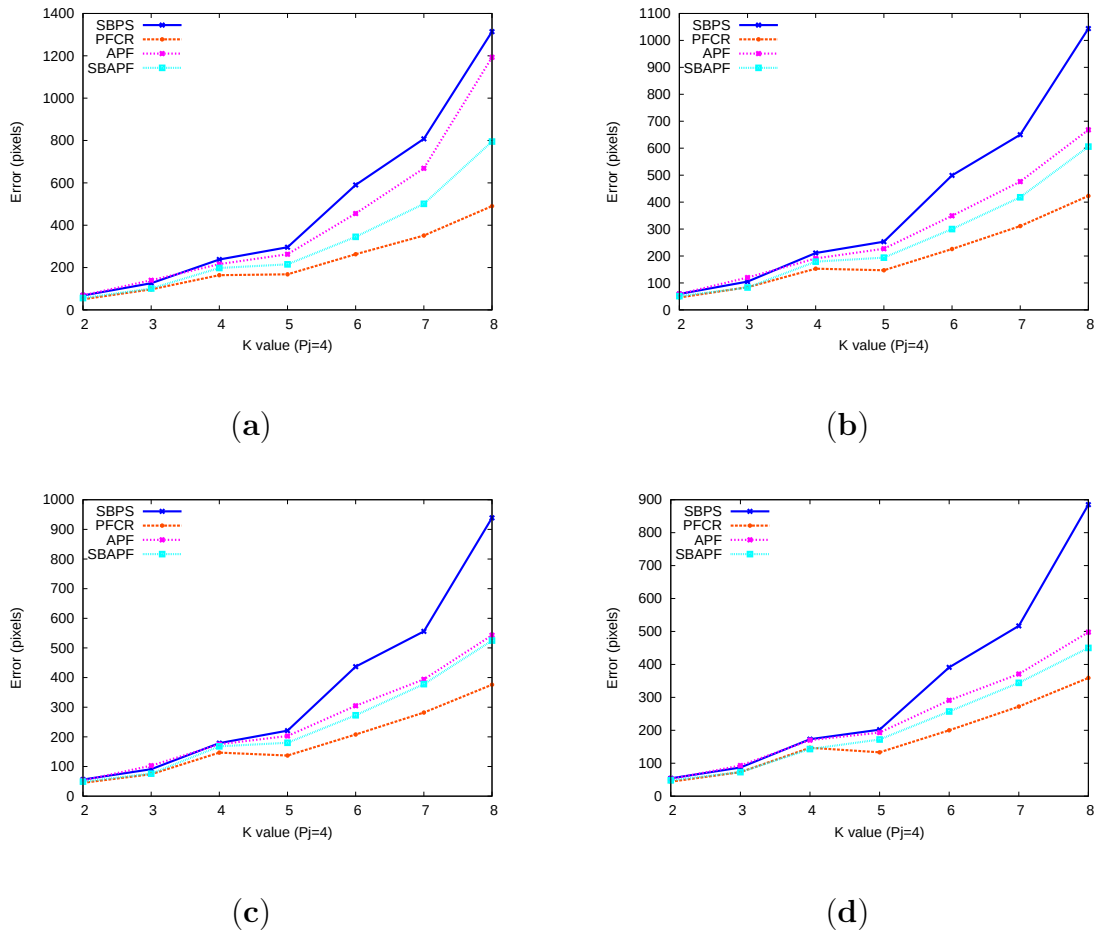


Figure 5-9: **Estimation errors of SBPS, APF, SBAPF and PFCR for tracking an object with $|P_j| = 4$, depending on K.** Tracking is performed with (**a**) $N = 50$, (**b**) $N = 100$, (**c**) $N = 200$ and (**d**) $N = 300$.

28. A drawback of SBPS we highlighted in the previous chapter is the fact that the performance of the swapping is highly dependent of the previous generated particle set: very good and very bad particles can be constructed, increasing the variance of the particle set. This problem is fixed by CR, as can be seen on the graphs of Figure 5-10. Finally, here again, we note that PFCR outperforms SBAPF, showing that using CR is more interesting than using a simple optimization approach followed by the swapping step. All these observations highlight the efficiency of PFCR and its capacity to generate the best particle set from all admissible permutations.

Resampling times. Resampling times of PS, SBPS, APF, SBAPF and PFCR are shown in Figure 5-11. PFCR requires more computation times than SBPS and

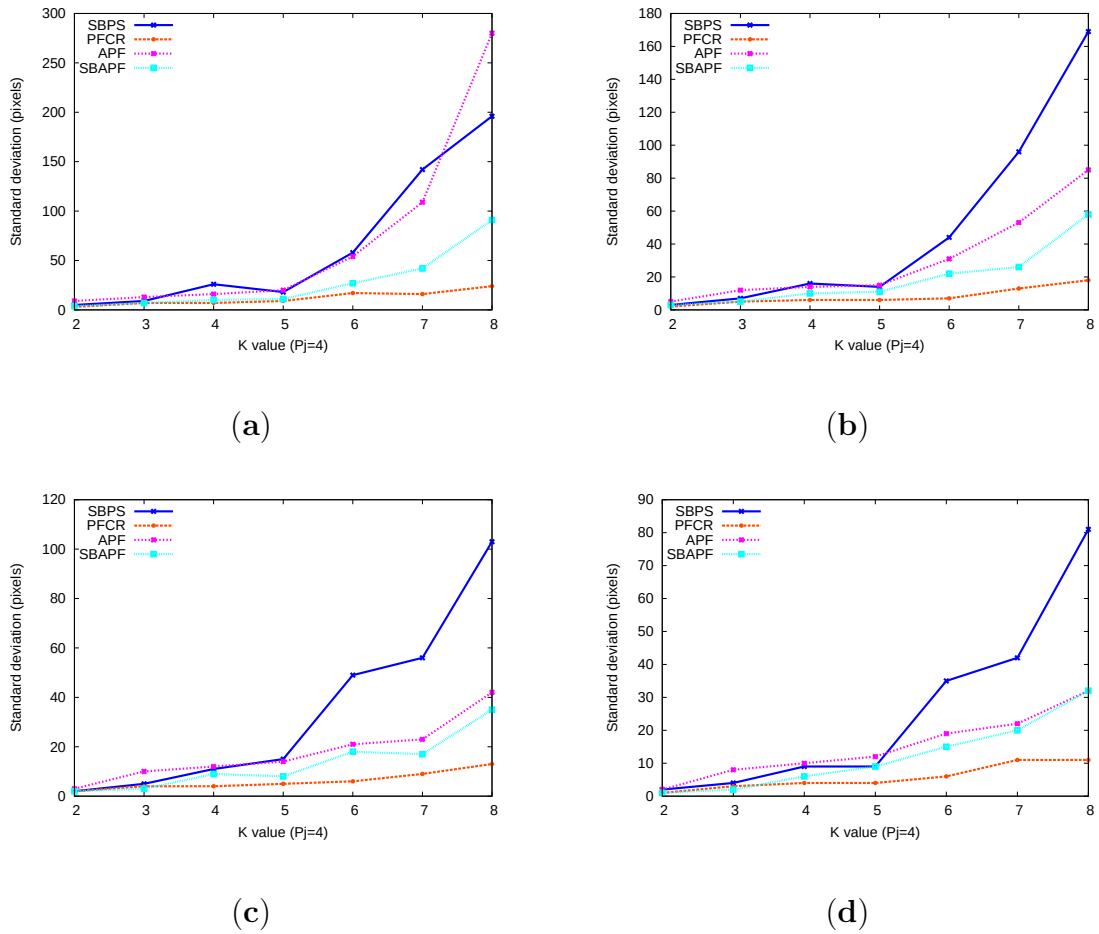


Figure 5-10: **Standard deviations of SBPS, APF, SBAPF and PFCR for tracking an object with $|P_j| = 4$, depending on K.** Tracking is performed with (**a**) $N = 50$, (**b**) $N = 100$, (**c**) $N = 200$ and (**d**) $N = 300$.

SBAPF, which are the fastest filters. However, one can notice that PFCR keeps reasonable resampling times compared to APF, and, in particular, it requires more resampling times than APF when $N = 200$ and $N = 300$, but less when $N = 50$ and $N = 100$. However, PFCR is still faster than PS, due to the reduced number of resampling steps it performs (i.e. its capacity to treat in parallel some parts).

5.2.2.2 Performances depending on $|P_j|$

In these experiments, K is fixed to 4, $|P_j|$ varies from 3 to 8, and the number of particles used for tracking are $N = 50$, $N = 100$, $N = 200$ and $N = 300$ respectively.

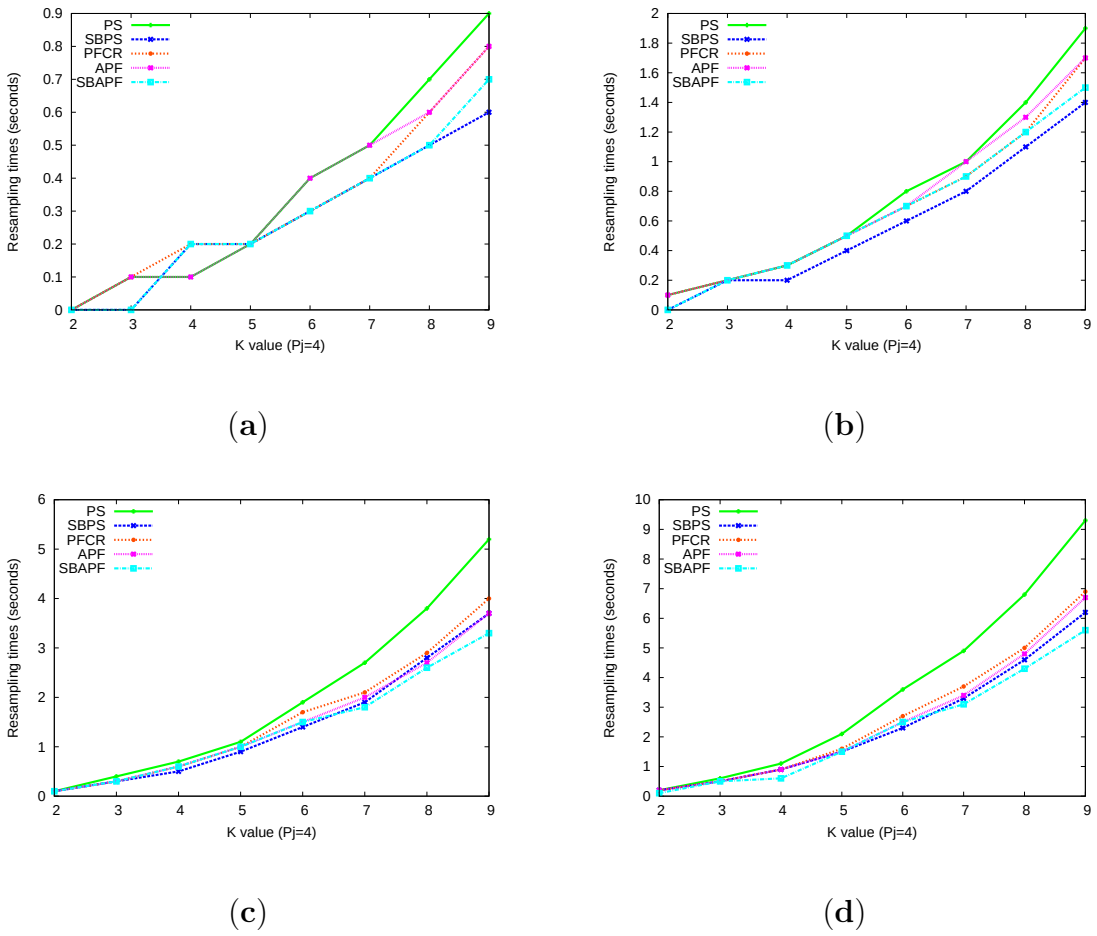


Figure 5-11: **Resampling times of SBPS, APF, SBAPF and PFCR for tracking an object with $|P_j| = 4$, depending on K.** Tracking is performed with (**a**) $N = 50$, (**b**) $N = 100$, (**c**) $N = 200$ and (**d**) $N = 300$.

Estimation errors. Figure 5-12 shows the estimation errors obtained by SBPS, SBAPF, APF and PFCR. Results show that PFCR gives lower estimation errors than the other methods. Again, the differences between the estimation errors of PFCR and those of SBPS, SBAPF, APF increase with $|P_j|$. Here, the worst approach is APF. When increasing $|P_j|$, we increase the number of parts that can be treated in parallel, also increasing the efficiency of the swapping process, that is why all the filters we proposed in this thesis are efficient in such cases. CR shows again its robustness compared to a "simple" swapping: it improves SBPS by reducing the estimation errors up to 91%, in particular when $|P_j|$ (and thus the dimension of the state space) becomes high. It also shows its capacity to outperform an optimization approach (APF), even if it is followed by a swapping step.

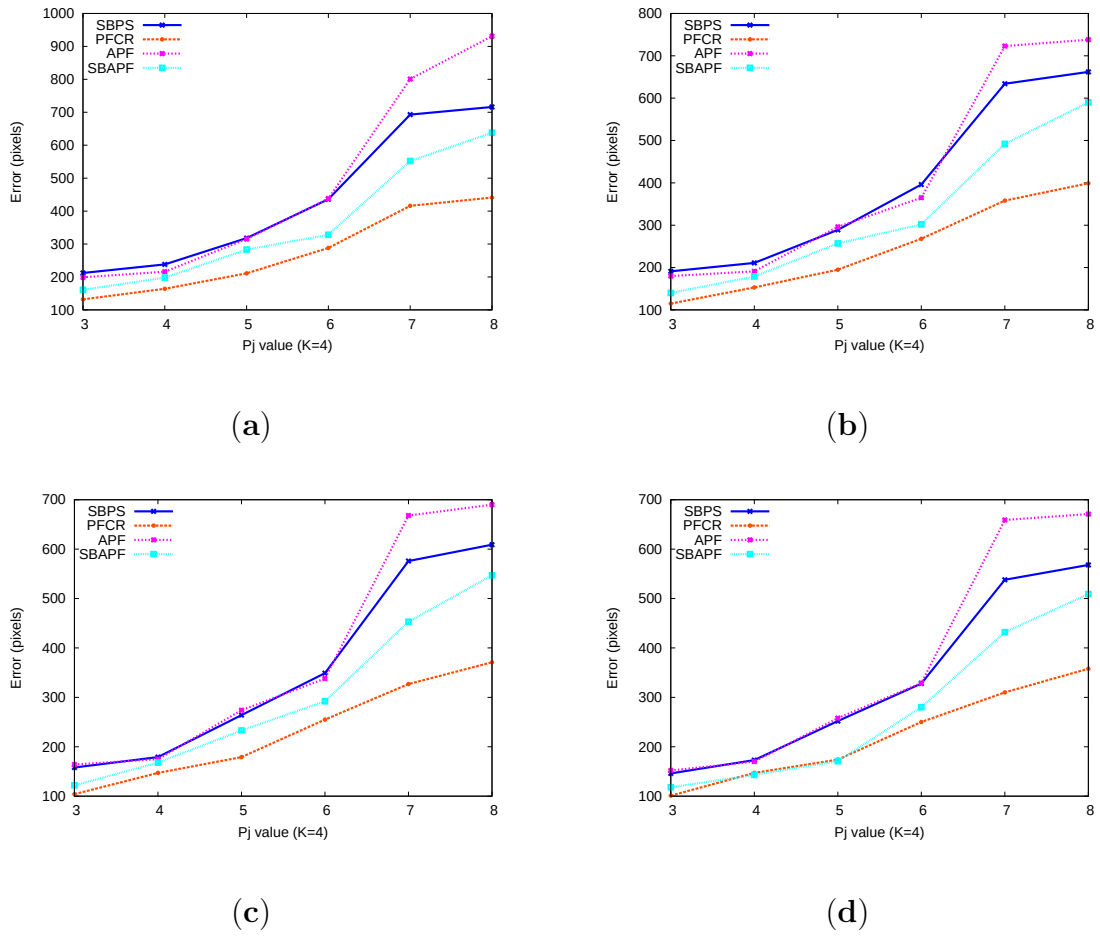


Figure 5-12: **Estimation errors of SBPS, APF, SBAPF and PFCR for tracking an object with $K = 4$, depending on $|P_j|$.** Tracking is performed with (**a**) $N = 50$, (**b**) $N = 100$, (**c**) $N = 200$ and (**d**) $N = 300$.

Standard deviations. Figure 5-13 shows the standard deviations obtained by SBPS, SBAPF, APF and PFCR. Here we observe results similar to those of Section 5.2.2.1: the standard deviations resulting from PFCR are always lower than those of the other compared filters, they increase slowly with $|P_j|$, and are more stable. Globally, APF gives the less stable standard deviations. This is due to the optimization process, during which the solution can be trapped into a local maxima, that can perturb the estimation process, and then increase the standard deviation. Here, we also observe the superiority of PFCR over SBPS, as we analyzed previously, because it drastically decreases the error's standard deviation.

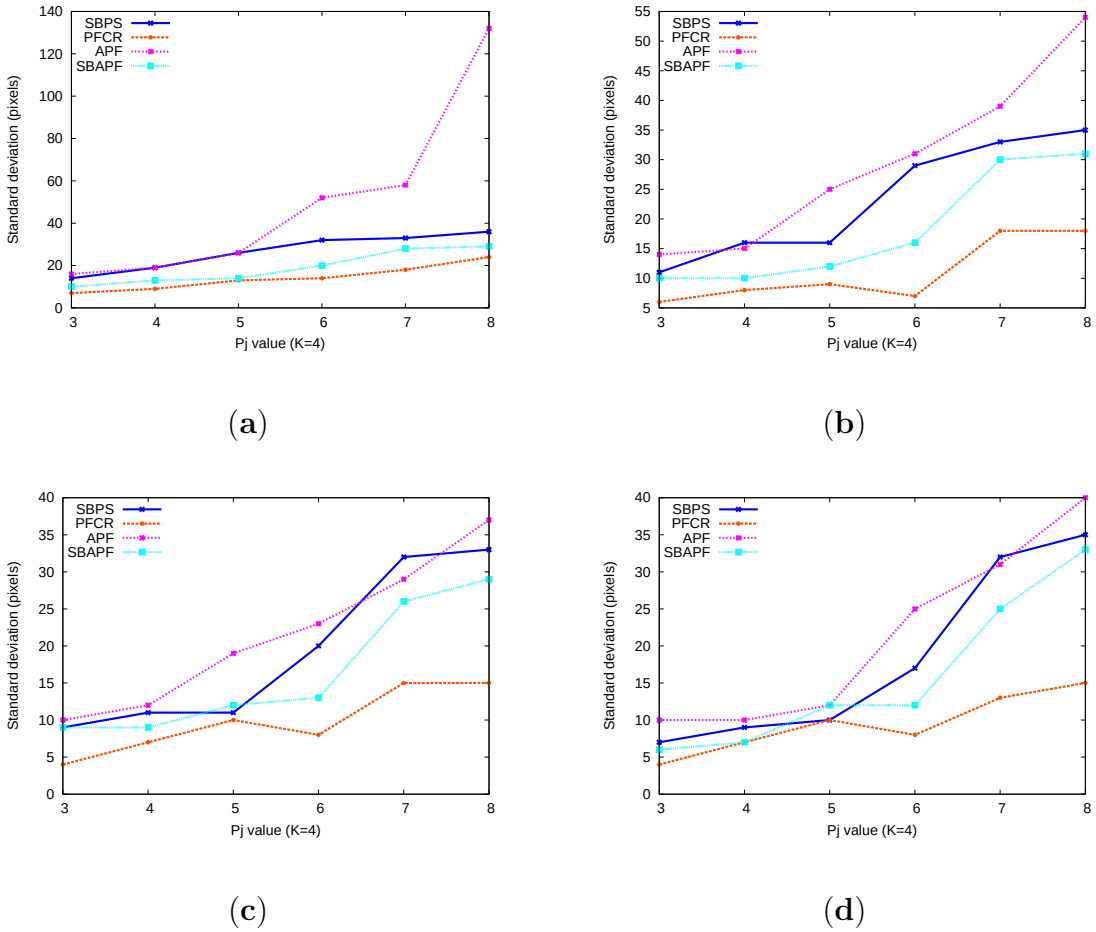


Figure 5-13: **Standard deviations of SBPS, APF, SBAPF and PFCR for tracking an object with $K = 4$, depending on $|P_j|$.** Tracking is performed with (**a**) $N = 50$, (**b**) $N = 100$, (**c**) $N = 200$ and (**d**) $N = 300$.

Computation times. Figure 5-14 shows the computation times of PS, SBPS, SBAPF, APF and PFCR. PS stays the slower approach. As previously, APF and PFCR give equivalent computation times, except that the curves are inverted: PFCR requires lower resampling times than APF when $N = 200$ and $N = 300$, but higher ones when $N = 50$ and $N = 100$. The resampling times for PFCR are higher than those of SBPS and SBAPF. When comparing with PS and APF, PFCR requires more computation times when $|P_j|$ is small. However, its resampling times increase more slower than those of PS and APF, and it becomes faster than PS and APF as $|P_j|$ increases. Especially, it is faster than PS when $|P_j|$ is high.

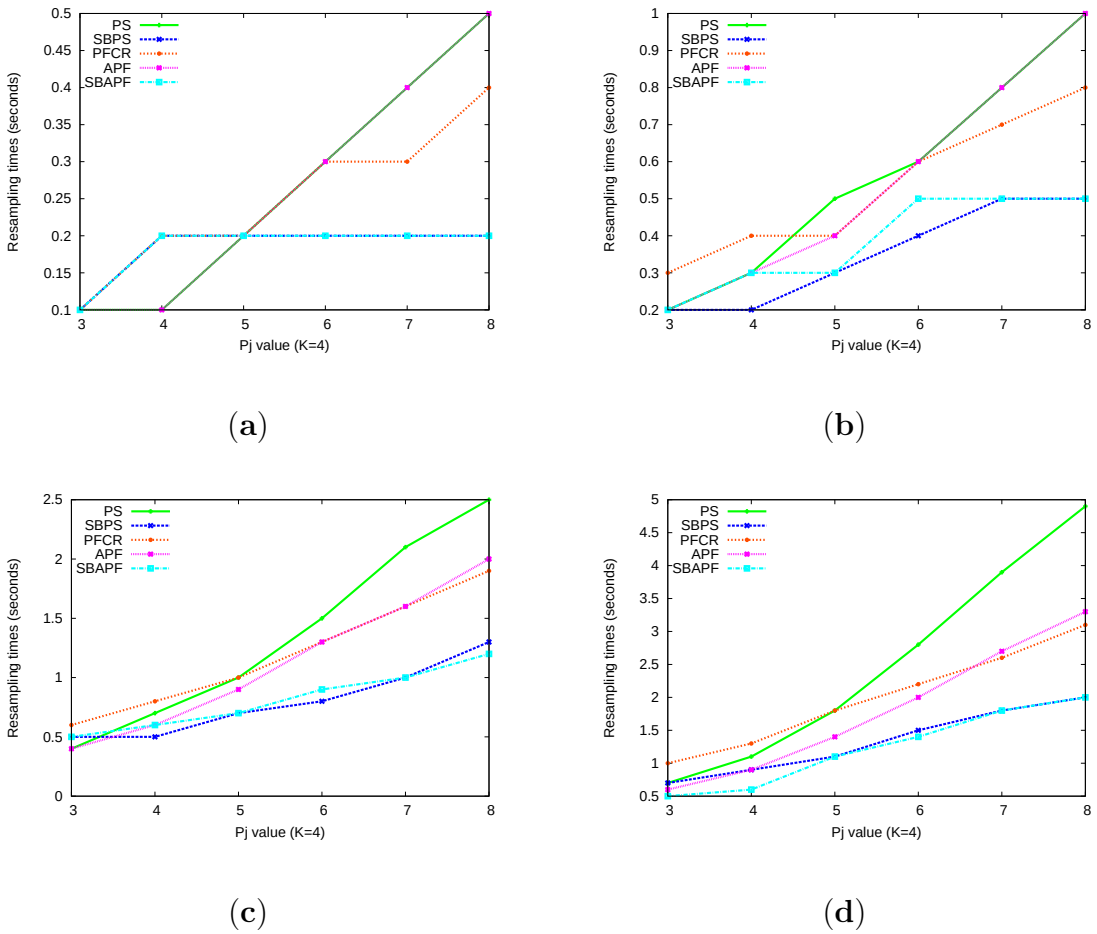


Figure 5-14: **Computation times of SBPS, APF, SBAPF and PFCR for tracking an object with $K = 4$, depending on $|P_j|$.** Tracking is performed with (**a**) $N = 50$, (**b**) $N = 100$, (**c**) $N = 200$ and (**d**) $N = 300$.

5.2.2.3 Performances depending on N (convergence study)

We studied the convergence of PFCR, SBPS, APF and SBAPF in terms of the evolution of the tracking errors and of the standard deviations w.r.t. the number of particles. Tests are made for two different tracked objects: one with long arms ($K = 8$ and $|P_j| = 4$), the other one with a lot of arms ($K = 4$ and $|P_j| = 6$).

Estimation errors. Figure 5-15 shows the estimation errors of PFCR in comparison with those of SBPS, SBAPF and APF for the two tracked objects. PFCR converges faster than the other methods in terms of both estimation errors. This is impressive to note that with only $N = 50$ particles, PFCR gives equivalent or lower estimation errors (340 pixels car the first object, 288 pixels for the second object)

that all the other filters with $N = 300$ particles (errors are 350 pixels for SBAPF, 370 pixels for APF and SBAPF 510 pixels for SBPS for the first object, and 280 pixels for SBAPF, 340 pixels for APF and SBAPF 330 pixels for SBPS for the second object). If we noticed that, in some cases, our PFCR can be slower than other filters when using the same number of particles for tracking, however its rapidity of convergence shows that it can perform well with only few particles, and then become faster.

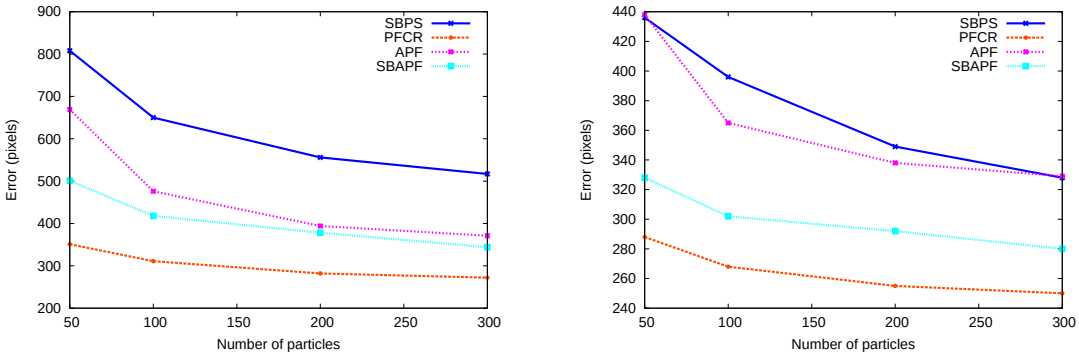


Figure 5-15: **Convergence study (estimation errors) for SBPS, SBAPF, APF and PFCR.** From left to right, an object with $K = 8$, $|P_j| = 4$ ($P = 33$, $|\mathcal{X}| = 35$), and an object with $K = 4$, $|P_j| = 6$ ($P = 25$, $|\mathcal{X}| = 27$).

Standard deviations. Figure 5-15 shows the standard deviations of PFCR in comparison with those of SBPS, SBAPF and APF when tracking two different objects. We remark that PFCR achieves low standard deviations with only a small number N of particles, and that they are relatively stable when N increases. This confirms that PFCR is reliable even with a reasonably small number of particles: particles are concentrated around the modes of the density to estimate, resulting in low standard deviations. This can be explained by the fact that PFCR's behavior guarantees an efficient exploration of the search space even with a small number of particles, by implicitly constructing a particle set of exponential size from all admissible permutations, and by selecting the best particles from this particle set.

Resampling times. Finally, Figure 5-17 reports the resampling computation times of PFCR in comparison with those of PS, SBPS, SBAPF and APF. The fastest approaches are SBPS and SBAPF, and PS stays the slowest. PFCR produces resampling times equivalent to those of APF for the object with a lot of arms (right of Figure 5-17). This confirms the ability of CR to perform very well in such cases. For long

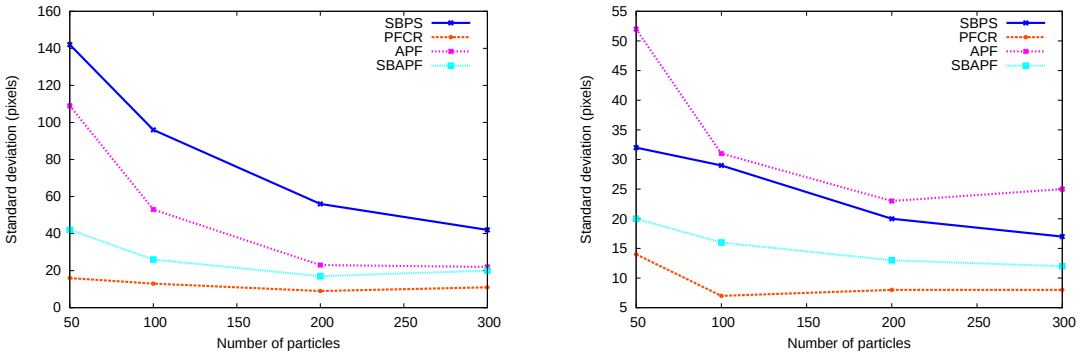


Figure 5-16: **Convergence study (standard deviations) for SBPS, SBAPF, APF and PFCR.** From left to right, an object with $K = 8$, $|P_j| = 4$ ($P = 33$, $|\mathcal{X}| = 35$), and an object with $K = 4$, $|P_j| = 6$ ($P = 25$, $|\mathcal{X}| = 27$).

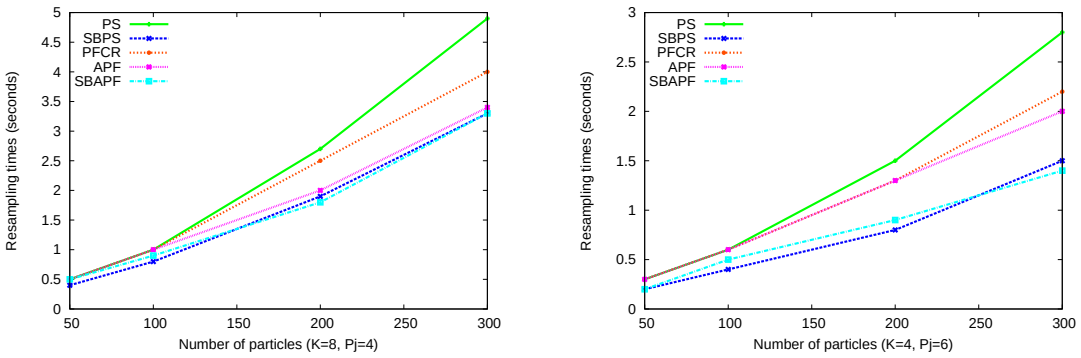


Figure 5-17: **Resampling times for PS, SBPS, SBAPF, APF and PFCR.** From left to right, an object with $K = 8$, $|P_j| = 4$ ($P = 33$, $|\mathcal{X}| = 35$), and an object with $K = 4$, $|P_j| = 6$ ($P = 25$, $|\mathcal{X}| = 27$).

arms (left of Figure 5-17), APF outperforms PFCR in terms of resampling times. This confirms the results we obtained in the preceding sections. Note that, except for PS, the slopes of the resampling time's curve is relatively equivalent for the other methods, i.e., for PFCR, SBPS, APF and SBAPF.

5.2.2.4 Combinatorial resampling versus annealing

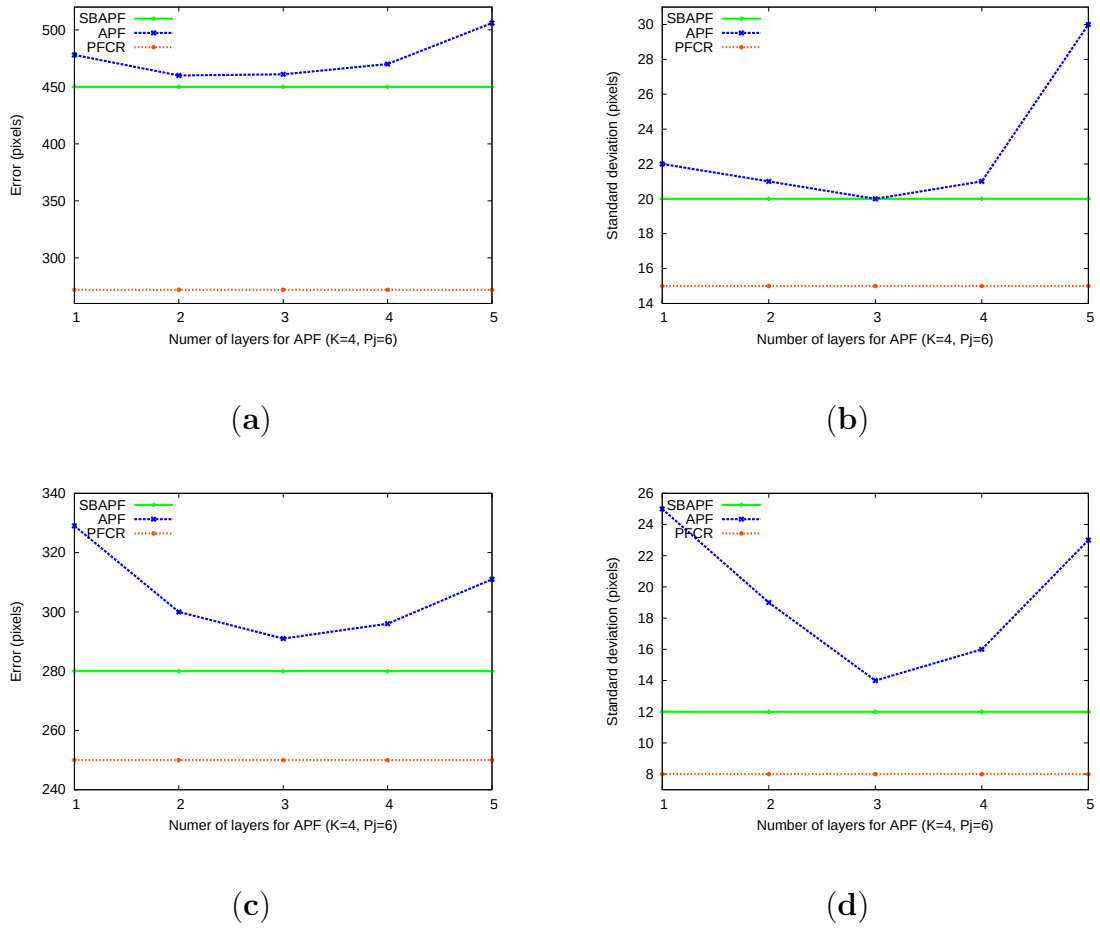


Figure 5-18: **Comparison of the performance of APF, SBAPF and PFCR, depending on the number of layers L.** (**a**) and (**b**) provide the estimation errors and standard deviations respectively of PFCR, APF and SBAPF when tracking an object with $K = 8$, $|P_j| = 4$. (**c**) and (**d**) provide the estimation errors and standard deviations respectively of PFCR, APF and SBAPF when tracking an object with $K = 4$, $|P_j| = 6$.

Similarly to Section 4.4.4.4, we study in this section the impact of the number L of layers of annealing on APF and compare the estimation errors and standard

deviations compared to those of SBAPF and PCFR. We added the estimation errors and the standard deviations of PFCR obtained when tracking the two objects mentioned in the previous section into Figure 4-25 to compare PFCR's performance with those of SBAPF and APF, which results in Figure 5-18. Recall that the results presented in this figure are obtained by fixing $N = 300$ and varying the number of layers of APF from 1 to 5. Results highlight the robustness of PFCR, because we note that APF never reaches the estimation errors nor the standard deviations provided by PFCR. Instead of trying to construct the best particles from a particle set resulting from *only one* admissible permutation, as SBPS does, PFCR stochastically draws particles from the particle set resulting from *all* admissible permutations. This results in an increased number of particles near the modes of the target density. As a consequence, PFCR achieves more accurate tracking than SBPS, and, *a fortiori* than APF, for any value of L. For example, the estimation error is reduced from 450 pixels to 260 between SBAPF and PFCR for the first object, and from 280 pixels to 250 between SBAPF and PFCR for the second object. Similarly, the standard deviation is reduced from 20 pixels to 15 between SBAPF and PFCR for the first object, and from 12 pixels to 8 between SBAPF and PFCR for the second object.

In this section, we have proposed a deep comparative study on synthetic video sequences with different filters to show the interest of CR and its particle filter's embedded version, namely PFCR. First, PFCR always gives lower estimation errors, and lower and more stable standard deviations. We have also shown that it does not require a high computation time, compared to APF, in particular when the number of parts that are treated in parallel is high (i.e. $|P_j|$ high). In the next section, we test our algorithm on a challenging real video sequence and show it is usable for real applications consisting of tracking human body modeled by an articulated object.

5.2.3 Tests on a real video sequence

We tested our approach on a sequence from the UCF50 dataset[1], 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. Qualitative results are given by superimposing on the frames of the sequences a red articulated object corresponding to the estimation derived from the weighted sum of the particles. For this test, we fixed $\sigma_x = \sigma_y = 2$ pixels and $\sigma_\theta = 0.08$ rad.

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

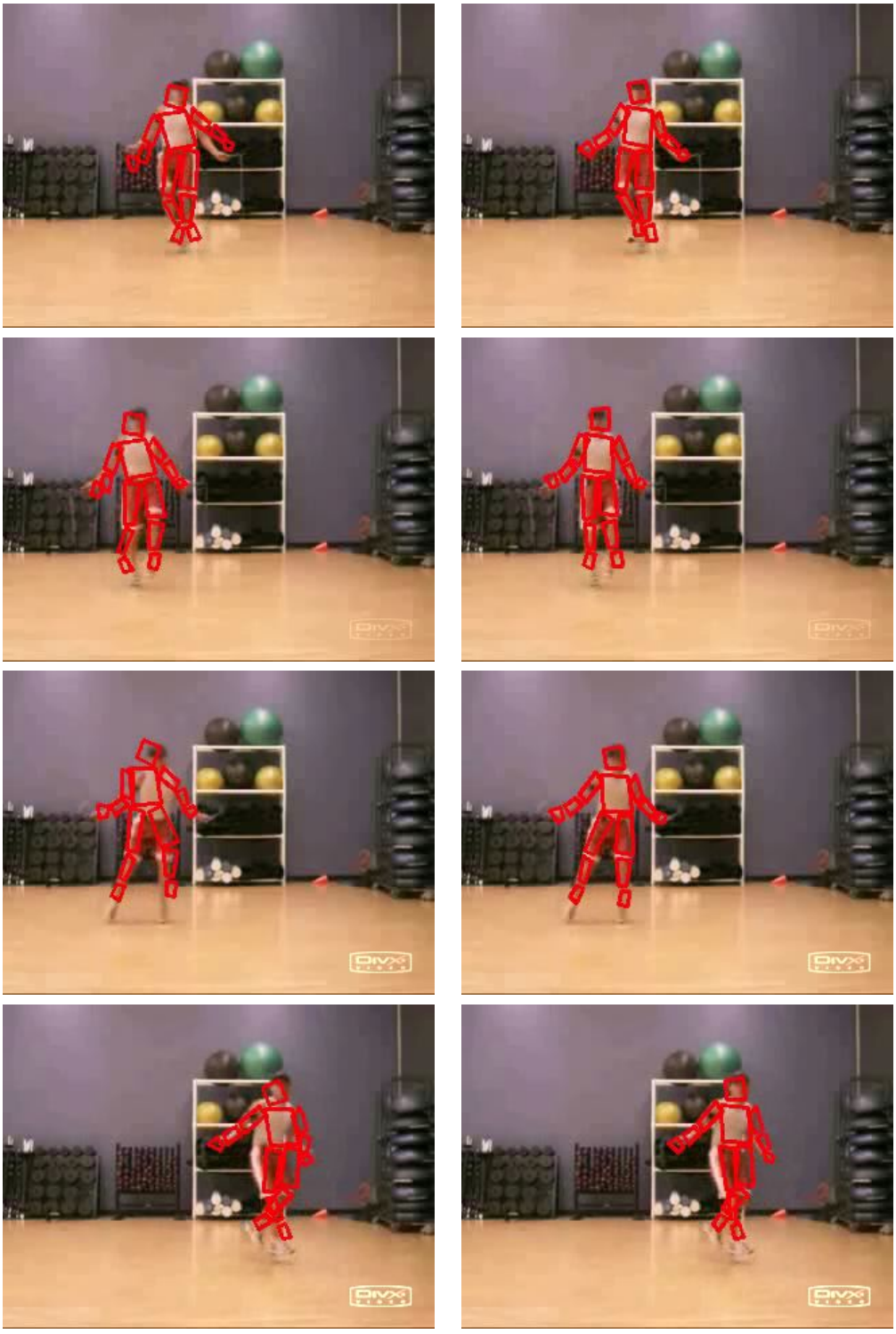


Figure 5-19: **Results on a real video sequence.** Tracking results on `JumpRope` sequence with $N = 500$ particles (frames 10, 50, 121 and 234). Left: using residual resampling, right: using our combinatorial resampling.

Figure 5-19 shows tracking results on the `JumpRope` sequence (containing 290 frames of 320×240 pixels) with $N = 500$ particles. In this sequence, a person is quickly moving from left to right while jumping, and crossing/uncrossing his arms and legs. For this test, we defined an articulated object with $P = 12$ parts, hence $|\mathcal{X}| = 36$, and we compared the estimations resulting from PS with a residual resampling (left column) with those resulting from our proposed resampling approach (right column). As can be observed, our approach produces better results: its estimations are more stable along the sequence. For example, on the images of the 2nd and 3rd lines, we can see that the estimation of the articulated object fails with residual resampling but is correct with our combinatorial resampling. For this sequence, on average over 20 runs, our method needed 16 seconds while residual resampling needed 22. In addition, our algorithm proved to be more robust and provided more accurate results. As for synthetic sequences, our tests show that the higher the number of particles, the more our algorithm outperforms residual resampling in terms of response time. It is also always more accurate.

5.3 Conclusion

In this chapter, we have introduced a new resampling method called *Combinatorial Resampling*. From a given sample $\mathcal{S}$, this algorithm constructs implicitly a new sample $\mathcal{S}'$ exponentially larger than $\mathcal{S}$. By construction, $\mathcal{S}'$ is more representative than $\mathcal{S}$ of the density over the whole state space and resampling from $\mathcal{S}'$ rather than $\mathcal{S}$ produces much better results, as confirmed by our experiments. We proved the mathematical correctness of the method and showed that it is effective for real time tracking. Comparisons with other resampling schemes show the ability of our CR to reduce the estimation errors, and to converge better and faster. When CR is embedded into the SBPS framework, resulting in PFCR, it gives lower estimation errors and standard deviations, without increasing drastically computation times.

For future researches, there remains to exhibit theoretical convergence results for SBPS combined with this new resampling scheme. A deeper study of the sampling quality should also be investigated. If we observed that CR and PFCR provide lower estimation errors and standard deviations, we are did not prove yet that the modes of the density to estimate are correctly sampled.

In this chapter and the preceding one, we investigated how classical particle filters could be improved by exploiting d-separation, the independence property at the core of OTDBNs. This property allowed us to introduce swappings within the particle

filter framework, leading usually to a significant tracking improvement. However, there exist cases where the likelihood functions provide misleading results, e.g., when there are occlusions or clutter and, in these cases, swapping may fail to improve tracking. In the next chapter, we investigate such cases and show how metaheuristics, especially Particle Swarm Optimization, can be enhanced.

Chapter 6

Hierarchical Annealed Particle Swarm Optimization for Articulated object tracking

In this chapter, we present a preliminary work that investigates the combination of a decomposition scheme and Particle Swarm Optimization (PSO): we propose a novel algorithm for articulated object tracking, based on a hierarchical search and particle swarm optimization. Our proposed approach aims to reduce the complexity induced by the high dimensional state space in articulated object tracking by decomposing the search space into subspaces and then using particle swarms to optimize the estimation into these subspaces hierarchically. Moreover, the intelligent search strategy proposed in [112] is integrated into each optimization step to provide a robust tracking algorithm under noisy observation conditions. We have compared the proposed algorithm with other common existing algorithms, including partitioned sampling with annealing (APF), the classical particle filter with annealing (PFAPF), hierarchical particle swarm optimization (HPSO) and annealed particle swarm optimization based particle filter (APSOPF). Our quantitative results on synthetic video sequences and on real video sequences show the efficiency of the proposed approach in comparison with these competing approaches.

6.1 Proposed approach

6.1.1 Motivation

The problem of noisy observation is known to be one of the most challenging in articulated object tracking, that should be taken into account to achieve a good tracking.

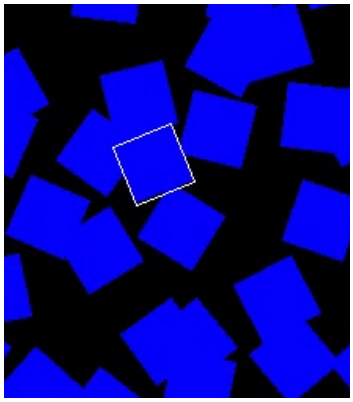


Figure 6-1: **Example of a rigid object tracking inside a noisy environment.** The tracked blue object's boundary is represented by a white rectangle.

Some recent works have shown that PSO is an effective optimization approach for dealing with noisy observations. Among these works, the approach proposed in [112], called annealed particle swarm optimization based particle filter (APSOPF), which is discussed in Section 3.2.2, is one of the most remarkable. In this approach, a sampling covariance and some annealing terms are introduced into the PSO update equation. The annealing terms help to reduce the impact of the global and local best particles on the particle swarm when the search is focused on global optima, since in such situation, the global and local best particles are no longer trustable due to noise. In this section, we first investigate the effectiveness of these terms. This leads to the conclusion that it is important to reduce the impact of the global and local best particles in PSO when noise is present. This motivates us to propose a novel approach based on APSOPF, called Hierarchical Annealed Particle Swarm Optimization (HAPSOPF), dedicated to the articulated object tracking into high dimensional state spaces and noisy environments, that improves considerably its computational cost as well as its tracking accuracy.

For this purpose, we conducted an experiment on a synthetic video sequence where a rigid object is moving over time. The rigid object is modeled by a rectangle whose state is defined by the coordinates of its center and its orientation. A cluttered background was generated by randomly drawing squares and/or rectangles having the same color as the rigid object (here blue) in the image sequences (see Figure 6-1). Color was used in this experiment to construct the likelihood function (see Section 2.2.5.3): the reference color histogram of the target object is computed in the first frame. In order to evaluate the benefit of incorporating the annealing terms into the PSO update equation, we compared APSOPF and one its variant in which

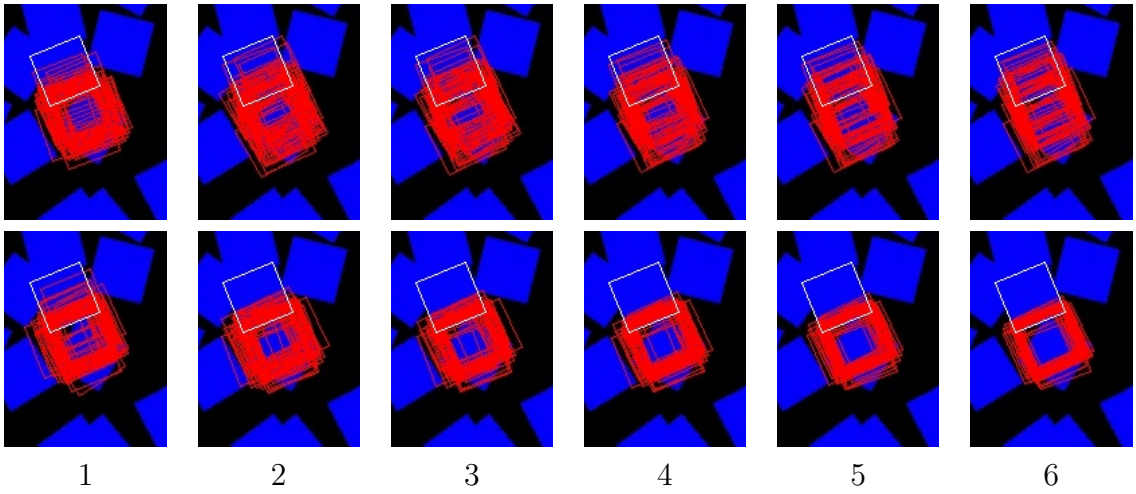


Figure 6-2: **The effectiveness of annealing terms in PSO update equation**. From top to bottom, tracking results given by APSOPF and CAPSOPF during the 6 iterations (left to right). In white: the true state, in red: the local and global best particles.

the annealing terms are replaced by a constant equal to 2, leading to the canonical PSO proposed in [48], which is referred as CAPSOPF in the following. The number of PSO iterations for both methods is 6. In Figure 6-2, the true object in different frames is represented by white rectangles, while the global and local best particles are symbolized by red rectangles. Results after each iteration given by APSOPF and CAPSOPF are shown in the first and second rows respectively. Both methods start the tracking with the same particle swarm, for which global and local best particles get stuck into wrong positions. After the first iteration (first column in Figure 6-2), the quality of particle swarm in both methods is not improved and the only difference between them is due to the random process. During the five next iterations (the *2*nd, *3*th, *4*th, *5*th, *6*th columns in Figure 6-2), the particle swarm of CAPSOPF (second line) is still stuck into wrong positions and cannot track the target object due to the wrong guide of the global and local best particles. In contrast, in APSOPF (first line), the particles follow their own searching strategy at the end of iterations, giving them more chance to get out of local maxima and to explore other regions of the search space.

The above example shows the interest of annealing terms in PSO when dealing with noise. We follow this idea to propose a new search strategy. In APSOPF, the inertial velocity is replaced by a sampling covariance. The reason for using this term instead of a inertial velocity, however, is less intuitive in [112] since the authors observed that it helped to produce more plausible human poses. We show in Section 6.2

that this sampling covariance helps to explore the search space more efficiently than the inertial velocity does. In cases where information about the movements of objects is available, and that motion priors can be learnt from it, an obvious advantage of using the sampling covariance is that it moves particles toward more promising regions of the search space, thus increasing the searching efficiency of PSO and reducing the number of particles required for successful tracking.

One limitation of APSOPF is that it performs PSO in the search space of the target object, which makes it computationally expensive when the dimension of the state space is very high. To alleviate this problem, we follow the methodology of decomposition approaches for articulated object tracking and propose to combine APSOPF with a hierarchical search. The proposed approach is introduced next.

6.1.2 Proposed algorithm

We propose to exploit the hierarchical nature of the kinematic structure of the articulated object to improve tracking. First, the state space of the target object is decomposed into lower dimensional subspaces. Then, optimal states are searched for into these subspaces in the hierarchical order of the kinematic structure using Partitioned Sampling (PS) [64]. These optimal states are then used to constrain the search in the next subspaces in the hierarchical order.

At time t, let $\mathbf{x}_t^{(i),k}$ (resp. $\mathbf{s}_t^{(i),k}$) denote the kth substate of the ith particle $\mathbf{x}_t^{(i)}$ (resp. the ith particle's best state $\mathbf{s}_t^{(i)}$) and let $\mathbf{s}_t^{\mathbf{g},k}$ be the kth substate of the global best state found so far. Recall that articulated objects are constituted of P parts. Then, at the mth iteration, $\mathbf{x}_{t,(m)}^{(i)} = \{\mathbf{x}_{t,(m)}^{(i),1}, ..., \mathbf{x}_{t,(m)}^{(i),P}\}$, $\mathbf{v}_{t,(m)}^{(i)} = \{\mathbf{v}_{t,(m)}^{(i),1}, ..., \mathbf{v}_{t,(m)}^{(i),P}\}$ and $\mathbf{s}_{t,(m)}^{(i)} = \{\mathbf{s}_{t,(m)}^{(i),1}, ..., \mathbf{s}_{t,(m)}^{(i),P}\}$. We follow the approach proposed in [112], and update the velocity and the position of particles at each time step as follows:

$$\mathbf{v}_{t,(m)}^{(i),k} = r_0 \mathbf{P}_{(m-1)} + \beta_1 r_1 (\mathbf{s}_t^{(i),k} - \mathbf{x}_{t,(m-1)}^{(i),k}) + \beta_2 r_2 (\mathbf{s}_t^{\mathbf{g},k} - \mathbf{x}_{t,(m-1)}^{(i),k}) \quad (6.1)$$

$$\mathbf{x}_{t,(m)}^{(i),k} = \mathbf{x}_{t,(m-1)}^{(i),k} + \mathbf{v}_{t,(m)}^{(i),k} \quad (6.2)$$

where r_0, r_1, r_2 are random numbers uniformly drawn from $[0, 1]$. Note that in [112], they are the absolute values of some random numbers drawn from the Gaussian distribution $\mathcal{N}(0, 1)$. However, we found that this way of generating r_0, r_1, r_2 does not give results as good as ours.

$\mathbf{P}_{(m-1)} = \alpha_0 * \mathbf{P}_{(m-2)}$, $m \geq 2$, is the sampling covariance, with α_0 a constant, and $\mathbf{P}_{(0)}$ a covariance matrix whose diagonal elements are fixed with respect to the model configuration parameters. We propose to compute factors β_1 and β_2 at each iteration

m using the annealing principle so that:

$$\beta_1 = \beta_2 = \beta_0\beta_{max}\left(\frac{\beta_{max}}{\beta_{min}}\right)^{-\frac{m}{M}} \tag{6.3}$$

where $\beta_0, \beta_{max}, \beta_{min}$ are constants, $0 < \beta_0 \leq 1$, $\beta_{min} < \beta_{max}$ and M is the maximal number of iterations.

The parameters β_{max}, β_{min} are used to control the annealing rate. β_1, β_2 start with the value $\beta_0\beta_{max}$ at the first iteration ($m = 0$), then they are gradually decreased until reaching the value $\beta_0\beta_{min}$ at the last iteration ($m = M$). Thus $\beta_0\beta_{min}$ should be set to a reasonably small value (e.g. < 1) to reduce the impact of global and local best particles at the end of iterations. In our experiments, we observed that the value of $\beta_0\beta_{max}$ should be set in accordance with the maximal number of iterations M since large values of $\beta_0\beta_{max}$ and M made the proposed approach unstable.

By combining PSO and hierarchical search, our approach aims to increase the tracking accuracy and to reduce the computational cost of the tracking algorithm by integrating the benefits of both methods. First, the search efficiency is improved by performing PSO within lower dimensional subspaces, thereby increasing tracking accuracy. Second, since the search is performed in the same way as PS, the number of particles required and thus the computational cost of the tracking algorithm is greatly reduced. Our proposed Hierarchical Annealed Particle Swarm Optimization Particle Filter (HAPSOPF) is described in Algorithm 6.1, where $\bar{\mathbf{x}}$ is the estimated state of the object at time slice t, $w(., \mathbf{y})$ is the cost function to be optimized by PSO, and $\mathbf{y}$ is the current observation.

6.2 Experimental results

We compared our approach with PFAPF [25], APF [7], APSOPF [112] and HPSO [47]. We also implemented a variant of our proposed approach (which we called CHAPSOPF, for Canonical Hierarchical Annealed Particle Swarm Optimization Particle Filter) in which we replaced the sampling covariance in Equation 6.1 with the inertia velocity proposed in [47]. The *cost function* $w(\mathbf{x}_{t,(m)}^{(i),k}, \mathbf{y})$ to be optimized by PSO measures how well a state hypothesis $\mathbf{x}_{t,(m)}^{(i),k}$ matches the true state w.r.t. the observed image $\mathbf{y}$, and is constructed using histogram and foreground silhouette [25]. An articulated object is described by a hierarchy of parts (a tree), each part being linked to its parent in the tree by an articulation point. For instance, in the top row of Figure 6-3, the blue polygonal parts are the root of the tree and the colored rectangles are the other nodes of the tree. The root is described by its center (x, y)

Input: $\{\mathbf{s}_{t-1}^{(i)}\}_{i=1}^{N}$, α_0, β_0, β_{max}, β_{min}, $\mathbf{P}_{(0)}$, M (number of iterations)
Output: $\{\mathbf{s}_{t}^{(i)}\}_{i=1}^{N}$

Set $\pi_t^{(i)} = 1$, $i = 1, \ldots, N$
for $k = 1$ **to** P **do**
  Sample: $\mathbf{x}_{t,(0)}^{(i),k} \sim \mathcal{N}(\mathbf{s}_{t-1}^{(i),k}, \mathbf{P}_{(0)})$, $i = 1...N$
  for $m = 0$ **to** M **do**
    if $m \geq 1$ **then**
      Compute $\mathbf{P}_{(m)}$ and update β_1, β_2
      Carry out the PSO iteration based on Equations 6.1 and 6.2
    Evaluate: $f(\mathbf{x}_{t,(m)}^{(i),k}) = w(\mathbf{x}_{t,(m)}^{(i),k}, \mathbf{y})$, $i = 1...N$
    Update $\{\mathbf{s}_t^{(i),k}\}_{i=1}^{N}$ and the kth part of the global best state $\mathbf{s}_t^{\mathbf{g},k}$
  Evaluate particle weights: $\pi_t^{(i)} = \pi_t^{(i)} \times w(\mathbf{s}_t^{(i),k}, \mathbf{y})$, $i = 1, \ldots, N$
Normalize particle weights: $\bar{\pi}_t^{(i)} = \frac{\pi_t^{(i)}}{\sum_{j=1}^{N} \pi_t^{(j)}}$, $i = 1...N$
return $\{\mathbf{s}_t^{(i)}\}_{i=1}^{N}, \bar{\mathbf{x}} = \sum_{i=1}^{N} \bar{\pi}_t^{(i)} \mathbf{s}_t^{(i)}$

Algorithm 6.1: Our HAPSOPF algorithm.

and its orientation θ whereas the other parts are only characterized by their angle θ. For all algorithms, particles are propagated using a random walk with standard deviations fixed to $\sigma_x = 2$, $\sigma_y = 2$ and $\sigma_\theta = 0.05$. For APSOPF and HAPSOPF, $\mathbf{P}_{(0)}$ is a diagonal matrix with the values of σ_x, σ_y and σ_θ. The values of β_0, β_{max} and β_{min} are empirically set as $\beta_0 = 0.5$, $\beta_{max} = 3$ and $\beta_{min} = 1$. Our comparisons are based on three criteria: estimation errors, standard deviations and computation times.

6.2.1 Tests on synthetic sequences

6.2.1.1 Video sequences

In order to demonstrate the robustness of the proposed approach to noise, we have generated two sets of various synthetic video sequences composed of 200 frames of 640×480 pixels (with ground truth). The video sequences in the first set contain no noise whereas, in the second set, cluttered backgrounds were generated. The clutter is made up of polygons and rectangles randomly positioned in the image. As defined in the previous chapters, an articulated object is defined by its number $|P_j|$ of arms, and their length $K - 1$: some examples are given in Figure 6-3.

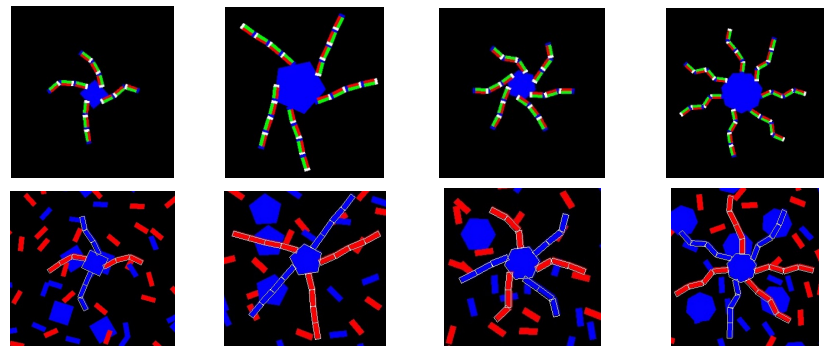


Figure 6-3: **Synthetic video sequences used for quantitative evaluation.** The number of arms $|P_j|$ and the length of arms $K - 1$ are given below. Sequence (**a**) without clutter and (**b**) with clutter.

6.2.1.2 Quantitative tracking results

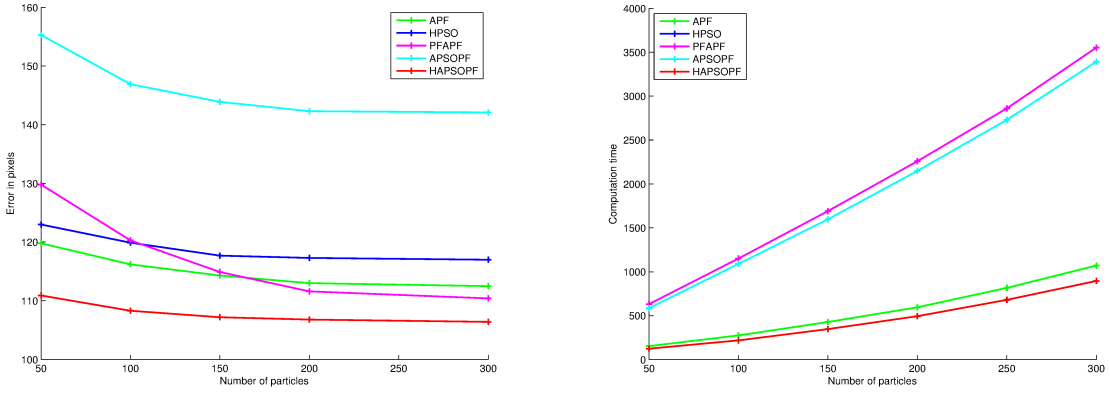
The tracking errors are given by the sum of the Euclidean distances between each corner of the estimated parts and their corresponding corner in the ground truth. We used $M = 3$ layers for APF and PFAPF since it produces stable results for both algorithms, and $M = 3$ maximal iterations for HAPSOPF, HPSO, APSOPF and CHAPSOPF. Table 6.1 gives the performances of the tested algorithms for sequences without or with noise (cluttered background). In our experiments, tracking in noisy sequences is challenging due to the background. All tracking algorithms can lost track of some parts of the target object during some interval of time providing tracking failures. This can happen when these parts and their underlying background have the same color: then the particle swarm gets stuck and cannot escape from that location. In such cases, the annealing process of APF forces the particle set to represent one

Table 6.1: Tracking errors in pixels (average over 30 runs) and standard deviations for synthetic video sequences, N is the number of particles used per filter.

		$\|P_j\| = 4, K = 4$		$\|P_j\| = 5, K = 5$		$\|P_j\| = 6, K = 4$		$\|P_j\| = 7, K = 5$	
	N	50	200	50	200	50	200	50	200
HAPSOPF	No noise	110(2)	106(1)	214(5)	195(2)	243(11)	211(9)	312(7)	271(4)
	Noise	204(39)	143(10)	227(56)	175(30)	322(67)	295(60)	553(194)	516(180)
APF	No noise	120(2)	114(1)	238(6)	208(4)	251(7)	218(3)	319(8)	278(4)
	Noise	309(109)	221(94)	281(78)	219(48)	432(86)	388(75)	1008(232)	914(213)
HPSO	No noise	125(5)	119(2)	252(9)	227(5)	254(11)	213(6)	382(5)	315(3)
	Noise	277(78)	194(65)	245(42)	201(26)	345(27)	295(10)	922(334)	731(259)
APSOPF	No noise	184(3)	169(2)	260(12)	241(10)	265(15)	257(12)	471(30)	439(21)
	Noise	254(16)	227(8)	308(33)	291(25)	490(68)	474(47)	817(223)	785(169)
PFAPF	No noise	128(3)	109(2)	246(11)	221(9)	270(13)	236(11)	487(35)	412(24)
	Noise	272(9)	258(5)	322(29)	309(18)	440(51)	429(40)	613(174)	592(156)
CHAPSOPF	No noise	129(4)	117(2)	265(16)	243(6)	280(51)	235(15)	432(29)	326(9)
	Noise	235(87)	206(60)	262(46)	229(24)	336(73)	299(66)	749(259)	721(216)

of the modes of the cost function, which also causes these parts to get stuck in that wrong location. This problem of annealing approaches has been previously reported in [16]. As can be observed, APF performs worst than HAPSOPF and HPSO in these tests. This shows the shortcoming of APF-based approaches and highlights the advantage of PSO-based approaches when tracking under noisy observations. This problem has less impact on our HAPSOPF method than on HPSO: as we can see, HAPSOPF outperforms HPSO and CHAPSOPF in terms of both estimation errors and standard deviations in most experiments. This can be explained by the contribution of two factors in HAPSOPF's update equation: the sampling covariance and the annealing terms. First, the annealing terms are gradually decreased during the PSO search, which helps the particle swarm to follow its own searching strategy at the end of iterations without being affected by any wrong guide of the local or global best states. Second, the use of the sampling covariance leads to an efficient exploration of the search space without losing the searching power of PSO. We can also observe the benefit of hierarchical search in HAPSOPF, as it clearly outperforms APSOPF in terms of both estimation errors and standard deviations.

Finally, Figure 6-4 gives comparative convergence results (error depending on the number N of particles) and computation times for a synthetic sequence. Note that our approach converges better and faster than the other methods. In particular, it is much faster than APSOPF. This shows the interest of decomposition technique when combining with APSOPF.



(**a**) (**b**)

Figure 6-4: **Comparison tests for convergence and computation time when tracking the object defined with** $|P_j| = 4, K = 4$**.** (**a**) Convergence and (**b**) Computation times (HPSO and our approach give the same curves).

6.2.2 Tests on real sequences

6.2.2.1 Dataset

We used sequences `S1 Gesture` and `S2 Throwcatch` of the HumanEva-I dataset [87]. In this dataset, the ground truth is provided in motion capture data, which were collected synchronously with the video data. Since cameras are calibrated, ground truth data points can be projected on the image sequences to evaluate quantitatively our proposed approach. Some extracted frames from these two sequences are given in Figure 6-5.

Sequence `S2 Throwcatch` contains self-occlusions (hands and torso, left and right hands, left and right legs) which complicate matters for the tracking algorithms. Moreover, in both sequences, the lower right hand of the subjects performing actions makes a lot of quick movements which makes it difficult to track.

(**a**) (**b**)

Figure 6-5: **Extracted frames from tested sequences.** (**a**) `S1 Gesture`. (**b**) `S2 Throwcatch`.

The searching order for APF, HPSO, and HAPSOPF is: torso, head, left thigh, right thigh, left upper arm, right upper arm, left leg, right leg, left forearm, right forearm. For a fair comparison, we fixed the number of evaluations of the weighting function at each frame for all the algorithms to 2000, and tuned parameters $\{N, M\}$, where N is the number of particles and M is the number of PSO's iterations for each method so that they achieve the best performance while satisfying the above constraint: $\{400, 5\}$ for PFAPF, $\{40, 5\}$ for APF, $\{200, 10\}$ for APSOPF and $\{20, 10\}$ for HPSO and HAPSOPF.

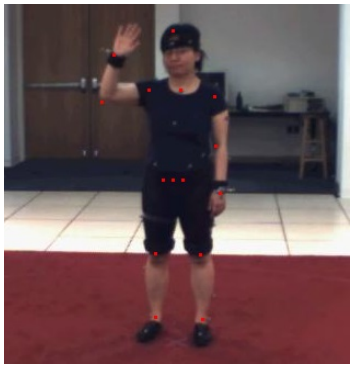


Figure 6-6: **Ground truth for a tested sequence.** The position of the 15 markers used for quantitative evaluation.

6.2.2.2 Evaluation measures

We used the evaluation measure proposed in [87, 16]. Let $\mathbf{x}$ represent the state of the body. We define $\mathcal{M} = 15$ virtual markers as $\{m_i(\mathbf{x})\}$, $i = 1, \ldots, \mathcal{M}\}$, where $m_i(\mathbf{x}) \in \mathbb{R}$ is a function of the body state that returns the position of the ith marker in the image (see Figure 6-6). The error between the estimated state $\bar{\mathbf{x}}$ and the ground truth state $\mathbf{x}$ is expressed as the Euclidean distance between all virtual markers:

$$D(\bar{\mathbf{x}}, \mathbf{x}) = \sum_{i=1}^{\mathcal{M}} ||m_i(\mathbf{x}) - m_i(\bar{\mathbf{x}})||$$

For the sequence of T frames we compute the average performance using the following:

$$\mu = \frac{1}{T} \sum_{i=1}^{T} D(\bar{\mathbf{x}}, \mathbf{x})$$

6.2.2.3 Tracking results

Table 6.2 provides tracking errors (e, in pixels) and total computation times (t in seconds) when tracking on the two sequences. As can be observed, our approach gives the same computation times as those of HPSO but reduces the estimation error and it outperforms the other approaches on both criteria. The difference of estimation errors between our approach and those of the other approaches are larger for Sequence `S2 Throwcatch` than for Sequence `S1 Gesture`, which suggests that our approach is also robust to fast and erratic movements.

Figures 6-7 and 6-8 provide the corresponding qualitative tracking results. For Sequence `S2 Throwcatch`, our approach always outperforms PSAPF and HPSO in

Table 6.2: Tracking errors for full body in pixels (average over 30 runs), and total computation times (in seconds), obtained when tracking on the two sequences.

	HAPSOPF		APF		HPSO		APSOPF		PFAPF	
	e	t	e	t	e	t	e	t	e	t
`S1 Gesture`	95(6)	287	99(11)	293	101(9)	287	102(4)	1348	105(2)	1412
`S2 Throwcatch`	212(10)	557	227(19)	579	232(12)	557	235(7)	2070	240(5)	2184

cases of self-occlusions (frames 275, 523) or fast motions (frames 160, 387), showing its robustness.

Because our approach incorporates the annealing into each searching stage of the hierarchical search, the problem of noisy observations is effectively alleviated. This makes our approach more robust to self-occlusions.

6.3 Conclusion

In this chapter, we have introduced a new algorithm for articulated object tracking based on particle swarm optimization and hierarchical search. We addressed the problem of articulated object tracking in high dimensional spaces by using a hierarchical search to improve search efficiency. Furthermore, the problem of noisy observation has been alleviated by incorporating the annealing factor terms into the velocity updating equation of PSO. Our experiments on synthetic and real video sequences demonstrate the efficiency and effectiveness of our approach compared to other common approaches, both in terms of tracking accuracy and computation time.

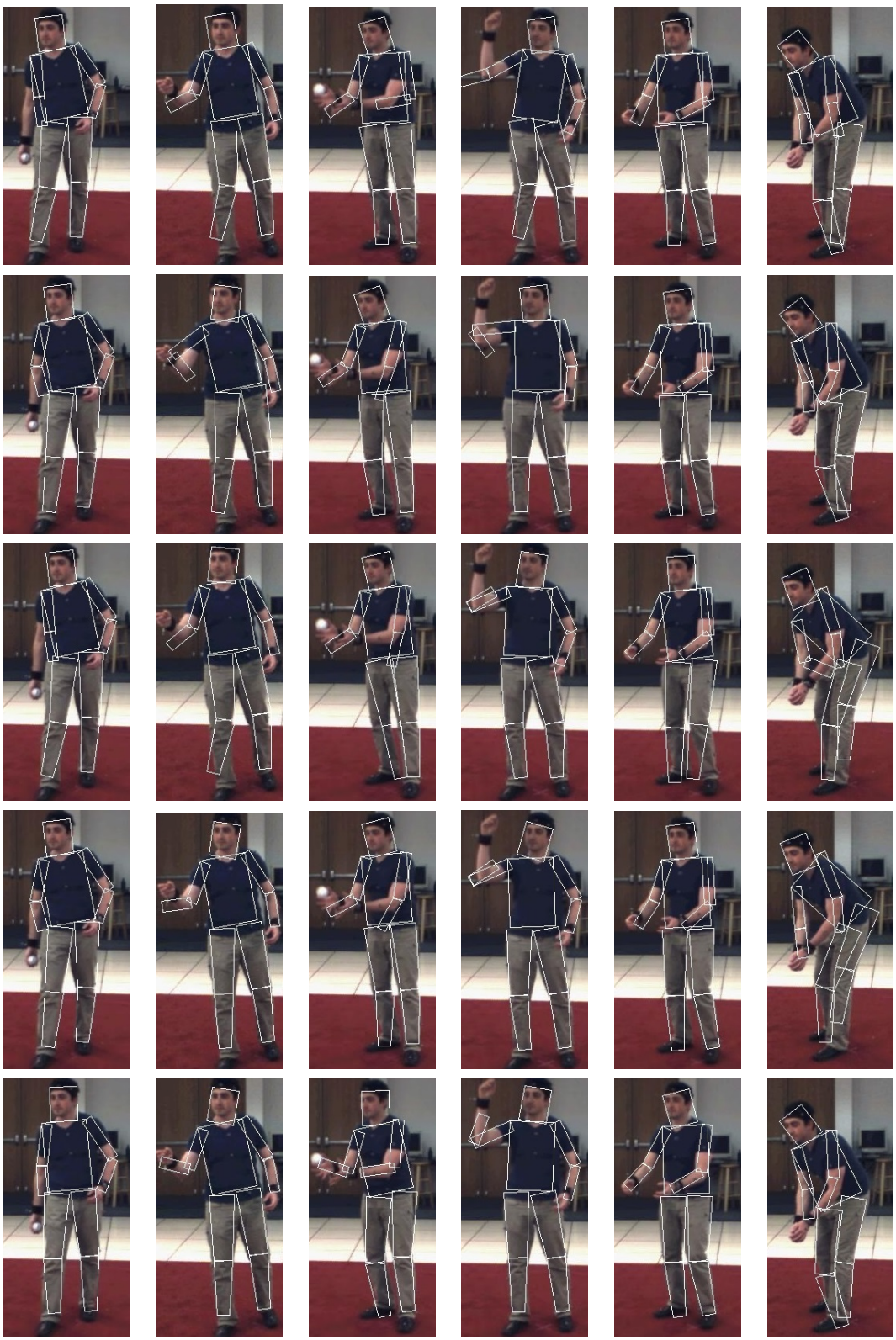


Figure 6-7: **Tracking results for frames 123,160,275,387,488,523 of sequence** `S2 Throwcatch`**.** PFAPF (*1*th row), APSOPF (*2*nd row), HPSO (*3*th row), APF (*4*th row) and HAPSOPF (*5*th row).

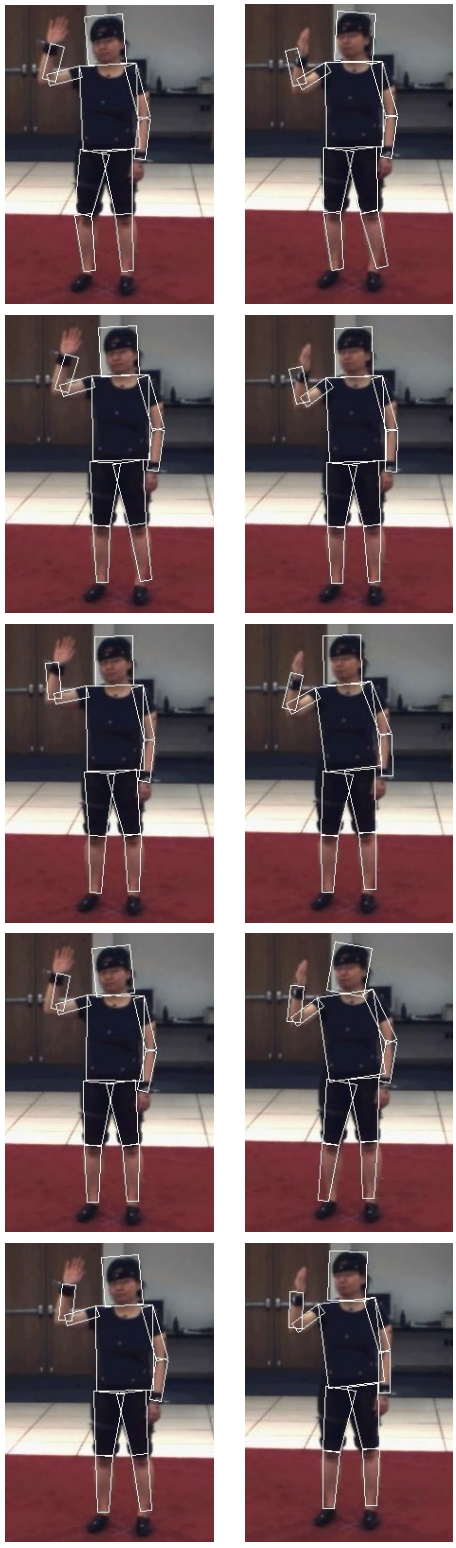


Figure 6-8: **Tracking results for frames 76 and 110 of sequence `S1 Gesture`.** PFAPF (*1*th row), APSOPF (*2*nd row), HPSO (*3*th row), APF (*4*th row) and HAPSOPF (*5*th row).

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Tracking articulated structures with accuracy and within a reasonable time is challenging. One of the main difficulties lies in the high complexity incurred by the high dimensional space of the problem. Moreover, an applicable and robust tracking algorithm must have not only real-time performances but also a high accuracy. Various existing approaches have shown their good performance in terms of accuracy, but their high computational cost makes them impractical. Therefore, it is crucial to reduce the complexity of the tracking algorithm for articulated objects to enable real-time performance.

In this thesis, we proposed to exploit decompositions within the particle filter framework to improve the performances of articulated object tracking. Particle filter provides an effective solution to the visual tracking problem since it can handle non-linear, non-Gaussian and multimodal distributions which often occur in such problems. The main advantage of the decomposition approaches is that their complexity, with respect to the number of parts of the target object to track, becomes linear instead of exponential, which reduces significantly the computation cost of the tracking algorithm. In this thesis, we proposed three algorithms relying on the decomposition principle.

Our first approach [31, 33], presented in Chapter 4, is directly inspired by the state-of-the-art algorithm for tracking in high dimensional state spaces, Partition Sampling (PS) [64], and by the independence property at the core of Bayesian networks, the so-called d-separation. Using a d-separation analysis, we show that PS can be accelerated by performing propagations/corrections for a group of parts in parallel, instead of part after part, while still estimating correctly the target distribution [39]. Furthermore, we introduced an operation called *swapping* to shift the particles toward the modes of

the target distribution by permuting some subsamples of the object parts processed in parallel so that the best subsamples (i.e. with highest weights) are combined to form new particles. This led us to develop a new tracking algorithm called Swapping-Based Partitioned Sampling (SBPS). SBPS first partitions the set of parts of the target object into subsets, that are then processed sequentially. These subsets are selected so that the parts of a subset are mutually independent conditionally to the parts that were previously processed. At each step, when a subset is processed, its parts are propagated, corrected and finally permuted using the swapping operation to produce better particles, located nearer the modes of the target density. In SBPS, the swapping operation is performed in a greedy manner, i.e., given a set of admissible permutations at any time, it aims to combine the best subsamples into one particle to produce the particle with highest weight. During the tests, SBPS gave most of the time lower estimation errors and computation times, compared to other filters such as APF or PS.

However, the major drawback of the swapping method in SBPS is that, by permuting subsamples w.r.t. only one possible permutation, it tends to produce small sets of different particles with very high weights, the other particles having much lower weights. As such, after the resampling step, only a few different high-weighted particles remain. This can cause problems in some situations, when the tracker needs to maintain multiple hypotheses to avoid tracking failure. To address this problem, we introduced the second tracking algorithm, presented in Chapter 5, which we called DBN-Based Combinatorial Resampling [32]. For this algorithm, the particle set produced by the swapping step is implicitly constructed by considering all the possible permutations. This swapping method has some advantages as compared to the one used in SBPS. It avoids the need for finding the best swapping. More importantly, it creates a large set of diverse particles. Resampling from it thus produces many particles near the modes of the target distribution while maintaining a certain amount of diversity among the particles. This new resampling scheme, introduced into the particle framework, and giving Particle Filter with Combinatorial Resampling (PFCR), performs better, in terms of estimation errors, without increasing the computation times, than other classical filters. Compared to SBPS, PFCR has shown its capacity to produce better sets of particles with lowest error's standard deviations. It has also demonstrated its ability to focus on the modes of the density to estimate.

During our tests, we noticed that there exist cases where the likelihood functions provide misleading results, e.g., when there are occlusions or clutter and, in these cases, swapping may fail to improve tracking. We thus introduced in Chapter 6 a PSO-based approach for articulated object tracking, called Hierarchical Annealed

Particle Swarm Optimization (HAPSOPF). In order to reduce the computational cost and increase the searching efficiency of PSO in high dimensional spaces, the proposed approach decomposes the search space into subspaces and then uses PSO to optimize these subspaces hierarchically. Moreover, a sampling covariance and some annealing terms are incorporated into the update equations of PSO at each optimization step to tackle the problem of noisy observations and cluttered background, thus making the proposed approach more robust in these situations.

7.2 Future Work

First, our different algorithms were essentially tested on synthetic video sequences. The main reason was that it allowed us to control the different parameters inducing the tracking complexity, such as the number of arms or their length, to analyze the robustness of the proposed approaches. However, it would of course be important to investigate more deeply the behavior and effectiveness of each of them in real-world problems, and thus in real video sequences. Many public datasets for articulated object tracking are now available, that would allow us to investigate the behavior of the proposed algorithms in different conditions in real-world problems, e.g. occlusion, cluttered background, lighting change, *etc.*, and to compare them with other existing approaches in such conditions. For example, the HumanEva dataset [87], which is used in Chapter 6, has been widely adopted for quantitative evaluation of human tracking approaches and could be considered for this purpose.

A nice property of our approaches described in Chapter 4 and Chapter 5 is that they are based on d-separation analysis in DBNs, which provides a rigorous justification for their correctness. As discussed previously, our proposed approaches maintain the target distribution they estimate. In other words, one is guaranteed that the distribution estimated by these approaches converges to the target distribution as the number of particles tends to infinity. However, their rates of convergence, which are important to perform a theoretical comparison of the performances among different PF-based approaches, have not been established in this thesis. Such a convergence result for PF has been found in literature [23], while the same convergence result for PS has never been stated. Therefore, one of the extensions of our work is to exhibit these convergence property for Partition Sampling as well as for SBPS and PFCR. Intuitively, PS consists of a sequential application of PF on subspaces of the original state space of the target object and, therefore, PS's convergence property should have a strong relation with that of PF. For the same reason, the convergence property of SBPS and PFCR should inherit from that of PF. Thus, the convergence result of PF

should be a good starting point to achieve our goal. Another convergence property of PF related to the variance of its estimations has been investigated in [21]. A study of this convergence property for our approaches is also desirable. Again, this convergence property of PF should provides a good basis for proving that of PS as well as those of SBPS and PFCR.

Another possible extension of our work is to incorporate the swapping operation into other existing decomposition approaches to improve their performance. In this case, the convergence of the obtained approaches depends on the conditional independence assumptions and also on the original approaches in which the swapping operation is combined with. For instance, a straightforward extension of our work is to combine the swapping operation into HAPSOPF to increase its tracking accuracy and computation time.

Finally, the works in this thesis in Chapter 4 and Chapter 5 exploit the property of the swapping operation and propose two swapping methods for better estimating the target distribution. The swapping method, proposed in Chapter 4, improves the tracking performances by generating the "best swap" at each time, while the one proposed in Chapter 5 improves the tracking performances by exploiting the diversity of the particle set generated from all possible permutations. Both methods have been shown to be effective in reducing the estimation errors and standard deviations of the tracking algorithm they were combined with. However, the questions of how to perform swapping in an optimal way and of what is the criterion for determining if a swapping method is optimal remain open research questions in this thesis and should be addressed in our future work. For example, it would be interesting to study the sampling properties, i.e. if the all the modes of the distribution are correctly sampled or not.

Bibliography

[1] Agarwal, A., Triggs, B., 2004. 3D human pose from silhouettes by relevance vector regression. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 882–888.

[2] Agarwal, A., Triggs, B., 2006. Recovering 3D human pose from monocular images. IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (1), 44–58.

[3] Alt, N., Hinterstoisser, S., Navab, N., 2010. Rapid selection of reliable templates for visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1355–1362.

[4] Andriluka, M., Roth, S., Schiele, B., 2009. Pictorial structures revisited: People detection and articulated pose estimation. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8.

[5] Arulampalam, M. S., Maskell, S., Gordon, N., Clapp, T., 2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on Signal Processing 50 (2), 174–188.

[6] Bajramovic, F., Deutsch, B., Grabl, C., Denzler, J., 2008. Efficient adaptive combination of histograms for real-time tracking. EURASIP Journal on Image and Video Processing 2008, 1–11.

[7] Bandouch, J., Engstler, F., Beetz, M., 2008. Evaluation of hierarchical sampling strategies in 3D human pose estimation. In: British Machine Vision Conference. pp. 925–934.

[8] Bergman, N., 1999. Recursive Bayesian estimation navigation and tracking applications. Ph.D. thesis, Linkopings University.

[9] Bernier, O., Cheung-Mon-Chan, P., Bouguet, A., 2009. Fast nonparametric belief propagation for real-time stereo articulated body tracking. Computer Vision and Image Understanding 113 (1), 29–47.

[10] Bhattacharyya, A., 1943. On a measure of divergence between two statistical populations defined by their probability distributions. Bulletin of the Calcutta Mathematical Society 35, 99–109.

[11] Bodor, R., Jackson, B., Papanikolopoulos, N., 2003. Vision-based human tracking and activity recognition. In: Mediterranean Conference on Control and Automation. pp. 18–20.

[12] Bray, M., Koller-Meier, E., Muller, P., Van Gool, L., Schraudolph, N. N., 2004. 3D hand tracking by rapid stochastic gradient descent using a skinning model. In: European Conference on Visual Media Production. pp. 59–68.

[13] Bray, M., Koller-Meier, E., Schraudolph, N. N., Gool, L. V., 2004. Stochastic meta-descent for tracking articulated structures. In: IEEE Conference on Computer Vision and Pattern Recognition Workshop. pp. 7–14.

[14] Brox, T., Rosenhahn, B., Cremers, D., Seidel, H.-P., 2006. High accuracy optical flow serves 3D pose tracking: exploiting contour and flow based constraints. In: European Conference on Computer Vision. pp. 98–111.

[15] Burges, C. J. C., 1998. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2 (2), 121–167.

[16] Blan, A. O., Sigal, L., Black, M. J., 2005. A quantitative evaluation of video-based 3d person tracking. In: International Workshop on Performance Evaluation of Tracking and Surveillance. pp. 349–356.

[17] Canton-Ferrer, C., Casas, J. R., Pardàs, M., 2011. Human motion capture using scalable body models. Computer Vision and Image Understanding 115 (10), 1363–1374.

[18] Chang, I.-C., Lin, S.-Y., 2010. 3D human motion tracking based on a progressive particle filter. Pattern Recognition 43 (10), 3621–3635.

[19] Chen, Z., 2003. Bayesian filtering: from Kalman filters to particle filters, and beyond. Tech. rep., McMaster University.

[20] Cheung, K.-M., Baker, S., Kanade, T., 2005. Shape-from-silhouette across time: Theory and algorithms. International Journal of Computer Vision 63, 225–245.

[21] Chopin, N., 2004. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. The Annals of Statistics 32 (6), 2385–2411.

[22] Comaniciu, D., Ramesh, V., Meer, P., 2000. Real-time tracking of non-rigid objects using mean shift. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 142–149.

[23] Crisan, D., Doucet, A., 2002. A survey of convergence results on particle filtering methods for practitioners. IEEE Transactions on Acoustics, Speech, and Signal Processing 50 (3), 736–746.

[24] Deutscher, J., Blake, A., Reid, I., 2002. Articulated body motion capture by annealed particle filtering. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 126–133.

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

[26] Doucet, A., 1998. On sequential simulation-based methods for Bayesian filtering. Tech. rep., Cambridge University Department of Engineering.

[27] Doucet, A., de Freitas, N., Gordon, N., 2001. Sequential Monte Carlo Methods in Practice. Information Science and Statistics. Springer.

[28] Doucet, A., de Freitas, N., Murphy, K. P., Russell, S. J., 2000. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In: Conference on Uncertainty in Artificial Intelligence. pp. 176–183.

[29] Doucet, A., Johansen, A. M., 2011. A tutorial on particle filtering and smoothing: fifteen years later. Tech. rep., Oxford University.

[30] Drew, M. S., Wei, J., Li, Z.-N., 1999. Illumination-invariant image retrieval and video segmentation. Pattern Recognition 32, 1369–1388.

[31] Dubuisson, S., Gonzales, C., N'Guyen, X. S., 2011. Swapping-based partitioned sampling for better complex density estimation: Application to articulated object tracking. In: International Conference on Scalable Uncertainty Management. pp. 525–538.

[32] Dubuisson, S., Gonzales, C., N'Guyen, X. S., 2012. DBN-based combinatorial resampling for articulated object tracking. In: Conference on Uncertainty in Artificial Intelligence. pp. 237–246.

[33] Dubuisson, S., Gonzales, C., Nguyen, X. S., To Appear. Sub-sample swapping for sequential Monte Carlo approximation of high-dimensional densities in the context of complex object tracking. International Journal of Approximate Reasoning.

[34] Efron, B., Tibshirani, R. J., 1994. An Introduction to the Bootstrap. Chapman and Hall/CRC.

[35] Efros, B., 2002. Adaptive color space switching for face tracking in multi-colored lighting environments. In: International Conference on Automatic Face and Gesture Recognition. pp. 249–253.

[36] Gall, J., 2005. Generalised annealed particle filter - mathematical framework, algorithms and applications. Master's thesis, University of Mannheim.

[37] Gall, J., Rosenhahn, B., Brox, T., Seidel, H.-P., 2010. Optimization and filtering for human motion capture. International Journal of Computer Vision 87 (1-2), 75–92.

[38] Glover, F., 1998. A template for scatter search and path relinking. In: European Conference on Artificial Evolution. pp. 3–54.

[39] Gonzales, C., Dubuisson, S., N'Guyen, X. S., 2011. Simultaneous partitioned sampling for articulated object tracking. In: International Conference on Advanced Concepts for Intelligent Vision Systems. pp. 150–161.

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

[41] Grochow, K., Martin, S. L., Hertzmann, A., Popović, Z., 2004. Style-based inverse kinematics. In: ACM Transactions on Graphics. pp. 522–531.

[42] Han, T. X., Ning, H., Huang, T. S., 2006. Efficient nonparametric belief propagation with application to articulated body tracking. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 214–221.

[43] Hauberg, S., Pedersen, K. S., 2010. Stick it! articulated tracking using spatial rigid object priors. In: Asian Conference on Computer Vision. pp. 758–769.

[44] Isard, M., 2003. PAMPAS: real-valued graphical models for computer vision. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 613–620.

[45] Isard, M., Blake, A., 1998. CONDENSATION - conditional density propagation for visual tracking. International Journal of Computer Vision 29, 5–28.

[46] Jiang, H., Martin, D. R., 2008. Global pose estimation using non-tree models. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8.

[47] John, V., Trucco, E., Ivekovic, S., 2010. Markerless human articulated tracking using hierarchical particle swarm optimization. Image and Vision Computing 28 (11), 1530–1547.

[48] Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: IEEE International Conference on Neural Networks. pp. 1942–1948.

[49] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., 1983. Optimization by simulated annealing. Science 220, 671–680.

[50] Kitagawa, G., 1996. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. Journal of Computational and Graphical Statistics 5 (1), 1–25.

[51] Kong, A., Liu, J. S., Wong, W. H., 1994. Sequential imputations and Bayesian missing data problems. Journal of the American Statistical Association 89 (425), 278–288.

[52] Krzeszowski, T., Kwolek, B., Wojciechowski, K. W., 2010. Articulated body motion tracking by combined particle swarm optimization and particle filtering. In: International Conference on Computer Vision and Graphics. pp. 147–154.

[53] Laguna, M., Marti, R., 2002. Scatter Search: Methodology and Implementations in C. Kluwer Academic Publishers, Norwell, MA, USA.

[54] Lawrence, N. D., 2004. Gaussian process latent variable models for visualisation of high dimensional data. In: Advances in Neural Information Processing Systems. Vol. 16. pp. 1–8.

[55] Lee, D. D., Seung, H. S., 1999. Learning the parts of objects by non-negative matrix factorization. Nature 401 (6755), 788–791.

[56] Leichter, I., Lindenbaum, M., Rivlin, E., 2009. Mean shift tracking with multiple reference color histograms. Image and Vision Computing 27 (5), 535–544.

[57] Liu, J., Chen, R., 1998. Sequential Monte Carlo methods for dynamic systems. Journal of the American Statistical Association 93, 1032–1044.

[58] Liu, J. S., Jun. 1996. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. Statistics and Computing 6 (2), 113–119.

[59] Liu, J. S., Chen, R., 1995. Blind deconvolution via sequential imputations. Journal of the American Statistical Association 90 (430).

[60] Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. International Journal on Computer Vision 60 (2), 91–110.

[61] Lu, W., Tan, Y.-P., 2001. A color histogram based people tracking system. In: IEEE International Symposium on Circuits And Systems. pp. 137–140.

[62] MacCormick, J., 2000. Probabilistic modelling and stochastic algorithms for visual localisation and tracking. Ph.D. thesis, Oxford University.

[63] MacCormick, J., Blake, A., 1999. A probabilistic exclusion principle for tracking multiple objects. In: IEEE International Conference on Computer Vision. pp. 572–587.

[64] MacCormick, J., Isard, M., 2000. Partitioned sampling, articulated objects, and interface-quality hand tracking. In: IEEE European Conference on Computer Vision. pp. 3–19.

[65] Mason, M., Duric, Z., 2001. Using histograms to detect and track objects in color video. In: Workshop on Applied Imagery Pattern Recognition. pp. 154–159.

[66] Meinhold, R. J., Singpurwalla, N. D., 1983. Understanding the Kalman filter. The American Statistician 37 (2), 123–127.

[67] Murphy, K., 2002. Dynamic Bayesian Networks: Representation, Inference and Learning. University of California, Berkeley.

[68] Oikonomidis, I., Kyriazis, N., Argyros, A. A., 2011. Efficient model-based 3D tracking of hand articulations using Kinect. In: British Machine Vision Conference. pp. 1–10.

[69] Okada, R., Soatto, S., 2008. Relevant feature selection for human pose estimation and localization in cluttered images. In: European Conference on Computer Vision. pp. 434–445.

[70] Pantrigo, J. J., Sanchez, A., Gianikellis, K., Montemayor, A., 2005. Combining particle filter and population-based metaheuristics for visual articulated motion tracking. Electronic Letters on Computer Vision and Image Analysis 5 (3), 68–83.

[71] Park, M., 2010. Efficient mean-shift belief propagation and its applications in computer vision. Ph.D. thesis, University Park, PA, USA.

[72] Pavlovic, V., Rehg, J. M., jen Cham, T., Murphy, K. P., 1999. A dynamic Bayesian network approach to figure tracking using learned dynamic models. In: International Conference on Computer Vision. pp. 94–101.

[73] Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufman Publishers.

[74] Pearl, J., 2000. Causality: Models, Reasoning, and Inference. Cambridge University Press.

[75] Pérez, P., Hue, C., Vermaak, J., Gangnet, M., 2002. Color-based probabilistic tracking. In: IEEE European Conference on Computer Vision. pp. 661–675.

[76] Pitt, M. K., Shephard, N., 1999. Filtering via simulation: Auxiliary particle filters. Journal of the American Statistical Association 94 (446), 590–599.

[77] Ramanan, D., Forsyth, D. A., 2003. Finding and tracking people from the bottom up. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. I467–474.

[78] Raskin, L., Rudzsky, M., Rivlin, E., 2007. Tracking and classifying of human motions with gaussian process annealed particle filter. In: Asian Conference on Computer Vision. pp. 442–451.

[79] Ren, X., Berg, A. C., Malik, J., 2005. Recovering human body configurations using pairwise constraints between parts. In: International Conference on Computer Vision. pp. 824–831.

[80] Rose, C., Saboune, J., Charpillet, F., 2008. Reducing particle filtering complexity for 3D motion capture using dynamic Bayesian networks. International Conference on Artificial Intelligence, 1396–1401.

[81] Rosipal, R., Kramer, N., 2006. Overview and recent advances in partial least squares. In: Subspace, Latent Structure and Feature Selection Techniques. pp. 34–51.

[82] Saboune, J., Charpillet, F., 2005. Using interval particle filtering for marker less 3D human motion capture. In: International Conference on Tools with Artificial Intelligence. pp. 621–627.

[83] Shen, C., van den Hengel, A., Dick, A., Brooks, M., 2004. 2D articulated tracking with dynamic Bayesian networks. pp. 130–136.

[84] Shi, Y., Eberhart, R., 1998. A modified particle swarm optimizer. In: IEEE International Conference on Evolutionary Computation Proceedings. pp. 69–73.

[85] Shin, J., Kim, S., Kang, S., Lee, S.-W., Paik, J., Abidi, B., Abidi, M., 2005. Optical flow-based real-time object tracking using non-prior training active feature model. Real-Time Imaging 11 (3), 204–218.

[86] Sigal, L., 2008. Continuous-state graphical models for object localization, pose estimation and tracking. Ph.D. thesis, Providence, RI, USA.

[87] Sigal, L., Balan, R. O., 2009. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. International Journal on Computer Vision 87 (1-2), 4–27.

[88] Sigal, L., Bhatia, S., Roth, S., Black, M. J., Isard, M., 2004. Tracking loose-limbed people. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 421–428.

[89] Song, K., Kittler, J., Petrou, M., 1996. Defect detection in random colour textures. Image and Vision Computing 14 (9), 667 – 683.

[90] Sudderth, E., Ihler, A., Freeman, W., Willsky, A., 2002. Nonparametric belief propagation. Communications of the ACM 53 (10), 95–103.

[91] Sudderth, E. B., Mandel, M. I., Freeman, W. T., Willsky, A. S., 2004. Visual hand tracking using nonparametric belief propagation. In: IEEE Conference on Computer Vision and Pattern Recognition Workshop. pp. 189–197.

[92] Sudderth, E. B., Mandel, M. I., Freeman, W. T., Willsky, A. S., 2005. Distributed occlusion reasoning for tracking with nonparametric belief propagation. In: Advances in Neural Information Processing Systems. pp. 1369–1376.

[93] Sun, M., Telaprolu, M., Lee, H., Savarese, S., 2012. An efficient branch-and-bound algorithm for optimal human pose estimation. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8.

[94] Taskar, B., Chatalbashev, V., Koller, D., Guestrin, C., 2005. Learning structured prediction models: a large margin approach. In: International Conference on Machine Learning. pp. 896–903.

[95] Tian, T.-P., Li, R., Sclaroff, S., 2005. Articulated pose estimation in a learned smooth space of feasible solutions. In: Worshop on Learning in Computer Vision and Pattern Recognition. pp. 50–58.

[96] Tian, T.-P., Li, R., Sclaroff, S., 2005. Tracking human body pose on a learned smooth space. In: IEEE Workshop on Learning in Computer Vision and Pattern Recognition. pp. 1–8.

[97] Tian, T.-P., Sclaroff, S., 2010. Fast globally optimal 2D human detection with loopy graph models. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 81–88.

[98] Tran, D., Forsyth, D., 2010. Improved human parsing with a full relational model. In: European Conference on Computer Vision. pp. 227–240.

[99] Umbaugh, S. E., 1997. Computer Vision and Image Processing: A Practical Approach Using Cviptools with Cdrom. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[100] Unal, G. B., Yezzi, A. J., Krim, H., 2005. Efficient incorporation of optical flow into visual motion estimation in tracking. In: Machine Learning and Robot Perception. Vol. 7. pp. 167–202.

[101] Urtasun, R., Fleet, D. J., Fua, P., 2006. 3D people tracking with Gaussian process dynamical models. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 238–245.

[102] Urtasun, R., Fleet, D. J., Hertzmann, A., Fua, P., 2005. Priors for people tracking from small training sets. In: IEEE International Conference on Computer Vision. pp. 403–410.

[103] Urtasun, R., Fua, P., 2004. 3D tracking for gait characterization and recognition. In: International Conference on Automatic Face and Gesture Recognition. pp. 17–22.

[104] Wang, J., Yagi, Y., 2008. Integrating color and shape-texture features for adaptive real-time object tracking. IEEE Transactions on Image Processing 17 (2), 235–240.

[105] Wang, J. M., Fleet, D. J., Hertzmann, A., 2008. Gaussian process dynamical models for human motion. IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2), 283–298.

[106] Wang, L., Tan, T., Ning, H., Hu, W., 2003. Silhouette analysis-based gait recognition for human identification. IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (12), 1505–1518.

[107] Wu, Y., Hua, G., Yu, T., 2003. Tracking articulated body by dynamic Markov network. In: International Conference on Computer Vision. pp. 1094–1101.

[108] Xu, X., Li, B., 2007. Learning motion correlation for tracking articulated human body with a Rao-Blackwellised particle filter. In: International Conference on Computer Vision. pp. 1–8.

[109] Yao, A., Gall, J., Gool, L. V., Urtasun, R., 2011. Learning probabilistic non-linear latent variable models for tracking complex activities. In: Advances in Neural Information Processing Systems. pp. 1359–1367.

[110] Yedidia, J. S., Freeman, W. T., Weiss, Y., 2003. Understanding belief propagation and its generalizations. Morgan Kaufmann Publishers Inc., pp. 239–269.

[111] Yilmaz, A., Li, X., Shah, M., 2004. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (11), 1531–1536.

[112] Zhang, X., Hu, W., Wang, X., Kong, Y., Xie, N., Wang, H., Ling, H., Maybank, S., 2010. A swarm intelligence based searching strategy for articulated 3D human body tracking. In: International Conference on Computer Vision and Pattern Recognition Workshops. pp. 45–50.