Subspace Hierarchical Particle Filter

Bruno Cedraz Brandão, Jacques Wainer, Siome Klein Goldenstein
Universidade Estadual de Campinas – Instituto de Computação
Av. Albert Einstein, 1251, Campinas, SP, Brazil
{bruno, wainer, siome}@ic.unicamp.br

Abstract

Particle filtering has become a standard tool for non-parametric estimation in computer vision tracking applications. It is an instance of stochastic search. Each particle represents a possible state of the system. Higher concentration of particles at any given region of the search space implies higher probabilities. One of its major drawbacks is the exponential growth in the number of particles for increasing dimensions in the search space. We present a graph based filtering framework for hierarchical model tracking that is capable of substantially alleviate this issue. The method relies on dividing the search space in subspaces that can be estimated separately. Low correlated subspaces may be estimated with parallel, or serial, filters and have their probability distributions combined by a special aggregator filter. We describe a new algorithm to extract parameter groups, which define the subspaces, from the system model. We validate our method with different graph structures withing a simple hand tracking experiment with both synthetic and real data.

1 Introduction

Non-parametric estimation has many advantages over its parametric counterpart, ubiquitously represented by the Kalman filter [13]. In many stochastic applications of computer vision, probability distributions assume arbitrary shapes over time, requiring solutions that are more powerful than the Gaussian/Linear estimation.

Particle filter is a tool for non-parametric estimation that is used extensively by the computer vision [15, 2] and machine learning [4, 10] communities. It can be seen as a stochastic search algorithm, where a set of particles, each one representing a possible system state, forms a distribution of probability that represents our knowledge of the system state. Roughly speaking, higher concentrations of particles at any given region implies higher probabilities that such region represents the real state of the system. Unfortunately, estimation under high dimensional search spaces usually requires exponentially more particles to assure convergence. This is known as the *curse of dimensionality* [11].

We propose a method to extract subspaces from the search space according to some implicit structure of the filter's observation function. For tracking applications involving hierarchical models, this function absorbs the hierarchical structure of the tracked object. Some of its parameters have a broader influence over the final configuration of the projected object at the observed image. It is the case of the wrist position and rotation parameters of a hand tracking application, for example. This information will be used to decide which subspaces can be tracked separately and which should be estimated in bundles. Then, a directed acyclic graph structure of specialized filters may be built. Specialized filters are responsible for tracking a given subspace. This graph-based divide and conquer approach has the potential to decrease exponentially the number of particles.

In the next section, we review briefly related work. In section 3, we present a overview of the standard particle filter. Section 4 contains the description of the Subspace Hierarchical Particle Filter. In section 5, we present the algorithm to extract the parameter subspaces. A case study is described in section 6, and, in section 7, experimental results, based on real and synthetic data, are presented and discussed. Section 8 summarizes our contributions.

2 Related Work

Much work has been done to extend the particle filter framework to multi-target tracking, many of which may be applied to hierarchical models to alleviate the dimensionality problem. In this section, we cover a small selection.

In [11, 12], MacCormick and Isard introduced the partitioned sampling method to articulated object tracking which made the dimensionality problem tractable. Their partition of the sample space assumes that the system dynamics $f()$

is decomposable as

$$f(\mathbf{x}_k|\mathbf{x}_{k-2}) = \int f_B(\mathbf{x}_k|\mathbf{x}_{k-1}) f_A(\mathbf{x}_{k-1}|\mathbf{x}_{k-2}) d\mathbf{x}_{k-1}. \quad (1)$$

Where f_A and f_B represent the dynamics of two different objects. Their method is similar to ours for serial topologies, but it considers characteristics of the dynamics function instead of the observation function. Their choice is more appropriate for multi-target tracking. They have also proposed a branched partitioned sampling that, unlikely ours, searches over every subspace in each branch, in different order, to improve tracking accuracy. They have illustrated their framework with the bidimensional contour tracking of the hand.

Deutscher and Reid [3] have presented the annealed particle filter which, through a series of gradually narrowing weighting functions, tries to efficiently assign particles to most promising regions of the sample space. This method is similar to well known methods [9]. They have introduced the crossover operation into the particle filter framework and have shown how it improves the convergence for articulated object tracking. The operation can combine the best parts from particles in different high probable regions, introducing some new states, not available before, that explore potentially new high probable regions. We have used a similar crossover operation to combine the probability distributions from separate filter branches. They have also illustrated their method with a 35-degree of freedom, multiple camera, body tracking application in uncluttered background.

To conclude this section, we point to a different approach to gradually assign higher probabilities to promising regions that is due to Stenger, the hierarchical Bayesian filter [16]. It performs a tree-search over an integral grid representation that is hierarchically partitioned as the search descends into each region. This technique requires multiple level discretization of the search space, and therefore, may not be suited for high dimensional tracking applications. It has been used for hand tracking over cluttered backgrounds, but no more than six degrees of freedom where estimated at any time.

3 Particle Filter Overview

Particle filtering is one of many Bayesian non-parametric estimation techniques. It is known by several names such as Condensation [8], Bootstrap filter [7], and Sequential Monte Carlo [5]. In this section, we describe briefly the algorithm's main elements. As in [7], we consider the simpler proposal distribution

$$q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{Y}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}). \quad (2)$$

In the Bayesian framework, we represent the knowledge about the system's state through the use of probability distributions. In the case of particle filters, we represent arbitrary distributions with sets of particles. Conversely, each particle may be thought as a sample from the represented distribution. Each particle may, or may not, have an associated weight. A smooth probability distribution might be extracted from the set of particles using a kernel density estimation algorithm [6].

A set of N particles is used to represent our distribution. The particle filter framework also requires a function that models the dynamics of the system, an observation function, a likelihood function, and a resampling procedure. If our system is n-dimensional, our particle will be a sample of $\mathbb{R}^n$. The dynamics function $f : \mathbb{R}^n \to \mathbb{R}^n$ should be able to make reasonable predictions on the near future state of the system given the current state, in other words

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \quad (3)$$

where $\mathbf{u}_k$ models the uncertainty of our prediction. This equation also implies a Markovian assumption.

The observation distribution, in $\mathbb{R}^m$, simulates the observation using distribution samples. We need this procedure to compare an observation with a particle within the likelihood function. It must model how the measurement noise is expected to affect the observation. This is important in order to implement the likelihood function correctly. The observation function $h : \mathbb{R}^n \to \mathbb{R}^m$ takes one particle and builds the corresponding observation

$$\mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{v}_k, \quad (4)$$

where $\mathbf{v}_k$ is the observation noise. The likelihood function assigns a value

$$w_i = p(\mathbf{y}_k|\mathbf{x}_k^i) / \sum_{j=1}^{N} p(\mathbf{y}_k|\mathbf{x}_k^j), \quad (5)$$

that measures how probably a given observation might have been generated if the system was in the state described by the particle $\mathbf{x}_k^i$.

This functions and representations are organized in three steps to perform the Bayesian estimation.

1. **Predict:** Each particle of the initial distribution goes through the dynamics function and a prediction distribution results.

2. **Update:** A weight, w_i, is assigned to each predicted particle according to the likelihood function. A new, weighted, particle distribution is formed.

3. **Reconfigure:** A final, unweighted, distribution is generated by resampling N particles from the weighted

distribution, with resampling probability corresponding to the weights. Particle with relatively high w_i have higher chances of being copied to the final distribution.

The first two steps are responsible for the Bayesian estimation. The last step is an optimization for particle filters. In the next section, we describe our proposed filter architecture.

4 Subspace Hierarchical Particle Filter

The Subspace Hierarchical Particle Filter consists of a group of particle filters arranged into an acyclic graph structure. Each individual particle filter is assigned to track a different subspace. With this construction, it is possible to reduce the impact of the exponential growth in the number of particles. In this section, we will explain how an already built filter performs tracking, the typical topologies and our solution to the issue of combining different distributions.

4.1 General Aspects and Topologies

The Subspace Hierarchical Particle Filter is a directed acyclic graph where the nodes are particle filters specialized on a subspace. All specialized filters have in common the observation and likelihood functions. Every filter has its own number of particles and search range. The search range is defined by a vector that stores the maximum displacement that any parameter of the particle may acquire. In other words, this limits the movement of the particles in the search space. A filter that has any parameter search range assigned to zero do not perform search in that direction. This means that the specialized filter does not change, or inject new values on, those parameters. Ideally, a specialized filter should be assigned as few as possible parameters in order to minimize the overall number of particles.

A special filter is needed when we combine distributions from different particle filters. We will call them aggregator filters. An aggregator filter usually performs a broader search but withing a very limited range. Their primary function is to reject incompatible particles that arise in the combination of different distributions. We will explore this latter. It may also be used to perform a last fitting step over the particle distribution.

There are two ways to organize the filters in terms of estimation. We may think of our acyclic graph structure as a combination series and parallel nodes. As stated in the section 2, our serial estimation is similar to what is described in previous works. This basic topology is illustrated in Figure 1.

Each filter, $\mathbf{A}$ to $\mathbf{E}$, has its own parameter subspace, $\mathbf{x}^A$ to $\mathbf{x}^E$, respectly. There is an aggregator, $\mathbf{F}$, for a last general fit. During a typical execution of the filter, $\mathbf{A}$ receives the

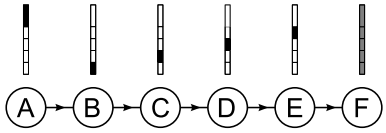

Figure 1. Serial topology and range vectors.

prior distribution, perfoms its estimation, and forwards the resulting distribution to filter $\mathbf{B}$. The posterior distribution is the resulting ditribution of filter $\mathbf{F}$.

Particle transfer between filters is performed simply by resampling the needed number of particles from the previous filter's weighted particle set.

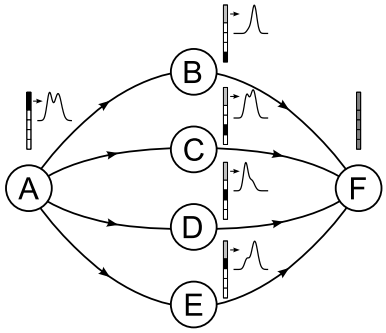

Figure 2. Parallel topology example.

Branches may arise at the observation function of hierarchical models, h_k, when some elements of the observation vector, $d = h(\mathbf{x})$, are not equally influenced by the same set of parameters, $\mathbf{x}$, that is,

$$\{\exists \mathbf{x}_\xi \neq \mathbf{x}_\psi, i \neq j | h_i(\mathbf{x}_\xi) = h_i(\mathbf{x}_\psi) \wedge h_j(\mathbf{x}_\xi) \neq h_j(\mathbf{x}_\psi)\} \tag{6}$$

Each branch of the observation function will result in the value of an element of the observation vector.

As we stated, there is the possibility to arrange the filters in parallel according to the observation function. This situation is illustrated in Figure 2. The execution of this filter is similar to that of the serial. Particle transfer from filter $\mathbf{A}$ to filters $\mathbf{B}$ to $\mathbf{E}$ is actually the same. How to combine the resulting distributions into filter $\mathbf{F}$ is a problem we will solve through the use of the crossover operation.

Let's assume a group of converging nodes, each one specialized into a disjoint set of parameters. We will employ a multi-parent crossover operation to build the new posterior distribution. As in Figure 2, we could name the subsets of parameters of our particle as $\mathbf{x} = (\mathbf{x}^A, \mathbf{x}^B, \mathbf{x}^C, \mathbf{x}^D, \mathbf{x}^E)$. The multi-parent crossover will choose some subregions of each filter, from $\mathbf{B}$ to $\mathbf{E}$ in order to build the particles that will form the initial distribution of $\mathbf{F}$. The operation should

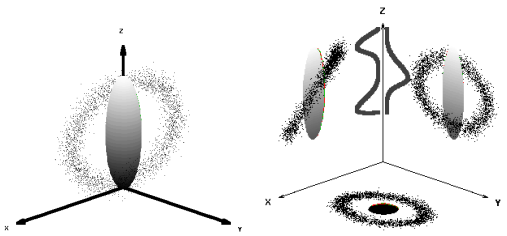

Figure 3. Incompatible hypothesis generated after separate estimation of high correlated parameters.

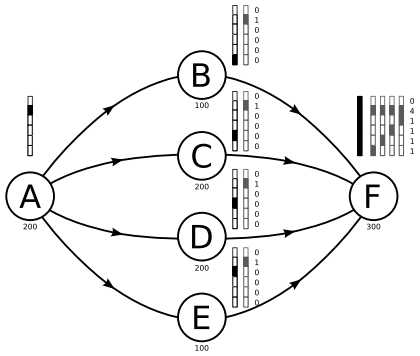

Figure 4. Range masks being forwarded.

choose from each filter, preferably, only updated regions. At subsection 4.3, we will show how to determine those regions automatically.

The crossover operation might generate an inconsistent set of particles. Therefore, we need the aggregator filter $\mathbf{F}$. It will naturally reject inconsistent, low probable, particles and perform a final fit.

When we execute the SHPF, filters are not queued for execution until they have their initial distributions completely filled by the previous filters.

4.2 Discussion

We may not simply unite every parallel distribution, i.e. by randomly sampling a particle from each converging filter, because the resulting distribution will be biased towards the prior. This happens since we usually assign the estimation of a given parameter to a single filter. therefore, one of the converging nodes will have an updated value while many others will have the prior's value.

When we approximate distributions through particles, tracking a subspace may change the distribution of orthogonal subspaces. Particles from any given specialized filter may only be moved in the directions that correspond to non-zero elements of the range vector. At the reconfigure step, particles with high probabilities will have priority to form the new distribution. This selection might unevenly exclude particles that are needed to represent orthogonal distributions. As a consequence, inherited subspaces, i.e. those that do not belong to any parallel filter, might have their distributions differently changed by the parallel filters. At the extreme case, each parallel filter will introduce an incompatible hypothesis, making it very difficult to combine the distributions.

This behavior is illustrated in figure 3. The left Figure shows a very correlated torus-like particle distribution next to an ellipsoid distribution from the observation. The parameters corresponding to the Z-axis were estimated previously. We then estimate X and Y separately. The figure to the right shows the projection from the particles and observation distribution at each plane. X estimation may only move the particles parallel to the X-axis. Similarly, estimation over Y may only perform movements parallel to the Y-axis. If we consider the XY plane projection and the fact that every filter shares the same likelihood function, we may agree with the resulting distributions plotted next to the Z-axis. The one to the left represents the distribution of the Z parameter after the estimation of the parameter X. Similarly, the one to the right, represents Z after Y estimation. Clearly those distributions are incompatible, since there is no overlap between the two.

This extreme case tells us that parameters that are very correlated should be estimated together. Slightly correlated parameters, as shown in Figure 2, might be estimated in parallel and then combined through the crossover operation.

4.3 Selecting Particle Subregions

In this subsection, we present an automatic algorithm to select particle subregions that will be used by the crossover operation. We will show how to use the range vectors, particle numbers and topology to preprocess a data structure containing that information.

The procedure consists of two stages. At the first one, ilustrated at Figure 4, we forward the **range masks** and generate the **occupation vectors**. The range mask is a binary mask that assumes one for non-zero regions of the range vector and zero for zero regions. The occupation vector records the quantities of each region it has received by each direct previous filter. When a filter receives a mask from its predecessor, it updates the occupation vector by incrementing the counter for every marked region of the mask, then it merges the received mask with its own. After receiving every predecessor's mask, the filter is able to forward its new, merged mask that represents every region it estimates and every region estimated by its predecessors.

At the second stage we will use the range mask, occupation vector, $Occ_{(\rho,j)}$, the number of predecessors, $\Pi_{(\rho)}$, and the number of particles, $n_{(\rho)}$, from each filter, ρ, to evaluate the proportion from each particle region each filter should forward for the crossover operation. There are two situations a filter should forward a region. If a filter tracks or inherits a region, $\mathbf{x}^j$ (region will be marked at the range mask), it will forward a quantity of $n_{(\rho)}/Occ_{(\rho,j)}$ of those regions for the adjacent filter ρ. If the occupation vector value for a given region is zero at the adjacent filter, implying that no predecessor filter estimates that region, every filter should forward an equal number of those regions. The number is $n_{(\rho)}/\Pi_{(\rho)}$. For exemple, filter $\mathbf{B}$ at the Figure 4, forwards the following number of each region: $\{n_{(\mathbf{F})}/\Pi_{(\mathbf{F},1)} = 75,\ n_{(\mathbf{F})}/Occ_{(\mathbf{F},2)} = 75,\ 0,\ 0,\ 0,\ n_{(\mathbf{F})}/Occ_{(\mathbf{F},6)} = 300\}$.

4.4 Finding Subspaces

In this section, we will show a procedure to partition the parameters in groups that will form the subspaces of the specialized filters, and, at the same time, suggest a topology. One possibility, commonly found in the neural network and control communities [1], is to evaluate the Jacobian Matrix

$$\mathrm{J}(\mathbf{X}) = \frac{\partial h_i(\mathbf{X})}{\partial \mathbf{x}^j} \tag{7}$$

of the observation function h_k. Those elements, $\mathbf{j}_{ij}$, of the matrix that are always zero represent parameters x_j, of the particle, that do not affect certain observation h_i. We want to groups the parameters that act on same set of observations to get a rough idea of which groups might be estimated in parallel. Naturally, when the model structure is clear, one might readily pick those groups, but in situations where the model is obtained by training, or the model is overly complex, an automatic algorithm is needed.

We will use the Jacobian matrix to generate a new matrix which elements are assigned to one, if they are non-zero for some value of $\mathbf{X}$, or zero otherwise. We will call this matrix the **Observability Matrix**. When we manually write h_k, the observability matrix may be extracted by simple inspection.

In order to extract the groups, we assign to each column of the observability matrix two values: the number of ones, $\mathbf{o}_j$, and the number resulting from the binary pattern of the column, $\mathbf{v}_j$. When we sort the rows by $\mathbf{o}_j$, and by $\mathbf{v}_j$ as secondary key, the resulting matrix will have a structure similar to

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

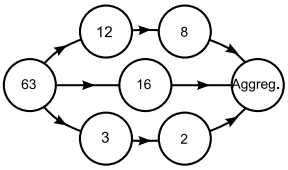

Figure 5. Suggested partition and topology.

in which columns with equal $\mathbf{v}_j$ belong to the same group. In this example, we have from left to right, the groups $\{63\}$, $\{12\}$, $\{3\}$, $\{16\}$, $\{8\}$, $\{2\}$, labeled after $\mathbf{v}_j$.

To decide which groups may be estimated in parallel, we perform a simple bitwise-and operation between the $\mathbf{v}_j$'s of each pair of groups. Those pairs that have no overlap might be estimated in parallel in our framework. The order of the serial estimation may be assumed to be of decreasing $\mathbf{o}_j$. Figure 5 illustrates the relations extracted from the above matrix. Note that this is a suggestion only, as groups with no overlapping might still be high correlated through their predecessors. It is up to the designer to choose one of the possible topologies. Along with the topology, the designer must choose the number of particles and define the actual limits of the dynamics model at each node. We think that this aspects might be automated by a learning approach [14] and we plan to investigate this possibility in future works.

4.5 Algorithm

In this section we provide the detailed Subspace Hierarchical Particle Filter algorithm. Before execution of the filter, we must run the *Preprocessing* function in order to generate a structure, *PT* (Particle Transfer), with the quantities of each regions each filter should forward, as described in section 4.3. $PT_{(\rho,\ell,k)}$ is the number of regions $\mathbf{x}^k$ the filter ρ should forward to filter ℓ. We will build this structure using the range vector *Range*, and the graph topology, given by the adjacent list, *Adj*, and the number of predecessors of each filter, Π. The preprocessing function needs some auxiliary structures: the range mask, *Mask*; the occupation vector, *Occ*; and the masks sent from every filter to every other filter, *MfP*.

Each filter ρ has its particle vector $P_{(\rho)}$ and weight vector $W_{(\rho)}$, so that

$$W_{(\rho,i)} = \sum_{j=1}^{i} w_j \quad \text{and} \quad W_{(\rho,n_{(\rho)})} = 1. \tag{8}$$

Observations are acquired in the form of the structure O. D is an auxiliary matrix that marks which regions from which particles have been filled, that is, $\beta = D_{(\rho,j)}$ particles of the filter ρ have their element $\mathbf{x}^j$ filled.

The parameter *limiting_filter* might be used to limit the tracking up to that filter. This is useful if we want our filter to work on a different mode. Normally *limiting_filter* $= m$, where m is the last filter. Particle vector, $P_{(1)}$, should have a valid distribution at the beginning. The posterior will be available at the same vector, $P_{(1)}$. At the algorithms below, we assume that logical terms are evaluated to 1 and 0, for *true* and *false*, respectly.

Partial_Particle_Filter (P, W, O, ρ)
1. *Transition_Function* $(P_{(\rho)})$
2. *Likelihood* $(P_{(\rho)}, O, W_{(\rho)})$

Guided_Reconfigure $(P, W, PT, D, \rho, \ell)$
1. **For each** *i* **from** 1 **to** *Max* $(PT_{(\rho,\ell)})$**:**
2. $r \Leftarrow$ *Uniform* $(0, 1)$
3. $k \Leftarrow$ *Binary_Search* $(W_{(\rho)}, r)$
4. **For each** *j* **from** 1 **to** *paticle_size***:**
5. **If** $\{PT_{(\rho,\ell,j)} > 0\}$**:**
6. $P_{(\ell,D_{(\ell,i)},j)} \Leftarrow P_{(\rho,k,j)}$
7. $PT_{(\rho,\ell,j)} \Leftarrow PT_{(\rho,\ell,j)} - 1$
8. $D_{(\ell,j)} \Leftarrow D_{(\ell,j)} + 1$

SHPF $(P, O, Adj, PT, limiting\_filter)$
1. $D \Leftarrow 0$
2. $W \Leftarrow 0$
3. $Q \leftarrow \{1\}$
4. **While** $\{Q \neq \emptyset\}$**:**
5. $\rho \leftarrow Q$
6. *Partial_Particle_Filter* (P, W, O, ρ)
7. **For each** ℓ **in** $Adj_{(\rho)}$**:**
8. *Guided_Reconfigure* $(P, W, PT, D, \rho, \ell)$
9. **If** $\{\nexists\, \alpha \in D_{(\ell)} | \alpha \neq n_{(\ell)}\}$**:**
10. $Q \leftarrow \{\ell\}$
11. *Guided_Reconfigure* $(P, W, FP, D, limiting\_filter, 1)$

Preprocessing $(PT, Adj, nAdj, \Pi, Range)$
1. **For each** i **from** 1 **to** m**:**
2. $\pi_{(i)} \Leftarrow \Pi_{(i)}$
3. **For each** k **from** 1 **to** *particle_size***:**
4. $Mask_{(i,k)} \Leftarrow \{Range_{(i,k)} > 0\}$
5. $Occ_{(i,k)} \Leftarrow 0$
6. $Q \leftarrow \{1\}$
7. **While** $\{Q \neq \emptyset\}$**:**
8. $\rho \leftarrow Q$
9. **For each** ℓ **from** 1 **to** $\Pi_{(\rho)}$**:**
10. **For each** k **from** 1 **to** *particle_size***:**
11. $Mask_{(\rho,k)} \Leftarrow \{Mask_{(\rho,k)} \vee MfP_{(\rho,\ell,k)}\}$
12. $Occ_{(\rho,k)} \Leftarrow Occ_{(\rho,k)} + MfP_{(\rho,\ell,k)}$

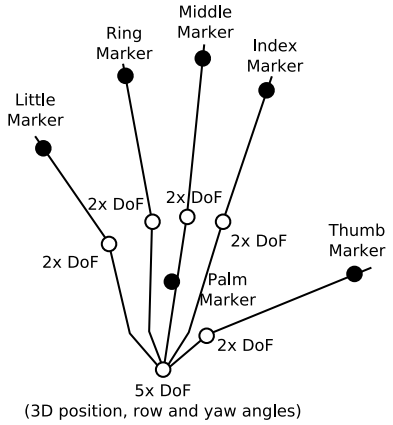

Figure 6. A simple hand model.

13. **For each** ℓ **in** $Adj_{(\rho)}$**:**
14. $MfP_{(\ell)} \Leftarrow Mask_{(\rho)}$
15. $\pi_{(\ell)} \Leftarrow \pi_{(\ell)} - 1$
16. **If** $\{\pi_{(\ell)} = 0\}$**:**
17. $Q \leftarrow \{\ell\}$
18. **For each** i **from** 1 **to** m**:**
19. **For each** j **from** 1 **to** $nAdj_{(i)}$**:**
20. **For each** k **from** 1 **to** *particle_size***:**
21. **If** $\{Mask_{(i,k)} > 0\}$**:**
22. $PT_{(i,j,k)} \Leftarrow n_{(j)}/Occ_{(Adj_{(i,j)},k)}$
23. **Else, if** $\{Occ_{(Adj_{(i,j)},k)} = 0\}$**:**
24. $PT_{(i,j,k)} \Leftarrow n_{(j)}/\Pi_{(j)}$
25. **Else:**
26. $PT_{(i,j,k)} \Leftarrow 0$
27. **For each** k **from** 1 **to** *particle_size***:**
28. $PT_{(m,1,k)} \Leftarrow n_{(1)}$

5 Case Study

We illustrate, and validate, our method with a simple hand tracking experiment. We designed a hand model with 15 degrees of freedom, five for wrist 3D position, row and yaw angles, and two for each rigid finger row and yaw angles. The observations are the positions of six unidentified markers, one at the center of the palm and one at each finger tip. Figure 6 illustrates our model. Our setup is clearly ill-conditioned since we should be able to track at most twelve parameters reliably. To cope partially with that, some parameters are given very limited range of variation, just enough to deal with initial fitting and small changes during the estimation. It is possible for the estimated parameters to assume considerably different poses from the ground truth for the same marker positions, but, we can still perform interesting experiments with such a simple setup.

All real data acquisition was performed with a main-

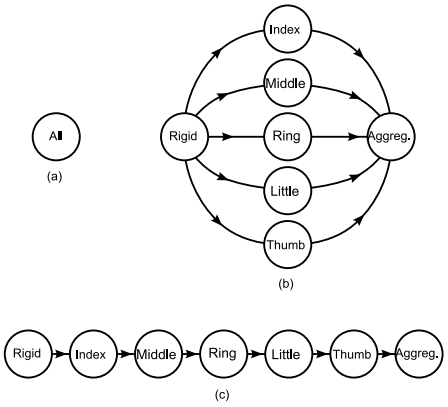

Figure 7. Standard particle filter (a). Serial (b) and parallel (c) SHPF topologies.

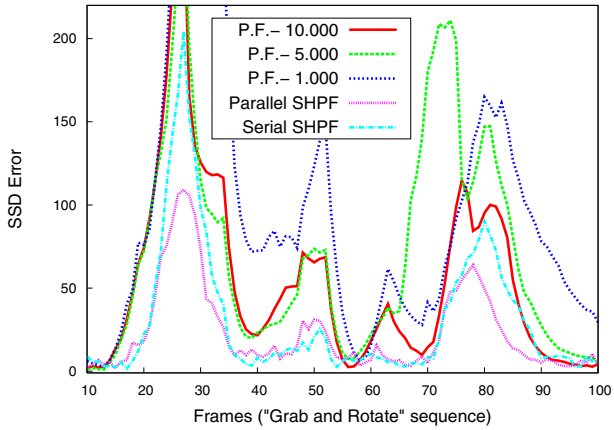

Figure 8. SSD graph of five filter configurations performing over a synthesized sequence.

stream webcam. We marked a medical glove with six unidentified markers painted in red from which we only extract the center position. Drugstore glove, painted with gouache, is still one of the fastest, and cheapest, ways of setting up a hand tracking application. The synthetic data was manually generated, and consists of sequences of hand model states from which we may extract synthetic observations. We employ a greedy proximity approach to match observed markers with those estimated through the model. This setup is not robust to occlusion. Extending the filter to track first derivatives of parameters might make these situations less ambiguous.

6 Experiments and Validation

For validation purposes, we created synthesized sequences involving 3D hand rotation, grabbing, translation and individual finger movement. We tested tree different topologies (Figure 7) and different parameter variances. The standard particle filter is tested with three different numbers of particles: 1.000, 5.000 and 10.000 particles. We have a serial and a parallel topology, that closely resembles the observation function, with 1.000 particles each. Since our setup has some "loose" degrees of freedom, that would induce unpredictable deviations, and make it impractical to compare the hidden states of any two given filters. Therefore we evaluated how close the filter matches the synthesized markers. All the sequences are of relatively fast movement, with a "grab" movement being performed in one second, for example.

Figures 8 and 9 display the sum of squared differences between each filter estimation and the synthesized data per frame for the period of a "grab and rotate" operation. As expected, the error for the SHPF was, most of the time, smaller than that for standard particle filters with similar particle number. Actually, in some situations, it performed better than a filter with ten times more particles. The parallel filter seemed to be more stable than the serial for the same number of particles. This is reasonable as the crossover operation forces a broader distribution of particles over the search space, while sequential filtering narrows the distribution at each step. Both require much less particles than the standard filter, but the parallel topology adapts faster to unpredictable changes in observation. Since we use the median of the distribution to estimate the filter state at any given time, our filter output is somewhat shakier than that of filter with much more particles, but is proportionally faster and more accurate.

We experimented with real data sequences where observation is sometimes missed or incorrectly assigned to a non-target region of the image. Fortunately, the filter maintains enough information in its distribution to be able to recover.

When we start tracking, the SHPF is set to run only at the first specialized filter. This way, it can match a predetermined pose by using rigid motion only. If the filter is allowed to perform complete estimation when fully immersed in regions of low probability, it may assume unpredictable states and never recover. Only when the likelihood function starts to evaluate probabilities above a certain threshold, the filter is allowed to perform full estimation. Figure 10 shows four frames from a 15 seconds sequence. The two images to the left show the initial fitting process. The remaining two are frames from a grab movement. With this setup, and estimating 15 parameters, the SHPF was capable of performing 30fps tracking on a Pentium 4 3.2Ghz system.

Figure 10. Frames from a real grab sequence. The pictures to the left are from the initial fitting step.

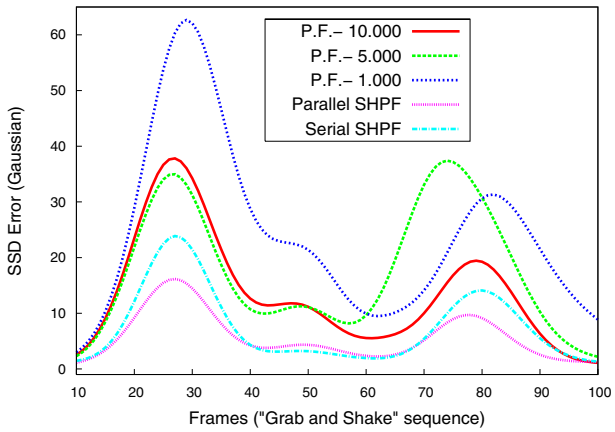

Figure 9. Previous graph, Gaussian filtered.

7 Conclusions and Future Work

We have proposed a generic DAG framework for particle filter design that employs the model's observation function to partition the parameter space into subspaces that may be tracked separately. We have argued that some groups may be tracked in parallel and combined by a multi-parent crossover operation. An easy way to partition the parameter space through the observation function Jacobian was shown. We have validated our method through a simple ill-conditioned hand tracking experiment with real and synthetic data.

In future work we intend to complete the mathematical model of the SHPF. We plan to employ a learning approach to the filter construction to automatically determine the topology, number of particles, and limits of the dynamics function of each node. We are also planning to apply dynamic reconfiguration techniques to the system model in order to make it capable of adapting to changes in the observation. This is the case of, for example, the unexpected disappearance of one on more markers in our current setup due to occlusion or resolution issues.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.

[2] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the CONDENSATION Algorithm for Robust, Vision-based Mobile Robot Localization. In *CVPR*, volume 2, pages 588–594, June 1999.

[3] J. Deutscher and I. Reid. Articulated Body Motion Capture by Stochastic Search. *IJCV*, 61(2):185–205, 2005.

[4] A. Doucet, N. de Freitas, K. P. Murphy, and S. J. Russell. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In *UAI*, pages 176–183, 2000.

[5] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Publishing, second edition, 2000.

[7] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *Proceedings-F Radar and Signal Processing*, volume 140, number 2, pages 107–113, 1993.

[8] M. Isard and A. Blake. CONDENSATION Conditional Density Propagation for Visual Tracking. *IJCV*, 29(1):5–28, 1998.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[10] M. Klaas, N. de Freitas, and A. Doucet. Toward Practical N^2 Monte Carlo: The Marginal Particle Filter. In *UAI*, 2005.

[11] J. MacCormick and A. Blake. A Probabilistic Exclusion Principle for Tracking Multiple Objects. *IJCV*, 39(1):57–71, 2000.

[12] J. MacCormick and M. Isard. Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracking. In *ECCV*, pages 3–19, 2000.

[13] P. S. Maybeck. *Stochastic Models, Estimating, and Control*. Academic Press, 1979.

[14] R. E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.

[15] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe. A Boosted Particle Filter: Multitarget Detection and Tracking. In *ECCV*, pages 28–39, 2004.

[16] B. Stenger. *Model-Based Hand Tracking Using a Hierarchical Bayesian Filter*. PhD thesis, University of Cambridge, March 2004.