Image and Vision Computing 29 (2011) 594–606

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

A hierarchical feature fusion framework for adaptive visual tracking ☆

Alexandros Makris [a,b,*], Dimitrios Kosmopoulos [a], Stavros Perantonis [a], Sergios Theodoridis [b]

[a] NCSR Demokritos, Institute for Informatics and Telecommunications, Computational Intelligence Laboratory, 15310, Aghia Paraskevi, Athens, Greece
[b] University of Athens, Department of Informatics, 15771 Athens, Greece

ARTICLE INFO

Article history:
Received 4 June 2009
Received in revised form 10 March 2011
Accepted 1 July 2011

Keywords:
Visual tracking
Particle filter
Sequential Monte-Carlo

ABSTRACT

A *Hierarchical Model Fusion* (HMF) framework for object tracking in video sequences is presented. The Bayesian tracking equations are extended to account for multiple object models. With these equations as a basis a particle filter algorithm is developed to efficiently cope with the multi-modal distributions emerging from cluttered scenes. The update of each object model takes place hierarchically so that the lower dimensional object models, which are updated first, guide the search in the parameter space of the subsequent object models to relevant regions thus reducing the computational complexity. A method for object model adaptation is also developed. We apply the proposed framework by fusing salient points, blobs, and edges as features and verify experimentally its effectiveness in challenging conditions.

1. Introduction

The problem of visual tracking consists in the localization of moving scene objects (targets) in consecutive frames acquired by static or moving sensors. It has a broad scope of applications ranging from human–computer interfaces, to surveillance. Its general solution might be very challenging especially when the targets are deformable, move abruptly in front of heavily cluttered background under varying illumination conditions and are partially or fully occluded.

A very popular approach is the probabilistic Bayesian tracking methods. The Sequential Monte Carlo (SMC) approximation methods [3–7] known as particle filters (PF) which belong to the Bayesian approach are among the most promising approaches for robust tracking. These methods treat the location of the target as a probability density function, which they attempt to estimate using a set of samples. Their main advantage lies in their ability to cope with multi-modal distributions, such as those emerging from a cluttered environment, due to the maintenance of multiple hypotheses. However, the application of particle filtering methods still has many problems to resolve before it can be considered robust for tracking targets in natural scenes in real-time. Among the most important issues are the efficient and information-rich *target representation* and the selection of the *proposal distribution* (hypothesis generation).

In this work we propose the *Hierarchical Model Fusion* (HMF) framework for fusing visual cues. The target is represented by several *object models* of increasing dimension, which are probabilistically linked. The parameter update for each object model takes place hierarchically so that the simpler object models, which are updated first, guide the search in the state space of the more complex object models to relevant regions. The most complicated object model (in terms of state dimension) and the last in hierarchy, is called *main model* and its parameters fully describe the target. The rest of the object models are referred to as *auxiliary* as the estimation of their state is not required by the application. A method to adapt the auxiliary object models to cope with target appearance changes is also proposed. The method deletes the auxiliary object models which seem to lose track based on a measure of their compatibility with the main object model. When the number of auxiliary models is low new ones are added.

A simple example (see Fig. 1) will clarify the proposed concept. Let us consider a case of a target of which we want to estimate the bounding box. We will use two object models, an auxiliary that tracks a salient point in the target and the main model, the bounding rectangle (blob). The state of the first object model has two parameters, the salient point's coordinates $x_s = [i_{s_x}; i_{s_y}]$, while the blob object model has three, the coordinates of its center and a scale parameter, $x_b = [i_{b_x}; i_{b_y}; s_b]$. When the tracking is initialized the relative position of x_b with respect to x_s is measured. If the tracked object is rigid this relative position should be almost constant between two consecutive frames. Thus if the location of the point is found on the next frame we can significantly narrow the search for the position of the blob thus the search in the three-dimensional space is simplified.

The contribution of our work consists in the following:

- We extended the Bayesian framework to allow the integration of multiple object models which may lead to a better *target representation*.
- We derived a particle filtering based approximation algorithm which leads to efficient *hypothesis generation*. This algorithm integrates

☆ This paper has been recommended for acceptance by Dimitris Samaras.

* Corresponding author. Tel.: +30 2106503141.
E-mail addresses: amakris@iit.demokritos.gr (A. Makris), dkosmo@iit.demokritos.gr (D. Kosmopoulos), sper@iit.demokritos.gr (S. Perantonis), stheodor@di.uoa.gr (S. Theodoridis).

0262-8856/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2011.07.001

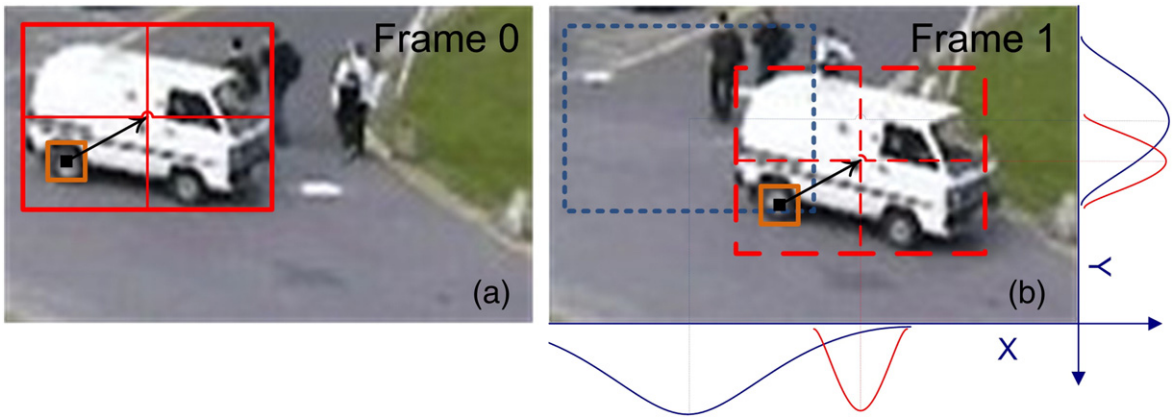

Fig. 1. In (a) the tracking is initialized with two object models describing the target, a blob and a salient point. The arrow shows the relative position of the point and the center of the blob. In (b) the position of the point is updated and using the stored relative distance the proposal for the blob given this position is shown in the x and y axis (red). This proposal is much closer to the target than the proposal derived by the state evolution model of the blob (blue).

multiple object models of different complexity with redundancy of information.
- We developed an *adaptation* technique, to automatically select appropriate auxiliary models.

To test the proposed HMF framework we implemented a tracker using the following cues: salient points within the target, color, and edge information. The object models used, in increasing state dimension order, are salient points and blobs within the target as auxiliary and the target's contour as the main model. A possible drawback of our method that we are going to handle in the future is the dependence on the main model.

The rest of the paper is structured as follows. Section 2 describes the merits of our approach compared to the related works. Section 3 provides the background knowledge to particle filtering methods, presents the HMF framework, and explains its requirements and constraints. Section 4 gives an implementation of the framework. Section 5 contains the experimental results, which demonstrate the merits of the proposed HMF framework using some challenging video shots and simulated data. Section 6 concludes this work.

2. Related work

In this section we overview the related works on tracking and how they treat the *target representation* (object model), the *hypothesis generation/evaluation* (proposal distribution/measurement model), and the *adaptation* compared to our approach. A recent survey on tracking methods can be found here [8].

The choice of the *object model* is crucial and depends on the speed and accuracy requirements of the application. Simple object models such as [9–13] with 3–4 state dimensions are efficiently calculated, but the amount of information they provide is low and requires significant post-processing for video understanding. On the other hand, object models with many state parameters have high computational cost. Examples are [14,1] with 6 state parameters, or target specific models with even more parameters [15–17]. The acquisition of those parameters is often required by the application and the high level knowledge they provide results in a detailed target representation, thus these object models are more difficult to be distracted by clutter. In this work we propose the use of several object models of varying complexity in an attempt to maintain the benefits of both simple and complicated models, by using a coarse to fine strategy.

Another important decision when designing a tracking algorithm is the choice of the *measurement model* and the *proposal distribution*. There are many works in the literature using particle filters with only a single cue (e.g. edges, color) and use the state evolution (dynamics) as a proposal [1,9]. However, using only one cue limits the robustness and sampling from dynamics is inefficient as is now acknowledged by many recent works [13]. The target's motion is hard to predict and its direction or velocity might change abruptly. To account for this kind of motion the range of search should be very wide. This results in inefficient search because many hypotheses are created in low likelihood regions. Furthermore, a large search range increases the probability of tracker distraction by similar objects in the target's neighborhood. Feature fusion is a popular approach to overcome these difficulties, and several methods which attempt it have recently appeared [10,18–22]. These approaches differ in the way they fuse the cues and can be classified in three categories: the first concerns methods which combine several cues during the measurement process to increase robustness. The methods of the second category try to improve the proposal distribution by using some of the cues to guide the new hypotheses in high likelihood areas. The third category contains methods which partition the state space and use different visual cues to sequentially update the resulting sub-states. In the following we are going to outline some methods from all three categories. The main challenge for these methods is to fuse the cues in a way that will increase robustness while maintaining a low computational cost since many tracking applications require on-line performance.

In [18,2] two frameworks for fusion during the measurement process are presented. In [2], a method to automatically estimate the reliability of each feature is also proposed. Similarly in [23], a method to evaluate and select the most suitable features for a given application is presented. Another relative approach is [24], which builds likelihood maps from each feature and combines them based on their classification confidence scores. A limitation of the aforementioned methods is that the object models that are coupled with the various cues must share the same state space. The number of particles required increases exponentially with the number of state parameters, rendering these methods inefficient. In [21], the authors overcome this drawback by using several object models to fuse different cues. This strategy maintains information redundancy and lower computational complexity by splitting the state into several subspaces, however, the cues are fused during the measurement process which does not improve the proposal distribution and thus might result in inefficient search of the state space. The same limitation holds for [25], where two object models are used for head tracking with the particles of each model updated using a Monte Carlo approximation to sequential belief propagation.

The second category of the fusion methods combine the cues during the hypothesis generation stage. Some of them, propose the use of some sort of low level information such as color [26], and

motion [13] to guide the search in high likelihood areas. These approaches make the first step towards better proposal distributions but they cannot efficiently bridge the gap between low level features and complex object models. In [14] shape and color information are fused. Information from one cue is used to construct a proposal distribution for the other and vice-versa. The authors claim that the use of several "rough" models increases the robustness of their method by avoiding clutter and keeps the computational cost low because a complicated model is not required. However, in cases where the background has objects similar to the target those rough models could fail. Additionally, several applications require a detailed target representation which is only achieved by using complex object models.

In the third category of fusion methods belongs the work in [10], which is more closely related to our method. Partitioned sampling is used to reduce the search in the state space and provides an efficient proposal distribution. Each cue provides information only for a subset of the state vector parameters. The new particles are constructed incrementally, each cue is used to update the sub-state which is assigned to it. This way the search in a high dimensional space is replaced by consecutive searches in lower dimensional spaces thus increasing the sampling efficiency. However, if one of the cues fails to locate the target region the rest will not be able to recover because, as mentioned above, only one cue is used for each subset of the state vector. Additionally, to use the partitioning method one has to find informative visual cues for each partition which is not always possible. The same approach is followed in [27] where the state of an articulated target (a hand) is partitioned and the search proceeds in a sequential fashion. If the sub-state that contains the parameters of the palm fails to locate it then tracking is lost (see Fig. 2). A similar method is presented in [28] for tracking multiple targets. In [29], partitioned sampling is combined with a method to dynamically change the order of the partitions. In [30], the state is partitioned and Rao–Blackwellization is used to analytically compute the appearance part of the state-vector while the location part is approximated using particles.

In our approach the information from various cues is gradually exploited through an object model hierarchy. It builds upon the method described in [31]. In [31], the object models are connected only through the proposal distribution which results in a loose connection. In this work, the object models are integrated by modifying the update equations of Bayesian filters. Each object model uses some of the cues to estimate its parameters and then is used to estimate a subset of the parameters of the consequent models. However, only the last object model (main) contains the required target parameters. This way the search in the parameter space of the complicated object models is narrowed but information redundancy is preserved. Our method is used for correlated fields of the state vector, to exploit information redundancy, while the partitioned sampling requires independent (uncorrelated) state vector fields. Fig. 3, shows the graphical models corresponding to the standard particle filter [1], the partitioned sampling method [27,10], and our method. Fig. 2, presents an example of an articulated model tracking to highlight the differences between the aforementioned methods.

The measurement model usually uses a reference template (e.g. for color based measurement models the reference template could be the color distribution of the target at the initialization frame), with which the various hypotheses are compared. However, due to illumination changes, target rotations and deformations, occlusions etc., the target's appearance changes during tracking, thus rendering the reference template invalid. Several approaches deal with these changes, by adapting the aforementioned template such as [18,32,33,12,11]. The main problem which these methods try to tackle is adapting the reference template to the background. Our method exploits the use of multiple object models to adapt the auxiliary object models while the main object model provides the information about the location of the target so that adaptation to background is minimized.

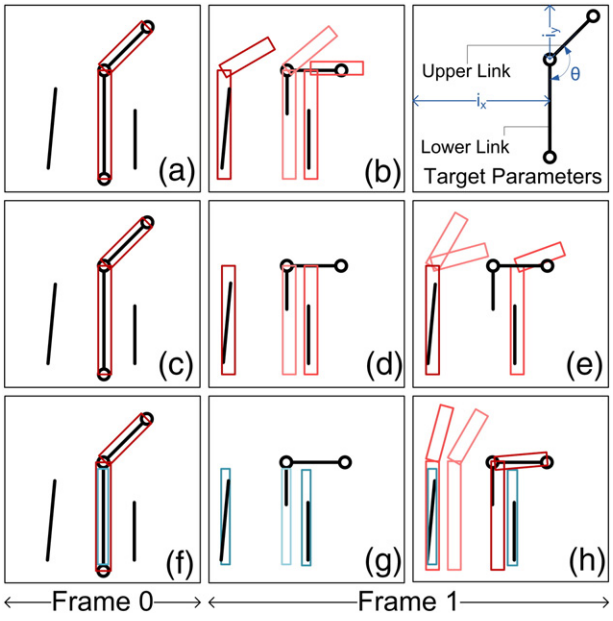

Fig. 2. Synthetic example of tracking an articulated object. The state vector is defined as $\mathbf{x}=[i_x;i_y;\vartheta]$ where i_x, i_y are the coordinates of the lower link. The position of the upper link is defined only by its angle ϑ (we assume constant link length for simplicity). In Frame 1 the lower link is occluded and some background clutter exists. The hypotheses are represented by bounding boxes and their color shows their weight (darker means higher weight). (a–b) Standard Particle Filter (SIR): the state is updated simultaneously for the 3 parameters, thus the computational cost is exponentially increased with the number of state dimensions. (c–e) Partitioned Sampling: the sub-state of the lower link $[i_x;i_y]$ is updated first at (d), followed by resampling and the update of the sub-state of the upper link $[\vartheta]$ at (e). The tracking of the lower link is distracted by the clutter which results in the tracker losing the target. (f–h) HMF: an auxiliary object model (blue rectangle) with state $\mathbf{x}_a=[i'_x;i'_y]$ tracks the lower link and is updated first at (g) using a different visual cue than the main model. The main object model with state $\mathbf{x}=[i_x;i_y;\vartheta]$ which tracks the whole target is updated at (h) by fusing information from the updated state of the auxiliary object model and the dynamics. Even though the auxiliary model is distracted by the clutter themain model successfully locates the target.

3. Proposed framework

3.1. Bayesian tracking/particle filters

Let $\{\mathbf{x}_t; t\in\mathbb{N}\}$ be the unobserved state of the target and $\{\mathbf{z}_t; t\in\mathbb{N}\}$ the measurements for every time step, t. The Bayesian Filtering consists

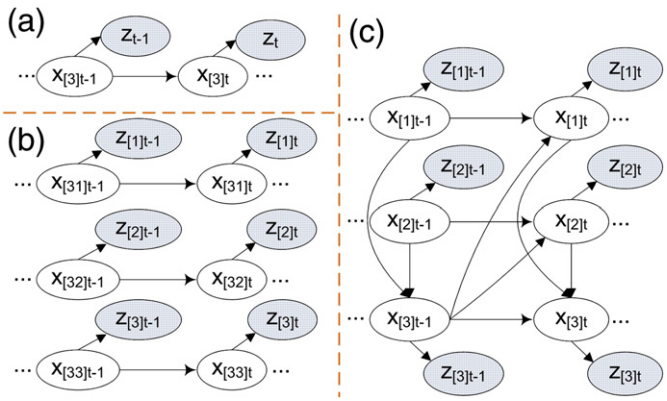

Fig. 3. Graphical models depicting only slices $t-1$ and t of the temporal dimension. The state to be approximated is denoted as $x_{[3]}$. (a) SIR [1]. (b) Partitioned Sampling [27,10], the state is partitioned in 3 parts ($x_{[31]}$, $x_{[32]}$, $x_{[33]}$), which are updated independently, each one depending on different measurements. (c)Proposed HMF model, using 2 auxiliary object models ($x_{[1]}$, $x_{[2]}$) which are linked to the main object model ($x_{[3]}$).

of calculating the posterior $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$ (where $\mathbf{x}_{0:t}=(\mathbf{x}_0 \ldots \mathbf{x}_t)$) at every step, given the measurements up to that step and a prior, $p(\mathbf{x}_0)$. The solution is expressed as:

$$p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) = p(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}) \frac{p(\mathbf{z}_t|\mathbf{x}_{0:t},\mathbf{z}_{1:t-1}) p(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})}. \quad (1)$$

In most practical problems the assumptions of Markovian dynamics $p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and the independence of measurements given the current state $p(\mathbf{z}_t|\mathbf{x}_{0:t}, \mathbf{z}_{1:t-1}) = p(\mathbf{z}_t|\mathbf{x}_t)$ are made.

Under these assumptions Eq. (1) becomes:

$$p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) = p(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}) \frac{p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})}. \quad (2)$$

The PF methods are used to approximate the probabilities involved in the Bayesian tracking equations. They use samples (particles) to estimate the involved pdf's [5,7]. Given N weighted particle trajectories $\{\mathbf{x}_{0:t-1}^{(n)}, w_{t-1}^{(n)}\}_{n=1}^{N}$, which approximate $p(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1})$ up to time $t-1$, the SMC methods compute N particles $\{\mathbf{x}_t^{(n)}\}_{n=1}^{N}$ which are combined with the previous trajectories to form $\{\mathbf{x}_{0:t}^{(n)}, w_t^{(n)}\}_{n=1}^{N}$, which approximate the posterior $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$, up to time t.

The particles are drawn from the proposal distribution $q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$ and are weighted by:

$$w_t^{(n)} = \frac{p\left(\mathbf{x}_{0:t}^{(n)}|\mathbf{z}_{1:t}\right)}{q\left(\mathbf{x}_{0:t}^{(n)}|\mathbf{z}_{1:t}\right)}. \quad (3)$$

The proposal distribution is selected to factorize as (using the Markov assumption):

$$q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) = q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{z}_t) q(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}) \quad (4)$$

to be able to update the weights recursively. To avoid wasting particles in low likelihood areas, a resampling step is added resulting to the *Sampling Importance Resampling* (SIR) [5] algorithm. The steps are shown in Algorithm 1.

A very common realization of this algorithm uses the prior $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ as proposal distribution which results in the weights updated by the likelihood $p(\mathbf{z}_t|\mathbf{x}_t)$ [1].

Algorithm 1. SIR Algorithm.

Input: $\{\mathbf{x}_{0:t-1}^{(n)}, w_t^{(n)}\}_{n=1}^{N}$

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

Sample $\mathbf{x}_t^{(n)}$ from $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(n)}, \mathbf{z}_t)$

Weight $\mathbf{x}_t^{(n)}$ by the following, which results if we replace Eqs.(2) and (4) in Eq.(3):

$$w_t'^{(n)} = \frac{p\left(\mathbf{z}_t|\mathbf{x}_t^{(n)}\right) p\left(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)}\right)}{q\left(\mathbf{x}_t^{(n)}|\mathbf{x}_{t-1}^{(n)},\mathbf{z}_t\right)} \quad (5)$$

end for

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

Normalize the weights by:

$$w_t^{(n)} = \frac{w_t'^{(n)}}{\sum_{n_1=1}^{N} w_t'^{(n_1)}} \quad (6)$$

end for

Resample the particle set according to its weights, so that the resulting particle set will be un-weighted and with the same number of particles.

Output: $\{\mathbf{x}_{0:t}^{(n)}, 1/N\}_{n=1}^{N}$

3.2. HMF framework description

In the general case M object models are used for target representation. The state can be written as[1]:

$$\mathbf{x} = \left[\mathbf{x}_{[1]}; \mathbf{x}_{[2]}; \ldots; \mathbf{x}_{[M]}\right] \quad (7)$$

where $\mathbf{x}_{[i]}$ are the state vectors of each object model. To each object model corresponds a measurement model with parameters $\mathbf{z}_{[i]}$. The graphical model of Fig. 3c encodes the architecture of our framework. It depicts the following assumptions which we make to derive an algorithm for the recursive calculation of the posterior:

- The total likelihood is given by multiplying the likelihoods of individual models: $p(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{M} p(\mathbf{z}_{[i]}|\mathbf{x}_{[i]})$.
- The state evolution is decomposed as: $p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \prod_{i=1}^{M} p(\mathbf{x}_{[i]t}|Pa(\mathbf{x}_{[i]t}))$.

where $Pa(\mathbf{x}_{[i]t})$ denotes the parent nodes of $\mathbf{x}_{[i]t}$.

To construct algorithms that will be able to update the posterior of each object model sequentially we derive the following equation which is an extension to the classical Bayesian tracking Eq. (1), but takes place in M steps. Each step updates the state of the corresponding object model. Using simple probability rules and the assumptions mentioned above the filtering equation for the *i-th* step is given by (the derivation steps are given in Appendix A):

$$p\left(\mathbf{x}_{[1:i]t}, \mathbf{x}_{0:t-1}|\mathbf{z}_{[1:i]t}, \mathbf{z}_{1:t-1}\right) = p\left(\mathbf{x}_{[1:i-1]t}, \mathbf{x}_{0:t-1}|\mathbf{z}_{[1:i-1]t}, \mathbf{z}_{1:t-1}\right) \frac{p\left(\mathbf{z}_{[i]t}|\mathbf{x}_{[i]t}\right) p\left(\mathbf{x}_{[i]t}|Pa\left(\mathbf{x}_{[i]t}\right)\right)}{p\left(\mathbf{z}_{[i]t}|\mathbf{z}_{[1:i-1]t}, \mathbf{z}_{1:t-1}\right)}. \quad (8)$$

As mentioned above, Eq. (8) can be used to construct Bayesian tracking algorithms that use multiple object models. Here, we use it to construct a PF based algorithm to iteratively update the posterior of each model.

We assume that at time $t-1$ the posterior $p(\mathbf{x}_{[1:M]0:t-1}|\mathbf{z}_{[1:M]1:t-1})$ is approximated by a weighted particle set comprised of N weighted sample trajectories: $\left\{\mathbf{x}_{[1:M]0:t-1}^{(n)}, w_{[M]t-1}^{(n)}\right\}_{n=1}^{N}$. To update the particle set that approximate the posterior at time t we proceed in a sequential fashion. Each object model is updated using the information from the already updated models at time t. The proposal distribution is selected to factorize as:

$$q\left(\mathbf{x}_{[1:i]t}, \mathbf{x}_{0:t-1}|\mathbf{z}_{[1:i]t}, \mathbf{z}_{1:t-1}\right) = q\left(\mathbf{x}_{[i]t}|Pa\left(\mathbf{x}_{[i]t}\right), \mathbf{z}_{[i]t}\right) q\left(\mathbf{x}_{[1:i-1]t}, \mathbf{x}_{0:t-1}|\mathbf{z}_{[1:i-1]t}, \mathbf{z}_{1:t-1}\right)\Big). \quad (9)$$

As in standard PF the samples are drawn from the first factor of Eq. (9).

The weights are given by:

$$w_{[i]t}^{(n)} = \frac{p\left(\mathbf{x}_{[1:i]t}^{(n)}, \mathbf{x}_{0:t-1}^{(n)}\middle|\mathbf{z}_{[1:i]t}, \mathbf{z}_{1:t-1}\right)}{q\left(\mathbf{x}_{[1:i]t}^{(n)}, \mathbf{x}_{0:t-1}^{(n)}\middle|\mathbf{z}_{[1:i]t}, \mathbf{z}_{1:t-1}\right)}. \quad (10)$$

By substituting Eqs. (8) and (9) into Eq. (10) we get the following weight update equation:

$$w_{[i]t}^{(n)} \propto w_{[i-1]t}^{(n)} \frac{p\left(\mathbf{z}_{[i]t}|\mathbf{x}_{[i]t}^{(n)}\right) p\left(\mathbf{x}_{[i]t}^{(n)}|Pa^{(n)}\left(\mathbf{x}_{[i]t}\right)\right)}{q\left(\mathbf{x}_{[i]t}^{(n)}|Pa^{(n)}\left(\mathbf{x}_{[i]t}\right), \mathbf{z}_{[i]t}\right)}. \quad (11)$$

[1] For notational clarity we avoid using the ^{T} superscript, instead we use the Matlab's ';' notation to concatenate vectors.

The steps of the iteration for the update of object model i at time t of the proposed algorithm are summarized in Algorithm 2 and a visual representation can be found in Fig. 4.

Algorithm 2. HMF Algorithm.

Input: $\left\{\mathbf{x}_{[1:i-1]t}^{(n)}, \mathbf{x}_{0:t-1}^{(n)}, w_{[i]t}^{(n)}\right\}_{n=1}^{N}$

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

Sample $\mathbf{x}_{[i]t}^{(n)}$ from $q\left(\mathbf{x}_{[i]t}^{(n)} | Pa\left(\mathbf{x}_{[i]t}\right)^{(n)}, \mathbf{z}_{[i]t}\right)$

Weight each particle by:

$$w_{[i]t}^{\prime(n)} \propto w_{[i-1]t}^{(n)} \frac{p\left(\mathbf{z}_{[i]t} | \mathbf{x}_{[i]t}^{(n)}\right) p\left(\mathbf{x}_{[i]t}^{(n)} | Pa^{(n)}\left(\mathbf{x}_{[i]t}\right)\right)}{q\left(\mathbf{x}_{[i]t}^{(n)} | Pa^{(n)}\left(\mathbf{x}_{[i]t}\right), \mathbf{z}_{[i]t}\right)}$$

end for

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

Normalize the weights by:

$$w_t^{(n)} = \frac{w_t^{\prime(n)}}{\sum_{n_1=1}^{N} w_t^{\prime(n_1)}}$$

end for

Resample the particle set according to its weights, so that the resulting particle set will be un-weighted and with the same number of particles.

Output: $\left\{\mathbf{x}_{[1:i]t}^{(n)}, \mathbf{x}_{0:t-1}^{(n)}, 1/N\right\}_{n=1}^{N}$

In Fig. 3, we show the graphical models for (a) the standard PF [1], (b) the partitioned sampling method [27,10] and, (c) the proposed HMF method. The standard PF uses a single object model for the target and therefore is modeled by a single Markov chain. The partitioned sampling method breaks the state into several sub-states and estimates each one separately using different visual cues. There are no interactions between the cues therefore the method depends on each one of them because if one fails then the sub-state that corresponds to it will not be estimated correctly and the target will be lost. In contrast, we use several auxiliary object models each one employing different cues to provide information to the main object model which contains the parameters that we need to estimate. Each auxiliary model provides information for a sub-state of the main model, this way information from more than one cue is fused to estimate the same sub-state. This interaction is graphically represented by the cross-model links of Fig. 3c. This strategy enables us to form better proposal distributions thus reducing the required number of particles and on the same time maintaining information redundancy which increases the robustness of the approach.

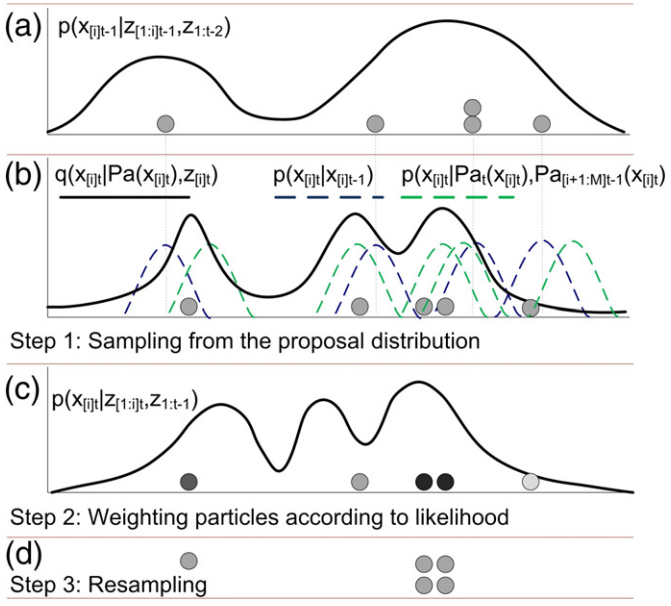

Fig. 4. Update for object model i at time t. (a) The pdf and particles at time $t-1$. (b) The proposal is formed by fusing information from the current object model's previous state and from the rest of the object models. $Pa_t(\mathbf{x}_{[i]t})$ denotes the parent nodes of $\mathbf{x}_{[i]t}$ from time t, $Pa_{[i+1:M]t-1}(\mathbf{x}_{[i]t})$ denotes the parent nodes of $\mathbf{x}_{[i]t}$ from time $t-1$ excluding $\mathbf{x}_{[i]t-1}$. (c) The new particles are weighted. Darker particles have higher weight. (d) Resampling.

3.3. Tracking models requirements

As mentioned above the proposed HMF framework requires several object models. The requirements for those models are presented in the following:

(i) The main model should contain the target parameters required by the application (e.g. location, shape, velocity).
(ii) Auxiliary models of different dimensionality should be available. Low dimensional models require less particles but they usually are not robust and do not offer a detailed target representation. High dimensional models, on the other hand, offer a very detailed target representation.
(iii) The models must be probabilistically linked. This means that we should be able to evaluate the conditional probability of the model state given the states of the ones linked to it.

For each object model, one or more visual cues are used to represent the target. The object models have to be updated in increasing state dimension order, so that the amount of processing can be minimized. The main model is the most complex one and the last to be updated. This strategy breaks the initial problem into M sub-problems. This way, fewer particles are required to search efficiently the state space, leading to lower computational complexity. As mentioned above the main object model contains all the required target parameters therefore the auxiliary object models are not restricted to describe the whole target. They might be used to describe only a part of it, such as a blob within the target. This way the resistance in partial occlusions is increased.

4. Tracker implementation

In this section we use the framework to build a tracker, which we applied in tracking various targets in challenging situations. We combined three different object models to represent the target which are in the order which are updated:

(i) A salient point tracking model. This model has only 2 position parameters.
(ii) A blob tracking model. The blob represents a rectangular region of the target with homogeneous color with 3 parameters which describe its position and scale.
(iii) The target's contour. This is the main object model. It is represented as a b-spline curve and contains 5 parameters which allow several geometric transformations.

The combined state vector is: $\mathbf{x} = [\mathbf{S};\mathbf{B};\mathbf{C}]$, where $\mathbf{S}$ represents the salient points $\mathbf{B}$ represents the blobs and $\mathbf{C}$ represents the contour curve. For a single target more than one salient point or blob object models can be used. To increase robustness and reduce the overall complexity we don't link together the object models of the same class as shown in Fig. 5 where there is no link between the two point object models. We also choose to link each point object model with one blob object model, by selecting the blob closest to it during the initialization. All the auxiliary models are directly connected to the main model. We denote by K_s the number of the tracked salient points and by K_b the number of blobs. The contour object model tracks only one curve representing the contour of the target. To link the models

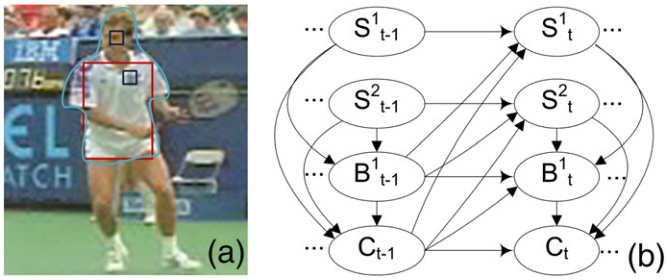

Fig. 5. (a) Sample image showing two salient points (dark), one blob (red) and one contour model (blue). (b) Graphical model of the implemented tracker, depicting slices $t-1$ and t of the temporal dimension for the aforementioned models. The evidence nodes are omitted for clarity.

together, each auxiliary model contains a set of parameters in their state that specify the relative position of each object that it tracks (salient points, blobs) to the higher object models. The initialization step is done manually. Once the target area is defined the contour curve is constructed. Within this area a search for salient points and blobs using standard corner and blob detection algorithms takes place to automatically initialize the other object models.

In the following we describe each object model in detail and define their measurement model $p(\mathbf{z}_{[i]t}|\mathbf{x}_{[i]t})$, and the proposal $q(\mathbf{x}_{[i]t}|Pa(\mathbf{x}_{[i]t}), \mathbf{z}_{[i]t})$, which are required to integrate them into the framework.

4.1. Salient points

We use K_s object models that track salient points of the target. The state vector for all the tracked points is $\mathbf{S} = [\mathbf{s}^1; \ldots; \mathbf{s}^{K_s}]$. The vector of a single salient point is defined as (in the rest of this section we refer to a single salient point without using the superscript denoting the number of the point for notational clarity):

$$\mathbf{s} = [\mathbf{sp}; \mathbf{sr}] \tag{12}$$

where:

- $\mathbf{sp} = [i_{s_x}; i_{s_y}]$ are the x and y-coordinates of the point.
- $\mathbf{sr} = [\mathbf{srb}; \mathbf{src}]$ are the parameters representing the relative position of the point and the rest of the object models and more specifically, **srb** is the relative position of the point with respect to the center of the blob with which is linked and **src** is the relative position of the point with respect to the center of gravity of the curve.

The likelihood of the n-th sample, of a point is determined by calculating the sum of squared differences(SSD) between a rectangle around a candidate feature and the template:

$$p\left(\mathbf{z}_{[S]}|\mathbf{s}^{(n)}\right) \propto \exp\left\{-\frac{d_{ssd}\left(\bar{\mathbf{s}}, \mathbf{s}^{(n)}\right)}{2\sigma_{s_i}^2}\right\} \tag{13}$$

where $d_{ssd}\left(\bar{\mathbf{s}}, \mathbf{s}^{(n)}\right)$ is the SSD between the template point, $\bar{\mathbf{s}}$, and the current hypothesis $\mathbf{s}^{(n)}$. The standard deviation, σ_{s_i}, is set empirically.

For each point, we select the proposal distribution to be $p(\mathbf{s}_t|\mathbf{x}_{t-1})$. The relative position **sr** is set during the initialization of the object model and remains unchanged during tracking. The proposal distribution for the position parameters of the n-th sample is defined by (the rest of the equations of this section refer to a single particle however the '(n)' superscript is omitted for notational clarity):

$$\begin{aligned} p(\mathbf{sp}_t|\mathbf{sp}_{t-1}, \mathbf{bp}_{t-1}, \mathbf{cp}_{t-1}, \mathbf{sr}_{t-1}) &\propto p(\mathbf{sp}_t|\mathbf{sp}_{t-1}) \\ &\times [p(\mathbf{sp}_t|\mathbf{bp}_{t-1}, \mathbf{srb}_{t-1}) + p(\mathbf{sp}_t|\mathbf{cp}_{t-1}, \mathbf{src}_{t-1})] \end{aligned} \tag{14}$$

where, $\mathbf{bp}_{t-1}$ and $\mathbf{cp}_{t-1}$ are the position parameters for the blob (the specific blob that is connected to the salient point) and the contour object models respectively. $p(\mathbf{sp}_t|\mathbf{sp}_{t-1})$ is the dynamic model of the salient points modeled by a Gaussian and $p(\mathbf{sp}_t|\mathbf{bp}_{t-1}, \mathbf{srb}_{t-1})$, $p(\mathbf{sp}_t|\mathbf{cp}_{t-1}, \mathbf{src}_{t-1})$ represent the influence from blob and contour object models which are also modeled by Gaussians:

$$p(\mathbf{sp}_t|\mathbf{sp}_{t-1}) = \mathcal{N}\left(\mathbf{sp}_t; \mathbf{sp}_{t-1}, \Sigma_{s_p}\right) \tag{15}$$

$$p(\mathbf{sp}_t|\mathbf{bp}_{t-1}, \mathbf{srb}_{t-1}) = \mathcal{N}\left(\mathbf{sp}_t; \mathbf{bs}_{t-1} + \mathbf{srb}_{t-1}, \Sigma_{p_{sb}}\right) \tag{16}$$

$$p(\mathbf{sp}_t|\mathbf{cp}_{t-1}, \mathbf{src}_{t-1}) = \mathcal{N}(\mathbf{sp}_t; \mathbf{cs}_{t-1}) + \mathbf{src}_{t-1}, \Sigma_{p_{sc}}) \tag{17}$$

where, Σ_{s_p}, $\Sigma_{p_{sb}}$ and $\Sigma_{p_{sc}}$ are the diagonal covariance matrices which are determined empirically, **bs** and **cs** denote the sub-vector of the parameters of blobs and contour which are linked to the parameters of the salient points. From the above equation we see that the proposal PDF is the product of a Gaussian dynamic model for the salient points and a Gaussian Mixture Model (GMM) resulting from adding the influence from blobs and contour. The motivation for the above formula is to restrict the particles by their dynamics while considering the influence from the rest of the object models. Therefore, the product rule is used to suppress regions not compatible with the previous target position while a sum rule is applied to the influence from the other object models to maintain multi-modality. The result is proportional to a GMM [34].

4.2. Blobs

The blob tracking object model is similar to the one described above. The model is used to describe rectangular homogeneously colored regions within the target. Similarly to the salient points, K_b object models of this type are used $\mathbf{B} = \left[\mathbf{b}^1; \ldots; \mathbf{b}^{K_b}\right]$. The state vector is each object model is defined as (in the rest of this section we refer to a single blob without using the superscript denoting the number of the blob for notational clarity):

$$\mathbf{b} = [\mathbf{bp}; \mathbf{br}] \tag{18}$$

where:

- $\mathbf{bp} = \left[i_{b_x}; i_{b_y}; s_b\right]$Z are the x and y-coordinates and scale of the blob.
- **br** are the parameters representing the relative position of the blob with respect to the center of gravity of the curve.

The likelihood of the n-th sample, of a blob is given by:

$$p\left(\mathbf{z}_{[B]}|\mathbf{b}^{(n)}\right) \propto \exp\left\{-\frac{d_{bht}^2\left(H\left(\bar{\mathbf{b}}\right), H\left(\mathbf{b}^{(n)}\right)\right)}{2\sigma_{b_i}^2}\right\} \tag{19}$$

where $H\left(\mathbf{b}^{(n)}\right)$ is the histogram for the n hypothesis of the blob, $H\left(\bar{\mathbf{b}}\right)$ is the k_b blob's template histogram and σ_{b_i} is the standard deviation. The $d_{bht}(\cdot)$ denotes the Bhattacharyya distance defined as:

$$d_{bht}(H_1, H_2) = \sqrt{1 - \sum_{u=1}^{m} \sqrt{\left[H_1^{(u)} H_2^{(u)}\right]}} \tag{20}$$

where the summation is over the m histogram bins.

For each blob, we define the proposal distribution as $p(\mathbf{b}_t|\mathbf{b}_{t-1}, \mathbf{C}_{t-1}, \mathbf{S}_t)$. The relative position **br** is set during the initialization of the model and remains unchanged during tracking. The proposal distribution for the position parameters is defined by (the rest of the

equations of this section refer to a single particle however the '(n)' superscript is omitted for notational clarity):

$$p(\mathbf{bp}_t|\mathbf{bp}_{t-1},\mathbf{sp}_t,\mathbf{cp}_{t-1},\mathbf{br}_{t-1}) \propto p(\mathbf{bp}_t|\mathbf{bp}_{t-1}) \times[p(\mathbf{bp}_t|\mathbf{sp}_t,\mathbf{srb}_t) + p(\mathbf{bp}_t|\mathbf{cp}_{t-1},\mathbf{br}_{t-1}))] \quad (21)$$

where, $\mathbf{sp}_t$ is the position of the salient point linked to this blob object model, $p(\mathbf{bp}_t|\mathbf{bp}_{t-1})$ is the dynamic model of the blobs modeled by a Gaussian distribution and $p(\mathbf{bp}_t|\mathbf{sp}_t, srb_t)$, $p(\mathbf{bp}_t|\mathbf{cp}_{t-1},\mathbf{br}_{t-1}))$ represent the influence from salient point and contour object models respectively which are also modeled by Gaussians:

$$p(\mathbf{bp}_t|\mathbf{bp}_{t-1}) = \mathcal{N}\left(\mathbf{bp}_t; \mathbf{bp}_{t-1}, \Sigma_{b_p}\right) \quad (22)$$

$$p(\mathbf{bs}_t|\mathbf{sp}_t,\mathbf{srb}_t) = \mathcal{N}\left(\mathbf{bs}_t; \mathbf{sp}_t - \mathbf{srb}_t, \Sigma_{p_{sb}}\right) \quad (23)$$

$$p(\mathbf{bp}_t|\mathbf{cp}_{t-1},\mathbf{br}_{t-1}) = \mathcal{N}(\mathbf{bp}_t; \mathbf{cb}_{t-1} + \mathbf{br}_{t-1}), \Sigma_{p_{br}}) \quad (24)$$

where, Σ_{b_p}, $\Sigma_{p_{sb}}$ and $\Sigma_{p_{bc}}$ are the covariances which are determined empirically and **cb** is a vector containing the parameters of the contour object model which are linked to the parameters of the blob model.

4.3. Contour

The contour of the target, modeled by a B-Spline curve, is the main object model. The curve's shape is initialized in the first frame or it can be selected from a pool of predefined shapes when the type of the tracked object is known in advance. In contrast with the other models only one curve is used for each target. The contour object model has 5 parameters which allow for several geometric transformations of the initial curve. The state vector is:

$$\mathbf{C} = \left[i_{c_x}; i_{c_y}; r_c; s_{c_x}; s_{c_y}\right] \quad (25)$$

where, i_{c_x}, i_{c_y} are the translation parameters, r_c the rotation, and s_{c_x}, s_{c_y} the scale parameters.

The likelihood, $p(\mathbf{z}_{[C]}|\mathbf{C})$, is calculated by using edge cues. More specifically, we measure how well the detected edges fit the current contour hypothesis using the method described in [1].

$$p\left(\mathbf{z}_{[C]}|\mathbf{C}^{(n)}\right) \propto \exp\left\{-\frac{f_{cl}^2\left(\mathbf{C}^{(n)}\right)}{2\sigma_{c_l}^2}\right\} \quad (26)$$

where $f_{cl}(\cdot)$ is the aforementioned metric (see [1] for details) and σ_{c_l} is the standard deviation which is determined empirically.

The proposal distribution for this object model is (the rest of the equations of this section refer to a single particle however the '(n)' superscript is omitted for notational clarity):

$$p(\mathbf{C}_t|\mathbf{C}_{t-1},\mathbf{B}_t,\mathbf{S}_t) \propto p(\mathbf{C}_t|\mathbf{C}_{t-1})\left(\sum_{k_b=1}^{K_b} p\left(\mathbf{C}_t|\mathbf{bp}_t^{k_b} - \mathbf{br}_t^{k_b}\right) + \sum_{k_s=1}^{K_s} p\left(\mathbf{C}_t|\mathbf{sp}_t^{k_s} - \mathbf{src}_t^{k_s}\right)\right). \quad (27)$$

The definitions for the probabilities follow:

$$p(\mathbf{C}_t|\mathbf{C}_{t-1}) = \mathcal{N}\left(\mathbf{C}_t; \mathbf{C}_{t-1}, \Sigma_{c_p}\right) \quad (28)$$

$$p(\mathbf{cs}_t|\mathbf{sp}_t - \mathbf{src}_t) = \mathcal{N}\left(\mathbf{cs}_{[t}; \mathbf{sp}_t - \mathbf{src}_t, \Sigma_{p_{cs}}\right) \quad (29)$$

$$p(\mathbf{cb}_t|\mathbf{bp}_t - \mathbf{br}_t) = \mathcal{N}\left(\mathbf{cb}_t; \mathbf{bp}_t - \mathbf{br}_t, \Sigma_{p_{cb}}\right) \quad (30)$$

where the covariances Σ_{c_p}, $\Sigma_{p_{cs}}$ and, $\Sigma_{p_{cb}}$ are determined empirically. We should note here that information from the current estimates of the salient points and blobs is used. For the parameters of the contour (e.g. rotation parameter) that the blob or the salient point object models cannot provide a prior only the dynamic model of the contour is used.

4.4. Adaptation procedure

So far we proposed a framework to fuse information from different cues using several object models which are initialized at the first frame. Here, we will expand the proposed framework to adapt the auxiliary object models during tracking using informationfrom the main object model. The adaptation consists of replacing auxiliary models which appear to lose track with new ones detected within the region defined by the main model. The fact that only the main model is required for the target representation allows to flexibly add or remove auxiliary models, which is very helpful in varying scene conditions. In this section we analyze how the adaptation is performed on the implemented object models.

The adaptation of the auxiliary models is integrated in the update equation. Each auxiliary model has a parameter that encodes the confidence of the object to belong to the target (target confidence). The adaptation consists of deleting an object if the target confidence is below a threshold and detecting and initializing new objects. If an auxiliary model is mislead by clutter, it has to be deleted otherwise it will mislead the tracker. For a salient point model the target confidence parameter $\mathbf{st}_t$ is initialized when the point is detected and updated during tracking by the following filtering equation:

$$\mathbf{st}_t = a_{s_{tc}}\mathbf{st}_{t-1} + \left(1 - a_{s_{tc}}\right) f_{stc}(\mathbf{s}_{t-1}, \mathbf{C}_{t-1}) \quad (31)$$

where $a_{s_{tc}}$ is the filtering parameter and t_{tc} is the threshold for deleting an auxiliary model. $f_{stc}(\cdot)$ is a metric measuring the compatibility between the points and the contour object model on the previous frame:

$$f_{stc}(\mathbf{s}_{t-1},\mathbf{C}_{t-1}) = \exp\left\{-\frac{d_{bht}^2\left(\mathbf{L}_{[S]},\mathbf{L}_{[C]}\right)}{2\sigma_{tl}}\right\} \quad (32)$$

where $\sigma_{p_{sc}}$ is the deviation, $\mathbf{L}_{[S]}$, $\mathbf{L}_{[C]}$ are the likelihood vectors for a salient point and the contour object model respectively. The vectors contain the likelihood values for the N particles and are defined as $\mathbf{L}_{[S]} = \left[p\left(\mathbf{z}_{[S]}|\mathbf{s}^{(1)}\right);\ldots;p\left(\mathbf{z}_{[S]}|\mathbf{s}^{(N)}\right)\right]$ and $\mathbf{L}_{[C]} = \left[p\left(\mathbf{z}_{[C]}|\mathbf{C}^{(1)}\right);\ldots;p\left(\mathbf{z}_{[C]}|\mathbf{C}^{(N)}\right)\right]$. The Bhattacharyya distance between the vectors, in this case $\mathbf{L}_{[S]}$, $\mathbf{L}_{[C]}$ of length N is defined as:

$$d_{bht}\left(\mathbf{L}_{[S]},\mathbf{L}_{[C]}\right) = \sqrt{1 - \sum_{n=1}^{N}\sqrt{\left[\mathbf{L}_{[S]}^{(n)}\mathbf{L}_{[C]}^{(n)}\right]}}. \quad (33)$$

This equation models the similarity between the likelihood vectors which is high when the two object models describe the same target (i.e. the particles with high salient point model likelihood also have high contour model likelihood). In that case the link from one object model to the other is meaningful. In contrast when one of the object models is distracted by clutter then the similarity between the two vectors is expected to be lower. The same equations hold for the initialization and updateof the target confidence parameter of the blob object model $\mathbf{bt}_{[t}$.

When $\mathbf{st}_{t-1}<t_{tc}$ or $\mathbf{bt}_{t-1}<t_{tc}$ for a salient point or blob object model respectively then the auxiliary model is deleted and a detection

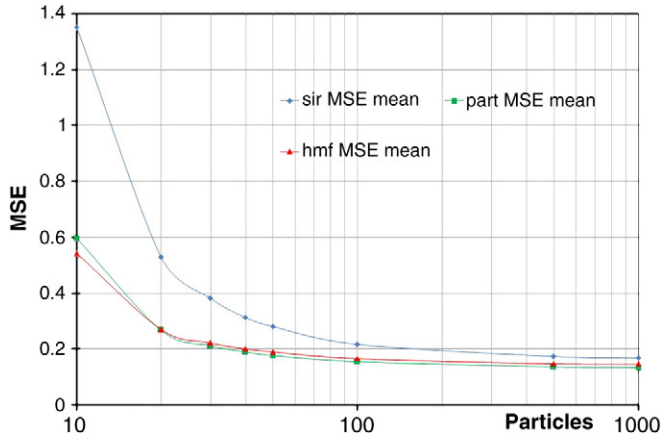

Fig. 6. Simulation Results Charts — the charts show the average MSE in relation with the number of particles for the SIR, PS, and HMF methods. As position estimator the mean particle position is used.

procedure searches for new salient points or blobs to re-initialize in the target region as defined by the main model:

$$p(\mathbf{s}_t|\mathbf{C}_{t-1}) = p(\mathbf{s}_t|D_s(\mathbf{C}_{t-1})) \tag{34}$$

$$p(\mathbf{b}_t|\mathbf{C}_{t-1}) = p(\mathbf{b}_t|D_b(\mathbf{C}_{t-1})) \tag{35}$$

where $D_s(\mathbf{C}_{t-1})$, $D_b(\mathbf{C}_{t-1})$ denotes the detection process within the area defined by the contour object model on the previous frame. This is the same process that is used during initialization. For the salient points we use the point detector from [35] and for the blobs we perform segmentation by pyramids using the functions provided by the OpenCV library.

5. Experiments

The experiments have been executed using both simulated data and several challenging video sequences and various objects have been tracked to verify our method. More specifically, we experimented with sequences containing deformable objects, abrupt motion, heavy clutter, partial occlusions and, short full occlusions.

5.1. Simulation experiments

In this section we use simulated data to compare our HMF method with the SIR algorithm [1] and the Partitioned Sampling (PS) method [10]. We consider a 2-dimensional state space $\mathbf{x}_t = [x_{t[x]}, x_{t[y]}]$ with state evolution equations:

$$x_{[x]t} = \mathbf{x}_{[x]t-1} + d_x \tag{36}$$

$$x_{[y]t} = x_{[y]t-1} + d_y sin\left(d_\phi t\right) \tag{37}$$

where d_x, d_y, d_ϕ are constants. The likelihood of a measurement $\mathbf{z}_t$ given the state $\mathbf{x}_t$ is given by:

$$p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{z}_t; \mathbf{x}_t + n_t, \Sigma_l) \tag{38}$$

Table 1
Simulation parameter values.

Parameter	Value	Parameter	Value
σ_l	0.2	σ_{se}	0.2
σ_{ln}	0.01	σ_{ca}	0.3
σ_{aux}	0.2	σ_{cm}	0.15

where Σ_l is the diagonal covariance matrix, and n_t is zero mean Gaussian noise with covariance matrix Σ_{ln}. For all the methods the particles are initially posed around the true target position. For the SIR method the state evolution model is a Gaussian around the previous position $p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \mathrm{x}_{t-1}, \Sigma_{se})$, where Σ_{se} is the diagonal covariance matrix with values σ_{se}. For the partitioned sampling method we update the y-dimension first using similarly a Gaussian with variance σ_{se} and after a weighted resambling step we update the x-dimension sampling again from a Gaussian with variance σ_{se}. For the HMF method we use an additional auxiliary state, $x'_{[y]t}$, that is related to the one ($x_{[y]t}$) of the two state dimensions by:

$$x'_{[y]t} = x_{[y]t} + p_t \tag{39}$$

where p_t is zero mean Gaussian noise with variance σ_{aux}. In the case of visual tracking the auxiliary state is used to track a part of the target and provides information for a sub-state of the main object model, however their relative position might change. In the simulated data case, we model this change by including the Gaussian noise p_t to create the auxiliary state. The update of the auxiliary state is performed by sampling from $p\left(x'_{[y]t}|\mathbf{x}_{t-1}, x'_{[y]t-1}\right) = \mathcal{N}\left(x'_{[y]t}; x'_{[y]t-1}, \sigma_{se}\right)\mathcal{N}\left(x'_{[y]t}; x_{[y]t-1}, \sigma_{ce}\right)$ where the two distributions are given by Gaussians as $p\left(x'_{[y]t}|x'_{[y]t-1}\right) = \mathcal{N}\left(x'_{[y]t}; x'_{[y]t-1}, \sigma_{se}\right)$, and $p\left(x'_{[y]t}; x_{[y]t-1}\right) = \mathcal{N}\left(x'[y]t; x_{[y]t-1}, \sigma_{ca}\right)$ with variances σ_{se} and σ_{ca} respectively. Similarly for the update of the main model the sampling is performed from: $p\left(x_t|\mathbf{x}_{t-1}, x'_{[y]t}\right) = p(\mathbf{x}_t|\mathbf{x}_{t-1})p\left(\mathbf{x}_t; x'_{[y]t}\right)$ where the probabilities are again Gaussians with variances σ_{se} and σ_{cm} respectively.

Using the above equations we generated 100 sequences each consisting of 1000 time instants. We run the trackers for each and calculated the Mean Square Error over the sequences using as estimator the mean particle position, and the higher weighted particle. The above procedure was repeated for various number of particles, and the results are shown in Fig. 6. The parameter values that were used are summarized in Table 1.

The results, depicted in Fig. 6, show that our HMF method has comparable performance with the partitioned sampling method. However, our method has more general applicability because it does not require a partitioning of the main model's state, which implies that appropriate cues for each partition have to be found. Both methods outperform the clasical SIR algorithm by requiring much less particles to achieve the same level of performance. For instance, the SIR algorithm requires almost 100 particles to reach the performance of our HMF algorithm with only 20 particles. As is expected when the number of particles increases all the methods converge towards the same level of performance.

5.2. Experiments with real video sequences

For the experiments with real video sequences we implemented the following trackers which we compare:

- **SIR** This is the original SIR algorithm described in Section 3.1. The state vector contains the object models described in Section 4.
- **HMF** The proposed tracker as described in Section 4, without adaptation.
- **aHMF** The proposed tracker using the adaptation method of Section 4.4.

The SIR tracker was built using the same object models and exploiting the same visual cues as the proposed HMF tracker. The states of all the used object models are combined to form a single state vector which is estimated using the standard equations described in Section 3.1. This way the comparison with HMF depends only on the update strategy since the measurement models are the same.

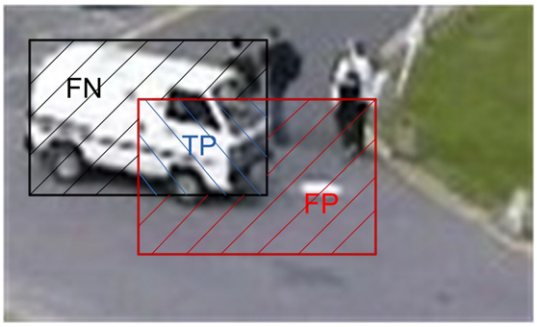

Fig. 7. Comparison metrics. The picture shows the ground truth bounding box (black) and the tracker's output (red). The *TP*, *FP*, and *FN* areas are highlighted.

Additionally, for fair comparison we compare the SIR tracker with HMF which does not use adaptation. Comparison to the standard particle filter is widely performed in the related literature. Comparing with other state-of-the art methods is difficult due to lack of standard datasets.

For each video category that we experimented with we selected the object models which were suitable for tracking and we built the aforementioned trackers using these models. In sports videos for example, the salient point object model is not very helpful so we used only the blob and contour object models. For surveillance videos we used all the three object models.

To compare the trackers we annotated several sequences by hand and we calculated the 'Tracker Detection Rate'(TDR) and 'False Alarm Rate'(FAR) measures [36]:

$$TDR = \frac{TP}{TP + FN}, FAR = \frac{FP}{FP + TP} \tag{40}$$

where *TP*, *FN* and *FP* denote the true positive, false negative and false positive area respectively (see Fig. 7). The ground truth target area was defined by a bounding box. For all trackers we calculated the bounding box of the output contour and we compared it with the ground truth bounding box. The output contour was calculated using the mean of the particles. The *TP* area is calculated as the area of the intersection between the ground truth and the output bounding boxes. The *FN* is the area of the ground truth bounding box excluding the *TP* area and similarly *FP* is the area of the tracker output bounding box excluding the *TP* area.

We made several series of experiments to enlighten various aspects of our method:

- The first series contains mainly qualitative experiments to evaluate the proposal distribution, the collaboration between the object models, and the model adaptation.
- The second series contains quantitative experiments to compare our method (HMF) with the standard particle filter (SIR).

In the previous sections describing the object models several parameters have appeared. The setting of these parameters was done empirically since some values worked well for most of the testing sequences. These parameters are summed up here:

- Standard deviations of the measurement models: $\sigma_{s_l}, \sigma_{b_l}, \sigma_{c_l}$.
- Covariances of the dynamic models: Σ_{s_p}, Σ_{b_p}, Σ_{c_p}. These parameters should be large enough to accommodate for abrupt motion but not excessively large because this increases the probability of the tracker to get distracted by clutter. We consider diagonal covariance matrices with each element being proportional to the range of the corresponding parameter (e.g. for the salient points $\Sigma_{s_p} = diag((\sigma_{s_p} * r_x)^2, (\sigma_{s_p} * r_y)^2)$ where r_x and r_y are the ranges of the x and y-coordinates of the salient points).
- Covariances linking the previous estimates of higher order object models to the current of lower order object models: $\Sigma_{p_{sb}}$, $\Sigma_{p_{sc}}$, $\Sigma_{p_{bc}}$. The values of these parameters are set larger than those of the dynamics above because they should cover situations where both the position of the object as well as the relative position to the higher order object changes between the previous and the current frame. These matrices are also diagonal and each element's value is proportional to the range of the corresponding parameter as in the case of dynamic model covariances.
- Covariances linking the current estimates of lower order object models to the higher order object models: $\Sigma_{p_{bs}}$, $\Sigma_{p_{cs}}$, $\Sigma_{p_{cb}}$. The values of these parameters are normally lower than those of the rest because they incorporate knowledge from the current frame which is more reliable and is not affected by abrupt motion.
- Filtering parameters for the update of target confidence of the auxiliary objects: $a_{s_{tc}}$, $a_{b_{tc}}$. Their values determine the 'memory' of the filtering function. Lower values mean that the target confidence adapts more rapidly to the current estimates.
- Threshold in the target confidence for deleting auxiliary objects t_{tc}, and standard deviation for the calculation of target confidence σ_{tl}.

The values that we used throughout the experiments are shown in Table 2.

Table 2
Parameter values. A_{sp}, is the area in pixels of the salient point, σ_{s_p}, σ_{b_p} etc. are the deviations which are multiplied by the ranges to form the diagonal covariance matrices Σ_{s_p}, Σ_{b_p} etc.

Parameter	Value	Parameter	Value
σ_{s_l}	$5A_{sp}$	$\sigma_{p_{bs}}, \sigma_{p_{cs}}, \sigma_{p_{cb}}$	0.02
σ_{b_l}	0.4	$\sigma_{s_p}, \sigma_{b_p}, \sigma_{c_p}$	0.03
σ_{c_l}	13	$\sigma_{p_{sb}}, \sigma_{p_{sc}}, \sigma_{p_{bc}}$	0.04
t_{tc}	0.2	$a_{b_{tc}}, a_{s_{tc}}$	0.6
σ_{tl}	0.3		

5.2.1. Qualitative experiments

In this section we are going to provide some qualitative experiments to show how our method behaves under various tracking conditions. More specifically, we are going to highlight the following points: i) The proposal distribution of the main object model has lower variance than that of SIR and the particles are more concentrated near the target even in cases with abrupt motion. ii) The adaptation mechanism is tested in sequences with target appearance changes (e.g. self occlusions, deformations) where the adaptation consists of replacing or completely removing salient point or blob object models.

In the experiment displayed in Fig. 8 we compare our HMF tracker to the SIR in a PETS 2006 surveillance sequence using the blob and contour object models. The contour hypotheses of our method are much more concentrated near the actual target than those of the SIR because of the strong prior provided by the blob model.

The experiment displayed in Fig. 9, illustrates the concept of the object model adaptation using the aHMF tracker with two object model types, salient points and contour. The salient point object models are deleted and re-initialized as the initial points are occluded due to the object rotation. The main model (contour), defines the target area and the search for new points is performed there.

Another experiment which illustrates the benefits of the adaptive tracker, aHMF, compared to HMF is presented in Fig. 10. One blob and the contour object model are used in a sequence where the color of the target changes due to illumination variations. The adaptive tracker re-initializes the blob model by detecting homogeneously colored regions within the area tracked by the contour as seen in the first row of Fig. 10. The HMF tracker fails when the color distribution of the target changes and the tracking performance deteriorates as seen in Fig. 10h.

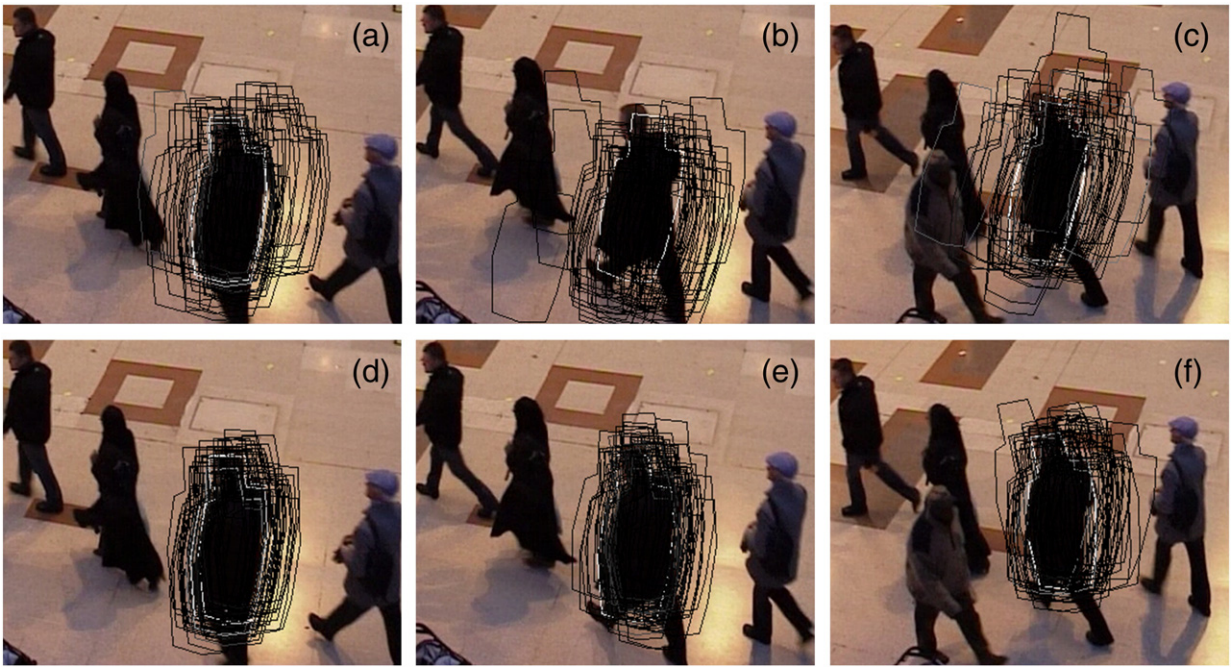

Fig. 8. Tracking results — surveillance sequence:(a), (b), (c)—SIR, 50 particles (d), (e), (f)—HMF, 50 particles. Frames 1, 35, 50. Both trackers use the blob and the contour object models, for image clarity we only show the contour particles. The particles of the HMF tracker are more concentrated near the target due to the better proposal distribution.

In several cases some type of auxiliary models do not provide valid information for the target. In such cases the adaptation mechanism discards these object models without replacing them. One such situation is observed in Fig. 11, where the target is textureless and all the salient point models are discarded for the largest part of the sequence. Another sequence is shown in Fig. 12 where the blob models are misled by similar background colors and are quickly discarded. The salient points are not helpful throughout the whole sequence and therefore are discarded for several frames as well.

5.2.2. Quantitative comparison

In this section we present quantitative experiments to test the accuracy and speed of our method (HMF) compared to the standard PF (SIR). To effectively compare the trackers we restricted the methods to equal resources by setting the same number of particles and using the same object models. Although we use a more complex sampling method the difference in computational complexity is negligible because the measurement stage consumes much more resources than the sampling stage. The following model setup is used in the experiments, however for some datasets we didn't use salient point models:

- Salient points: 4 grayscale point models with masks sized at about 1/10th the target area.
- Blobs: 2 blob models with area at about 1/3rd the target area.
- Contour: a b-spline curve with 18 control points.

We measured the TDR and FAR for each method for different number of particles. On a standard PC with 2 GHz processor and for frame size of 720×576 pixels (DVD quality) real-time performance (25 fps) was achieved with 30 particles.

Three challenging datasets where used for the quantitative experiments. They contain sequences from soccer games, tennis games and moving vehicles. The length of the sequences varies from 50 to 400 frames. Their parameters are summarized in Table 3. In Fig. 13, we show several screen-shots from the sequences of each dataset. In sports datasets we used only the blobs and the contour object model as the salient points were not helpful. In the vehicles dataset all the three object model types were used. Fig. 14 shows the mean TDR and FAR values for different number of particles for the 'soccer', 'tennis' and, 'vehicles' dataset. Our HMF tracker performs better having higher TDR and lower FAR compared to the SIR tracker at all the datasets. As is expected the difference is larger when the number of particles is low. After a certain number both methods converge to a standard TDR and FAR value which seems to be the limit for the given features. Our method reaches that value with much fewer particles. In 'soccer' dataset, even with 10 particles our method reaches the performance of the SIR tracker with 80 particles. In 'tennis', a very

Fig. 9. Tracking results — 7up sequence: aHMF, 60 particles, frames 1, 180, 300.

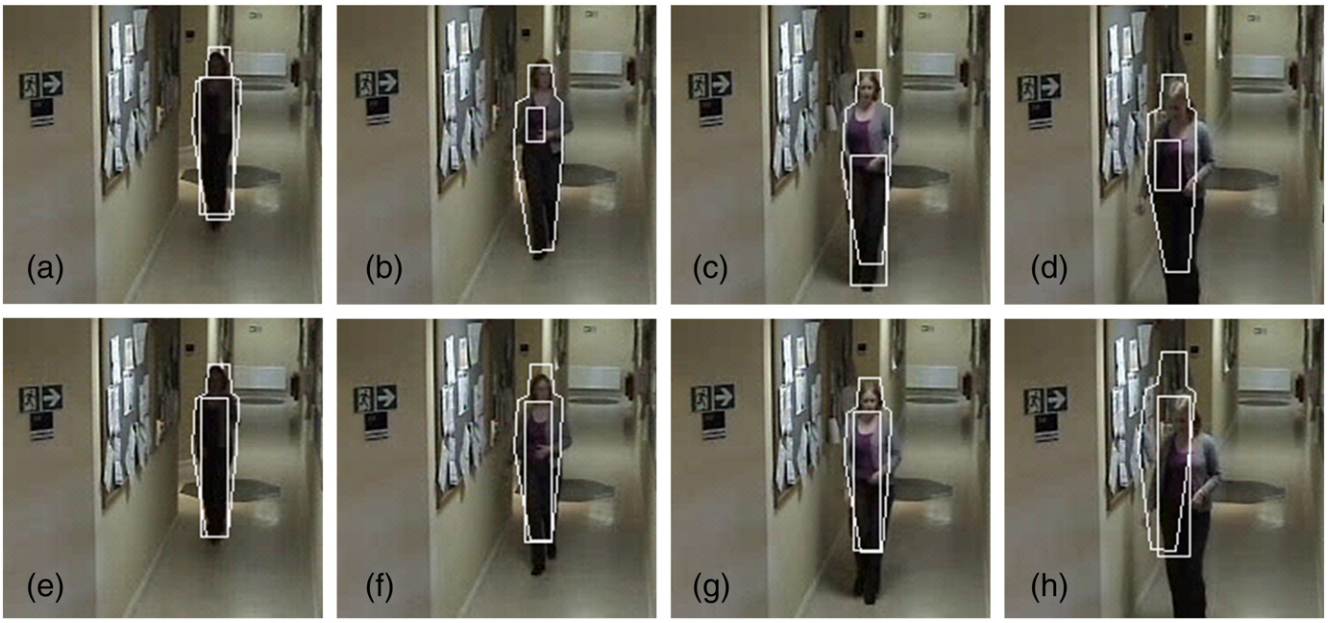

Fig. 10. Tracking results — walking sequence:(a)–(d) aHMF, 60 particles. (e)–(h) HMF, 60 particles. Frames 1, 15, 35, 45.

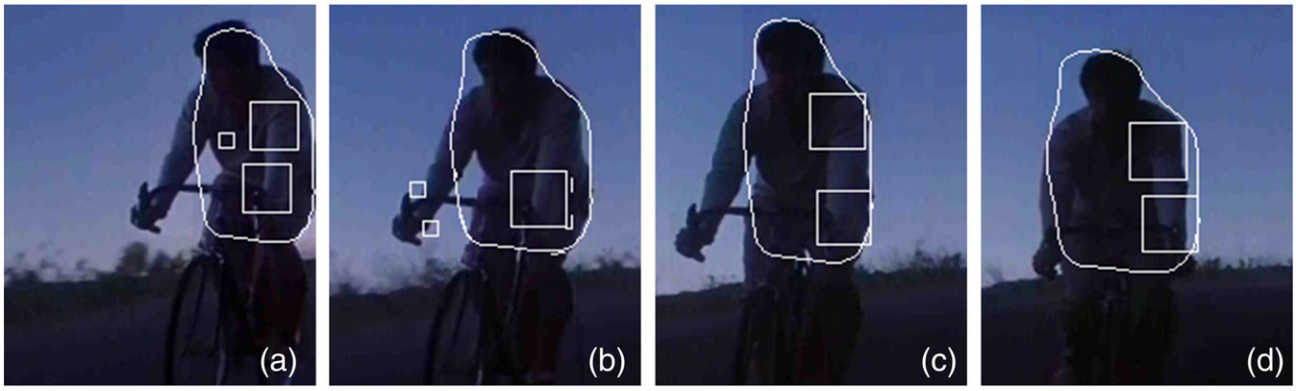

Fig. 11. Tracking results — textureless sequence: aHMF, 50 particles, frames 1, 25, 50, 75. The adaptation mechanism completely discards the salient point object models. In (c), (d) only the contour and two blob object models are active.

adequate performance is also met with 10 particles by our tracker whereas the SIR requires 60 particles to reach the same performance levels. A similar situation is observed in 'vehicles' where the SIR requires six times more particles to reach the same performance.

6. Conclusions

In this paper we proposed a framework for building particle filtering based tracking algorithms. These algorithms enhance significantly previous approaches in terms of robustness and speed by fusing several cues hierarchically in order to achieve robust tracking in various challenging situations including occlusions, deformations, abrupt motion, illumination changes and cluttered background.

The problem of dimensionality is a very challenging problem for sample-based methods. Our work acknowledges the problem and tries to tackle it by building priors using low dimensional object models at the beginning and using them to guide the most complex

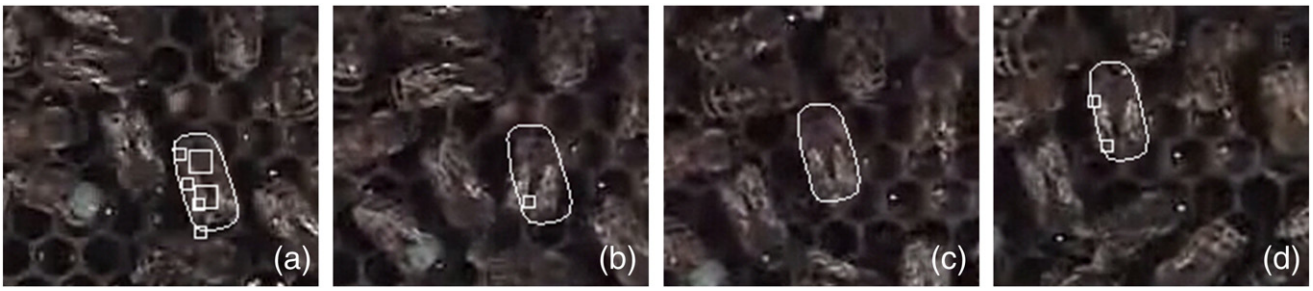

Fig. 12. Tracking results — sequence without blobs: aHMF, 50 particles, frames 1, 15, 30, 50. The blob object models, although initialized in (a), are quickly discarded because the background contains similar colors that distract it. In (c), the salient point object models are also discarded and only the contour is used. In (d), several new salient points are detected.

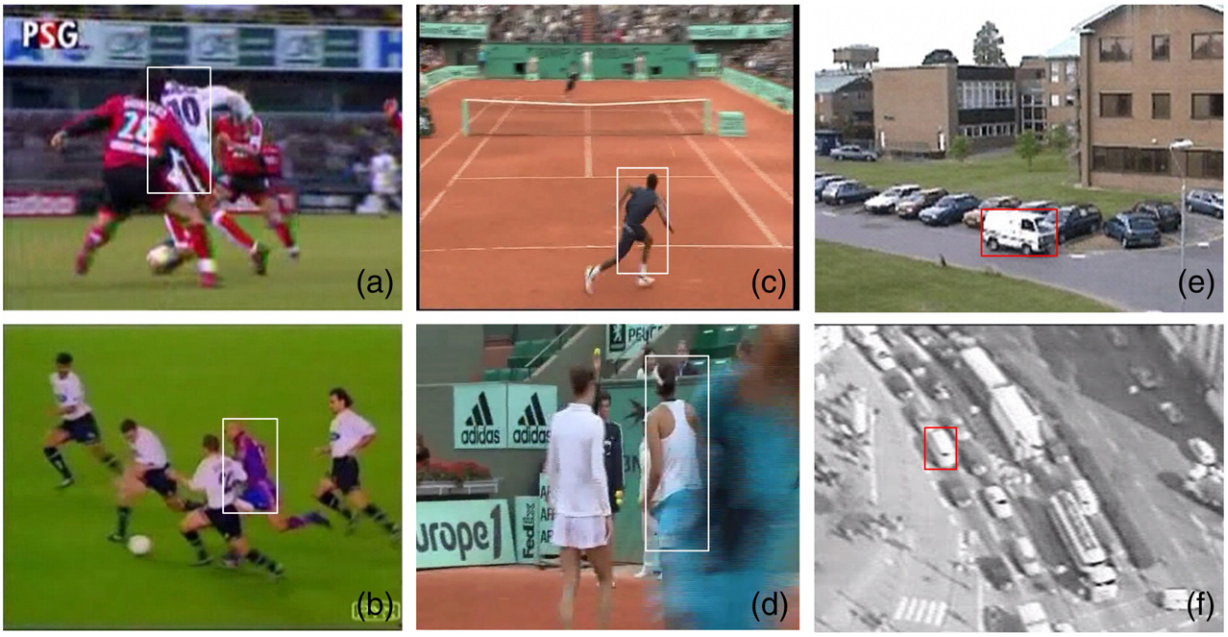

Fig. 13. Dataset screen-shots: 'soccer', 'tennis' and, 'vehicles' datasets.

Table 3
Datasets parameters: number of clips and total frames.

Dataset	Clips	Frames
Soccer	15	1880
Tennis	20	2000
Vehicles	10	1600

ones. The more complex object models use the better priors offered by simple models and thus require significantly less particles. Our experiments both with real or simulated data demonstrate that this technique works and we show that we succeded to achieve a reduction in the number of required particles by a factor of 5 to 10 which is not negligible. This allows our method to use higher dimensional models up to some extend, however, we do not claim that this completely solves the problem of dimensionality which might arise if we use very high dimensional models.

A limitation of the proposed method that we are going to handle in the future is the dependence on the main model. A more flexible representation would use several object models to describe the target and would not depend on a single main model. Another open issue is the development of a more elaborate way to link the different object models together. Scale variations, rotations or gradual change in the relative position could be taken into account to form a more general cross-model link. The framework could also be used with the human body or other object specific object models which usually have many parameters; therefore we believe that the use of our search strategy could be beneficial.

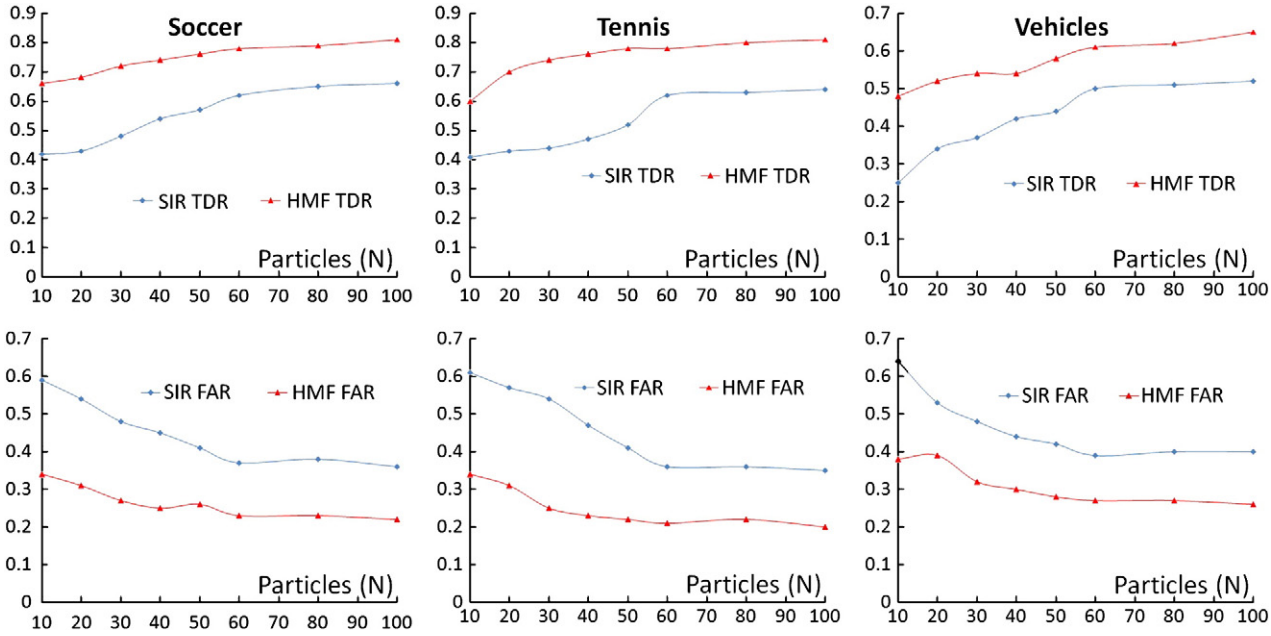

Fig. 14. Tracking results charts — (a) soccer dataset, (b) tennis dataset, (c) vehicles dataset. The charts show the average TDR and FAR for different number of particles.

Appendix A. Bayesian filtering equation

The steps for the derivation of the bayesian filtering equation for M object models are:

$$
\begin{aligned}
p\left(\mathbf{x}_{[1:i]t},\mathbf{x}_{0:t-1}\,|\,\mathbf{z}_{[1:i]t},\mathbf{z}_{1:t-1}\right) &= \frac{p\left(\mathbf{z}_{[1:i]t},\mathbf{z}_{1:t-1}\,|\,\mathbf{x}_{[1:i]t},\mathbf{x}_{0:t-1}\right)p\left(\mathbf{x}_{[1:i]t},\mathbf{x}_{0:t-1}\right)}{p\left(\mathbf{z}_{[1:i]t},\mathbf{z}_{1:t-1}\right)}\\
&= \frac{p\left(\mathbf{z}_{[i]t}\,|\,\mathbf{z}_{[1:i-1]t},\mathbf{z}_{1:t-1},\mathbf{x}_{[1:i]t},\mathbf{x}_{0:t-1}\right)p\left(\mathbf{z}_{[1:i-1]t},\mathbf{z}_{1:t-1}\,|\,\mathbf{x}_{[1:i]t},\mathbf{x}_{0:t-1}\right)p\left(\mathbf{x}_{[1:i]t},\mathbf{x}_{0:t-1}\right)}{p\left(\mathbf{z}_{[1:i]t},\mathbf{z}_{1:t-1}\right)}\\
&= \frac{p\left(\mathbf{z}_{[i]t}\,|\,\mathbf{x}_{[i]t}\right)p\left(\mathbf{x}_{[1:i]t},\mathbf{x}_{0:t-1}\,|\,\mathbf{z}_{[1:i-1]t},\mathbf{z}_{1:t-1}\right)}{p\left(\mathbf{z}_{[i]t}\,|\,\mathbf{z}_{[1:i-1]t},\mathbf{z}_{1:t-1}\right)}\\
&= \frac{p\left(\mathbf{z}_{[i]t}\,|\,\mathbf{x}_{[i]t}\right)p\left(\mathbf{x}_{[i]t}\,|\,\mathbf{x}_{[1:i-1]t},\mathbf{x}_{0:t-1},\mathbf{z}_{[1:i-1]t},\mathbf{z}_{1:t-1}\right)p\left(\mathbf{x}_{[1:i-1]t},\mathbf{x}_{0:t-1}\,|\,\mathbf{z}_{[1:i-1]t},\mathbf{z}_{1:t-1}\right)}{p\left(\mathbf{z}_{[i]t}\,|\,\mathbf{z}_{[1:i-1]t},\mathbf{z}_{1:t-1}\right)}\\
&= p\left(\mathbf{x}_{[1:i-1]t},\mathbf{x}_{0:t-1}\,|\,\mathbf{z}_{[1:i-1]t},\mathbf{z}_{1:t-1}\right)\frac{p\left(\mathbf{z}_{[i]t}\,|\,\mathbf{x}_{[i]t}\right)p\left(\mathbf{x}_{[i]t}\,|\,Pa\left(\mathbf{x}_{[i]t}\right)\right)}{p\left(\mathbf{z}_{[i]t}\,|\,\mathbf{z}_{[1:i-1]t},\mathbf{z}_{1:t-1}\right)}.
\end{aligned}
\tag{A.1}
$$

References

[1] M. Isard, A. Blake, Condensation — conditional density propagation for visual tracking, Int. J. Comput. Vis. 29 (1) (1998) 5–28, URL: citeseer.ist.psu.edu/isard98condensation.html.

[2] E. Maggio, F. Smerladi, A. Cavallaro, Adaptive multifeature tracking in a particle filtering framework, Circuits and Systems for Video Technology, IEEE Transactions on, 17 (10), Oct. 2007, pp. 1348–1359, doi:10.1109/TCSVT.2007.903781.

[3] R.M. Neal, Probabilistic inference using Markov chain Monte Carlo methods, Tech. Rep. CRG-TR-93-1, University of Toronto, (1993), URL: citeseer.ist.psu.edu/neal93probabilistic.html.

[4] J.S. Liu, R. Chen, Sequential Monte Carlo methods for dynamic systems, J. Am. Stat. Assoc. 93 (1998) 1032–1044, URL, citeseer.ist.psu.edu/article/liu98sequential.html.

[5] A. Doucet, S. Godsill, C. Andrieu, On sequential Monte Carlo sampling methods for Bayesian filtering, Stat. Comput. 10 (3) (2000) 197–208, doi: http://dx.doi.org/10.1023/A:1008935410038.

[6] C. Andrieu, N. de Freitas, A. Doucet, M.I. Jordan, An introduction to mcmc for machine learning, Mach. Learn. V50 (1) (2003) 5–43, doi:10.1023/A:1020281327116, URL, http://dx.doi.org/10.1023/A:1020281327116.

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

[8] A. Yilmaz, O. Javed, M. Shah, Object tracking: a survey, ACM Comput. Surv. 38 (4) (2006) 13, doi: http://doi.acm.org/10.1145/1177352.1177355.

[9] P. Perez, C. Hue, J. Vermaak, M. Gangnet, Color-based probabilistic tracking, ECCV '02: Proceedings of the 7th European Conference on Computer Vision—Part I, Springer-Verlag, London, UK, 2002, pp. 661–675.

[10] P. Perez, J. Vermaak, A. Blake, Data fusion for visual tracking with particles, Proc. IEEE 92 (3) (2004) 495–513.

[11] J. Vermaak, P. Perez, M. Gangnet, A. Blake, Towards improved observation models for visual tracking: selective adaptation, Proc. Eur. Conf. Computer Vision, ECCV '02, Vol. 1, Copenhagen, Denmark, 2002, pp. 645–660.

[12] A.D. Jepson, D.J. Fleet, T.F. El-Maraghi, Robust online appearance models for visual tracking, IEEE Trans. Pattern Anal. Mach. Intell. 25 (10) (2003) 1296–1311.

[13] J. Odobez, D. Gatica Perez, S. Ba, Embedding motion in model-based stochastic tracking, IEEE Trans. Image Process. 15 (11) (2006) 3514–3530.

[14] Y. Wu, T.S. Huang, Robust visual tracking by integrating multiple cues based on co-inference learning, Int. J. Comput. Vis. 58 (1) (2004) 55–71, doi: http://dx.doi.org/10.1023/B:VISI.0000016147.97880.cd.

[15] M. Isard, J. MacCormick, Bramble: a Bayesian multiple-blob tracker, Proceedings International Conference on Computer Vision, 2001, pp. 34–41, URL: citeseer.ist.psu.edu/isard01bramble.html.

[16] D.M. Gavrila, The visual analysis of human movement: a survey, Comput. Vis. Image Underst. 73 (1) (1999) 82–98, doi: http://dx.doi.org/10.1006/cviu.1998.0716.

[17] T.B. Moeslund, E. Granum, A survey of computer vision-based human motion capture, Comput. Vis. Image Underst.: CVIU 81 (3) (2001) 231–268, URL: citeseer.ist.psu.edu/moeslund01survey.html.

[18] M. Spengler, B. Schiele, Towards robust multi-cue integration for visual tracking, ICVS '01: Proceedings of the Second International Workshop on Computer Vision Systems, Springer-Verlag, London, UK, 2001.

[19] C. Shen, A. van den Hengel, A.R. Dick, Probabilistic Multiple Cue Integration for Particle Filter Based Tracking, in: C. Sun, H. Talbot, S. Ourselin, T. Adriaansen (Eds.), DICTA, CSIRO Publishing, 2003, pp. 399–408.

[20] F. Moreno-Noguer, A. Sanfeliu, D. Samaras, Fusion of a multiple hypotheses color model and deformable contours for figure ground segmentation in dynamic environments, In Proceedings 3rd IEEE Workshop on Articulated and Nonrigid Motion, ANM '04, (in conjunction with CVPR '04), 2004.

[21] F. Moreno-Noguer, A. Sanfeliu, D. Samaras, Dependent multiple cue integration for robust tracking, IEEE Trans. Pattern Anal. Mach. Intell. 30 (4) (2008) 670–685, doi: http://dx.doi.org/10.1109/TPAMI.2007.70727.

[22] P. Li, F. Chaumette, Image cues fusion for contour tracking based on particle filter, in: J. Perales, B. Draper (Eds.), Int. Workshop on articulated motion and deformable objects, AMDO '04, Vol. 3179 of Lecture Notes in Computer Science, Palma de Mallorca, Spain, 2004, pp. 99–107.

[23] B. Stenger, T. Woodley, R. Cipolla, Learning to track with multiple observers, 2009, pp. 2647–2654.

[24] Z. Yin, F. Porikli, R. Collins, Likelihood map fusion for visual object tracking, 2008, pp. 1–7.

[25] Y. Li, H. Ai, C. Huang, S. Lao, Robust head tracking based on a multi-state particle filter, Automatic Face and Gesture Recognition, 2006, FGR 2006. 7th International Conference on, 2006, pp. 335–340, doi:10.1109/FGR.2006.96.

[26] M. Isard, A. Blake, ICONDENSATION: unifying low-level and high-level tracking in a stochastic framework, Lect. Notes Comput. Sci. 1406 (1998) 893–908, URL: citeseer.ist.psu.edu/isard98icondensation.html.

[27] J. MacCormick, M. Isard, Partitioned sampling, articulated objects, and interface-quality hand tracking, ECCV '00: Proceedings of the 6th European Conference on Computer Vision—Part II, Springer-Verlag, London, UK, 2000, pp. 3–19.

[28] K. Branson, S. Belongie, Tracking multiple mouse contours (without too many samples), CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1, IEEE Computer Society, Washington, DC, USA, , 2005, pp. 1039–1046, doi: http://dx.doi.org/10.1109/CVPR.2005.349.

[29] S. Duffner, J. Odobez, E. Ricci, Dynamic Partitioned Sampling for Tracking with Discriminative Features, Proceedings of the British Maschine Vision Conference, 2009.

[30] Z. Khan, T. Balch, F. Dellaert, A Rao–Blackwellized particle filter for eigentracking, computer vision and pattern recognition, CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on 2 (27 June–2 July 2004) II–980–II–986, 2, 2004, doi:10.1109/CVPR.2004.1315271.

[31] A. Makris, D.I. Kosmopoulos, S.J. Perantonis, S. Theodoridis, Hierarchical feature fusion for visual tracking, ICIP, 6, 2007, pp. 289–292.

[32] K. Nummiaro, E. Koller-Meier, L.J.V. Gool, An adaptive color-based particle filter, Image Vis. Comput. 21 (1) (2003) 99–110.

[33] J. Li, C.-S. Chua, Transductive local exploration particle filter for object tracking, Image Vision Comput. 25 (5) (2007) 544–552, doi: http://dx.doi.org/10.1016/j.imavis.2006.05.001.

[34] M.J.F. Gales, S.S. Airey, Product of Gaussians for speech recognition, Comput. Speech Lang. 20 (1) (2006) 22–40.

[35] J. Shi, C. Tomasi, Good features to track, IEEE Conference on Computer Vision and Pattern Recognition (CVPR '94), Seattle, 1994, URL: citeseer.ist.psu.edu/shi94good.html.

[36] D. Hall, J. Nascimento, P. Ribeiro, E. Andrade, P. Moreno, S. Pesnel, T. List, R. Emonet, R.B. Fisher, J.S. Victor, J.L. Crowley, Comparison of target detection algorithms using adaptive background models, Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2nd Joint IEEE International Workshop on, 15–16 Oct. 2005, 2005, pp. 113–120, doi: 10.1109/VSPETS.2005.1570905.