Dependent Multiple Cue Integration for Robust Tracking

Francesc Moreno-Noguer, Alberto Sanfeliu, *Member*, *IEEE*, and Dimitris Samaras, *Member*, *IEEE*

Abstract—We propose a new technique for fusing multiple cues to robustly segment an object from its background in video sequences that suffer from abrupt changes of both illumination and position of the target. Robustness is achieved by the integration of appearance and geometric object features and by their estimation using Bayesian filters, such as Kalman or particle filters. In particular, each filter estimates the state of a specific object feature, conditionally dependent on another feature estimated by a distinct filter. This dependence provides improved target representations, permitting us to segment it out from the background even in nonstationary sequences. Considering that the procedure of the Bayesian filters may be described by a "hypotheses generation-hypotheses correction" strategy, the major novelty of our methodology compared to previous approaches is that the mutual dependence between filters is considered during the feature observation, that is, into the "hypotheses-correction" stage, instead of considering it when generating the hypotheses. This proves to be much more effective in terms of accuracy and reliability. The proposed method is analytically justified and applied to develop a robust tracking system that adapts online and simultaneously the color space where the image points are represented, the color distributions, the contour of the object, and its bounding box. Results with synthetic data and real video sequences demonstrate the robustness and versatility of our method.

Index Terms—Bayesian tracking, multiple cue integration.

1 Introduction

Tracking and figure-ground segmentation of image sequences is a topic of great interest in a wide variety of computer vision applications, extending from video compression to mobile robot navigation. It has been observed that the simultaneous use of redundant and complementary cues for describing the target noticeably improves the performance of the tracking algorithms [2], [4], [7], [8], [14], [18], [24], [25], [26], [27].

Unfortunately, most of these approaches are still not robust enough and suffer from various limitations. For instance, they are usually tailored to specific applications (frequently under controlled environments) and do not represent a general integration methodology that might be generalized to new experimental conditions. Most importantly, most of the works referenced above do not take advantage of the existing relationships between different object cues. For instance, Leichter et al. [15] present an approach where several Bayesian filter algorithms are integrated for tracking tasks. However, in [15], it is assumed that the methods are conditionally independent, that is, each algorithm estimates the state of a target feature based on some measurements that are conditionally independent of the measurements used by the other algorithms. That is, if Bayesian filter $\mathcal{BF}_1$ is based on measurements (observations) $\mathbf{z}_1$ to estimate the state vector $\mathbf{x}_1$ (representing one object feature) and Bayesian filter $\mathcal{BF}_2$ uses measurements $\mathbf{z}_2$ to estimate $\mathbf{x}_2$ (representing another object feature), for each complete state vector of the object $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2\}$, it is assumed that the joint observation model can be separately considered for each feature, that is, $p(\mathbf{z}_1, \mathbf{z}_2|\mathbf{X}) = p(\mathbf{z}_1|\mathbf{x}_1)p(\mathbf{z}_2|\mathbf{x}_2)$.

Nevertheless, this assumption is very restrictive since it assumes that the measurements used to estimate feature $\mathbf{x}_1$ are independent of the measurements used to estimate feature $\mathbf{x}_2$, which often is not satisfied. For instance, a standard method to weigh the samples of a contour particle filter is based on the ratio of the number of pixels inside the contour having an object color versus the number of pixels outside the contour having a background color. This means that the contour feature is not independent of the color feature. In this situation, if $\mathbf{z}_1$ represents the observations for the color feature and $\mathbf{z}_2$ represents the corresponding observations for the contour, the latter will be a function of both $\mathbf{x}_1$ and $\mathbf{z}_1$, that is, $\mathbf{z}_2 = \mathbf{z}_2(\mathbf{x}_1, \mathbf{z}_1)$. Based on the definition of *conditional probability*, it is straightforward to rewrite the previous equation as

$$p(\mathbf{z}_1, \mathbf{z}_2|\mathbf{X}) = p(\mathbf{z}_1|\mathbf{x}_1)p(\mathbf{z}_2|\mathbf{z}_1, \mathbf{x}_1, \mathbf{x}_2),$$

where we have assumed independence of $\mathbf{z}_1$ with respect to $\mathbf{x}_2$ and $\mathbf{z}_2$, that is, $\mathbf{z}_1 \neq \mathbf{z}_1(\mathbf{x}_2, \mathbf{z}_2)$. This formulation allows us to simultaneously adapt both features, performing more robustly than the "independent" case.

In this work, we introduce a probabilistic framework to integrate several object cues, which produces a detailed representation of both the object of interest and the

- F. Moreno-Noguer is with the Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne, BC 307 (Building BC) Station 14, 1015 Lausanne, Switzerland. E-mail: fmorenoguer@gmail.com.
- A. Sanfeliu is with the Institut de Robòtica i Informàtica Industrial, UPC-CSIC, Parc Tecnològic de Barcelona, c/ Llorens i Artigas 4-6, 08028 Barcelona, Spain. E-mail: sanfeliu@iri.upc.edu.
- D. Samaras is with the Image Analysis Lab, Computer Science Department, 2429 Computer Science, Stony Brook University, Stony Brook, NY 11794-4400. E-mail: samaras@cs.sunysb.edu.

Manuscript received 18 July 2006; revised 26 Jan. 2007; accepted 17 May 2007; published online 20 June 2007.
Recommended for acceptance by L. Van Gool.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0531-0706.
Digital Object Identifier no. 10.1109/TPAMI.2007.70727.

background. This enhanced representation allows us to robustly segment the object from the rest of the image in dynamically changing sequences, despite abrupt changes of illumination, cluttered backgrounds, and nonlinear dynamics of the target movement or deformation. The key point of our approach and the main contribution of this paper is the consideration of the cue dependence discussed above and the representation of each one of the features by a different Bayesian filter. As will be explained in the following sections, the update procedure of the Bayesian filters may be decomposed into two stages: an initial "hypothesis/es generation" and a subsequent "hypothesis/es correction." We will show how the consideration of the cue dependence during the latter stage provides better results in terms of reliability and accuracy than considering the dependence during the "hypothesis/es-generation" stage. Furthermore, it is worth noting that the approach presented here is general, both in the sense that it applies to all Bayesian filters and in the sense that it does not restrict the total number of features to integrate, and the complexity of the system does not increase noticeably when integrating additional features. In the rest of the paper, we will refer to the proposed approach as Dependent Bayesian Filter Integration (DBFI).

The proposed framework is theoretically proven and validated in a tracking example of synthetically generated data. The method is subsequently applied to develop a novel and robust tracking system that simultaneously 1) adapts the color space where image points are represented, 2) updates the distributions of the object and background color points, and 3) accommodates the contour of the object. The tracking results obtained in a wide variety of nonstationary environments demonstrate the strength of our method.

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 introduces the mathematical framework. In Section 4, a comprehensible example for one-dimensional cues will be explained which will be used as a benchmark to compare the performance of our method to that of other approaches. The features used in the "real-world" operation of the method and their dynamic models are described in Section 5. Section 6 depicts details about the complete tracking algorithm. Results and conclusions are given in Sections 7 and 8. Part of this work was presented to the computer vision community at the 10th International Conference on Computer Vision (ICCV '05) [19].

2 Related Work

Clearly, this is not the first work to consider multiple cue integration for tracking tasks from a Bayesian point of view. The simplest approach to integrating several cues is to consider an extended state vector including the parameterization of all the cues. For instance, Isard and Blake [9] use a single state vector to integrate appearance and shape in a particle-filter framework. However, as observed by Khan et al. in [12], to proceed by simply augmenting the state space is problematic since it causes an exponential expansion of the region of possible state vector configurations and the tracking becomes extremely complex. Khan et al. [12] suggest using a Rao-Blackwellized particle filter, where some "appearance-related" coefficients are integrated out of the extended state vector. This procedure considerably reduces the size of the search space and, as a consequence, reduces the cost of the tracking as well. Unfortunately, the generalization of this formulation to include additional features is not feasible. Generalization may be achieved by associating a different filter to each feature. Along these lines, Rasmussen and Hager [22] introduced the Joint Probability Data Association Filter (JPDAF) for tracking several targets (note that multiple target tracking can be compared to multiple cue and single target tracking). Nevertheless, the work in [22] estimates each target state independently of other targets, that is, the JPDAF formulation does not permit us to represent the dependence between different state vectors. As we have mentioned in Section 1, a similar approach is presented by Leichter et al. [15]. In particular, their work [15] integrates Kalman and particle filters for tracking tasks, although again assuming independence between filters, which limits the performance of the tracking system.

The partitioned sampling technique introduced by MacCormick et al. [16], [17] and the related approaches of Wu and Huang [28] and Branson and Belongie [3] are probably the works that are closest to the methodology presented in this paper. Partitioned sampling is specifically designed for particle filters and allows the reduction of the curse of dimensionality problem affecting this kind of Bayesian filters. This method applies the "hypotheses generation" and "hypotheses correction" stages separately for different parts of the state vector. The key difference with respect to our method is that, in these partitioned sampling-based approaches, cue dependence is considered during the hypotheses generation stage, whereas we consider it during hypotheses correction. We will show through synthetic experiments that, by proceeding this way, tracking accuracy and reliability are significantly improved.

3 Mathematical Framework

In this section, we will define the mathematical background for the proposed framework. We will start by describing the integration process of conditionally dependent features. Next, we will review the basic concepts for Bayesian filters, in particular, for the particle filters and Kalman filter. Finally, the algorithm we implemented to track an object based on various dependent features will be explained in detail.

3.1 Integration Process

In the general case, let us describe the object being tracked by a set of F features, the configuration of which is specified by the state vectors $\mathbf{x}_1, \ldots, \mathbf{x}_F$, which are sequentially conditionally dependent, that is, feature i depends on feature $i-1$ (later, we will see that the integration of independent cues is straightforward). These features have an associated set of measurements $\mathbf{z}_1, \ldots, \mathbf{z}_F$, where measurement $\mathbf{z}_i$ allows us to update the state vector $\mathbf{x}_i$ of the ith feature. The conditional a posteriori probability $\tilde{p}_1 = p(\mathbf{x}_1|\mathbf{z}_1), \ldots, \tilde{p}_F = p(\mathbf{x}_F|\mathbf{z}_F)$ is estimated using a corresponding Bayesian filter $\mathcal{BF}_1, \ldots, \mathcal{BF}_F$ such as a Kalman filter or a particle filter. For the whole set of variables, we assume that the dependence is only in one direction:

$$\{\mathbf{z}_k = \mathbf{z}_k(\mathbf{z}_i, \mathbf{x}_i), \mathbf{x}_k = \mathbf{x}_k(\mathbf{x}_i, \mathbf{z}_i)\} \Longleftrightarrow i < k. \tag{1}$$

Considering this dependence relationship, we can add extra terms to the a posteriori probability computed for each Bayesian filter. In particular, the expression for the a posteriori probability computed $\mathcal{BF}_i$ will be $p_i = p(\mathbf{x}_i|\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{z}_1, \ldots, \mathbf{z}_i)$. Keeping this in mind and introducing the notation for the *cue-augmented state vector* $\mathbf{X}_{1:k} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ and *cue-augmented measurement vector* $\mathbf{Z}_{1:k} = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$, we proceed to prove that the whole a posteriori probability can be computed sequentially as follows:

$$\begin{aligned} P &= p(\mathbf{X}_{1:F}|\mathbf{Z}_{1:F}) \\ &= p(\mathbf{x}_1|\mathbf{Z}_1)p(\mathbf{x}_2|\mathbf{X}_1, \mathbf{Z}_{1:2}) \cdots p(\mathbf{x}_F|\mathbf{X}_{1:F-1}, \mathbf{Z}_{1:F}) \\ &= p_1 p_2 \cdots p_F. \end{aligned} \tag{2}$$

Proof. We will prove this by induction, applying the definition of *conditional probability* and (1):

- The proof for two features is given by

$$\begin{aligned} p(\mathbf{X}_{1:2}|\mathbf{Z}_{1:2}) &= \frac{p(\mathbf{X}_{1:2}, \mathbf{Z}_{1:2})}{p(\mathbf{Z}_{1:2})} = \frac{p(\mathbf{x}_2|\mathbf{X}_1, \mathbf{Z}_{1:2})p(\mathbf{x}_1, \mathbf{Z}_{1:2})}{p(\mathbf{Z}_{1:2})} \\ &= p(\mathbf{x}_1|\mathbf{Z}_{1:2})p(\mathbf{x}_2|\mathbf{X}_1, \mathbf{Z}_{1:2}). \end{aligned}$$

- For $F-1$ features, we assume that

$$\begin{aligned} p(\mathbf{X}_{1:F-1}|\mathbf{Z}_{1:F-1}) = p(\mathbf{x}_1|\mathbf{Z}_1)p(\mathbf{x}_2|\mathbf{X}_1, \mathbf{Z}_{1:2}) \cdots \\ \cdots p(\mathbf{x}_{F-1}|\mathbf{X}_{1:F-2}, \mathbf{Z}_{1:F-1}). \end{aligned} \tag{3}$$

- The proof for F features is given by

$$\begin{aligned} p(\mathbf{X}_{1:F}|\mathbf{Z}_{1:F}) &= \frac{p(\mathbf{X}_{1:F}, \mathbf{Z}_{1:F})}{p(\mathbf{Z}_{1:F})} \\ &= \frac{p(\mathbf{x}_F|\mathbf{X}_{1:F-1}, \mathbf{Z}_{1:F})p(\mathbf{X}_{1:F-1}|\mathbf{Z}_{1:F})p(\mathbf{Z}_{1:F})}{p(\mathbf{Z}_{1:F})} \\ (\text{by Eq. 3}) &= p(\mathbf{x}_1|\mathbf{Z}_1)p(\mathbf{x}_2|\mathbf{X}_1, \mathbf{Z}_{1:2}) \ldots \\ &\quad p(\mathbf{x}_F|\mathbf{X}_{1:F-1}, \mathbf{Z}_{1:F}). \end{aligned}$$

□

Equation (2) tells us that the whole a posteriori probability density function can be computed sequentially, starting with $\mathcal{BF}_1$ to generate $p(\mathbf{x}_1|\mathbf{Z}_1)$, which is then used to estimate $p(\mathbf{x}_2|\mathbf{X}_1, \mathbf{Z}_{1:2})$ with $\mathcal{BF}_2$, and so on. Note that the inclusion of an extra feature $\mathbf{x}_G$ (with the corresponding measurement vector $\mathbf{z}_G$) independent of the rest is straightforward. We just need to multiply (2) by the posterior $p(\mathbf{x}_G|\mathbf{Z}_G)$.

Until now, we have only considered the fusion of several Bayesian filters from the static point of view. However, in the iterative performance of the method, $\mathcal{BF}_i$ receives as input at iteration t the output PDF of its state vector $\mathbf{x}_i$ at the iteration $t-1$. We write the time-expanded version of the PDF for $\mathcal{BF}_i$ as

$$p_i^t = p\left(\mathbf{x}_i^t|\mathbf{X}_{1:i-1}^t, \mathbf{Z}_{1:i}^t, p_i^{t-1}\right).$$

The expression for the complete PDF from (2) may be expanded as

$$\begin{aligned} P^t &= p(\mathbf{x}_1^t, \ldots, \mathbf{x}_F^t|\mathbf{z}_1^t, \ldots, \mathbf{z}_F^t) \\ &= p(\mathbf{x}_1^t|\mathbf{Z}_1^t, p_1^{t-1}) \cdots p\left(\mathbf{x}_F^t|\mathbf{X}_{1:F-1}^t, \mathbf{Z}_{1:F}^t, p_F^{t-1}\right) \\ &= p_1^t p_2^t \cdots p_F^t. \end{aligned} \tag{4}$$

This equation represents the basis for the DBFI method proposed here and, in Section 3.3, we will explain the algorithm we implemented to approximate it. We next review the particle-filter and Kalman-filter procedures, explaining them in terms of Bayesian filtering. The intention of the following section is to introduce the notation that will be used in the rest of the paper.

3.2 Bayesian Filtering

Let us now briefly describe how the kth Bayesian filter $\mathcal{BF}_k$ computes the posterior $p(\mathbf{x}_k^t|\mathbf{Z}_k^{t_0:t})$. For simplicity, here, we assume that the measurements are obtained based only on observations of feature $\mathbf{x}_k$. In the next section, we will consider the dependence of feature $\mathbf{x}_k$ with respect to features $\mathbf{x}_i$, $\forall i < k$.

The formulation of the tracking problem in terms of a Bayes filter consists of recursively updating the posterior distribution $p(\mathbf{x}_k^t|\mathbf{Z}_k^{t_0:t})$ according to

$$p(\mathbf{x}_k^t|\mathbf{Z}_k^{t_0:t}) \propto p(\mathbf{z}_k^t|\mathbf{x}_k^t) \int_{\mathbf{x}_k^{t-1}} p(\mathbf{x}_k^t|\mathbf{x}_k^{t-1})p(\mathbf{x}_k^{t-1}|\mathbf{Z}_k^{t_0:t-1})d\mathbf{x}_k^{t-1}, \tag{5}$$

where $p(\mathbf{z}_k^t|\mathbf{x}_k^t)$ is the *observation (or measurement) model*, that is, the probability of making the observation $\mathbf{z}_k^t$ given that the target state at time t is $\mathbf{x}_k^t$. The *dynamic model* $p(\mathbf{x}_k^t|\mathbf{x}_k^{t-1})$ predicts the state $\mathbf{x}_k^t$ at time t given the previous state $\mathbf{x}_k^{t-1}$.

Although $\mathcal{BF}_k$ may take different forms (Kalman filter, extended Kalman filter (EKF), particle filter, Rao-Blackwellized particle filter, and so forth), the Bayes filter equation (5) is updated in all of the cases through a "hypotheses generation-hypotheses correction" scheme. Initially, based on the *dynamic model* $p(\mathbf{x}_k^t|\mathbf{x}_k^{t-1})$ and the a posteriori distribution at the previous time step $p(\mathbf{x}_k^{t-1}|\mathbf{Z}_k^{t_0:t-1})$, the state of the target is predicted as follows:

$$p(\mathbf{x}_k^t|\mathbf{Z}_k^{t_0:t-1}) = \int_{\mathbf{x}_k^{t-1}} p(\mathbf{x}_k^t|\mathbf{x}_k^{t-1})p(\mathbf{x}_k^{t-1}|\mathbf{Z}_k^{t_0:t-1})d\mathbf{x}_k^{t-1}. \tag{6}$$

This likelihood is subsequently corrected by the *observation model* $p(\mathbf{z}_k^t|\mathbf{x}_k^t)$:

$$p(\mathbf{x}_k^t|\mathbf{Z}_k^{t_0:t}) = \alpha^t p(\mathbf{z}_k^t|\mathbf{x}_k^t)p(\mathbf{x}_k^t|\mathbf{Z}_k^{t_0:t-1}), \tag{7}$$

where α^t is a normalizing constant.

Next, we give an overview of two different implementations of Bayesian filters, namely, the Kalman filter and particle filters, which are representative examples for the continuous and discrete methodologies to approximate the posterior densities (and which will be used to design the "real-world" tracking algorithm).

3.2.1 Kalman Filter

In the particular case where the observation density is assumed to be Gaussian and the dynamics are assumed to be linear with additive Gaussian noise, (6) and (7) result in the Kalman filter [1], [11]. The expressions for the densities of the dynamic model and observation model are

$$\begin{aligned} p(\mathbf{x}_k^t|\mathbf{x}_k^{t-1}) &= \mathcal{N}(\mathbf{H}_k^t\mathbf{x}_k^{t-1};\boldsymbol{\Sigma}_{k,h}^t) \\ p(\mathbf{z}_k^t|\mathbf{x}_k^t) &= \mathcal{N}(\mathbf{M}_k^t\mathbf{x}_k^t;\boldsymbol{\Sigma}_{k,m}^t), \end{aligned} \tag{8}$$

where the $\mathbf{H}_k^t$ and $\mathbf{M}_k^t$ matrices are the deterministic components of the models and $\boldsymbol{\Sigma}_{k,h}^t$ and $\boldsymbol{\Sigma}_{k,m}^t$ are the covariance matrices of the white and normally distributed noise assumed for the models.

We now plug these expressions into the Bayes-filter equations, (6) and (7), which can be analytically solved. The hypothesis-generation stage provides the following Gaussian likelihood:

$$\begin{aligned} p(\mathbf{x}_k^t|\mathbf{Z}_k^{t_0:t-1}) &= \mathcal{N}(\mathbf{x}_{k,-}^t;\boldsymbol{\Sigma}_{k,-}^t) \\ &= \mathcal{N}\left(\mathbf{H}^t\mathbf{x}_k^{t-1};\boldsymbol{\Sigma}_{k,h}^t + \mathbf{H}_k^t\boldsymbol{\Sigma}_k^{t-1}(\mathbf{H}_k^t)^T\right). \end{aligned}$$

Similarly, the hypothesis correction stage generates the following Gaussian posterior density:

$$\begin{aligned} p(\mathbf{x}_k^t|\mathbf{Z}_k^{t_0:t}) &= \mathcal{N}(\mathbf{x}_k^t,\boldsymbol{\Sigma}_k^t) \\ &= \mathcal{N}(\mathbf{x}_{k,-}^t + \mathbf{K}^t[\mathbf{z}_k^t - \mathbf{M}_k^t\mathbf{x}_{k,-}^t],[\boldsymbol{\Sigma}_{k,-}^t - \mathbf{K}^t\mathbf{M}_k^t\boldsymbol{\Sigma}_{k,-}^t]), \end{aligned}$$

where matrix $\mathbf{K}^t$ is the Kalman gain.

3.2.2 Particle Filter

In noisy scenes with cluttered backgrounds, observations usually have non-Gaussian multimodal distributions and the models estimated using the Kalman filter formulation are no longer valid. Particle filtering [5] offers an approximate solution for these cases by approximating the posterior $p(\mathbf{x}_k^{t-1}|\mathbf{Z}_k^{t_0:t-1})$ as a set of weighted samples $\{\mathbf{s}_{kj}^{t-1},\pi_{kj}^{t-1}\}_{j=1}^{n_k}$, where π_{kj}^{t-1} is the weight associated to particle $\mathbf{s}_{kj}^{t-1}$. Therefore, the Bayes-filter equation (5) is now represented by

$$p(\mathbf{x}_k^t|\mathbf{Z}_k^{t_0:t}) \approx p(\mathbf{z}_k^t|\mathbf{x}_k^t)\sum_{j=1}^{n_k}\pi_{kj}^{t-1}p(\mathbf{x}_k^t|\mathbf{s}_{kj}^{t-1}),$$

which is recursively approximated using a "hypotheses generation-hypotheses correction" strategy. Note that, now, the dynamic model is represented by the distribution $p(\mathbf{x}_k^t|\mathbf{s}_{kj}^{t-1})$.

During the hypotheses generation stage, a set of n_k samples $\mathbf{s}_{kj}^t$ is drawn from the distribution:

$$\left\{\mathbf{s}_{kj}^t\right\}_{j=1}^{n_k} \sim \sum_{j=1}^{n_k}\pi_{kj}^{t-1}p(\mathbf{x}_k^t|\mathbf{s}_{kj}^{t-1}).$$

For implementation purposes, this stage is usually split into two subprocesses. Initially, the set $\{\mathbf{s}_{kj}^{t-1},\pi_{kj}^{t-1}\}_{j=1}^{n_k}$ is resampled (sampling with replacement) according to the weights π_{kj}^{t-1}. We obtain the new set $\{\tilde{\mathbf{s}}_{kj}^{t-1},\pi_{kj}^{t-1}\}_{j=1}^{n_k}$, which is propagated to the set $\{\mathbf{s}_{kj}^t\}_{j=1}^{n_k}$ based on the probabilistic dynamic model.

Finally, based on the observation function $p(\mathbf{z}_k^t|\mathbf{x}_k^t)$, the set of samples $\{\mathbf{s}_{kj}^t\}_{j=1}^{n_k}$ is weighted:

$$\pi_{kj}^t = p(\mathbf{z}_k^t|\mathbf{x}_k^t = \mathbf{s}_{kj}^t).$$

The set $\{\mathbf{s}_{kj}^t,\pi_{kj}^t\}_{j=1}^{n_k}$ approximates the posterior $p(\mathbf{x}_k^t|\mathbf{Z}_k^{t_0:t})$.

3.3 Approximation of the Dependent Bayesian Filter Integration Model

We now describe the algorithm used to approximate the DBFI model of (4). For ease of explanation, let us assume that our target is represented by a set of k features estimated by particle filters and we are given the posteriors $p_1^t,\ldots,p_{k-1}^t$ at time t for features $1\ldots k-1$. Our goal is to compute p_k^t for the feature k for which we know its posterior p_k^{t-1} at the previous time step, approximated by a set of n_k weighted samples $\{\mathbf{s}_{kj}^{t-1},\pi_{kj}^{t-1}\}_{j=1}^{n_k}$. Similarly, the distribution $P_{1:k-1}^t = p_1^tp_2^t\ldots p_{k-1}^t$ is approximated by a set $\{\mathbf{s}_{k-1,j}^t,\pi_{k-1,j}^t\}_{j=1}^{n_{k-1}}$ of weighted samples. Then, the process to update p_k^{t-1} may be summarized as follows:

1. *Resampling the posterior for features* $1\ldots k-1$. The set $\{\mathbf{s}_{k-1,j}^t,\pi_{k-1,j}^t\}_{j=1}^{n_{k-1}}$ is sampled with replacement n_k times in such a way that the probability for each particle $\mathbf{s}_{k-1,j}^t$ of being selected is determined by its weight $\pi_{k-1,j}^t$. Note that, by doing this resampling, we are selecting a subset $\{\mathbf{s}_{k-1,j}^*\}_{j=1}^{n_k}$, containing the best hypotheses of $P_{1:k-1}^t$.
2. *Hypotheses generation for feature k*. The initial operation performed over the posterior p_k^{t-1} of feature k is based on the usual "hypotheses generation" step performed on particle filters. That is, set $\{\mathbf{s}_{kj}^{t-1},\pi_{kj}^{t-1}\}_{j=1}^{n_k}$ is resampled according to the weights π_{kj}^{t-1} and propagated to the set $\{\mathbf{s}_{kj}^t\}_{j=1}^{n_k}$ based on the dynamical model associated to feature k.
3. *Hypotheses correction for feature k*. The set of particles $\{\mathbf{s}_{kj}^t\}_{j=1}^{n_k}$ then needs to be corrected according to some observation model. One important property of our model is that this correction is performed according to the posterior distribution of the features $1,\ldots,k-1$ previously considered in the algorithm. For this purpose, we design a weighting function that evaluates each sample for feature k assuming that the state of features $1,\ldots,k-1$ is described by $P_{1:k-1}^t$. For the most accurate approximation of (4), each sample $\mathbf{s}_{kj}^t$ should be evaluated according to the complete posterior distribution $P_{1:k-1}^t$. However, this procedure would be extremely computationally expensive since each sample of feature k should be evaluated with all the samples of the previous features.

To reduce the computational load, we found it adequate to evaluate each particle $\mathbf{s}_{kj}^t$ using a single element $\mathbf{s}_{k-1,j}^*$ from the previously resampled set $\{\mathbf{s}_{k-1,j}^t,\pi_{k-1,j}^t\}_{j=1}^{n_{k-1}}$. Note that $P_{1:k-1}^t$ is now approximated by the set $\{\mathbf{s}_{k-1,j}^*\}_{j=1}^{n_k}$ containing repeated copies of those samples of features $1,\ldots,k-1$ having larger weights (high probability), whereas those samples having a low probability may not be represented at all. As we will show in the following section, apart from a significant reduction in the computational load, this procedure permits concentration of the samples around the more likely regions in the configuration

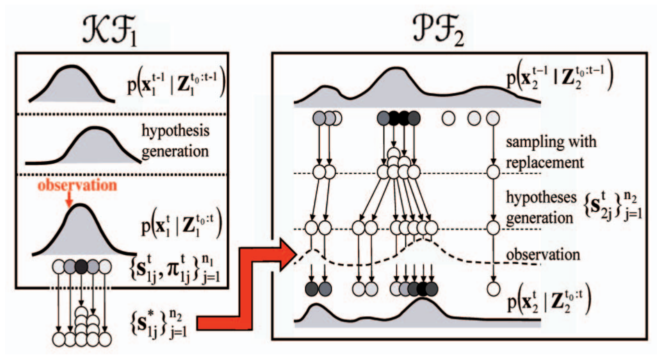
$\mathcal{KF}_1$
$\mathcal{PF}_2$
$p(x_1^{t-1}|Z_1^{t_0:t-1})$
hypothesis generation
observation
$p(x_1^t|Z_1^{t_0:t})$
$\{s_{1j}^t, \pi_{1j}^t\}_{j=1}^{n_1}$
$\{s_{1j}^*\}_{j=1}^{n_2}$
$p(x_2^{t-1}|Z_2^{t_0:t-1})$
sampling with replacement
hypotheses generation $\{s_{2j}^t\}_{j=1}^{n_2}$
observation
$p(x_2^t|Z_2^{t_0:t})$

Fig. 1. **Introducing cue dependence into the observation model.** Example of how cue dependence is handled in the proposed DBFI framework in a case dealing with two features, one estimated by a Kalman filter and the other estimated by a particle filter. The posterior of feature $\mathbf{x}_1$, computed by $\mathcal{KF}_1$, is represented by a set of weighted samples $\{\mathbf{s}_{1j}^t, \pi_{1j}^t\}_{j=1}^{n_1}$. These particles are resampled n_2 times (according to their weights) in order to obtain the set $\{\mathbf{s}_{1j}^*, \pi_{1j}^*\}_{j=1}^{n_2}$. Finally, each sample $\{\mathbf{s}_{2j}^t\}_{j=1}^{n_2}$ of feature $\mathbf{x}_2$ is weighted according to the configuration of the corresponding sample $\mathbf{s}_{1j}^*$.

space, avoiding an unnecessary waste of low probability samples.[1]

The integration of features estimated by filters yielding a continuous PDF (such as a Kalman filter or EKF) is immediate. For example, when the information of a Kalman filter feeds into a particle filter, the Gaussian posterior of the Kalman filter is discretized and represented by a set of samples and weights. On the other hand, when a particle filter feeds into a Kalman filter, the posterior of the particle filter represented by a set of weighted samples is approximated by its mean and covariance matrix.

Fig. 1 shows an example of how cue dependence is handled in a case with two features: $\mathbf{x}_1$ estimated by a Kalman filter $\mathcal{KF}_1$ and $\mathbf{x}_2$ estimated by a particle filter $\mathcal{PF}_2$. For this example, $\mathbf{x}_2$ depends on feature $\mathbf{x}_1$. During the observation phase of $\mathcal{PF}_2$, the multiple hypotheses $\{\mathbf{s}_{2j}\}_{j=1}^{n_2}$ need to be weighted according to some external measurement. This measurement will be based on the posterior of feature $\mathbf{x}_1$ estimated by $\mathcal{KF}_1$. For this purpose, the distribution $p(\mathbf{x}_1^t|\mathbf{Z}_1^{t_0:t})$ is discretized into n_1 weighted particles $\{\mathbf{s}_{1j}^t, \pi_{1j}^t\}_{j=1}^{n_1}$. Subsequently, this set is resampled with replacement n_2 times and a set $\{\mathbf{s}_{1j}^*, \pi_{1j}^*\}_{j=1}^{n_2}$ is obtained. Finally, each sample $\mathbf{s}_{2j}^t$ of the feature $\mathbf{x}_2$ is weighted using the configuration of feature $\mathbf{x}_1$ represented by $\mathbf{s}_{1j}^*$.

Observe in Fig. 1 that samples $\{\mathbf{s}_{1j}^t\}$ with higher weights have a higher chance of being selected several times when evaluating the hypotheses $\{\mathbf{s}_{2j}^t\}$, allowing the more likely samples of feature $\mathbf{x}_1$ with the more likely samples of feature $\mathbf{x}_2$ to be grouped together. Also, it is important to note that not all the features need to be approximated by the same number of samples. In the example just presented, $\mathbf{x}_1$ is estimated by $n_1 = 5$ samples, whereas $\mathbf{x}_2$ is estimated by $n_2 = 10$ samples. This is an important advantage of the proposed framework, especially when dealing with particle filters, since it permits adaptation of the number of necessary samples to estimate each feature in light of its particular requirements.

1. If one of the features is not observable, all of its configurations in the vector space will be equally probable. This means that the posterior distribution of the feature will be represented by a constant PDF.

To illustrate all the mathematical foundations, in the next section, we will apply this method to a simulated case, with only two 1D particle filters.

4 Dependent Object Features in 1D

Let us assume that we want to track a single point that changes its position and color. Both features lie on a 1D space, that is, the point is moving between the $[-1, 1]$ coordinates, and the color is also represented by a single value in the $[0, 1]$ interval. The movement is simulated with a random dynamic model (centered in μ_{pos} and scaled by α_{pos}). We also simulate an observation model, adding Gaussian noise to the simulated position:

$$\text{pos}^t = (\text{pos}^{t-1} - \mu_{\text{pos}})\alpha_{\text{pos}} + \mathcal{N}(\mu_{\text{noise,pos}}, \sigma_{\text{noise,pos}})$$
$$\text{obs\_pos}^t = \text{pos}^t + \mathcal{N}(0, \sigma_{\text{noise,obs\_pos}}).$$

Similar equations generate the models for color change (col^t) and its observation (obs_col^t).

The state of each feature will be estimated by particle filters. We will use $\mathcal{PF}_1$ to track the color, with $\mathbf{x}_1$ representing the color state vector and $\mathcal{PF}_2$ and $\mathbf{x}_2$ the corresponding particle filter and state vector assigned to the position.

At the starting point of iteration t, $\mathcal{PF}_1$ receives as its input p_1^{t-1}, the PDF of the color at time $t-1$, approximated with n_1 weighted samples $\{\mathbf{s}_{1j}^{t-1}, \pi_{1j}^{t-1}\}_{j=1}^{n_1}$. This set is resampled and propagated according to a random dynamic model of Gaussian noise, that is, $\mathbf{s}_{1j}^t = \tilde{\mathbf{s}}_{1j}^{t-1} + \mathcal{N}(0, \sigma_{\text{dyn,col}})$, where $\tilde{\mathbf{s}}_{1j}^{t-1}$ are the resampled particles.

Each one of these propagated samples is weighted based on its proximity to the color observation: $\pi_{1j}^t \sim e^{-(\|\mathbf{s}_{1j}^t - \text{obs\_col}^t\|)}$.

The set $\{\mathbf{s}_{1j}^t, \pi_{1j}^t\}_{j=1}^{n_1}$ is the output of $\mathcal{PF}_1$ and approximates the distribution p_1^t. This PDF, jointly with p_2^{t-1}, feeds into $\mathcal{PF}_2$, the particle filter responsible for estimating the position of the point. As in the previous particle filter, p_2^{t-1} is approximated by a set of n_2 samples and weights $\{\mathbf{s}_{2j}^{t-1}, \pi_{2j}^{t-1}\}_{j=1}^{n_2}$, which are resampled and propagated using a random Gaussian dynamic model, that is, $\mathbf{s}_{2j}^t = \tilde{\mathbf{s}}_{2j}^{t-1} + \mathcal{N}(0, \sigma_{\text{dyn,pos}})$.

We then evaluate the several hypothesized target positions based on the color feature. With this purpose, the set $\{\mathbf{s}_{1j}^t, \pi_{1j}^t\}_{j=1}^{n_1}$ is initially sampled with replacement n_2 times, where, for each particle, the probability of being selected is determined by its weight π_{1j}^t. This sampling procedure yields a subset $\{\mathbf{s}_{1j}^*, \pi_{1j}^*\}_{j=1}^{n_2}$ containing the best hypotheses of feature $\mathbf{x}_1$. Subsequently, each position sample $\mathbf{s}_{2j}^t$ is associated to a color sample $\mathbf{s}_{1j}^*$. The samples $\mathbf{s}_{2j}^t$ are weighted based on the function $\pi_{2j}^t \sim e^{-(\|\mathbf{s}_{1j}^* - \text{obs\_col}^t\| + \|\mathbf{s}_{2j}^t - \text{obs\_pos}^t\|)}$, which considers both position and color. The set $\{\mathbf{s}_{2j}^t, \pi_{2j}^t\}_{j=1}^{n_2}$ approximates p_2^t.

Finally, the complete a posteriori probability of the system at time t may be computed by

$$P^t = p(\mathbf{x}_1^t, \mathbf{x}_2^t | \mathbf{z}_1^t, \mathbf{z}_2^t) = p_1^t p_2^t.$$

4.1 Comparison with Other Approaches

The simple example just presented will be considered as a benchmark to compare the efficiency of the DBFI method proposed in this paper with that of previous approaches, specifically with the conventional Condensation algorithm [9] assuming independent cues and with the partitioned sampling algorithm [16], [17] assuming the dependence in the propagation stage. The comparison will be performed in terms of the tracking accuracy (distance between the estimated position and color features and the true values) and in terms of the *survival diagnostic* [17]. The survival diagnostic $\mathcal{D}$ for a particle set $\{\mathbf{s}_i, \pi_i\}_{i=1}^n$ is defined as

$$\mathcal{D} = \left(\sum_{i=1}^{n} \pi_i^2 \right)^{-1}.$$

This random variable may be interpreted as the number of particles that would survive a resampling operation and, therefore, it is an indicator of whether the tracking performance is reliable or not.[2] A low value of $\mathcal{D}$ means that the tracker may lose the target. For instance, if $\pi_1 = 1$ and $\pi_2 = \pi_3 = \ldots = \pi_n = 0$, then $\mathcal{D} = 1$. In these circumstances, only one particle might survive the resampling and tracking would probably fail. On the other hand, if all of the particles have the same weight, $\pi_1 = \pi_2 = \ldots = \pi_n = 1/n$ results in $\mathcal{D} = n$. This indicates that all the n particles would survive an ideal resampling and the tracker has significant chances of succeeding. With this clarification, we proceed to study the performance of different algorithms in the tracking problem proposed in this section.

In the first experiment, the problem has been addressed by the conventional Condensation algorithm, assuming that cues are independent. $\mathbf{x}_1$ and $\mathbf{x}_2$ are represented into a common state vector and the hypotheses generation and correction stages are applied simultaneously to both features. Since the dynamic model of a specific feature has no information about the state of the other feature, the samples are spread over a wide area of the state space and, as a consequence, only a few particles will be closely located to the true state. Fig. 2a shows the a posteriori distribution obtained in one iteration of the algorithm. The dots represent the different samples (in the "color-position" space) and the crosses are the true (black) and observed (blue) values. The gray level of the particles is proportional to their likelihood (darker levels are more probable particles). Observe that only a small number of particles have a large weight. As a consequence, the survival diagnostic for this approach will have low values.

A better approach may be obtained through the partitioned sampling algorithm. In this case, the dynamics and measurements are not applied simultaneously but are partitioned into two components. First, the dynamics are applied in the $\mathbf{x}_1$ direction and, therefore, the particles are rearranged so that they concentrate around the color observation (by a process called *weighted resampling* [16], which keeps the distribution unchanged). This arrangement enhances the estimation by concentrating more particles around the true state. Note in Fig. 2b this effect on the posterior distribution. Although particles are spread in the $\mathbf{x}_2$ direction, their variability along the $\mathbf{x}_1$ direction is highly reduced. As a result, the number of particles having a large weight is considerably bigger than when using the conventional Condensation.

2. Comparing the performance of particle filters is a difficult task. Although the survival diagnostic used here is not an irrefutable proof that the tracker will or will not get lost, it gives an idea of its efficiency and reliability.

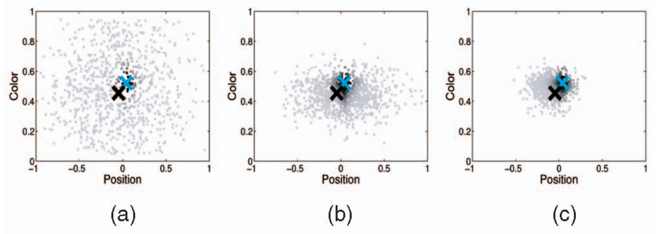

Fig. 2. **A posteriori probability distributions for different particle filter-based algorithms.** Comparison of the posterior obtained for three algorithms in the tracking example presented in Section 4 corresponding to a point moving in the "color-position" space. The results are for a particular iteration and show how the filters approximate the true value (black cross) based on a set of weighted particles (gray level dots). The gray level is proportional to the probability of the sample in such a way that darker gray levels indicate more likely samples. Since the true value is only ideally available, the correction of the hypotheses is done based on the observation (blue cross), which we have simulated to be the true value plus a Gaussian noise. The three experiments use the same number of particles $(n = 1,000)$ and the same dynamic models. However, note that the DBFI approach proposed in this paper is the method that concentrates a maximum number of samples around the true value. (a) Condensation. (b) Partitioned sampling. (c) DFBI.

It is worth noting that, in the partitioned sampling technique, particles are propagated in the direction $\mathbf{x}_2$ according to the likelihood of the samples of feature $\mathbf{x}_1$. Thus, the best hypotheses of feature $\mathbf{x}_1$ have more chances of being propagated in the direction $\mathbf{x}_2$. Although this approach outperforms the conventional Condensation algorithm, it still has a limitation in that the best samples of feature $\mathbf{x}_1$ do not need to be the best samples of feature $\mathbf{x}_2$. Therefore, the common association of the best samples of feature $\mathbf{x}_1$ with the best samples of feature $\mathbf{x}_2$ is not guaranteed.

This issue is addressed by the DBFI algorithm proposed in the paper. The key difference with respect to the previous approaches is that we assume a different state vector for each feature and the hypotheses generation and correction stages are also applied separately. In particular, the propagation of the particles for feature $\mathbf{x}_i$ is performed according to the particles resampling the feature's own probability distribution in the previous time step $p(\mathbf{x}_i^{t-1}|\mathbf{Z}_i^{t-1})$ and not according to the particles that better approximate another feature, avoiding the aforementioned issue in partitioned sampling. In Fig. 2c, we see that, proceeding this way, the samples are much more concentrated around the true value than they were for the other approaches, which noticeably improves the survival diagnostic.

Furthermore, although partitioned sampling considers the feature dependence during the hypotheses generation stage, we consider it in the hypotheses correction phase, where the posterior of a specific feature is used to weigh the samples of another feature. This permits us to update all of the features representing the target in the same iteration.

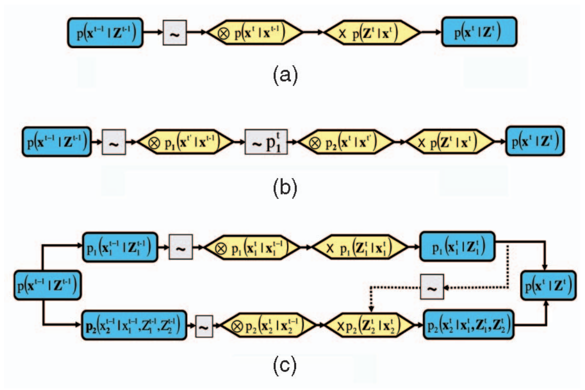

Fig. 3. **Whole process diagrams of the conventional Condensation, the partitioned sampling, and the DBFI algorithms.** The symbology used in these diagrams is adapted from [17]: $\sim$ denotes the resampling operation, $\sim p_1^t$ indicates a weighted resampling operation with respect to the importance function p_1^t, $\otimes$ represents a convolution operation with the dynamics, and $\times$ is the multiplication by the observation density (see [17] for details). (a) Conventional condensation $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$. (b) Partitioned sampling $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$. (c) DBFI.

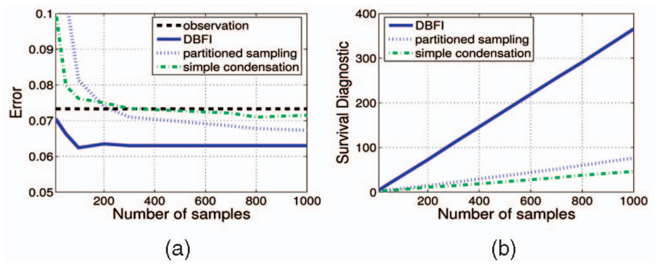

Fig. 4. **Tracking results obtained for the conventional Condensation, partitioned sampling and the proposed DBFI method.** Analysis of the three algorithms when applied to the tracking example explained in Section 4, which was a 20-iteration sequence. The analysis is done (a) in terms of the error in the tracking (distance between the true state and the state estimated by the algorithm) and (b) in terms of the survival rate. In both cases, the experiments have been realized for different numbers of samples and, for each specific number of samples, 25 repetitions of the simulation have been performed. The results we show correspond to the mean of these 25 repetitions, with 20 iterations each. Observe that the results agree with the a posteriori distributions plotted in Fig. 2 as DBFI outperforms both the Condensation and the partitioned sampling algorithms.

Following the diagram symbology used in [17] to describe particle filter processes (which explains the particle filter operation through convolution and multiplication of PDFs), Fig. 3 depicts one time step of the conventional Condensation, the partitioned sampling, and the DBFI algorithms. These diagrams clearly reflect the difference between the algorithms.

The plots in Fig. 4 show the tracking results obtained for the three algorithms compared in this section. In Fig. 4a, the algorithms are compared in terms of the tracking error, where the error is computed as the distance between the filter estimate and the true value. For instance, given a posterior approximated by the set $\{\mathbf{s}_j, \pi_j\}_{j=1}^n$ and the true state of the tracked point given by $\mathbf{x}_{true}$, the value of the error is

$$\mathcal{E}(n) = \|E(\mathbf{x}) - \mathbf{x}_{true}\|,$$

where $E(\mathbf{x})$ is the expected value approximated by the filter, that is, $E(\mathbf{x}) = \sum_{j=1}^n \mathbf{s}_j \pi_j$, and $\|\cdot\|$ refers to the euclidean norm. Observe that the error produced using DBFI is clearly smaller than the one produced by the other algorithms.

Analyzing the survival diagnostic for the same experiments, we reach similar conclusions. Fig. 4b shows that the largest survival rates and, hence, the most reliable tracking results are obtained when using the integration technique presented in this paper.

A final remark for this section regarding the number of particles necessary to achieve a desired level of performance: It is well known that the *curse of dimensionality* is one of the main problems affecting particle filters, that is, when the dimensionality of the state space increases, the number of required samples increases exponentially [12], [16], [28]. Intuitively, the number of samples is proportional to the volume of the search space. For instance, if a 1D space is sampled by n particles, the same sampling density in a two-dimensional space will require n^2 particles and so on. Nevertheless, in the proposed method, the high-dimensional state vectors are decomposed into various small state vectors and the sampling is particularized for each low-dimensional configuration space. The final number of required particles corresponds to the sum of the particles used in each of these low-dimensional spaces. For example, if a two-dimensional state vector can be split into two one-dimensional state vectors, the number of samples may be reduced from n^2 (required in the two-dimensional configuration space) to $2n$ (required in the two one-dimensional spaces). Furthermore, as we have previously pointed out, the number of samples may be adapted for the particular requirements of each particular component of the whole state vector.

5 Features Used for Robust Tracking

In the preceding sections, the integration framework has been presented from a general point of view and applied to a simple example involving 1D features, which has allowed us to highlight the important properties of the method and compare it with other approaches. The rest of the paper will describe a particular application of the proposed framework for designing a tracking system able to work in real and dynamic environments. The target is going to be represented by both appearance (normal to the Fisher plane [20] and color distribution of the object) and geometric attributes (contour and bounding box). In the following sections, we will describe these features, as well as their parameterizations and dynamic models.

5.1 Object Bounding Box

The bounding box of the object is simply a rectangular shape that gives a rough estimate about the target position. It will be parameterized by

$$\mathbf{x}_1 = [u_1, v_1, a_1, b_1, \theta_1]^T \in \mathbb{R}_{5\times 1},$$

where (u_1, v_1) are the coordinates of the center, a_1 and b_1 are the lengths of the sides of the rectangle, and θ_1 is the angle between a_1 and the horizontal axis.

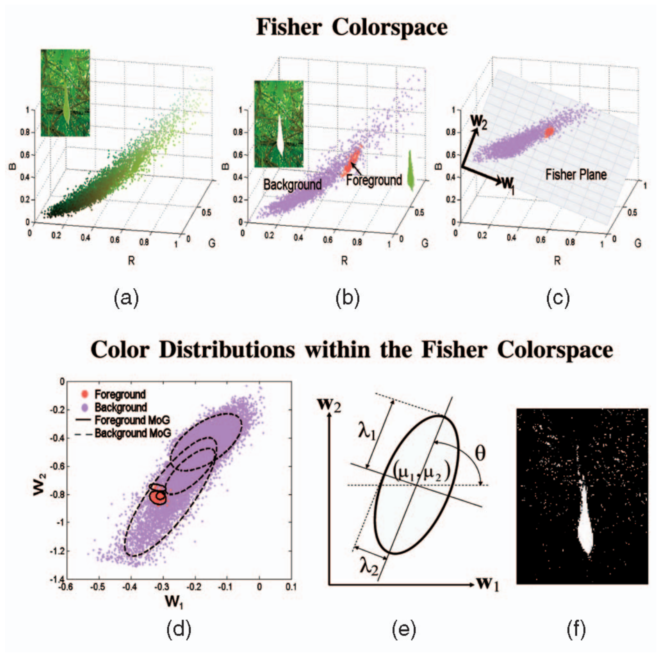

Fig. 5. **Color model.** (a) Representation of all image points in the RGB color space. In the upper left corner of the figure, the original image is shown. (b) Manual classification of image points into foreground ($\mathcal{O}$) and background ($\mathcal{B}$) classes. The foreground (target to track) is the leaf appearing in the center of the image. (c) Projection of $\mathcal{O}$ and $\mathcal{B}$ on the Fisher plane. The Fisher plane is determined from the training points. This plane maximizes the separation of the projected classes while keeping a low variance. (d) Mixture of Gaussians (MoG) components of $\mathcal{O}$ and $\mathcal{B}$ in the Fisher color space. (e) Detail of the parameters used to represent a single Gaussian component. (f) $p(\mathcal{O}|\mathbf{c}^{Fisher})$. Brighter points are more likely pixels.

5.2 Normal to the Fisher Plane

In [20], the concept of Fisher color space was introduced and it was suggested that, for tracking purposes, the best color space is the one that maximizes the distance between the object and background color points. Let the sets $\mathbf{C}_{\mathcal{O}}^{RGB} = \{\mathbf{c}_{\mathcal{O},i}^{RGB}\}_{i=1}^{n_{\mathcal{O}}}$ and $\mathbf{C}_{\mathcal{B}}^{RGB} = \{\mathbf{c}_{\mathcal{B},j}^{RGB}\}_{j=1}^{n_{\mathcal{B}}}$ be the color points of the object and background, respectively, represented in the 3D RGB color space. We define as the optimal color space the one resulting from the projection of the RGB color points onto the plane $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2]^T \in \mathbb{R}_{2\times3}$ (Fisher plane), computed by applying the nonparametric Linear Discriminant Analysis technique [6] over the sets $\mathbf{C}_{\mathcal{O}}^{RGB}$ and $\mathbf{C}_{\mathcal{B}}^{RGB}$. An RGB color point $\mathbf{c}^{RGB}$ is transformed into the 2D Fisher color space by $\mathbf{c}^{Fisher} = \mathbf{W}\mathbf{c}^{RGB}$ (see Fig. 5, top).

The Fisher plane will be parameterized by its normal direction:

$$\mathbf{x}_2 = \frac{\mathbf{w}_1 \times \mathbf{w}_2}{\|\mathbf{w}_1 \times \mathbf{w}_2\|} \in \mathbb{R}_{3\times1}.$$

5.3 Color Distribution of the Target and Background

In order to represent the color distribution of the foreground and background in the Fisher color space, we use a Mixture of Gaussians (MoG) model. With this model, the conditional probability for a pixel $\mathbf{c}^{Fisher}$ belonging to a multicolored object $\mathcal{O}$ may be expressed as a sum of $m_{\mathcal{O}}$ Gaussian components:

$$p(\mathbf{c}^{Fisher}|\mathcal{O}) = \sum_{j=1}^{m_{\mathcal{O}}} p(\mathbf{c}^{Fisher}|j)P(\mathcal{O}, j),$$

where $P(\mathcal{O}, j)$ corresponds to the a priori probability that pixel $\mathbf{c}^{Fisher}$ was generated by the jth Gaussian component of the object color distribution. The likelihood $p(\mathbf{c}^{Fisher}|j)$ is a Gaussian distribution.

Similarly, the background color will be represented by a mixture of $m_{\mathcal{B}}$ Gaussians. Given the foreground ($\mathcal{O}$) and background ($\mathcal{B}$) classes, we will use the Bayes rule to compute the a posteriori probability that a pixel $\mathbf{c}^{Fisher}$ belongs to the object $\mathcal{O}$ (Fig. 5, bottom):

$$p(\mathcal{O}|\mathbf{c}^{Fisher}) = \frac{p(\mathbf{c}^{Fisher}|\mathcal{O})P(\mathcal{O})}{p(\mathbf{c}^{Fisher}|\mathcal{O})P(\mathcal{O}) + p(\mathbf{c}^{Fisher}|\mathcal{B})P(\mathcal{B})}, \tag{9}$$

where $P(\mathcal{O})$ and $P(\mathcal{B})$ represent the a priori probabilities of $\mathcal{O}$ and $\mathcal{B}$, respectively.

The configurations of the MoGs for $\mathcal{O}$ and $\mathcal{B}$ will be parameterized by the vector

$$\mathbf{g}_\varepsilon = [\mathbf{p}_\varepsilon, \boldsymbol{\mu}_\varepsilon, \boldsymbol{\lambda}_\varepsilon, \boldsymbol{\theta}_\varepsilon]^T \in \mathbb{R}_{6m_\varepsilon\times1}, \tag{10}$$

where $\varepsilon = \{\mathcal{O}, \mathcal{B}\}$, m_ε is the number of Gaussian components for the class ε, $\mathbf{p}_\varepsilon \in \mathbb{R}_{m_\varepsilon\times1}$ contains the priors for each Gaussian component, $\boldsymbol{\mu}_\varepsilon \in \mathbb{R}_{2m_\varepsilon\times1}$ contains the centroids, $\boldsymbol{\lambda}_\varepsilon \in \mathbb{R}_{2m_\varepsilon\times1}$ contains the eigenvalues of the principal directions, and $\boldsymbol{\theta}_\varepsilon \in \mathbb{R}_{m_\varepsilon\times1}$ contains the angles between the principal directions and the horizontal. In Fig. 5e, all of these parameters for a single Gaussian are depicted.

The state vector representing the color model will be

$$\mathbf{x}_3 = [\mathbf{g}_{\mathcal{O}}^T, \mathbf{g}_{\mathcal{B}}^T]^T \in \mathbb{R}_{6m_T\times1},$$

where $m_T = m_{\mathcal{O}} + m_{\mathcal{B}}$.

5.4 Object Contour

Since color segmentation usually gives a rough estimation of the target location, we use the contour of the object to obtain a more precise tracking. In particular, the contour will be represented by a discrete set of n_c points in the image, $\mathbf{R} = [(u_1, v_1)^T, \dots, (u_{n_c}, v_{n_c})^T]^T$. We assign these values to the contour state vector:

$$\mathbf{x}_4 = [(u_1, v_1)^T, \dots, (u_{n_c}, v_{n_c})^T]^T \in \mathbb{R}_{n_c\times2}.$$

5.5 Dynamic Models

The behavior over time of all of the previous state vectors will be predicted by simple stochastic dynamic models. In particular, the state of the bounding box $\mathbf{x}_1$ will be estimated by a Kalman filter based on a Gaussian linear dynamic model with additive white noise:

$$\mathbf{x}_1^t = \mathbf{H}_1\mathbf{x}_1^{t-1} + \mathbf{q}_{1,h},$$

where $\mathbf{H}_1$ is a deterministic component and $\mathbf{q}_{1,h}$ is a random variable distributed as a Gaussian with zero mean and diagonal covariance matrix $\boldsymbol{\Sigma}_{1,h}$.

The rest of features $\mathbf{x}_i$, $i = \{2, 3, 4\}$, will be estimated by particle filters with dynamic models consisting of a random scaling and translation:

$$\mathbf{x}_i^t = (\mathbb{I} + \mathbf{S}_i)\mathbf{x}_i^{t-1} + \mathbf{q}_i,$$

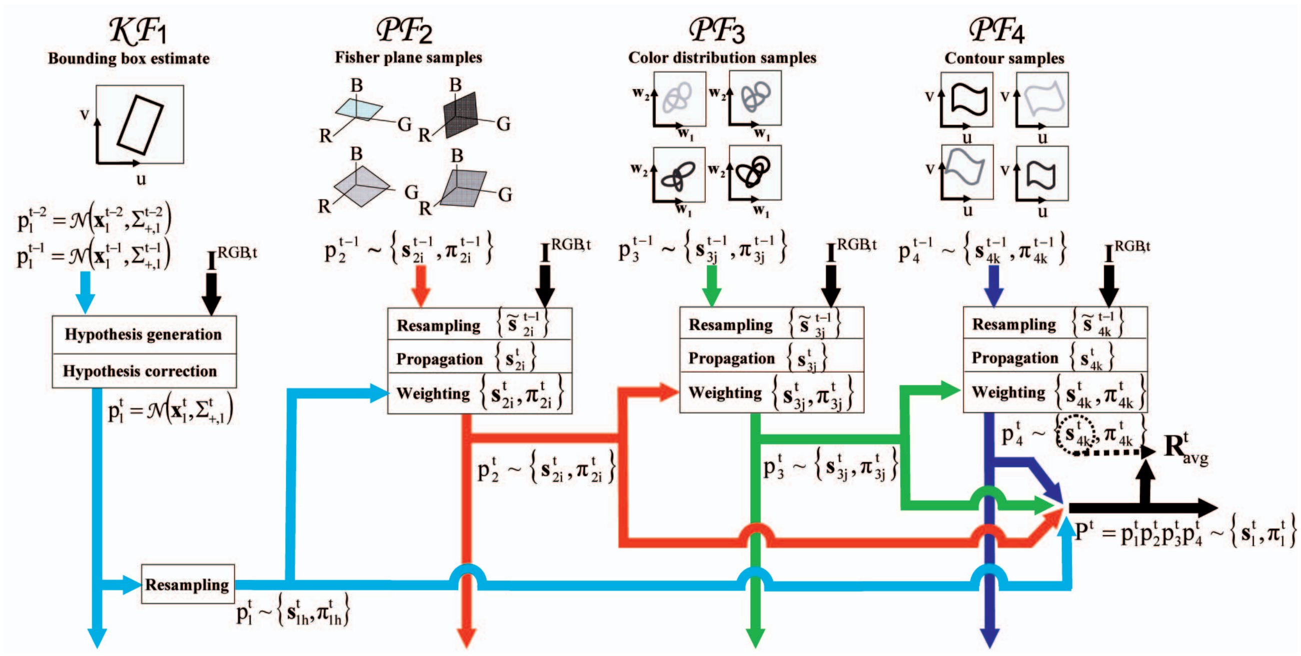

Fig. 6. **Flow diagram of one iteration of the complete algorithm.** Different color lines and arrows show the paths of each feature. Observe how the output of each filter feeds into a subsequent filter.

where $\mathbb{I}$ is the identity matrix, $\mathbf{S}_i$ is a random scaling matrix, and $\mathbf{q}_i$ a random translation vector.

In all of these dynamic models, there are certain parameters that control the stochastic performance of the model. Their value will determine the level of dispersion of the samples in the configuration space and, although they are an important factor to consider when designing the tracker, they do not need to be estimated with high accuracy. In particular, when using particle filters, poor estimates of these parameters may be compensated for by selecting a larger number of particles. On the other hand, a Kalman filter might be more sensitive to the value of the covariance matrix $\mathbf{\Sigma}_{1,h}$ defined in its dynamic model since the prediction is based on a single hypothesis. Nevertheless, in the tracker system explained in the following section, the role of the Kalman filter will be to provide a coarse estimate about the bounding box surrounding the target. Therefore, poor estimates of $\mathbf{\Sigma}_{1,h}$ are not that critical.[3]

With these considerations, for the experiments that will be presented at the end of the paper, the parameters providing the random behavior to the dynamic models have been learned offline, based on a simple least squares procedure over a set of hand-segmented training sequences.

6 The Complete Tracking Algorithm

In this section, we will integrate the tools described previously and analyze in detail the complete method for tracking rigid and nonrigid objects in cluttered environments under changing illumination. Specifically, the target is going to be tracked based on the four features just defined: the bounding box (estimated by a Kalman filter $\mathcal{KF}_1$), the Fisher color space (estimated through a particle filter $\mathcal{PF}_2$), the color distribution (estimated through $\mathcal{PF}_3$), and the object contour (estimated using $\mathcal{PF}_4$). In the following sections, the algorithm will be described step by step. For a better understanding of the method, the reader is encouraged to follow the flow diagram in Fig. 6.

6.1 Input at Iteration t

At time t, for the bounding box feature, the mean and covariance parameters from the previous iteration are available, which can be used to estimate its posterior probability p_1^{t-1}. For the rest of the $\mathbf{x}_i$ features, $i = \{2, 3, 4\}$, estimated through particle filters, a set of n_i samples $\{\mathbf{s}_{ij}^{t-1}\}_{j=1}^{n_i}$ is available from the previous iteration. The structure of these samples is the same as the corresponding state vector $\mathbf{x}_i$. Each sample has an associated weight π_{ij}^{t-1}. The whole set approximates the a posteriori PDF of the system $P^{t-1} = p(\mathbf{X}^{t-1}|\mathbf{Z}^{t-1})$, as defined in (4), where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ contains the state vectors of all the cues utilized to represent the object and $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4\}$ refers to the observations measured to evaluate these features. Obviously, the input RGB image at time t, denoted by $\mathbf{I}^{\mathrm{RGB},t}$, is also available.

6.2 Updating the Bounding Box PDF

The bounding box is estimated through a Kalman filter, which basically relies on the prediction term and for which the correction introduced by the observation has a low significance. The reason why we do not rely on the bounding box observation is because we wish to deal with highly cluttered sequences and, hence, the observation done by a single cue will probably be inaccurate. The robustness of the system comes from the integration over all of the cues and not from of a single cue. Therefore, the estimate of the

3. For the experiments shown in the paper, we do not expect abrupt changes of position. In these circumstances, a Kalman filter works properly. In order to deal with sequences including abrupt changes of position (besides abrupt changes of illumination), the Kalman filter estimating the bounding box should be replaced by a particle filter.

bounding box state will mostly come from the prediction done by the dynamic model.

In order to obtain a Kalman filter with such a behavior, a large value is assigned to the covariance associated with the measurement noise $\mathbf{\Sigma}^t_{m,1}$. Next, let us see in detail how the Kalman filter behaves under these specifications. The basic steps for a single iteration are detailed in Section 3.2 and are repeated here for convenience.

6.2.1 Input Data

$\mathcal{KF}_1$, the Kalman filter associated with state vector $\mathbf{x}_1$, receives the bounding box estimate of the previous state, that is, $p^{t-1}_1 = \mathcal{N}(\mathbf{x}^{t-1}_1, \mathbf{\Sigma}^{t-1}_1)$, where $\mathbf{x}_1$ and $\mathbf{\Sigma}_1$ correspond to the a posteriori estimates of the mean and covariance.

6.2.2 Hypothesis Generation

Using the Kalman filter equation (8), the state vector and covariance matrix are propagated to

$$\mathbf{x}^t_{1,-} = \mathbf{H}_1 \mathbf{x}^{t-1}_1,$$
$$\mathbf{\Sigma}^t_{1,-} = \mathbf{\Sigma}_{1,h} + \mathbf{H}_1 \mathbf{\Sigma}^{t-1}_1 \mathbf{H}^T_1,$$

where the matrix $\mathbf{H}_1 \in \mathbb{R}_{5\times5}$ corresponds to the deterministic component of the dynamic model and $\mathbf{\Sigma}_{1,h} \in \mathbb{R}_{5\times5}$ is the covariance matrix of the process noise. Here, the subscript symbol "$-$" indicates that the estimate is a priori.

6.2.3 Hypothesis Correction

In order to correct the predicted state, an observation $\mathbf{z}^t_1$ is computed by a simple correlation method. Let us call the rectangular window defined by $\mathbf{x}^{t-1}_1 = [u^{t-1}_1, v^{t-1}_1, a^{t-1}_1, b^{t-1}_1, \theta^{t-1}_1]^T$ $\mathcal{W}^{t-1}$. The observation $\mathbf{z}^t_1$ will be the same window but with its centroid translated according to the parameters (du, dv) minimizing the following *Sum of Squared Differences* (SSD) criterion:

$$\arg\min_{du,dv} \sum_{u,v\in\mathcal{W}^{t-1}} \left(\mathbf{I}^{\mathrm{RGB},t-1}(u,v) - \mathbf{I}^{\mathrm{RGB},t}(u+du, v+dv)\right)^2.$$

Subsequently, the value of the observation vector is defined as

$$\mathbf{z}^t_1 = \left[u^{t-1}_1 + du, v^{t-1}_1 + dv, a^{t-1}_1, b^{t-1}_1, \theta^{t-1}_1\right]^T.$$

Following (9), the observation $\mathbf{z}^t_1$ is used to correct the predicted state vector and covariance matrix:

$$\mathbf{x}^t_1 = \mathbf{x}^t_{1,-} + \mathbf{K}^t[\mathbf{z}^t_1 - \mathbf{M}^t \mathbf{x}^t_{1,-}],$$
$$\mathbf{\Sigma}^t_1 = \mathbf{\Sigma}^t_{1,-} - \mathbf{K}^t \mathbf{M}^t_1 \mathbf{\Sigma}^t_{1,-},$$

where $\mathbf{M}_1$ is the matrix denoting the deterministic component of the measurement model and $\mathbf{K}^t = \mathbf{\Sigma}^t_{1,-}(\mathbf{M}^t_1)^T[\mathbf{M}^t_1 \mathbf{\Sigma}^t_{1,-}(\mathbf{M}^t_1)^T + \mathbf{\Sigma}_{1,m}]^{-1}$ is the Kalman gain, with the matrix $\mathbf{\Sigma}_{1,m}$ being the covariance associated to the observation noise.

As we have previously commented, the bounding box observation is highly sensitive to the presence of clutter or lighting changes since the SSD operator is not robust under this kind of artifact. Hence, a low responsibility needs to be assigned to the observation measure within its contribution to the final decision of the a posteriori probability. Kalman filtering allows us to control the relative contribution of the prediction term and the observation term through the values of the dynamic model covariance matrix $\mathbf{\Sigma}_{1,h} \in \mathbb{R}_{10\times10}$ and the measurement covariance matrix $\mathbf{\Sigma}_{1,m} \in \mathbb{R}_{5\times5}$. In particular, these matrices have been selected offline such that they satisfy $\mathbf{\Sigma}_{1,m} \gg \mathbf{\Sigma}_{1,h}$. Note that a large measurement covariance matrix implies a small Kalman gain, that is, $\mathbf{\Sigma}_{1,m} \uparrow\uparrow \Rightarrow \mathbf{K} \downarrow\downarrow$. As a consequence, the innovation terms introduced by the observation $\mathbf{z}_1$ will have a small responsibility and the filter will mostly rely on the prediction terms.

6.2.4 Output Data

The variables $\mathbf{x}^t_1$ and $\mathbf{\Sigma}^t_1$ approximate a normal distribution $p^t_1 = \mathcal{N}(\mathbf{x}^t_1, \mathbf{\Sigma}^t_1)$, which estimates the state of the bounding box feature at the output of the filter $\mathcal{KF}_1$. Since this distribution is going to feed into subsequent particle filters based on discrete and weighted samples of the state vector, it is necessary to discretize p^t_1. Thus, the normal density p^t_1 is uniformly sampled and approximated by a set of n_1 weighted particles:

$$p^t_1 = \mathcal{N}(\mathbf{x}^t_1, \mathbf{\Sigma}^t_1) \cong \sum_{j=1}^{n_1} \mathbf{s}_{1j}\pi_{1j}.$$

6.3 Updating the Fisher-Plane PDF

Whereas the bounding-box feature is approximately estimated through a Kalman filter mostly relying on its prediction component, the rest of the object cues are going to be estimated through particle filters. In this section, the particle filter responsible for the Fisher plane feature, $\mathcal{PF}_2$, will be described.

6.3.1 Input Data

At the starting point of iteration t, $\mathcal{PF}_2$, the particle filter associated with $\mathbf{x}_2$, receives p^{t-1}_2, the PDF of the state vector $\mathbf{x}_2$ at time $t-1$, approximated by n_2 weighted samples, $\{\mathbf{s}^{t-1}_{2j}, \pi^{t-1}_{2j}\}^{n_2}_{j=1}$. In addition, it also receives the output of the previous filter $\mathcal{KF}_1$ estimating the feature $\mathbf{x}_1$ by a set of n_1 weighted samples, $\{\mathbf{s}^t_{1j}, \pi^t_{1j}\}^{n_1}_{j=1}$.

6.3.2 Hypotheses Generation

Using the standard particle filter procedure [9], the set of particles $\{\mathbf{s}^{t-1}_{2j}, \pi^{t-1}_{2j}\}^{n_2}_{j=1}$ is resampled (sampling with replacement) and propagated to the set $\{\mathbf{s}^t_{2j}\}$ according to the dynamic model defined in Section 5.5. Each sample represents a different configuration of the Fisher plane $\mathbf{W}_j$, $j = 1, \ldots, n_2$. Fig. 7 (top left) shows some samples of Fisher planes obtained after the hypotheses generation stage.

6.3.3 Hypotheses Correction

The key point of the proposed DFBI approach is that cue dependence is considered during the hypotheses correction stage. In particular, in order to assign a weight to the propagated samples $\{\mathbf{s}^t_{2j}\}^{n_2}_{j=1}$, the information provided from the output p^t_1 of $\mathcal{KF}_1$ is used. The discretized samples $\{\mathbf{s}^t_{1j}, \pi^t_{1j}\}^{n_1}_{j=1}$, approximating p^t_1 are resampled n_2 times, resulting in the set $\{\tilde{\mathbf{s}}^t_{1j}\}^{n_2}_{j=1}$. Note that this set may contain repeated copies of the more likely samples of the bounding box. Then, every Fisher plane sample $\mathbf{s}^t_{2j}$ is associated with a

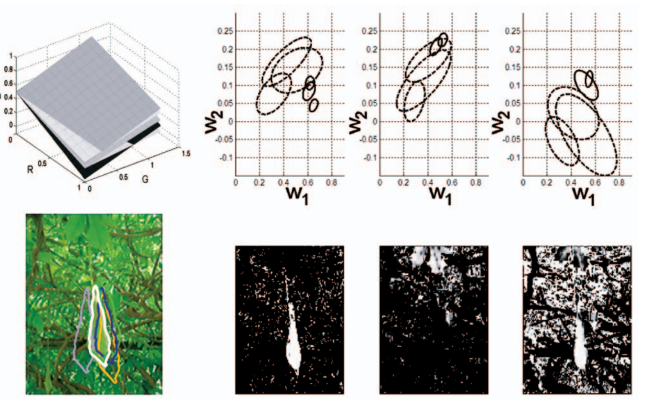

Fig. 7. **Generation of multiple hypotheses for each feature.** Top left: Fisher plane. Bottom left: Hypothesized contours. Right: Color distribution. Top right: Several hypothesized MoGs parameterizing the foreground ($\mathcal{O}$) and the background ($\mathcal{B}$) color distributions. Solid line ellipses and dashed line ellipses belong to the foreground and background MoGs, respectively. Bottom right: A posteriori probability maps of the object class, obtained using the corresponding color configurations above them. Note that some of the color configurations are appropriate to discriminate the target (central leaf) from the rest of the background, whereas, using other configurations, the $\mathcal{O}$ and $\mathcal{B}$ regions are undistinguishable.

bounding box sample $\tilde{\mathbf{s}}_{1j}^t$. Let us call $\mathcal{W}_j^t$ the rectangular bounding box defined by $\tilde{\mathbf{s}}_{1j}^t$.

Once we have defined a bounding box $\mathcal{W}_j^t$ for each Fisher plane $\mathbf{s}_{2j}^t$, the basic idea is to weigh the latter depending on how well it permits us to discriminate the points inside $\mathcal{W}_j^t$ from the points outside $\mathcal{W}_j^t$.

To this end, we randomly select two sets of RGB color points, $\mathbf{C}_{\mathcal{W}}^{\text{RGB}}$ and $\mathbf{C}_{\overline{\mathcal{W}}}^{\text{RGB}}$, inside and outside $\mathcal{W}_j^t$, respectively. These sets and the image $\mathbf{I}^{\text{RGB},t}$ are projected onto the n_j Fisher planes, generating the n_j triplets $\{\mathbf{C}_{\mathcal{W},j}^{\text{Fisher}}, \mathbf{C}_{\overline{\mathcal{W}},j}^{\text{Fisher}}, \mathbf{I}_j^{\text{Fisher},t}\}$. For each triplet, we use the EM algorithm to fit an MoG to the sets $\mathbf{C}_{\mathcal{W},j}^{\text{Fisher}}$ and $\mathbf{C}_{\overline{\mathcal{W}},j}^{\text{Fisher}}$.

Based on these MoGs, we compute the a posteriori probability map $p(\mathcal{W}_j^t|\mathbf{I}_j^{\text{Fisher},t})$ for all of the (u,v) pixels of image $\mathbf{I}_j^{\text{Fisher},t}$ using the Bayes rule (9). According to this probability map, we assign the following weight to each sample:

$$\pi_{2j}^t \sim \frac{\sum_{(u,v)\in\mathcal{W}_j^t} p\left(\mathcal{W}_j^t|\mathbf{I}_j^{\text{Fisher},t}\right)}{n_{\mathcal{W}}} - \frac{\sum_{(u,v)\notin\mathcal{W}_j^t} p\left(\mathcal{W}_j^t|\mathbf{I}_j^{\text{Fisher},t}\right)}{n_{\overline{\mathcal{W}}}},$$

where $n_{\mathcal{W}}$ and $n_{\overline{\mathcal{W}}}$ are the number of image pixels in and out of $\mathcal{W}_j^t$, respectively.

6.3.4 Output Data

The output of $\mathcal{PF}_2$ is the set $\{\mathbf{s}_{2j}^t, \pi_{2j}^t\}_{j=1}^{n_2}$ approximating the estimate of the a posteriori probability function p_2^t for the normal to the Fisher plane.

6.4 Updating the PDFs of the Foreground and Background Color Distributions

6.4.1 Input Data

$\mathcal{PF}_3$, the particle filter associated with the state vector $\mathbf{x}_3$, receives as its input $p_3^{t-1} \sim \{\mathbf{s}_{3j}^{t-1}, \pi_{3j}^{t-1}\}_{j=1}^{n_3}$, approximating the PDF of the color distributions in the previous iteration, and $p_2^t \sim \{\mathbf{s}_{2j}^t, \pi_{1j}^t\}_{j=1}^{n_2}$, an approximation of the PDF of the Fisher planes at time t.

6.4.2 Hypotheses Generation

Particles $\{\mathbf{s}_{3j}^{t-1}\}$ are resampled and propagated (using the dynamic model associated with $\mathbf{x}_3$) to the set $\{\mathbf{s}_{3j}^t\}_{j=1}^{n_3}$. A sample $\mathbf{s}_{3j}^t$ represents a MoG configuration of the foreground and background color points projected onto the Fisher color space. Fig. 7 (top right) shows the appearance of different MoG configurations resulting from the random propagation generated by the dynamic model.

6.4.3 Hypotheses Correction

Again, in order to assign a weight to these samples, we use the information provided from the output p_2^t of $\mathcal{PF}_2$. Through a sampling with replacement procedure, the set $\{\mathbf{s}_{2j}^t\}_{j=1}^{n_2}$ is resampled n_3 times, providing the set $\{\tilde{\mathbf{s}}_{2j}^t\}_{j=1}^{n_3}$. This allows us to assign the most likely samples $\mathbf{s}_{2j}^t$ of Fisher planes to the samples $\mathbf{s}_{3j}^t$ of MoGs.

The rest of the weighting process is similar to the one described in the previous section: For a given sample $\mathbf{s}_{3j}^t$, $j = 1, \ldots, n_3$, we project image $\mathbf{I}^{\text{RGB},t}$ to its associated Fisher plane $\mathbf{W}_j$ parameterized by $\tilde{\mathbf{s}}_{2j}^t$. The new image will be $\mathbf{I}_j^{\text{Fisher},t} = \mathbf{I}^{\text{RGB},t}\mathbf{W}_j^T$.

Using the MoGs of the object and background parameterized by the sample $\mathbf{s}_{3j}^t$, the a posteriori probability map $p(\mathcal{O}|\mathbf{I}_j^{\text{Fisher},t})$ is computed for all of the pixels of $\mathbf{I}_j^{\text{Fisher},t}$ and the weight π_{3j}^t is assigned by

$$\pi_{3j}^t \sim \frac{\sum_{(u,v)\in\mathcal{W}_j^t} p\left(\mathcal{O}|\mathbf{I}_j^{\text{Fisher},t}\right)}{n_{\mathcal{W}}} - \frac{\sum_{(u,v)\notin\mathcal{W}} p\left(\mathcal{O}|\mathbf{I}_j^{\text{Fisher},t}\right)}{n_{\overline{\mathcal{W}}}},$$

where $\mathcal{W}_j^t$, $n_{\mathcal{W}}$, and $n_{\overline{\mathcal{W}}}$ were defined above.

In Fig. 7 (bottom right), the a posteriori probability maps of the target (the central leaf) are depicted. Notice how some of the MoG configurations provide probability maps where the target is clearly distinguished from the background.

6.4.4 Output Data

The set $\{\mathbf{s}_{3j}^t, \pi_{3j}^t\}_{j=1}^{n_3}$ approximates the estimate of the a posteriori probability function p_3^t for the foreground and background color distributions.

6.5 Updating the Contour PDF

6.5.1 Input Data

The last particle filter $\mathcal{PF}_4$ receives as its input $p_4^{t-1} \sim \{\mathbf{s}_{4j}^{t-1}, \pi_{4j}^{(t-1)}\}_{j=1}^{n_4}$, which approximates the PDF of the contours in the previous iteration, and $p_3^t \sim \{\mathbf{s}_{3j}^t, \pi_{3j}^t\}_{j=1}^{n_3}$, an approximation of the PDF of the foreground and background color distributions at time t.

6.5.2 Hypotheses Generation

Similarly to the procedure utilized for $\mathcal{PF}_2$ and $\mathcal{PF}_3$, particles $\{\mathbf{s}_{4j}^{t-1}\}$ are resampled and propagated to the set $\{\mathbf{s}_{4j}^t\}_{j=1}^{n_4}$, according to the dynamic model described in Section 5.5. This dynamic model produces affinely deformed and translated copies of the original contours (see some examples in Fig. 7 (bottom left) for the leaf tracking example).

6.5.3 Hypotheses Correction

The set $\{\mathbf{s}^t_{4j}\}$ is weighted based on p^t_3 through a similar process as the one described for $\mathcal{PF}_2$ and $\mathcal{PF}_3$: Initially, samples $\{\mathbf{s}^t_{3j}, \pi^t_{3j}\}^{n_3}_{j=1}$ are resampled according to the weights π^t_{3j}, resulting in a new set $\{\tilde{\mathbf{s}}^t_{3j}\}^{n_4}_{j=1}$. Then, each color sample $\tilde{\mathbf{s}}^t_{3j}$, $j = 1, \dots, n_4$, is associated to a contour sample $\mathbf{s}^t_{4j}$.

The a posteriori probability map $p(\mathcal{O}|\mathbf{I}^{\text{Fisher},t}_j)$ assigned to $\tilde{\mathbf{s}}^t_{3j}$ in the previous time step and the contour $\mathbf{R}_j$ represented by $\mathbf{s}^t_{4j}$ are used to compute the weights for the contour samples as follows:

$$\pi^t_{4j} \sim \frac{\sum_{(u,v)\in\mathbf{R}_j} p\left(\mathcal{O}|\mathbf{I}^{\text{Fisher},t}_j\right)}{n_{\mathbf{R}_j}} - \frac{\sum_{(u,v)\notin\mathbf{R}_j} p\left(\mathcal{O}|\mathbf{I}^{\text{Fisher},t}_j\right)}{n_{\overline{\mathbf{R}}_j}},$$

where $n_{\mathbf{R}_j}$ and $n_{\overline{\mathbf{R}}_j}$ are the number of image pixels inside and outside the contour $\mathbf{R}_j$.

6.5.4 Output Data

Finally, the set of samples and weights $\{\mathbf{s}^t_{4j}, \pi^t_{4j}\}^{n_4}_{j=1}$ approximates the estimate of the a posteriori probability function p^t_4 for the contours of the object.

6.6 Algorithm Output Generation

As we have shown in Section 3.1, the complete a posteriori probability function can be determined by

$$\begin{aligned} P^t &= p(\mathbf{X}^t_{1:4}|\mathbf{Z}^t_{1:4}) = p^t_1 p^t_2 p^t_3 p^t_4 \\ &= p_1(\mathbf{x}^t_1|\mathbf{Z}^t_1) p_2(\mathbf{x}^t_2|\mathbf{X}^t_1, \mathbf{Z}^t_{1:2}) p_3(\mathbf{x}^t_3|\mathbf{X}^t_{1:2}, \mathbf{Z}^t_{1:3}) p_4(\mathbf{x}^t_4|\mathbf{X}^t_{1:3}, \mathbf{Z}^t_{1:4}) \\ &\sim \left\{\left\{\mathbf{s}^t_{4k}\left(\mathbf{s}^t_{3j}(\mathbf{s}^t_{2i}(\mathbf{s}^t_{1h}))\right)\right\}, \left\{\pi^t_{1h}\pi^t_{2i}\pi^t_{3j}\pi^t_{4k}\right\}\right\} = \{\mathbf{s}^t_l, \pi^t_l\}, \end{aligned} \quad (11)$$

where $l = 1, \dots, n_4$. Equation (11) reflects the fact that samples of the state vector $\mathbf{x}_4$ are computed by taking into account samples of $\mathbf{x}_3$ (that is, $\mathbf{s}^t_{4k} \equiv \mathbf{s}^t_{4k}(\mathbf{s}^t_{3j})$) and these have been computed by considering samples of $\mathbf{x}_2$ (that is, $\mathbf{s}^t_{3j} \equiv \mathbf{s}^t_{3j}(\mathbf{s}^t_{2i})$) and these have considered samples of $\mathbf{x}_1$ (that is, $\mathbf{s}^t_{2i} \equiv \mathbf{s}^t_{2i}(\mathbf{s}^t_{1h})$). Observe that the final number of samples to approximate the whole probability P^t is determined by n_4. Considering the final weights, the average contour is computed as

$$\mathbf{R}^t_{avg} = \sum_{l=1}^{n_4} \mathbf{s}^t_{4l}\pi^t_l. \quad (12)$$

Since all of the contour samples have been generated with an affine deformation model, we need to add an extra final stage in order to deal with nonrigid deformations of the object boundary. We use $\mathbf{R}^t_{avg}$ to initialize a deformable contour that is fitted to the object boundary using the traditional *snake* formulation [13]. This adjustment is highly simplified by using the target position estimated by the color particle filter. This is shown in Fig. 8, where the a posteriori probability map of the color module allows us to eliminate noisy edges from the original image, which might disrupt the fitting procedure of the snake.

Note the advantage of using the color module: Traditional snake algorithms need to adjust a given curve to the edges of an image. However, if the image contains a high level of clutter (such as the image shown in Fig. 8a), a standard edge detector may detect a lot of noisy edges that might disturb the snake during the fitting procedure. For instance, Fig. 8b shows the edges detected by a Canny filter in the previous image. Under this type of edge image, traditional snake algorithms are likely to fail. Nevertheless, by applying simple morphological operations on the a posteriori probability map of the target provided by the color module (Fig. 8c), most of the noisy edges may be eliminated from the image (Fig. 8d). Then, the fitting procedure is made considerably easier. Figs. 8e and 8f show the initialization of the snake (by the averaged contour $\mathbf{R}^t_{avg}$) and the final result of the adjustment, respectively.

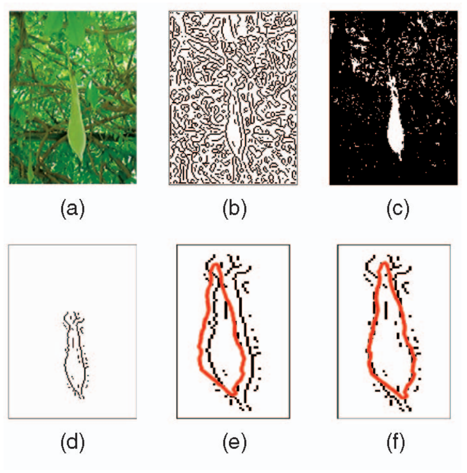

Fig. 8. **Simplification of the snake fitting procedure using color information.** (a) Original cluttered image. (b) Edge features of the image obtained with a Canny edge detector. Observe the large quantity of noisy spurious edges detected, which might disrupt a traditional snake procedure from converging to the true object contour. (c) Foreground a posteriori probability map obtained using the color module. (d) Refined edge image, where most of the noisy edges have been removed by considering a mask obtained by applying simple morphological operations on image (c). (e) Contour $\mathbf{R}^t_{avg}$ used as initialization for a snake fitting procedure. (f) Results of snake fitting.

7 Experimental Results

In this section, we present the results of different experiments on both synthetic and real video sequences and examine the robustness of our system to several changing conditions of the environment in situations where other algorithms may fail.

Before discussing the obtained results, we would like to point out that, in the particular case of integrating several particle filters, the structure of the DBFI framework allows us to considerably reduce the number of samples necessary to approximate the PDF that represents the state of the target. As we have previously argued in Section 4.1, this ability addresses the problem of the curse of dimensionality undergone by particle filters when the size of the state vector is increased. That is, if we integrate N features using n particles for each one, the complexity of the proposed methodology would be $\mathcal{O}(nN)$, whereas if all the features were integrated in a single particle filter, its complexity would be $\mathcal{O}(n^N)$. In terms of computation times, it is worth mentioning that the algorithm takes less than a minute per

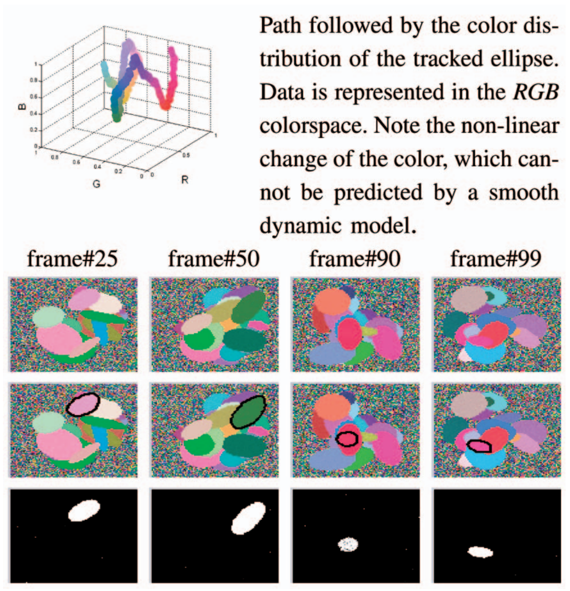
Path followed by the color distribution of the tracked ellipse. Data is represented in the *RGB* colorspace. Note the non-linear change of the color, which cannot be predicted by a smooth dynamic model.

Fig. 9. **Experiment 1: Tracking of a synthetic ellipse that randomly changes its color, position, and shape.** Top: Path followed by the color distribution of the ellipse. Bottom: Some sequence frames: original frames (first row), tracking results (second row), and a posteriori PDF maps of the color module (third row). The proposed method of integrating position prediction, optimal color space selection, color distribution estimate, and contour estimate is able to segment the tracked ellipse even when the background contains highly disturbing elements. Observe in the frame before the last how the tracked ellipse is surrounded by another ellipse with a similar appearance. In spite of that, the tracker does not lose the target.

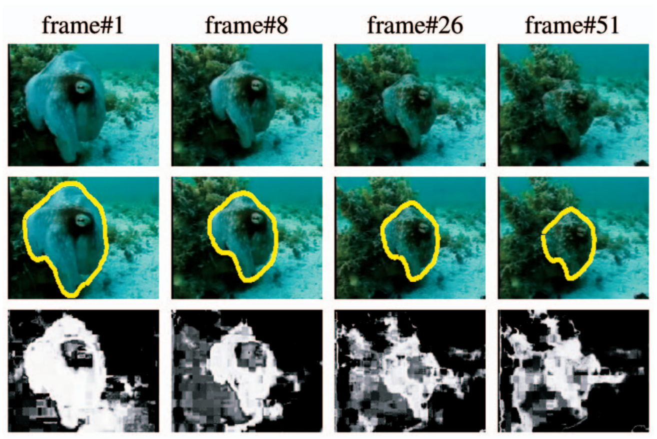

Fig. 10. **Experiment 2: Tracking a camouflaging octopus.** Top row: Original sequence. Middle row: Results using the proposed method. Bottom row: A posteriori foreground PDF maps obtained by the color module $(\mathcal{PF}_3)$.

frame (Pentium IV, 2.0 GHz), implemented in an interpretative language (Matlab) and using about 100 samples per feature. The most time-consuming part of the algorithm corresponds to the Fisher plane update since it requires running the EM algorithm for each particle. However, several approximations might reduce the computation time for this step, such as using a relatively small number of data points or predefining the number of Gaussians that are used to fit the MoG models.

In the following sections, some experimental results will be reported. The first set of experiments deals with sequences where the lighting conditions or the appearance of the target changes continuously. In the last group of experiments, abrupt illumination changes will be considered. In both cases, examples of targets that deform rigidly and nonrigidly are included.

7.1 Tracking under Continuous Lighting Changes

The first experiment corresponds to the tracking of a synthetically generated sequence of an ellipse that randomly changes its position, color, and shape in a cluttered background. In Fig. 9 (top), we depict the path followed by the color cue. Observe the nonlinearity of the trajectory. As was shown in [9], this kind of nonlinear path may be estimated by filters based on multihypothesis, such as particle filters. Results show that the DBFI method proposed in this paper, based on multiple-multihypothesis algorithms, allows us to segment and track the ellipse, even when the background and target have similar colors (observe the frame before the last).

In the second experiment (Fig. 10), we show how our method performs in a real video sequence of an octopus changing its appearance while camouflaging. Observe that the foreground a posteriori probability maps of the color module give a rough estimate about the target position, especially when the octopus appearance is quite similar to that of the background. Nevertheless, a detailed detection of the target may be obtained by correcting the color estimate using the shape module.

In order to emphasize the importance of simultaneously adapting color and contour features using particle filters, in the rest of the experiments, the performance of the discussed algorithm will be compared to a tracking technique that uses multiple hypotheses to predict the contour of the object and accommodates the color with a predictive filter based on a simple smooth dynamic model such as

$$\mathbf{g}^t = (1-\beta)\mathbf{g}^{t-2} + \beta\mathbf{g}^{t-1}, \tag{13}$$

where g is the parameterization of the color distribution (with the same structure as in (10)) and β is a mixing factor. Actually, this approach is quite similar to the ICondensation technique described in [10].

Experiment 3 corresponds to the tracking of the nonrigid boundary of a bending book in a video sequence, where the lighting conditions smoothly change from natural lighting to yellow lighting. Fig. 11 shows some frames of the tracking results. Note that, despite the smooth change of illuminant, the smooth dynamic model is unable to track the contour of the object. The reason for the failure is that the smooth dynamic model cannot cope with the effect of self-shadowing produced during the movement of the book.

7.2 Tracking under Abrupt Lighting Changes

In Experiment 4, the color distribution of the bending book sequence previously presented is manually modified in order to simulate an abrupt change of illumination. The top row in Fig. 12 shows three consecutive frames presented to the algorithm. Note that the abrupt illumination changes occurred between frames $t-1$ and t. Results prove the inability of the smooth color model to predict such a change since the a posteriori probability map of the foreground

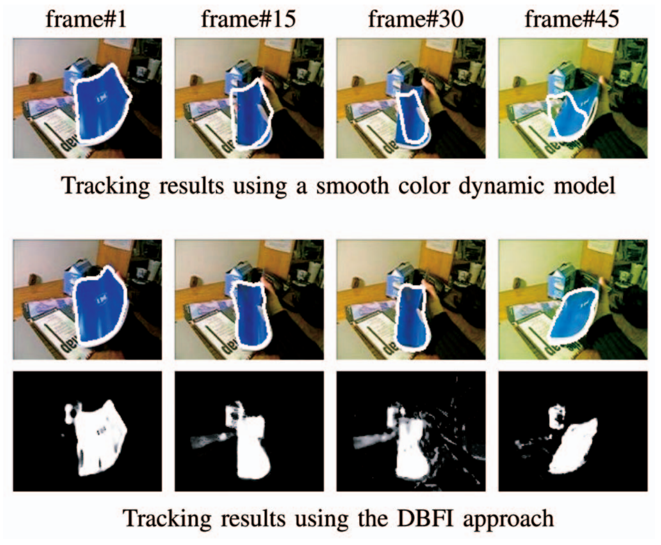
frame#1
frame#15
frame#30
frame#45
Tracking results using a smooth color dynamic model
Tracking results using the DBFI approach

Fig. 11. **Experiment 3: Tracking results of a bending book in a sequence with a smooth change of illumination**. Top row: Results using only a contour particle filter and assuming a smooth change of color. The method fails. Middle row: Results using the proposed method. Bottom row: A posteriori object probability maps of the color module $(\mathcal{PF}_3)$.

region depicted in Fig. 12 (bottom left) does not discriminate between the foreground and background. On the other hand, a good result is obtained with the DBFI approach proposed is this paper (Fig. 12, bottom center and bottom right).

In the final experiment (Experiment 5), we have tested the algorithm with the sequence of a moving leaf used as an example in previous sections. Although this is a challenging sequence because it is highly cluttered, the illumination changes abruptly and the target moves unpredictably, we can perform tracking using the proposed method. Fig. 13 shows some frames of the tracking results. Observe the abrupt change of illumination between the first and second frames, which leads to failure when we try to track using a contour particle filter with a smooth color prediction.

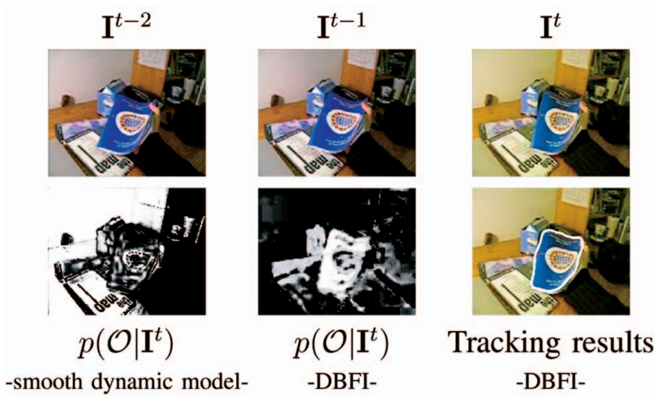

Fig. 12. **Experiment 4: Tracking results of a nonrigid object (a bending book) in a sequence with abrupt changes of illumination.** Top row: $\mathbf{I}^{t-2}$, $\mathbf{I}^{t-1}$, and $\mathbf{I}^{t}$ are three consecutive images. Note the abrupt change in illuminant between frames $t-1$ and t. Bottom left: $p(\mathcal{O}|\mathbf{I}^t)$ map obtained assuming a smooth dynamic model of the color feature. There is no good discrimination between the foreground and background. Bottom center: $p(\mathcal{O}|\mathbf{I}^t)$ map provided by the proposed framework. The foreground and background discrimination is clearly enhanced with respect to the smooth dynamic model case. Bottom right: Tracking results obtained after using $p(\mathcal{O}|\mathbf{I}^t)$ to eliminate false edges from the image and fitting a deformable contour to the object boundary.

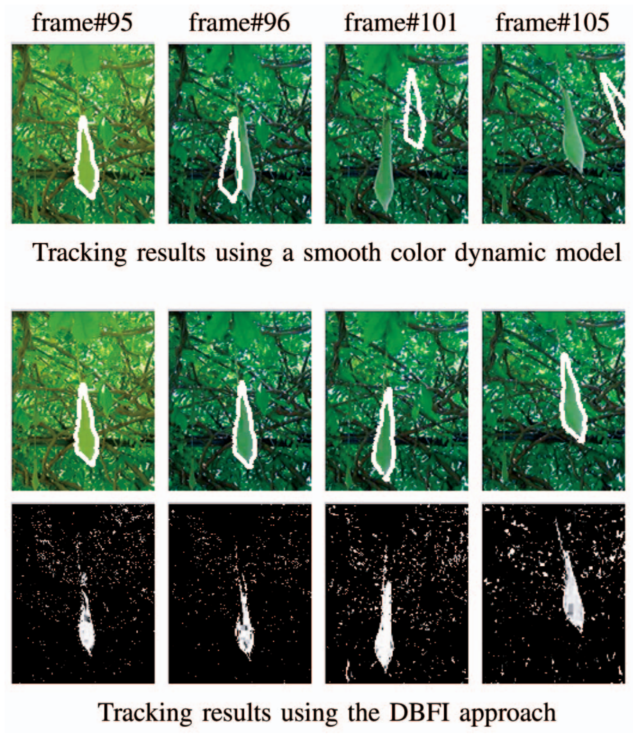
frame#95
frame#96
frame#101
frame#105
Tracking results using a smooth color dynamic model
Tracking results using the DBFI approach

Fig. 13. **Experiment 5: Tracking results of a leaf.** Tracking results of a cluttered sequence, where the target moves following unexpected paths. Furthermore, the sequence suffers from an abrupt change of illumination (observe it between frame #95 and frame #96). Top row: Results using a contour-based particle filter and assuming a smooth change of the color feature. The method fails. Middle row: Successful results obtained using DBFI. Bottom row: A posteriori PDF maps of the color module $(\mathcal{PF}_3)$. Observe how the tracked leaf is clearly detected and the unexpected illumination change does not destabilize the tracker.

8 Conclusions and Future Work

Enhancing target representation by using multiple cues has been a common strategy for improving the performance of tracking techniques. However, most of these algorithms are based on heuristics and ad hoc rules that only work for specific applications.

In this paper, we describe a general probabilistic framework allowing to integrate any number of object features. The state of the features may be estimated by any algorithm based on a "hypotheses generation-hypotheses correction" strategy (for instance particle filters or a Kalman filter). The key point of the approach is that it permits us to consider cue dependence and obtain precise estimates for each of the cues.

The proposed framework has been theoretically proven and validated in a tracking example with synthetic data, which has been used as a benchmark to compare the performance of our method with other well-known algorithms from the field. The best results in terms of accuracy and reliability are obtained by the DBFI method presented here. Furthermore, in the specific case where the integrated features are estimated by particle filters, our method does not suffer from the *curse of dimensionality* problem, which usually affects particle filter formulations, producing exponential increases in computation time when the dimensionality of the state space increases.

Furthermore, this framework has allowed us to design a tracking algorithm that simultaneously accommodates the color space where the image points are represented: the color distributions of the object and background and the contour of the object. The effectiveness of the method has been proven by successfully tracking objects from synthetic and real sequences presenting high content of clutter, nonrigid object boundaries, unexpected target movements, and abrupt changes of illumination.

In the proposed approach, we only have considered the integration of multiple cues for single object tracking. In future research, we plan to extend this formulation to multiple object and multiple cues integration. Further integration of other features into the framework, such as texture, contrast, depth, and key point features, is also part of future work. It is also worth mentioning that the sequential ordering of the features needs to be selected in an "a priori" phase where the algorithm is designed. We are currently exploring ways to incorporate automatic methods for selecting the most appropriate features and its ordering in function of the scenarios where the tracking is going to be applied. We also believe that it is interesting to extend the algorithm in order to deal with reciprocal dependencies between cues and avoid the assumption of sequential dependency. For this purpose, the estimation at each frame could be computed iteratively and the posterior of all of the cues might be looped back into the system for a second refinement.

Acknowledgments

This research was conducted at the Institut de Robòtica i Informàtica Industrial of the Technical University of Catalonia and Consejo Superior de Investigaciones Científicas. It was partially supported by Consolider Ingenio 2010, project CSD2007-00018, CICYT project DPI2007-614452, and IST-045062 of the European Community Union, by a fellowship from the Spanish Ministry of Science and Technology, and by grants from the US Department of Justice (2004-DD-BX-1224), Department of Energy (MO-068), and US National Science Foundation (ACI-0313184 and IIS-0527585).

References

[1] Y. Bar-Shalom, X.R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, 2001.

[2] S. Birchfield, "Elliptical Head Tracking Using Intensity Gradients and Color Histograms," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 232-237, 1998.

[3] K. Branson and S. Belongie, "Tracking Multiple Mouse Contours (without Too Many Samples)," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 1039-1046, 2005.

[4] T. Darrel, G. Gordon, M. Harville, and J. Woodfill, "Integrated Person Tracking Using Stereo, Color, and Pattern Detection," *Int'l J. Computer Vision,* vol. 37, no. 2, pp. 175-185, 2000.

[5] *Sequential Monte Carlo in Practice,* A. Doucet, N. de Freitas, and N. Gordon, eds. Springer, 2001.

[6] K. Fukunaga, *Introduction to Statistical Pattern Recognition,* second ed. Academic Press, 1990.

[7] G. Hager and P. Belhumeur, "Efficient Region Tracking with Parametric Models of Geometry and Illumination," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 10, pp. 1125-1139, Oct. 1998.

[8] E. Hayman and J.O. Eklundh, "Probabilistic and Voting Approaches to Cue Integration for Figure-Ground Segmentation," *Proc. Seventh European Conf. Computer Vision,* pp. 469-486, 2002.

[9] M. Isard and A. Blake, "Condensation-Conditional Density Propagation for Visual Tracking," *Int'l J. Computer Vision,* vol. 29, no. 1, pp. 5-28, 1998.

[10] M. Isard and A. Blake, "ICondensation: Unifying Low-Level and High-Level Tracking in a Stochastic Framework," *Proc. Fifth European Conf. Computer Vision,* pp. 893-908, 1998.

[11] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Trans. ASME-J. Basic Eng.,* pp. 35-45, 1960.

[12] Z. Khan, T. Balch, and F. Dellaert, "A Rao-Blackwellized Particle Filter for Eigentracking," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 980-986, 2004.

[13] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *Int'l J. Computer Vision,* vol. 1, pp. 321-331, 1987.

[14] S. Khan and M. Shah, "Object Based Segmentation of Video Using Color, Motion, and Spatial Information," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 746-751, 2001.

[15] I. Leichter, M. Lindenbaum, and E. Rivlin, "A Probabilistic Framework for Combining Tracking Algorithms," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 445-451, 2004.

[16] J. MackCormick and A. Blake, "Probabilistic Exclusion and Partitioned Sampling for Multiple Object Tracking," *Int'l J. Computer Vision,* vol. 39, pp. 57-71, 2000.

[17] J. MacCormick and M. Isard, "Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracking," *Proc. Sixth European Conf. Computer Vision,* vol. 2, pp. 3-19, 2000.

[18] J. Malik, S. Belongie, J. Shi, and T. Leung, "Textons, Contours and Regions: Cue Integration in Image Segmentation," *Proc. Int'l Conf. Computer Vision,* pp. 918-925, 1999.

[19] F. Moreno-Noguer, A. Sanfeliu, and D. Samaras, "Integration of Conditionally Dependent Object Features for Robust Figure/Background Segmentation," *Proc. 10th Int'l Conf. Computer Vision,* pp. 1713-1720, 2005.

[20] F. Moreno-Noguer, A. Sanfeliu, and D. Samaras, "A Target Dependent Colorspace for Robust Tracking," *Proc. 18th Int'l Conf. Pattern Recognition,* vol. 3, pp. 43-46, 2006.

[21] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "An Adaptive Color-Based Particle Filter," *Image and Vision Computing,* vol. 2, no. 1, pp. 99-110, 2003.

[22] C. Rasmussen and G.D. Hager, "Probabilistic Data Association Methods for Tracking Complex Visual Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 6, pp. 560-576, June 2001.

[23] H. Sidenbladh, M.J. Black, and D.J. Fleet, "Stochastic Tracking of 3D Human Figures Using 2D Image Motion," *Proc. Sixth European Conf. Computer Vision,* pp. 702-718, 2000.

[24] M. Spengler and B. Schiele, "Towards Robust Multi-Cue Integration for Visual Tracking," *Machine Vision and Applications,* vol. 14, no. 1, pp. 50-58, 2003.

[25] P. Torr, R. Szelinski, and P. Anandan, "An Integrated Bayesian Approach to Layer Extraction from Image Sequences," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 3, pp. 297-303, Mar. 2001.

[26] K. Toyama and E. Horvitz, "Bayesian Modality Fusion: Probabilistic Integration of Multiple Vision Cues for Head Tracking," *Proc. Fourth Asian Conf. Computer Vision,* 2000.

[27] J. Triesch and C. von der Malsburg, "Democratic Integration: Self-Organized Integration of Adaptive Cues," *Neural Computation,* vol. 13, no. 9, pp. 2049-2074, 2001.

[28] Y. Wu and T.S. Huang, "Robust Visual Tracking by Integrating Multiple Cues Based on Co-Inference Learning," *Int'l J. Computer Vision,* vol. 58, no. 1, pp. 55-71, 2004.

Francesc Moreno-Noguer received the MS degree in industrial engineering from the Technical University of Catalonia in 2001, the MS degree in electronic engineering from the University of Barcelona in 2002, and the PhD degree (with highest honors) from the Technical University of Catalonia in 2005. In 2006, he was a research scientist at Columbia University. Currently, he is a postdoctoral researcher in the Computer Vision Laboratory at the École Polytechnique Fédérale de Lausanne. His research interests include robust techniques for 2D/3D camera tracking and computational cameras, with applications in both computer vision and graphics.

Alberto Sanfeliu received the BSEE and PhD degrees from the Technical University of Catalonia (UPC), Spain, in 1978 and 1982, respectively. He joined the faculty of UPC in 1981, where he is currently a full professor of computational sciences and artificial intelligence. He is the director of the Automatic Control Department at UPC, director of the Artificial Vision and Intelligent System Group (VIS), and past president of AERFAI. He is doing research at the Institut de Robòtica i Informàtica Industrial (IRI). He has worked on various theoretical aspects on pattern recognition, computer vision, and robotics and on applications on vision defect detection, tracking, object recognition, robot vision, and SLAM. He has several patents on quality control based on computer vision. He has authored books in pattern recognition and SLAM and published more than 200 papers. He is (or has been) a member of the editorial boards of *Computer Vision and Image Understanding*, the *International Journal on Pattern Recognition and Artificial Intelligence*, *Pattern Recognition Letters*, *Computación y Sistemas*, and *Electronic Letters on Computer Vision*. He received the technology prize given by the Generalitat de Catalonia. He is a fellow of the International Association for Pattern Recognition. He is member of the IEEE.

Dimitris Samaras received the diploma degree in computer science and engineering from the University of Patras in 1992, the MSc degree in computer science from Northeastern University in 1994, and the PhD degree from the University of Pennsylvania in 2001. He is currently an associate professor in the Department of Computer Science at Stony Brook University, where he has been teaching since 2000. He specializes in deformable model techniques for 3D shape estimation and motion analysis, illumination modeling and estimation for recognition and graphics, and biomedical image analysis. He is a member of the ACM and the IEEE.

▹ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.