Integration of Conditionally Dependent Object Features for Robust Figure/Background Segmentation

Francesc Moreno-Noguer, Alberto Sanfeliu
Institut de Robòtica i Informàtica, UPC-CSIC
Llorens Artigas 4-6, 08028, Barcelona, Spain
fmoreno,asanfeliu@iri.upc.es

Dimitris Samaras
Computer Science Department
SUNY Stony Brook, NY 11794-4400, USA
samaras@cs.sunysb.edu

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 description using particle filters. Previous approaches assume independence of the object cues or apply the particle filter formulation to only one of the features, and assume a smooth change in the rest, which can prove is very limiting, especially when the state of some features needs to be updated using other cues or when their dynamics follow non-linear and unpredictable paths. Our technique offers a general framework to model the probabilistic relationship between features. The proposed method is analytically justified and applied to develop a robust tracking system that adapts online and simultaneously the colorspace where the image points are represented, the color distributions, and the contour of the object. Results with synthetic data and real video sequences demonstrate the robustness and versatility of our method.

1. Introduction

The integration of several visual features has been commonly used to improve the performance of techniques for figure-ground segmentation and tracking in video sequences [1, 2, 4, 5, 11, 8, 15, 17, 16, 18]. However, most of these methods suffer from the lack of a robust dynamic model that can adapt the state of the features and cope with abrupt and unexpected changes of the target's position or appearance. Particle filters have been demonstrated to be sufficiently robust to track complex dynamics. In computer vision tasks, particle filters have been restricted to adapt geometric object cues, such as position or shape [6, 13, 14]. In [12] particle filters were used effectively to parameterize and predict object color distributions.

In this work, we introduce a probabilistic framework that integrates several object cues, allowing to robustly segment the object from the rest of the image, in dynamically changing sequences as demonstrated in the results section. In order to predict these kind of complex dynamics, each one of the features is represented by a different particle filter, and we consider a conditional dependence between cues. A similar approach is presented in [9], where several particle filter algorithms are integrated for tracking tasks. However, [9] assumes that the methods are conditionally independent, i.e, each algorithm estimates the state of a target feature based on some measurements which are conditionally independent of the measurements used by the other algorithms. That is, if particle filter $\mathcal{PF}_1$ is based on measurements $\mathbf{z}_1$ to estimate the state vector $\mathbf{x}_1$ (representing one object feature) and particle filter $\mathcal{PF}_2$ 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 $p(\mathbf{z}_1, \mathbf{z}_2|\mathbf{X}) = p(\mathbf{z}_1|\mathbf{x}_1)p(\mathbf{z}_2|\mathbf{x}_2)$.

Nevertheless, this assumption is very restrictive and many times is not satisfied. For instance, a usual method to weigh the samples of a contour particle filter, is based on the ratio of the number of pixels inside the contour with object color versus the number of pixels outside the contour with 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 of color features and $\mathbf{z}_2$ the corresponding for the contour, the latter will be a function of both $\mathbf{x}_1$ and $\mathbf{z}_1$, i.e., $\mathbf{z}_2 = \mathbf{z}_2(\mathbf{x}_1, \mathbf{z}_1)$, and the previous equation should be rewritten as $p(\mathbf{z}_1, \mathbf{z}_2|\mathbf{X}) = p(\mathbf{z}_1|\mathbf{x}_1)p(\mathbf{z}_2|\mathbf{z}_1, \mathbf{x}_1, \mathbf{x}_2)$. This formulation allows to simultaneously adapt both features, performing more robustly than the 'independent' case.

In a closely related work presented in [10], the common association of the best samples for each feature is not guaranteed, in contrast to our method. Furthermore, our formulation is more general, in the sense that any individual module represented by a particle filter, could be substituted by another algorithm providing a probability density function (*pdf*), for instance any algorithm using Kalman filtering.

Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)
1550-5499/05 $20.00 © 2005 **IEEE**

The main contributions of our work are the following:

1. We propose a probabilistic framework to fuse several conditionally dependent cues. In particular, the state of the features is represented by *pdf*'s, and approximated using particle filters, although the method is general for any algorithm that outputs a *pdf*. The proposed framework is theoretically proven and validated in a tracking example of synthetically generated data.
2. The method is applied to develop a robust tracking system that simultaneously: **(a)** Adapts the colorspace where image points are represented. **(b)** Updates the distributions of the object and background colorpoints. **(c)** Accommodates the contour of the object.
3. The representation of the color feature by particle filters and the online adaption of the colorspace are novel contributions of our work and make our system capable to track objects in complex and highly cluttered environments, altered by unexpected changes of color.

The rest of the paper is organized as follows: Section 2 introduces the mathematical framework. Section 3 demonstrates the operation of the algorithm, in a simple example for 1D cues . The features used in the real tracker, and their dynamic models are described in Section 4. Section 5 depicts details about the complete tracking algorithm. Results and conclusions are given in Sections 6 and 7.

2. Mathematical framework

In the general case, let us describe the object being tracked by a set of F features, $\mathbf{x}_1, \ldots, \mathbf{x}_F$, that are sequentially conditionally dependent, i.e., feature i depends on feature $i-1$. These features have an associated set of measurements $\mathbf{z}_1, \ldots, \mathbf{z}_F$. The conditional a posteriori probability $p_1 = p(\mathbf{x}_1|\mathbf{z}_1), \ldots, p_F = p(\mathbf{x}_F|\mathbf{z}_F)$ is estimated using a corresponding particle filter $\mathcal{PF}_1, \ldots, \mathcal{PF}_F$. For the whole set of variables we assume that the dependence is only in one direction:

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

Considering this dependence relationship we can add extra terms to the a posteriori probability computed for each particle filter. In particular, the expression for the a posteriori probability computed by $\mathcal{PF}_i$ 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, next we will prove that the whole a posteriori probability can be computed sequentially, as follows:

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

Proof. We will prove this by induction, and use of the Bayes' rule [3] and the assumptions from Eq. 1:

• Proof for 2 features:

$$\begin{aligned} &p(\mathbf{x}_1, \mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2) = \frac{p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{z}_1, \mathbf{z}_2)}{p(\mathbf{z}_1, \mathbf{z}_2)} = \frac{p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}_1, \mathbf{z}_2)p(\mathbf{x}_1, \mathbf{z}_1, \mathbf{z}_2)}{p(\mathbf{z}_1, \mathbf{z}_2)} \\ &= p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}_1, \mathbf{z}_2)p(\mathbf{x}_1|\mathbf{z}_1, \mathbf{z}_2) = p(\mathbf{x}_1|\mathbf{z}_1)p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}_1, \mathbf{z}_2) \end{aligned}$$

• For $F-1$ features we assume that

$$\begin{aligned} &p(\mathbf{x}_1, \ldots, \mathbf{x}_{F-1}|\mathbf{z}_1, \ldots, \mathbf{z}_{F-1}) = p(\mathbf{x}_1|\mathbf{z}_1)p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}_1, \mathbf{z}_2) \\ &\cdots p(\mathbf{x}_{F-1}|\mathbf{x}_1, \ldots, \mathbf{x}_{F-2}, \mathbf{z}_1, \ldots \mathbf{z}_{F-1}) \end{aligned} \qquad (3)$$

• Proof for F features:

$$\begin{aligned} &p(\mathbf{x}_1, \ldots, \mathbf{x}_F|\mathbf{z}_1, \ldots, \mathbf{z}_F) = \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_F, \mathbf{z}_1, \ldots, \mathbf{z}_F)}{p(\mathbf{z}_1, \ldots, \mathbf{z}_F)} \\ &= \frac{p(\mathbf{x}_F|\mathbf{x}_1, \ldots, \mathbf{x}_{F-1}, \mathbf{z}_1, \ldots, \mathbf{z}_F)p(\mathbf{x}_1, \ldots, \mathbf{x}_{F-1}|\mathbf{z}_1, \ldots, \mathbf{z}_F)p(\mathbf{z}_1, \ldots, \mathbf{z}_F)}{p(\mathbf{z}_1, \ldots, \mathbf{z}_F)} \\ &= p(\mathbf{x}_F|\mathbf{x}_1, \ldots, \mathbf{x}_{F-1}, \mathbf{z}_1, \ldots, \mathbf{z}_F)p(\mathbf{x}_1, \ldots, \mathbf{x}_{F-1}|\mathbf{z}_1, \ldots, \mathbf{z}_{F-1}) \\ &\overset{Eq.\ 3}{=} p(\mathbf{x}_1|\mathbf{z}_1)p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}_1, \mathbf{z}_2) \cdots p(\mathbf{x}_F|\mathbf{x}_1, \ldots, \mathbf{x}_{F-1}, \mathbf{z}_1, \ldots \mathbf{z}_F) \end{aligned}$$

□

Eq. 2 tells us that the whole a posteriori probability density function can be computed sequentially, starting with $\mathcal{PF}_1$ to generate $p(\mathbf{x}_1|\mathbf{z}_1)$ which is then used to estimate $p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}_1, \mathbf{z}_2)$ with $\mathcal{PF}_2$, and so on.

Until now we have only considered the fusion of several particle filters from the static point of view. But in the iterative performance of the method, $\mathcal{PF}_i$ 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 $\mathcal{PF}_i$ as $p_i^{(t)} = p(\mathbf{x}_i^{(t)}|\mathbf{x}_1^{(t)}, \ldots, \mathbf{x}_{i-1}^{(t)}, \mathbf{z}_1^{(t)}, \ldots, \mathbf{z}_i^{(t)}, p_i^{(t-1)})$. We expand the expression of the complete *pdf* from Eq. 2:

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

We now describe in detail the update procedure of the $i-th$ particle filter, $\mathcal{PF}_i$. At time t, the filter receives $p_i^{(t-1)}$, the *pdf* of the state vector $\mathbf{x}_i$ at time $t-1$. This distribution is approximated by a set of samples $\mathbf{s}_{ij}^{(t-1)}$, $j = 1, \ldots, N_i$, with associated weights $\pi_{ij}^{(t-1)}$. Given the set $\{\mathbf{s}_{ij}^{(t-1)}, \pi_{ij}^{(t-1)}\}$ the value of $p_i^{(t)}$ is estimated using the standard particle filter procedure:

1. The set $\{\mathbf{s}_{ij}^{(t-1)}, \pi_{ij}^{(t-1)}\}$, $j = 1, \ldots, N_i$ is resampled (sampling with replacement) according to the weights $\pi_{ij}^{(t-1)}$. We obtain the new set $\{\mathbf{s'}_{ij}^{(t-1)}, \pi_{ij}^{(t-1)}\}$.
2. Based on a probabilistic dynamic model, particles $\mathbf{s'}_{ij}^{(t-1)}$ are propagated, providing the new set $\{\mathbf{s}_{ij}^{(t)}\}$, $j = 1, \ldots, N_i$.
3. Finally, using some external measure on the feature $\mathbf{z}_i^{(t)}$ (updated with the values of the set of features $\{\mathbf{z}_k^{(t)}\}$, $k < i$ and its corresponding state vectors $\{\mathbf{x}_k^{(t)}\}$) , samples $\mathbf{s}_{ij}^{(t)}$ are weighted in order to obtain the output of iteration t, that is $\{\mathbf{s}_{ij}^{(t)}, \pi_{ij}^{(t)}\}$, $j = 1, \ldots, N_i$, approximating $p_i^{(t)}$.

To make all the mathematical foundations more clear, in the next Section we will apply this method for a simulated case, with only two one-dimensional particle filters.

Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)
1550-5499/05 $20.00 © 2005 **IEEE**

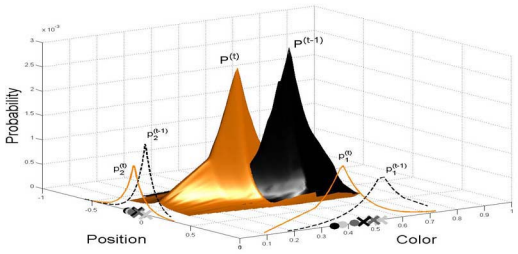


Figure 1. A posteriori *pdf*s that take part in one iteration of the method: $p_1^{(t-1)}$, $p_2^{(t-1)}$ and $P^{(t-1)}$ are the input *pdf's* and $p_1^{(t)}$, $p_2^{(t)}$ and $P^{(t)}$ are the output *pdf's* of the iteration. Crosses and circles represent the data values at time $t-1$ and t, respectively. The gray level of crosses and circles indicates if the data corresponds to the real value (black), to the estimation done by the filter (dark gray) or the to observation (light gray).

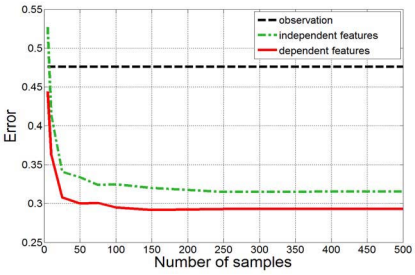


Figure 2. Errors obtained when estimating the state of the point (color and position), considering that both features are independent or dependent. Observe that the error is reduced when considering cue dependence.

3. 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 in a one-dimensional space, that is, the point is moving on the horizontal axis, between the $[-1, 1]$ coordinates, and the color is also represented by a single value in the $[0, 1]$ interval. The movement of the point is simulated with a random dynamic model (centered in μ_{pos} and scaled by α_{pos}). Furthermore, we simulate an observation model, adding gaussian noise to the simulated position :

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

Similar equations generate the models for color change and for observation.

We will use $\mathcal{PF}_1$ to track the color, with $\mathbf{x}_1$ and $\mathbf{z}_1$ representing the color state vector and its measurement, and $\mathcal{PF}_2$, $\mathbf{x}_2$ and $\mathbf{z}_2$ the corresponding particle filter, state vector and measurement assigned to the position. At the starting point of iteration t, $\mathcal{PF}_1$ receives at its input $p_1^{(t-1)}$, the *pdf* of the color at time $t-1$, approximated with N_1 weighted samples $\{\mathbf{s}_{1j}^{(t-1)}, \pi_{1j}^{(t-1)}\}$, $j = 1, \ldots, N_1$. This set is resampled and propagated according to a random dynamic model of Gaussian random noise: $\mathbf{s}_{1j}^{(t)} = \mathbf{s'}_{1j}^{(t-1)} + \mathcal{N}(0, \sigma_{dyn,col})$, where $\mathbf{s'}_{1j}^{(t-1)}$ are the resampled particles.

Each one of these propagated samples is weighted taking into account its proximity to the observed value of the color: $\pi_{1j}^{(t)} \sim e^{-(\|\mathbf{s}_{1j}^{(t)} - obs\_col^{(t)}\|)}$.

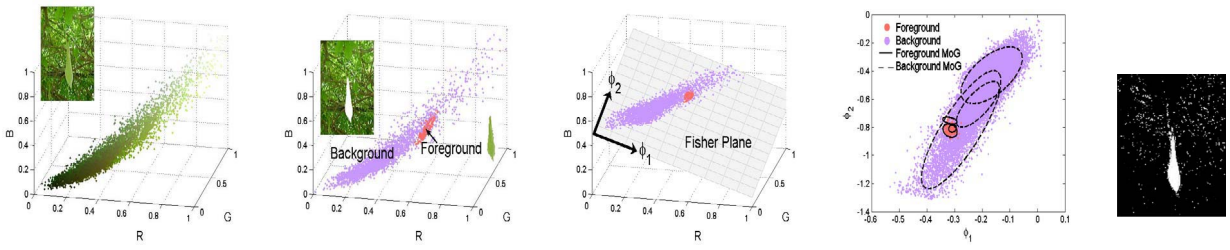
The set $\{\mathbf{s}_{1j}^{(t)}, \pi_{1j}^{(t)}\}$, $j = 1, \ldots, N_1$, is the output of the $\mathcal{PF}_1$ and represents an approximation to the a posteriori probability distribution $p_1^{(t)}$. This *pdf*, jointly with $p_2^{(t-1)}$ feeds into $\mathcal{PF}_2$, the particle filter responsible 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 $\{\mathbf{s}_{2j}^{(t-1)}, \pi_{2j}^{(t-1)}\}, j = 1, \ldots, N_2$, that are resampled and propagated using a random Gaussian dynamic model: $\mathbf{s}_{2j}^{(t)} = \mathbf{s'}_{2j}^{(t-1)} + \mathcal{N}(0, \sigma_{dyn,pos})$.

In real trackers, it is common to evaluate several target positions based on some appearance measure of the object, in our case, color. So we will proceed in a similar way for the weighting stage. To each sample $\mathbf{s}_{2j}^{(t)}$, representing a position of the point in space, we associate a sample $\mathbf{s}_{1k}^{(t)}$, representing a color state in the color-space, based on the weight $\pi_{1k}^{(t)}$. This means, that those color samples having higher weights (high probability) will be used more times than those having low probability. Finally, the weight assigned to each sample $\mathbf{s}_{2j}^{(t)}$ is computed with the function $\pi_{2j}^{(t)} \sim e^{-(\|\mathbf{s}_{1k}^{(t)} - obs\_col^{(t)}\| + \|\mathbf{s}_{2j}^{(t)} - obs\_pos^{(t)}\|)}$.

The set $\{\mathbf{s}_{2j}^{(t)}, \pi_{2j}^{(t)}\}$, $j = 1, \ldots, N_2$, represents the approximation to $p_2^{(t)}$, and we can compute the complete a posteriori probability of the system at time t by $P^{(t)}(\mathbf{x}_1, \mathbf{x}_2|\mathbf{z}_1, \mathbf{z}_2) = p_1^{(t)} p_2^{(t)}$.

In Fig. 1 we show all the a posteriori *pdf*'s in the 'color-position' space, that are generated in one iteration. Fig. 2 compares the error obtained when estimating the state of the point assuming that its features are either dependent or independent. To represent the independence of the features, color samples $\mathbf{s}_{1j}^{(t)}$ distributed uniformly over the color space, are assigned to the position samples $\mathbf{s}_{2k}^{(t)}$ independently of their weight $\pi_{1j}^{(t)}$, that is, we have no a priori knowledge about the state of the color variable when computing the weight of the position samples. From Fig. 2 we conclude that although cue integration (both in the independent and dependent case) corrects the observation, it is better to assume feature dependence.

Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)
1550-5499/05 $20.00 © 2005 **IEEE**



(a) (b) (c) (d) (e)

Figure 3. Color model. (a) Representation of all image points in the RGB colorspace. In the upper left corner of the figure the original image is shown. (b) Manual classification of image points in foreground ($\mathcal{O}$) and background ($\mathcal{B}$). (c) Projection of $\mathcal{O}$ and $\mathcal{B}$ on the Fisher plane. (d) *MoG* components of $\mathcal{O}$ (central leaf) and $\mathcal{B}$ in the Fisher colorspace. (e) $p\left(\mathcal{O}|\mathbf{c}^{Fisher}\right)$. Brighter points are more likely pixels.

4. Features used for robust tracking

In order to design a system able to work in real, dynamic environments we define a set of cues including both appearance (normal to the Fisher plane [12] and the color distribution of the object) and geometric attributes (contour) of the object. Next we describe these features:

4.1. Normal to the Fisher plane

In [12] the concept of Fisher colorspace was introduced, and it was suggested that for tracking purposes the best colorspace is the one that maximizes the distance between the object and background colorpoints. Let the sets $\mathcal{C}_{\mathcal{O}}^{RGB} = \{\mathbf{c}_{\mathcal{O},i}^{RGB}\}$, $i = 1, \ldots, N_{\mathcal{O}}$ and $\mathcal{C}_{\mathcal{B}}^{RGB} = \{\mathbf{c}_{\mathcal{B},j}^{RGB}\}$, $j = 1, \ldots, N_{\mathcal{B}}$ be the colorpoints of the object and background respectively, represented in the 3D RGB colorspace. We define as the optimal colorspace as the one resulting from the projection of the RGB colorpoints on the plane $\Phi = [\phi_1, \phi_2] \in \mathcal{M}_{3\times 2}$ (Fisher plane), computed by applying the nonparametric Linear Discriminant Analysis technique [3] over the sets $\mathcal{C}_{\mathcal{O}}^{RGB}$ and $\mathcal{C}_{\mathcal{B}}^{RGB}$. An RGB colorpoint $\mathbf{c}^{RGB}$ is transformed to the 2D Fisher colorspace by $\mathbf{c}^{Fisher} = \Phi^T \mathbf{c}^{RGB}$ (see Fig. 3).

The Fisher plane is adapted online, through the particle filter formulation presented above, with the following state vector and dynamic model:

• **State vector:** The Fisher plane is parameterized by its normal vector, $\mathbf{x}_1 = \phi_1 \times \phi_2$.

• **Dynamic model:** A sample $\mathbf{s}'^{(t-1)}_{1,j}$ of the state vector $\mathbf{x}_1$ is propagated using the random dynamic model, $\mathbf{s}^{(t)}_{1,j} = \mathcal{H}_1 \mathbf{s}'^{(t-1)}_{1,j} + \mathbf{p}_1$, where $\mathcal{H}_1 \sim \mathcal{A}_{3\times 3}(0, \sigma_{H_1})$ and $\mathbf{p}_1 \sim \mathcal{T}_{3\times 1}(\mu_{p_1}, \sigma_{p_1})$, and where:

$$\mathcal{A}_{m\times m}(\mu_A, \sigma_A) = \begin{bmatrix} 1+a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & 1+a_{mm} \end{bmatrix} \qquad (5)$$
$$\mathcal{T}_{m\times 1}(\mu_t, \sigma_t) = [t_1, \ldots, t_m]^T$$

Variables a_{ij} and t_i are approximated by normal random values, $a_{ij} \sim \mathcal{N}\left(\mu_{A_{ij}}, \sigma_{A_{ij}}\right)$, $t_i \sim \mathcal{N}\left(\mu_{t_i}, \sigma_{t_i}\right)$.

4.2. Color distribution of target and background

In order to represent the color distribution of the foreground and background in the Fisher colorspace, we use a *Mixture of Gaussians* (*MoG*) model. With this model, the conditional probability for a pixel $\mathbf{c}^{Fisher}$ belonging to a multi-colored object $\mathcal{O}$ is expressed as a sum of M_o Gaussian components: $p\left(\mathbf{c}^{Fisher}|\mathcal{O}\right) = \sum_{j=1}^{M_o} p\left(\mathbf{c}^{Fisher}|j\right) P\left(j\right)$. Similarly, the background color will be represented by a mixture of M_b gaussians. Given the foreground ($\mathcal{O}$) and background ($\mathcal{B}$) classes, the a posteriori probability that a pixel $\mathbf{c}^{Fisher}$ belongs to object $\mathcal{O}$ is computed using the Bayes rule (Fig. 3d,e):

$$p\left(\mathcal{O}|\mathbf{c}^{Fisher}\right) = \frac{p\left(\mathbf{c}^{Fisher}|\mathcal{O}\right) P\left(\mathcal{O}\right)}{p\left(\mathbf{c}^{Fisher}|\mathcal{O}\right) P\left(\mathcal{O}\right) + p\left(\mathbf{c}^{Fisher}|\mathcal{B}\right) P\left(\mathcal{B}\right)} \qquad (6)$$

where $P\left(\mathcal{O}\right)$, $P\left(\mathcal{B}\right)$ represent the a priori probabilities of $\mathcal{O}$ and $\mathcal{B}$, respectively. The state vector and dynamic model for the color distribution feature are:

• **State vector:** The configurations of the *MoG* for $\mathcal{O}$ and $\mathcal{B}$ are parameterized by the vector $\mathcal{G}_\varepsilon = [\mathbf{p}_\varepsilon, \mu_\varepsilon, \lambda_\varepsilon, \theta_\varepsilon]$ where $\varepsilon = \{\mathcal{O}, \mathcal{B}\}$, $\mathbf{p}_\varepsilon$ contains the priors for each Gaussian component, μ_ε the centroids, λ_ε the eigenvalues of the principal directions and θ_ε the angles between the principal directions and the horizontal. $\mathbf{x}_2 = \{\mathcal{G}_\mathcal{O}, \mathcal{G}_\mathcal{B}\}$ will be the state vector representing the color model.

• **Dynamic model:** For each one of the components of a state vector sample $\mathbf{s}'^{(t-1)}_{2,j} = \{\mathbf{s}'^{(t-1)}_{2\mathcal{O},j}, \mathbf{s}'^{(t-1)}_{2\mathcal{B},j}\}$ we apply the dynamic model: $\mathbf{s}'^{(t)}_{2\varepsilon,j} = \mathcal{H}_2 \mathbf{s}'^{(t-1)}_{2\varepsilon,j} + \mathbf{p}_2$, where $\mathcal{H}_2 \sim \mathcal{A}_{6N_\varepsilon\times 6N_\varepsilon}(0, \sigma_{H_2})$, $\mathbf{p}_2 \sim \mathcal{T}_{6N_\varepsilon\times 1}(\mu_{p_2}, \sigma_{p_2})$, and N_ε is the number of gaussian components of the color distribution of ε. Matrix $\mathcal{A}$ and vector $\mathcal{T}$ are defined in Eq. 5.

4.3. Contour of the object

Since color segmentation usually gives a rough estimation of the object location, we use the contour of the object, to obtain a more precise tracking. The state vector and dynamic model used to represent the contour are:

• **State vector:** The contour will be represented by N_c

Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)
1550-5499/05 $20.00 © 2005 **IEEE**

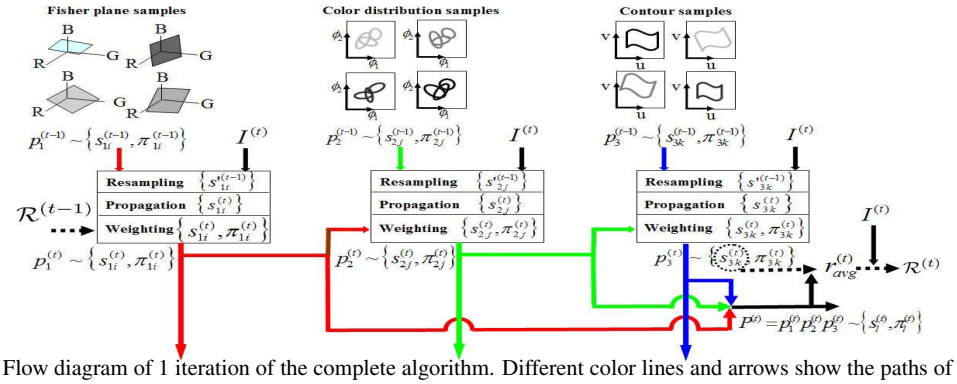


Figure 4. Flow diagram of 1 iteration of the complete algorithm. Different color lines and arrows show the paths of each feature. Observe how the output of each particle filter, feeds into a subsequent particle filter.

points in the image, $\mathbf{r} = [(u_1, v_1), \ldots, (u_{N_c}, v_{N_c})]^T$. We assign these values to the state vector, $\mathbf{x}_3 = \mathbf{r}$.

• **Dynamic model:** A contour sample $\mathbf{s}'^{(t-1)}_{3,j}$ is propagated according to the random affine deformation, $\mathbf{s}^{(t)}_{3,j} = \mathcal{H}_3 \mathbf{s}'^{(t-1)}_{3,j} + \mathbf{p}_3$, where $\mathcal{H}_3 \sim \mathcal{A}_{2\times2}(0, \sigma_{H_3})$ and $\mathbf{p}_3 \sim \mathcal{T}_{2\times1}(\mu_{p_3}, \sigma_{p_3})$.

A note about the parameters of the dynamic models. *In all the dynamic models defined above there are certain parameters ($\{\sigma_{H_i}, \sigma_{p_i}, \mu_{p_i}\}$, $i = \{1, 2, 3\}$) that control the random performance of the model. Their value will determine the level of dispersion of the samples in the state vector space, and have been learned off-line from a training hand-segmented sequence using a least squares technique.*

5. The complete tracking algorithm

This Section integrates the tools described previously and analyzes in detail the complete method for tracking rigid and non-rigid objects in cluttered environments, under changing illumination. Let us describe the algorithm step by step (for a better understanding of the method, the reader is encouraged to follow the flow diagram in Fig. 4):

5.1. Input at iteration t

At time t, for each i-feature, $i = \{1, 2, 3\}$, a set of N_i samples $\mathbf{s}^{(t-1)}_{ij}$, $j = 1, \ldots, N_i$, 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 $\pi^{(t-1)}_{ij}$. 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 Eq. 4, where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ contains the state vectors parameterizing the features, and $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3\}$ refers to the observations measured to evaluate each one of the cues. Also available is the set of image points $\mathcal{R}^{(t-1)}$ that discretize the contour of the object, and the input RGB image at time t, $I^{RGB,(t)}$.

5.2. Updating the Fisher plane *pdf*

As described in Section 3, at the starting point of iteration t, $\mathcal{PF}_1$, the particle filter associated to $\mathbf{x}_1$, receives $p^{(t-1)}_1$, the *pdf* of the state vector $\mathbf{x}_1$ at time $t-1$, approximated by N_1 weighted samples $\{\mathbf{s}^{(t-1)}_{1j}, \pi^{(t-1)}_{1j}\}$, $j = 1, \ldots, N_1$. These particles are resampled and propagated to the set $\{\mathbf{s}^{(t)}_{1j}\}$ according to the dynamic model defined in Section 4.1 (see Fig. 5). Each sample represents a different Fisher plane, Φ_j, $j = 1, \ldots, N_1$. In order to assign a weight to each propagated sample, we define a region W in the image $I^{RGB,(t)}$, where we expect the object will be. This region does not need to be estimated with precision, it can be just a bounding box around $\mathcal{R}^{(t-1)}$ with a tolerance big enough to cope with unexpected object movements.

Once we have defined W, the basic idea is to weight each Fisher plane Φ_j depending on how well it discriminates the points inside W from the points outside W. To this end we select randomly two sets of RGB colorpoints, $\mathcal{C}^{RGB}_W$ and $\mathcal{C}^{RGB}_{\overline{W}}$, inside and outside W, respectively. These sets and the image $I^{RGB,(t)}$ are projected on the N_j Fisher planes, generating the N_j triplets $\{\mathcal{C}^{Fisher}_{W,j}, \mathcal{C}^{Fisher}_{\overline{W},j}, I^{Fisher,(t)}_j\}$. For each triplet we use the *EM* algorithm to fit a *MoG* to the sets $\mathcal{C}^{Fisher}_{W,j}$ and $\mathcal{C}^{Fisher}_{\overline{W},j}$.

Based on these *MoG*s we compute the a posteriori probability map $p(W|I^{Fisher,(t)}_j)$ for all the (u, v) pixels of image $I^{Fisher,(t)}_j$, using the Bayes rule (Eq. 6). According to this probability map, we assign the following weight to each sample:

$$\pi^{(t)}_{1j} \sim \frac{\sum_{(u,v)\in W} p\left(W|I^{Fisher,(t)}_j\right)}{N_W} - \frac{\sum_{(u,v)\notin W} p\left(W|I^{Fisher,(t)}_j\right)}{N_{\overline{W}}}$$

where N_W and $N_{\overline{W}}$ are the number of image pixels in and out of W, respectively.

The set $\{\mathbf{s}^{(t)}_{1j}, \pi^{(t)}_{1j}\}$, $j = 1, \ldots, N_1$ approximates the estimate of the a posteriori probability function $p^{(t)}_1$ for the normal to the Fisher plane.

Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)
1550-5499/05 $20.00 © 2005 **IEEE**

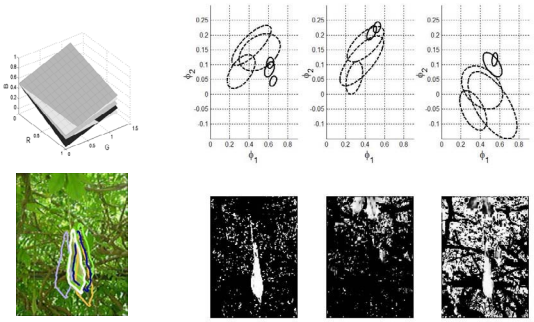


Figure 5. Generation of multiple hypotheses for each feature. Top left: Fisher plane. Bottom left: Contour of the object. Right: Color distributions (and the corresponding a posteriori *pdf*s maps).

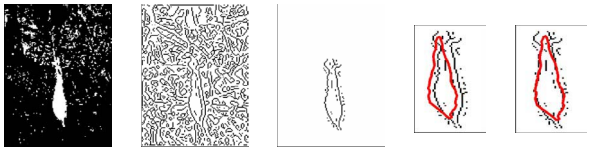
5.3. Updating the foreground and background color distributions *pdf*'s

$\mathcal{PF}_2$, the particle filter associated to the state vector $\mathbf{x}_2$, receives at its input $p_2^{(t-1)} \sim \{\mathbf{s}_{2j}^{(t-1)}, \pi_{2j}^{(t-1)}\}$, $j = 1, \ldots, N_2$, approximating the *pdf* of the color distributions in the previous iteration, and $p_1^{(t)} \sim \{\mathbf{s}_{1k}^{(t)}, \pi_{1k}^{(t)}\}$, $k = 1, \ldots, N_1$, an approximation to the *pdf* of the Fisher planes at time t. Particles $\{\mathbf{s}_{2j}^{(t-1)}\}$ are resampled and propagated (using the dynamic model associated with $\mathbf{x}_2$) to the set $\{\mathbf{s}_{2j}^{(t)}\}$. A sample $\mathbf{s}_{2j}^{(t)}$ represents a *MoG* configuration for the foreground and background. The keypoint is that now in order to assign the weight to these samples we use the information provided from the output $p_1^{(t)}$ of $\mathcal{PF}_1$. We associate a sample $\mathbf{s}_{1k}^{(t)}$ to each sample $\mathbf{s}_{2j}^{(t)}$, according to the weight $\pi_{1k}^{(t)}$, in such a way that those samples $\mathbf{s}_{1k}^{(t)}$ of Fisher planes having higher probabilities will be assigned more times to the samples $\mathbf{s}_{2j}^{(t)}$ of *MoG*s. The rest of the weighting process is similar to the one described in Sect. 5.2: for a given sample $\mathbf{s}_{2j}^{(t)}$ we project image $I^{RGB,(t)}$ to the Fisher plane Φ_j of the associated sample $\mathbf{s}_{1k}^{(t)}$. The new image will be $I_j^{Fisher,(t)} = \Phi_j^T I^{RGB,(t)}$. Using the *MoG*s of the object and background parameterized by the sample $\mathbf{s}_{2j}^{(t)}$, the a posteriori probability map $p(\mathcal{O}|I_j^{Fisher,(t)})$ is computed (see Fig. 5) for all the pixels of $I_j^{Fisher,(t)}$, and the weight $\pi_{2j}^{(t)}$ is assigned by:

$$\pi_{2j}^{(t)} \sim \frac{\sum_{(u,v)\in W} p\left(\mathcal{O}|I_j^{Fisher,(t)}\right)}{N_W} - \frac{\sum_{(u,v)\notin W} p\left(\mathcal{O}|I_j^{Fisher,(t)}\right)}{N_{\overline{W}}}$$

where W, N_W and $N_{\overline{W}}$ were defined above.

The set $\{\mathbf{s}_{2j}^{(t)}, \pi_{2j}^{(t)}\}$, $j = 1, \ldots, N_2$ approximates the estimate of the a posteriori probability function $p_2^{(t)}$ for the fore/background color distributions.



(a) (b) (c) (d) (e)

Figure 6. Simplification of the snake fitting procedure using color information. (a) Foreground a posteriori *pdf* map obtained using the color module. (b) Edge features image. (c) Refined edge image using (a). (d) Contour $\mathbf{r}_{avg}^{(t)}$ used as initialization for a snake fitting. (e) Snake fitting results.

5.4. Updating the contour *pdf*

The last particle filter, $\mathcal{PF}_3$, receives at its input $p_3^{(t-1)} \sim \{\mathbf{s}_{3j}^{(t-1)}, \pi_{3j}^{(t-1)}\}$, $j = 1, \ldots, N_3$, that approximates the *pdf* of the contours in the previous iteration, and $p_2^{(t)} \sim \{\mathbf{s}_{2k}^{(t)}, \pi_{2k}^{(t)}\}$, $k = 1, \ldots, N_2$, an approximation to the *pdf* of the color distributions of foreground and background at time t. The set $\{\mathbf{s}_{3j}^{(t)}\}$ (the resampled and propagated particles, see Fig. 5) are weighted based on $p_2^{(t)}$ through a similar process to the one described for $\mathcal{PF}_2$: first we associate a sample $\mathbf{s}_{2k}^{(t)}$ to each sample $\mathbf{s}_{3j}^{(t)}$, according to the weight $\pi_{2k}^{(t)}$. Then we use the a posteriori probability map $p(\mathcal{O}|I_j^{Fisher,(t)})$ assigned to $\mathbf{s}_{2k}^{(t)}$ in the previous step, and the contour $\mathbf{r}_j$ represented by $\mathbf{s}_{3j}^{(t)}$ to compute the weight as follows:

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

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

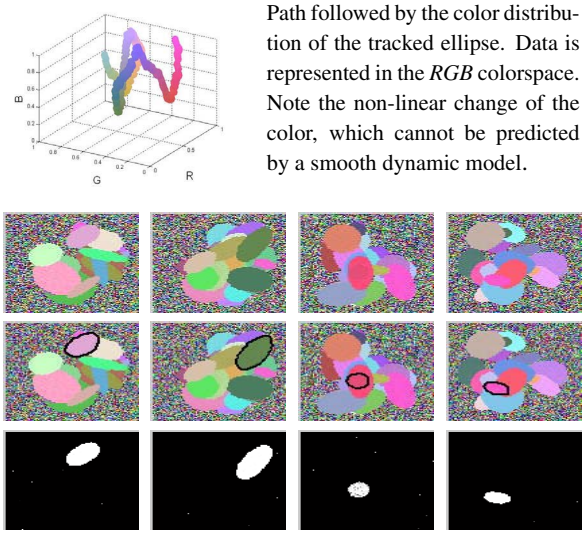
Finally, the set of samples and weights $\{\mathbf{s}_{3j}^{(t)}, \pi_{3j}^{(t)}\}$, $j = 1, \ldots, N_3$ approximate the estimate of the a posteriori probability function $p_3^{(t)}$ for the contours of the object.

5.5. Algorithm output generation

As we have demonstrated in Section 2, the complete a posteriori probability function, can be computed as

$$P^{(t)} = P^{(t)}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3|\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) = p_1^{(t)} p_2^{(t)} p_3^{(t)} \sim \left\{\left\{s_{3k}^{(t)}\left(s_{2j}^{(t)}(s_{1i}^{(t)})\right)\right\}, \left\{\pi_{1i}^{(t)}\pi_{2j}^{(t)}\pi_{3k}^{(t)}\right\}\right\} = \{s_l^{(t)}, \pi_l^{(t)}\} \quad (7)$$

where $l = 1, \ldots, N_3$. Eq. 7 reflects the fact that samples of state vector $\mathbf{x}_3$ are computed while taking into account samples of the state vector $\mathbf{x}_2$, which have been computed considering samples of $\mathbf{x}_1$. Observe that the final number of samples to approximate the whole probability $P^{(t)}$ is determined by N_3. Considering the final weights, the average contour is computed as $\mathbf{r}_{avg}^{(t)} = \sum_{l=1}^{N_3} s_{3l}^{(t)} \pi_l^{(t)}$.



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.

Figure 7. Tracking results of a synthetic sequence. 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* map of the color module (third row).

Since all the contour samples have been constructed with an affine deformation model, we need to introduce an additional final stage to cope with non-linear deformations of the object boundary. We use $\mathbf{r}_{avg}^{(t)}$ to initialize a deformable contour ($\mathcal{R}^{(t)}$) that is fitted to the contours of the object using the traditional *snake* formulation [7]. This adjustment is highly simplified by using the target position estimated by the color particle filter, as it is shown in Fig. 6, where the a posteriori probability map of the best color hypothesis allows to eliminate noisy edges from the original image.

6. 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. In the first experiment, we segment a synthetically generated sequence of an ellipse that changes randomly its position, color and shape in a cluttered background. In Fig. 7(top) we depict the path followed by the color cue. Observe the non-linearity of the trajectory. In [6] it is shown that paths of this kind can not be estimated by filters based on smooth dynamic models, but instead we need to use filters based on multihypotheses, such as particle filters. Results show that our method based on multiple-multihypotheses algorithms allows to segment and track the ellipse, even when the background has a similar color to the target (observe the frame before last).

In the second experiment (Fig. 8) we show how our method performs in a real video sequence of an octopus changing its appearance while camouflaging. Observe that the a posteriori *pdf* maps of the color module give a rough estimation of the target position (especially when the octopus appearance is quite similar to the background), which is corrected by introducing the shape module.

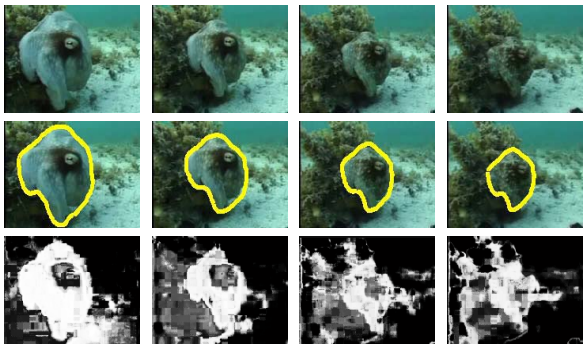


Figure 8. Tracking results of a camouflaging octopus. Top row: Original sequence. Middle row: Results using the proposed method. A posteriori *pdf* map of the color module.

In the last two experiments, we compare the performance of our algorithm to a common tracking technique [6] that uses multiple hypotheses to predict the contour of the object and accommodates the color with a smooth dynamic model written as $\mathcal{G}_t = (1-a)\,\mathcal{G}_{t-2} + a\mathcal{G}_{t-1}$, where $\mathcal{G}$ is the parameterization of the color distribution and a is a mixing factor. The first of these experiments corresponds to the tracking of the non-rigid boundary of a bending book in a video sequence, where the lighting conditions change smoothly from natural lighting to yellow lighting. Fig. 9 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 of the failure is that the smooth dynamic model cannot cope with the effect of self-shadowing produced during the movement of the book.

In the final experiment we have tested the algorithm with a sequence of a moving leaf. Although this is a challenging sequence because it is highly cluttered, the illumination changes abruptly and the target moves unpredictably, we can perform the tracking with the proposed method. Fig. 10 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 smooth color prediction.

7. Conclusions

In this paper we have presented a new technique to integrate different object features that are conditionally dependent. This framework has allowed us to design a track-

Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)
1550-5499/05 $20.00 © 2005 **IEEE**

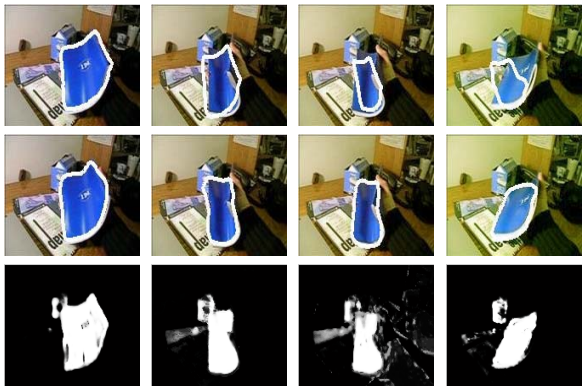


Figure 9. Tracking results of a bending book in a sequence with smooth change of illumination. Top row: Results using only a contour particle filter and assuming smooth change of color. The method fails. Middle row: Results using the proposed method. Bottom row: A posteriori *pdf* map of the color module ($\mathcal{PF}_2$).

ing algorithm that accommodates simultaneously the colorspace where the image points are represented, the color distributions of the object and background and the contour of the object. We have demonstrated the effectiveness of the method both analytically and experimentally, first tracking sequences of synthetically generated data and then tracking real sequences presenting high content of clutter, non-rigid objects, non-expected target movements and abrupt changes of illumination. Further integration of other features into the framework, such as texture, contrast or depth, and other algorithms apart from particle filters, is part of future work.

Acknowledgments

This work was partially supported by CICYT projects DPI2004-05414 and DPI2003-05193-C02-01, by a fellowship from the Spanish Ministry of Science and Technology, and by the grants from U.S Department of Justice (2004-DD-BX-1224), Department of Energy (MO-068) and National Science Foundation (ACI-0313184).

References

[1] S.Birchfield, "Elliptical head tracking using intensity gradients and color histograms", *CVPR*, pp.232-237, 1998.

[2] T.Darrel, G.Gordon, M.Harville, J.Woodfill, "Integrated person tracking using stereo, color, and pattern detection", *IJCV*, Vol.37(2), pp.175-185, 2000.

[3] K.Fukunaga, "Introduction to statistical pattern recognition", *2nd ed., Academic Press*, 1990.

[4] G.Hager, P.Belhumeur, "Efficient region tracking with parametric models of geometry and illumination", *PAMI*, Vol.20(10), pp.1125-1139, 1998.

[5] E.Hayman, J.O.Eklundh, "Probabilistic and voting approaches to cue integration for figure-ground segmentation", *ECCV*, 2002.

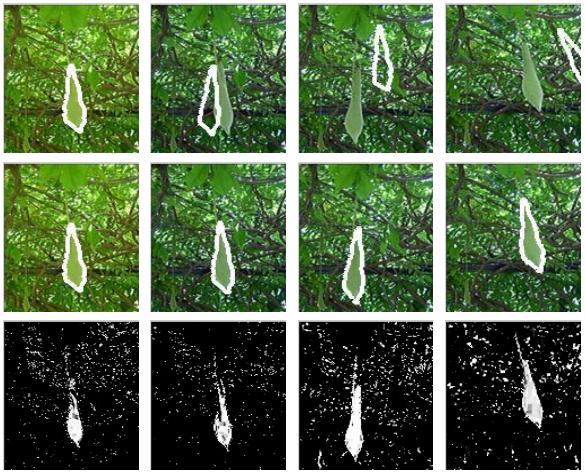


Figure 10. Tracking results of a leaf. Top row: Results using only a contour particle filter and assuming smooth change of color. The method fails. Middle row: Results using the proposed method. Bottom row: A posteriori *pdf* map of the color module ($\mathcal{PF}_2$). Observe how the tracked leaf is clearly detected.

[6] M.Isard, A.Blake, "Condensation-conditional density propagation for visual tracking", *IJCV*, Vol.29(1), pp.5-28, 1998.

[7] M.Kass, A.Witkin, D.Terzopoulos,"Snakes: Active contour models", *IJCV*, Vol.1, pp.321-331, 1987.

[8] S.Khan, M.Shah, "Object based segmentation of video using color, motion, and spatial information", *CVPR*, Vol.2, pp.746-751, 2001.

[9] I.Leichter, M.Lindenbaum, E.Rivlin, "A probabilistic framework for combining tracking algorithms", *CVPR*, 2004.

[10] J.MackCormick, A.Blake, "Probabilistic exclusion and partitioned sampling for multiple object tracking", *IJCV*, Vol.39(1), pp.57-71, 2000.

[11] J.Malik, S.Belongie, J.Shi, T.Leung, "Textons, contours and regions: Cue integration in image segmentation", *ICCV*, 1999.

[12] F.Moreno-Noguer, A.Sanfeliu, D.Samaras, "Fusion of a Multiple Hypotheses Color Model and Deformable Contours for Figure Ground Segmentation in Dynamic Enviroments", *ANM04 (in conjunction with CVPR'04)*, 2004.

[13] K.Nummiaro, E.Koller-Meier, L.Van Gool,"An adaptive color-based particle filter", *IVC*, Vol.2(1), pp.99-110, 2003.

[14] H.Sidenbladh, M.J.Black, D.J.Fleet, "Stochastic tracking of 3D human figures using 2D image motion", *ECCV*, 2000.

[15] M.Spengler, B.Schiele, "Towards robust multi-cue integration for visual tracking", *Machine Vision and Applications,* Vol. 14(1), pp. 50-58, 2003

[16] P.Torr, R.Szelinski, P.Anandan, "An integrated bayesian approach to layer extraction from image sequences", *PAMI*, Vol.23(3), pp.297-303, 2001.

[17] K.Toyama, E.Horvitz, "Bayesian modality fusion: Probabilistic integration of multiple vision cues for head tracking", *ACCV*, 2000.

[18] J.Triesch, C.von der Malsburg, "Democratic integration: self-organized integration of adaptive cues", *Neural Computation,* Vol.13(9), pp.2049-2074, 2001.