A Probabilistic Integrated Object Recognition and Tracking Framework

Francesc Serratosa[a], René Alquézar[b] & Nicolás Amézquita[a]

[a] Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, Av. Països Catalans 26, 43007 Tarragona, Spain

[b] Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens Artigas 4-6, 08028 Barcelona, Spain

Abstract. This paper describes a probabilistic integrated object recognition and tracking framework called PIORT, together with two specific methods derived from it, which are evaluated experimentally in several test video sequences. The first step in the proposed framework is a static recognition module that provides class probabilities for each pixel of the image from a set of local features. These probabilities are updated dynamically and supplied to a tracking decision module capable of handling full and partial occlusions. The two specific methods presented use RGB colour features and differ in the classifier implemented: one is a Bayesian method based on maximum likelihood and the other one is based on a neural network. The experimental results obtained have shown that, on one hand, the neural net based approach performs similarly and sometimes better than the Bayesian approach when they are integrated within the tracking framework. And on the other hand, our PIORT methods have achieved better results when compared to other published tracking methods in video sequences taken with a moving camera and including full and partial occlusions of the tracked object.

Keywords: Object tracking; object recognition; occlusion; performance evaluation; probabilistic methods; video sequences; dynamic environments

1. Introduction

One of the most general and challenging problems a mobile robot has to confront is to identify, locate and track objects that are common in its environment. To this end, object models have to be defined or learned in conjunction with some associated recognition and tracking procedures. There are several issues that have to be considered while dealing with object locating and tracking which deserve some previous discussion.

The first important issue is to determine the type of object model to learn, which usually depends on the application environment. For instance, in [1], the target was an aerial vehicle. And in [2,3,4,5] the targets were people. In [6], they used specific parameters of the object to be tracked. In [7], they track hands using textures.

In [8], they developed and implemented a real system for simultaneous localization and mapping (SLAM) algorithm for mobile robots based on an extended Kalman filter. It was applied to indoor environments and used stereo vision based on two web-cam. The system diverges from ours in that we like to track objects captured from the mobile-robot cameras, instead of localize the position of our robot.

While tracking the object, for instance people walking in the street, the system could try to recognize person the through a face recognition system. There is a lot of literature related to this field [9]. Moreover, other systems not only indentify subjects but detect the mood of these subjects or detect specific pathologies [10]. Face identification or recognition is not the scope of this paper.

Finally, other field related to object tracking is automatic hand-gesture recognition [11]. In this kind of systems, hands have to be tracked and the trajectory (position, speed, acceleration) has to be analyzed to conclude the meaning of this movement.

In our case, we want a mobile robot equipped with a camera to locate and track general objects (people, other robots, balls, wastepaper bins …) in both indoor and outdoor environments. A useful object model should be relatively simple and easy to acquire from the result of image processing steps. For instance, the result of a colour image segmentation process, consisting of a set of regions or spots, characterized by simple features related to colour, may be a good starting point to learn the model [7, 12]. Although structured models like attributed graphs or skeletons can be synthesized for each object from several segmented images [13, 14], we have decided to investigate a much simpler approach in which the object is just represented as an unstructured set of pixels. Other methods detect some characteristic points of the object to be tracked [15]. At a learning phase, the most repeatable object keypoints for the specific object are learned. Another interesting work is [16], in which the algorithm search for different region tracks. These methods have been proven to have a good performance when there is low variability of the features of the object. Nevertheless, with deformable objects, it is difficult to extract some representative points.

One of the main drawbacks of structural methods is that the segmented images can be quite different from one frame to the other, and therefore it is difficult to match the structure in the current frame with the previous ones. In [17], the model was specially designed to segment and track objects from video sequences that suffer from abrupt changes. The starting point of our approach is to accept these differences between segmented images and use a more rudimentary model in which the basic element is not the spot or region of the segmented image but its pixels. An example of structural method was reported in [14], where the object model was based on the skeleton of the object obtained in the segmented images. Since the skeletons resulting from two almost equal images can be very different, the applicability of such approach is limited. The tracking step was performed in [14] by an extension of the Kalman filter in which the skeleton and other geometrical features were considered. Other options has been [18,19] where the model specifically incorporated the relation between position and time. Finally, other methods are based on keeping the information of the silhouette of the object to be tracked. In [5], the method is based on learning a dynamic and statistical model of the silhouette of the object. In our case, we cannot use this system since we assume that the deformation of the object to be tracked is not predictable.

A second significant issue is to provide the tracking procedure with the capacity of determining occlusions and re-emergencies of tracked objects, i.e. occlusion handling. Over recent years, much research has been developed to solve the problem of object tracking under occlusions [4], because, in real-world tracking, a target being partly or entirely covered by other objects for an uncertain period of time is common. Occlusions pose two main challenges to object tracking systems. The first challenge is how to determine the beginning and the end of an occlusion. The second challenge is how to predict the location of the target during and at the end of the occlusion.

Determining occlusion status is very hard for the trackers where the only knowledge available on the target is its initial appearance. When some parts of an occluder are similar to those of the target, the occluder and the target are mistaken. Various approaches that analyze occlusion situations have been proposed. The most common one is based on background subtraction [4, 19, 20]. Although this method is reliable, yet it only works with a fixed camera and a known background, which is not our case in mobile robotics. In [4], they used several cameras, and tracking and occlusion of people is solved by a multi-view approach. In [20], they achieve real-time tracking with small images. Evidence is gathered from all of the cameras into a synergistic framework. Other approaches are based on examining the measurement error for each pixel [22, 23]. The pixels that their measurement error exceeds a certain value are considered to be occluded. These methods are not very appropriate in outdoor scenarios, where the variability of the pixel values between adjacent frames may be high. A mixture of distributions is used in [24] to model the observed value of each pixel, where the occluded pixels are characterised by having an abrupt difference with respect to a uniform distribution. Contextual information is exploited in [25, 27]. These methods have better performance in terms of analysing occlusion situations but tracking errors are observed to frequently occur and propagate away. In addition, in the case of using these approaches in a mobile robot application, there is a need of knowing a priori the robot surroundings.

Determining the re-emergence of the target and recapture its position after it is completely occluded for some time is the other main challenge. Setting a similarity threshold is one method, yet the optimal threshold value is difficult to determine. This problem is circumvented in [22], where the image region that matches the best with the template over a prefixed duration is assumed to be the reappearing target. In [23], an observation model and a velocity motion model were defined. The observation model was based on an adaptive appearance model, and the velocity motion model was derived using a first-order linear predictor. Both approaches are defined in the framework of particle filter, with provisions for handling occlusion.

In the scenarios where the motion of the target is not smooth neither predictable most of the aforementioned methods would fail. Recently, new object tracking methods that are robust to occlusion have been reported with very promising results [27, 28]. The method reported in [27] relies on background subtraction (it works only for static cameras) and a *k*-NN classifier to segment foreground regions into multiple objects using on-line samples of object's appearance local features taken before the occlusion. The method described in [28] relies on an adaptive template matching but it only handles partial occlusions and the matching process seems to be computationally costly.

A third relevant issue, which generally is not so mentioned, is to integrate the recognition and tracking steps in a common framework that helps to exploit some feedback between them. To the best of our knowledge there are few existing works that combine recognition and tracking in an integrated framework [29, 30]. Object recognition and tracking are usually performed sequentially and without any feedback from the tracking to the recognition step [14]. These tasks often are treated separately and/or sequentially on intermediate representations obtained by the segmentation and grouping algorithms [31-33]. Sometimes, they are applied in a reverse order, with a first tracking module supplying inputs to the recognition module, as, for instance, in gesture recognition [34].

An integrated framework for tracking and recognising faces was presented in [30]. Conventional video-based face recognition systems are usually embodied with two independent components: the recognition and the tracking module. In contrast, an architecture was defined in [30] that tightly couples these two components within a single framework. The complex and nonlinear appearance manifold of each registered person was partitioned into a collection of sub-manifolds where each models the face appearances of the person in nearby poses. The sub-manifolds were approximated by a low-dimensional linear subspace computed by PCAs. Finally, Artificial Intelligence was applied to tracking objects in [35].

This paper describes thoroughly and in detail the current state of a *probabilistic integrated object recognition and tracking* (PIORT) methodology that we have developed in the latest years, as well as two particular methods derived from it. It also presents a collection of experimental results in test video sequences obtained by PIORT methods and alternative tracking methods. Previous stages in the development of PIORT, together with preliminary results, have been partially reported elsewhere [36-39].

In the experimental evaluation carried out, PIORT methods have been compared to six state-of-the-art tracking methods of which we were able to get and apply their program codes to the test video sequences:

- Template Match by Correlation (TMC) [40];
- Basic Meanshift (BM) [41];
- Histogram Ratio Shift (HRS) [42];
- Variance Ratio Feature Shift (VRFS) [42];
- Peak Difference Feature Shift (PDFS) [42]; and
- Graph-Cut Based Tracker (GCBT) [43, 44].

Their codes were downloaded from the VIVID tracking evaluation web site www.vividevaluation.ri.cmu.edu, which unfortunately seems not to be accessible anymore. We briefly summarise these methods next.

In the TMC method [40], the features of the target object are represented by histograms. These histograms are regularised by an isotropic kernel which produce spatially smooth functions suitable for gradient-based optimisation. The metric used to compare these functions is based on the Bhattacharyya distance and the optimisation is performed by the mean-shift procedure.

In [41] a general non-parametric framework is presented for the analysis of a multimodal feature space and to separate clusters. The mean-shift procedure (localisation of stationary points in the distributions) is used to obtain the clusters. Throughout this framework, a segmentation application is described.

In [42] three different tracking methods are presented. They are based on the hypothesis that the best feature values to track an object are the ones that best discriminate between the object and the present background. Therefore, with several sample densities of the object and also of the background, the system computes the separability of both classes and obtains new features. The feature evaluation mechanism is embedded in a mean-shift tracking system that adaptively selects the top-ranked discriminative features for tracking. In the first method, Histogram Ratio Shift (HRS), the weights applied to each feature are dynamically updated depending on the histograms of the target and also of the background. In the second one, Variance Ratio Feature Shift (VRFS), the ratio between the variance of the target and the surrounding background is computed and considered for selecting the features. Finally, the Peak Difference Feature Shift (PDFS) softens the histogram of the features by a Gaussian kernel; moreover, it considers possible distracter objects near the target and dynamically changes the feature selection.

And finally, in [43, 44], a method for direct detection and segmentation of foreground moving objects is presented called Graph-Cut Based Tracker (GCBT). The method first obtains several groups of pixels with similar motion and photometric features. The mean-shift procedure is used to validate the motion and bandwidth. And then, the system segments the objects based on a MAP framework.

Our PIORT methodology is based on the iterative and adaptive processing of consecutive frames by a system that integrates recognition and tracking in a probabilistic framework. The system uses object recognition results provided by a classifier, e.g. a Bayesian classifier or a neural net, which are computed from colour features of image regions for each frame. The location of tracked objects is represented through probability images that are updated dynamically using both recognition and tracking results. The tracking procedure is capable of handling quite long occlusions. In particular, object occlusion is detected automatically and the prediction of the object's apparent motion and size takes into account the cases of occlusion entering, full occlusion and occlusion exiting. In contrast with [27], our tracking method does not rely on background subtraction and a fixed camera and, to the contrary of [28], it can cope with complete occlusion and it does not involve any template to match and update.

In our approach, the following assumptions are made:
i) target objects may be distinguished from other objects and the background based on

colour features of their appearance, though these features may experiment slight variations during the image sequence; in fact, this is a requirement of the classifiers we currently use for static recognition and could be relaxed or changed if the classifier in this module were replaced or used a different set of object's appearance features;
ii) target object's shape, apparent motion and apparent size can all vary smoothly between consecutive frames (non-rigid deformable objects are thus allowed); we think this is not a strong assumption if a typical video acquisition rate is used, as large changes in shape, motion and size are allowed for the whole sequence;
iii) image sequences may be obtained either from a fixed or a slow moving camera (this is also quite realistic for most applications in practice);
iv) target objects may be occluded during some frames, but their motion does not change abruptly during occlusion; this last assumption is certainly stronger and may fail in some cases, but it is caused by the need of predicting an approximate position of the object during occlusion based on its previous trajectory.

The rest of the paper is organized as follows. A formal description of our probabilistic framework for object recognition and tracking is given in Section 2. As shown in Fig. 1, the system involves three modules: static recognition, dynamic recognition and tracking decision modules. The methods used for the static recognition module are specified in Section 3. The dynamic recognition module is explained in Section 4. The tracking decision module is described in detail in Section 5. The experimental results are presented in Section 6. Finally, conclusions and future work are discussed in Section 7.

2. A probabilistic framework for integrated object recognition and tracking

Let us assume that we have a sequence of 2D color images $I^t(x,y)$ for $t=1,...,L$, and that there are a maximum of N objects of interest in the sequence of different types (associated with classes $c=1,...,N$, where $N\geq 1$), and that a special class $c=N+1$ is reserved for the background. Furthermore, let us assume that the initial position of each object is known and represented by N binary images, $p_c^0(x,y)$, for $c=1,...,N$, where $p_c^0(x,y)=1$ means that the pixel (x,y) belongs to a region covered by an object of class c in the first image. If less than N objects are actually present, some of these images will be all-zero and they will not be processed further, so, without loss of generality, we consider in the sequel that N is the number of present objects to track.

Hence, we would like to obtain N sequences of binary images $T_c^t(x,y)$, for $c=1,...,N$, that mark the pixels belonging to each object in each image; these images are the desired output of the whole process and can also be regarded as the output of a tracking process for each object. We can initialize these tracking images (for t=0) from the given initial positions of each object, this is

$$T_c^0(x,y) = p_c^0(x,y) \tag{1}$$

In our approach, we divide the system in three modules. The first one performs object recognition in the current frame (static recognition) and stores the results in the form of probability images (one probability image per class), that represent for each pixel the probabilities of belonging to each one of the objects of interest or to the background, according only to the information in the current frame. This can be achieved by using a classifier that has been trained previously to classify image regions of the same objects using a different but similar sequence of images, where the objects have been segmented and labeled. Hence, we assume that the classifier is now able to produce a sequence of class probability images $Q_c^t(x,y)$ for $t=1,...,L$ and $c=1,...,N+1$, where the value $Q_c^t(x,y)$ represents the estimated probability that the pixel (x,y) of the image $I^t(x,y)$ belongs to the class c, which has been computed taking into account a local feature vector (see Section 3). In general, the probability images $Q_c^t(x,y)$ can be regarded as the output of a static recognition module defined by some function r on the current image:

$$Q_c^t(x,y) = r\left(I^t(x,y)\right) \tag{2}$$

In the second module (dynamic recognition), the results of the first module are used to update a second set of probability images, p_c , with a meaning similar to that of Q_c but now taking into account as well both the recognition and tracking results in the previous frames through a dynamic iterative rule. More precisely, we need to store and update $N+1$ probability images $p_c^t(x,y)$, for $c=1,...,N+1$, where the value $p_c^t(x,y)$ represents the probability that the pixel (x,y) in time t belongs to an object of class c (for $c=1,...,N$) or to the background (for $c=N+1$). In general, these dynamic probabilities should be computed as a certain function f of the same probabilities in the previous step, the class probabilities given by the classifier for the current step (which have been obtained from the actual measurements) and the tracking images resulting from the previous step:

$$p_c^t(x,y) = f\left(p^{t-1}(x,y), Q^t(x,y), T^{t-1}(x,y)\right) \tag{3}$$

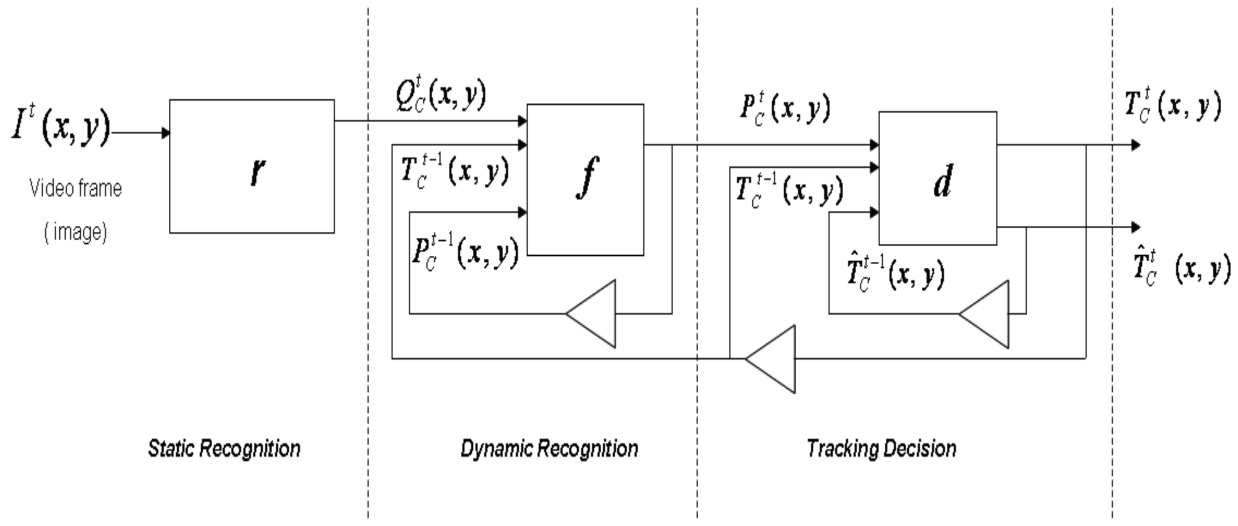


Figure 1: Block diagram of the dynamic object recognition and tracking process.

The update function *f* used in our system is described in Section 4, which incorporates some additional arguments coming from the tracking module to adapt its parameters.

Finally, in the third module (tracking decision), tracking binary images are determined for each object from the current dynamic recognition probabilities, the previous tracking image of the same object and some other data, which contribute to provide a prediction of the object's apparent motion in terms of translation and scale changes as well as to handle the problem of object occlusion. Formally, the tracking images $T_c^t(x,y)$ for the objects ($1 \leq c \leq N$) can be calculated dynamically using the pixels probabilities $p^t(x,y)$ according to some decision function *d*:

$$T_c^t(x,y) = d\left(p^t(x,y), T_c^{t-1}(x,y)\right) \qquad (4)$$

in which some additional arguments and results may be required (see (12) and Section 5 for a detailed description of the tracking decision module).

3. Static recognition module

In our PIORT (Probabilistic Integrated Object Recognition and Tracking) framework, the static recognition module is based on the use of a classifier that is trained from examples and provides posterior class probabilities for each pixel from a set of local features. The local features to be used may be chosen in many different ways. A possible approach consists of first segmenting the given input image $I^t(x,y)$ in homogeneous regions (or spots) and computing some features for each region that are afterwards shared by all its constituent pixels. Hence, the class probabilities $Q_c^t(x,y)$ are actually computed by the classifier once for each spot in the segmented image and then replicated for all the pixels in the spot. For instance, RGB color averages can be extracted for each spot after color segmentation and used as feature vector $v(x,y)$ for a classifier. In the next two subsections we present two specific classifiers that have been implemented and tested within the PIORT framework using this type of information.

3.1 A simple Bayesian method based on maximum likelihood and background uniform conditional probability

Let *c* be an identifier of a class (between 1 and *N*+1), let *B* denote the special class *c*=*N*+1 reserved for the background, let *k* be an identifier of an object (non-background) class between 1 and *N*, and let *v* represent the value of a feature vector. Bayes theorem establishes that the posterior class probabilities can be computed as

$$P(c \mid v) = \frac{P(v \mid c)P(c)}{P(v)} = \frac{P(v \mid c)P(c)}{P(v \mid B)P(B) + \sum_{k=1}^{N} P(v \mid k)P(k)} \tag{5}$$

Our simple Bayesian method for static recognition is based on imposing the two following assumptions:

a) equal priors: all classes, including *B*, will have the same prior probability, i.e. *P*(*B*)=1/(*N*+1) and *P*(*K*)=1/(*N*+1) for all *k* between 1 and *N*.
b) a uniform conditional probability for the background class, i.e. *P*(*v*|*B*)=1/*M*, where *M* is the number of values (bins) in which the feature vector *v* is discretized.

Note that the former assumption is that of a maximum likelihood classifier, whereas the latter assumes no knowledge about the background. After imposing these conditions, equation (5) turns into

$$P(c \mid v) = \frac{P(v \mid c)}{\frac{1}{M} + \sum_{k=1}^{N} P(v \mid k)} \tag{6}$$

and this gives the posterior class probabilities we assign to the static probability images, i.e. $Q_c^t(x,y) = P(c \mid v(x,y))$ for each pixel (*x*,*y*) and time *t*.

It only remains to set a suitable *M* constant and to estimate the class conditional probabilities *P*(*v* | *k*) for all *k* between 1 and *N* (object classes). To this end, class histograms H_k are set up using the labeled training data and updated on-line afterwards using the tracking results in the test data.

For constructing the histograms, let *v(x,y)* be the feature vector consisting of the original RGB values of a pixel (*x*,*y*) labeled as belonging to class *k*. We uniformly discretize each of the R, G and B channels in 16 levels, so that $M = 16\times16\times16 = 4096$. Let *b* be the bin in which *v(x,y)* is mapped by this discretization. To reduce discretization effects, a smoothing technique is applied when accumulating counts in the histogram as follows:

$$\begin{aligned} H_k(b) &:= H_k(b) + (10 - \#neighbors(b)) \\ H_k(b') &:= H_k(b') + 1 \quad \text{if } b' \text{ is a neighbor of } b \end{aligned} \tag{7}$$

where the number of neighbors of *b* (using non-diagonal connectivity) varies from 3 to 6, depending on the position of *b* in the RGB space. Hence, the total count C_k of the histogram is increased by ten (instead of one) each time a pixel is counted and the conditional probability is estimated as $P(v \mid k) = H_k(b) \,/\, C_k$ where b is the bin corresponding to v. The above smoothing technique is also applied when updating the histogram from the tracking results; in that case the RGB value $v(x,y)$ in the input image $I^t(x,y)$ of a pixel (x,y) is used to update the histogram H_k (and the associated count C_k) if and only if $T_k^t(x,y)=1$.

3.2 A neural net based method

In this method, a neural net classifier (a multilayer perceptron) is trained off-line from the labeled training data. The RGB color averages extracted for each spot after color segmentation are used as feature vector $v(x,y)$ and supplied as input to the network in both training and test phases. To the contrary of the Bayesian method described previously, training data for the background class are also provided by selecting some representative background regions in the training image sequence, because the network needs to gather examples for all classes including the background. The network is not retrained on-line using the tracking results in the test phase (this is another difference with respect to the Bayesian method described).

It's well known that using a 1-of-c target coding scheme for the classes, the outputs of a network trained by minimizing a sum-of-squares error function approximate the posterior probabilities of class membership (here, $Q_c^t(x,y)$), conditioned on the input feature vector [45]. Anyway, to guarantee a proper sum to unity of the posterior probabilities, the network outputs (which are always positive values between 0 and 1) are divided by their sum before assigning the posterior probabilities.

4. Dynamic recognition module

Even though the static recognition module can be applied independently to each image in the sequence, this does not exploit the dynamic nature of the problem and the continuity and smoothness properties that are expected in the apparent motion of the objects through the sequence. Hence, a dynamic update of the pixel class probabilities $p_c^t(x,y)$ is desired that takes into account these properties. To this end, not only the previous probabilities $p_c^{t-1}(x,y)$ and the results of the current static recognition $Q_c^t(x,y)$ have to be combined but also the binary results of the tracking decision in the previous step $T_c^{t-1}(x,y)$ have to be considered, since this permits to filter some possible misclassifications made by the static classifier. Typically, some background spots are erroneously classified as part of an object and this can be detected if these spots are situated far from the last known position of the object.

Therefore, the update function f for the dynamic class probabilities can be defined as follows (for some adaptive parameters α_c^t, $0<\alpha_c^t<1$):

$$p_c^t(x,y) = \frac{\alpha_c^t T_c^{t-1}(x,y) p_c^{t-1}(x,y) + (1-\alpha_c^t) Q_c^t(x,y)}{\sum_{k=1}^{N+1} \left(\alpha_k^t T_k^{t-1}(x,y) p_k^{t-1}(x,y) + (1-\alpha_k^t) Q_k^t(x,y) \right)} \qquad (8)$$

A tracking image for the background, which is required in the previous equation, can be defined simply as

$$T_{N+1}^{t}(x,y)=\begin{cases}1 & \text{if } T_c^t(x,y)=0 \quad \forall c: 1\le c\le N\\ 0 & \text{otherwise}\end{cases} \tag{9}$$

and computed after the tracking images for the objects.

The parameter α_c^t that weights the influence of the previous probabilities must be adapted depending on the apparent motion of the tracked object of class *c*. If this motion is very slow, α_c^t should reach a maximum α_{max} closer to 1, whereas if the motion is very fast, α_c^t should reach a minimum α_{min} closer to 0. In order to a set a proper value for α_c^t the areas (A_c^{t-1} and A_c^{t-2}) and mass centers (C_c^{t-1} and C_c^{t-2}) of the object in the two previous tracking images are used in the following way.

Let $r_c^{t-1}=\sqrt{A_c^{t-1}/\pi}$ and $r_c^{t-2}=\sqrt{A_c^{t-2}/\pi}$ be the estimates of the object radius in the two previous frames obtained by imposing a circular area assumption. Let $d_c=\left|C_c^{t-1}-C_c^{t-2}\right|$ be the estimated displacement of the object in the 2D image and let $d_c^{\max}=r_c^{t-1}+r_c^{t-2}$ be the maximum displacement yielding some overlapping between the two former circles. If $d_c \ge d_c^{\max}$ we would like to set $\alpha_c^t=\alpha_{\min}$, whereas if $d_c=0$ then the value of α_c^t should be set according to the change of the object apparent size: Let $s_c=\left|r_c^{t-1}-r_c^{t-2}\right|/\max\left(r_c^{t-1},r_c^{t-2}\right)$ be a scale change ratio. If $s_c=0$ (unchanged object size) then we would like to set $\alpha_c^t=\alpha_{\max}$ whereas in the extreme case $s_c=1$ then we would set $\alpha_c^t=\alpha_{\min}$ again. Combining linearly both criteria, displacement and scale change, we define the prior value

$$\hat{\alpha}_c^t=\alpha_{\max}-\left(\alpha_{\max}-\alpha_{\min}\right)d_c/d_c^{\max}-\left(\alpha_{\max}-\alpha_{\min}\right)s_c \tag{10}$$

which satisfies $\hat{\alpha}_c^t \le \alpha_{\max}$. Note that the value $\hat{\alpha}_c^t=\alpha_{\max}$ (maximum weight for the previous probabilities) is obtained when both $d_c=0$ and $s_c=0$, what means that both the centers and the areas of the object are the same in the last two observations (no displacement and no scale change have occurred). Finally the parameter α_c^t is set as follows:

$$\alpha_c^t=\begin{cases}\alpha_{\min} & \text{if } d_c \ge d_c^{\max}\\ \alpha_{\min} & \text{if } d_c<d_c^{\max}\wedge\hat{\alpha}_c^t\le\alpha_{\min}\\ \hat{\alpha}_c^t & \text{if } d_c<d_c^{\max}\wedge\hat{\alpha}_c^t>\alpha_{\min}\end{cases} \tag{11}$$

The constants α_{min} and α_{max} were set to 0.1 and 0.6, respectively, in our experiments (see Section 6).

5. Tracking decision module

As depicted in Figure 1, the tracking images $T_c^t(x,y)$ for the objects ($1 \le c \le$ N) can be calculated dynamically using the pixels probabilities $p^t(x,y)$ according to some decision function d. However, this function involves some additional arguments and results, as explained next.

To give an initial estimate of the foreseen translation and rescaling of the object in the current step, the measurements of both the object mass center and area in the tracking images of the two previous steps are required. Hence, the areas A_c^{t-1} and A_c^{t-2} and the mass centers C_c^{t-1} and C_c^{t-2}, already used in the dynamic recognition module as we have seen, must also be supplied here. The application of the estimated transformation to the previous tracking image $T_c^{t-1}(x,y)$ will serve to reduce the image area to explore using the class probabilities while filtering (blacking) the rest. This strategy alone permits to track visible objects reasonably well [26, 27] but it fails completely if the object becomes occluded for some frames [28].

In order to cope with occlusion, more information is needed in the decision function d. The key point is to distinguish between the *a posteriori* tracking image $T_c^t(x,y)$ and an *a priori* prediction $\hat{T}_c^t(x,y)$, which could maintain some relevant information of the object before the occlusion such as area and movement. The object mass center C_c^t and area A_c^t needed for tracking should be measured either from $T_c^t(x,y)$ or $\hat{T}_c^t(x,y)$ depending on whether the object is visible or occluded. Hence, an occlusion flag O_c^t has to be determined as an additional result. Moreover, the two previous flags O_c^{t-1} and O_c^{t-2} help to know whether the object is entering or exiting an occlusion. In addition, $\vec{m}_c^t$ is a movement weighted average vector that represents the past trajectory direction of the tracked object, which is useful to solve some ambiguous cases that happen when the object crosses or exits an occlusion by another object with a similar appearance (same-class occlusion). Finally, it should be taken into account that the uncertainty in the prediction $\hat{T}_c^t(x,y)$ grows as the number of consecutive frames the object is occluded increases. In the original method described in [27], two constant parameters ε and δ were used to define an uncertainty region around each pixel transformation. Since we want to adjust the level of uncertainty based on the duration of the occlusion, these parameters have to be adaptive for each object, i.e. ε_c^t and δ_c^t. Summarizing, the decision function d involves the following arguments and results:

$$\left\langle T_c^t(x,y), \hat{T}_c^t(x,y), \vec{m}_c^t, O_c^t, C_c^t, A_c^t, \varepsilon_c^t, \delta_c^t \right\rangle = d\begin{pmatrix} p^t(x,y), T_c^{t-1}(x,y), \hat{T}_c^{t-1}(x,y), \\ \vec{m}_c^{t-1}, O_c^{t-1}, O_c^{t-2}, C_c^{t-1}, C_c^{t-2}, \\ A_c^{t-1}, A_c^{t-2}, \varepsilon_c^{t-1}, \delta_c^{t-1} \end{pmatrix} \tag{12}$$

This function is described in detail in the next subsections, which cover the different independent sub-modules of the tracking decision module. Figure 2 illustrates graphically some of the calculations that are explained in what follows.

5.1 A priori prediction of the tracked objects

The first step is to give *a priori* estimates of the mass center and area of the object in time t. The mass center is predicted as follows:

$$\hat{C}_c^t = \begin{cases} C_c^{t-1} & \text{if } O_c^{t-2} \wedge \neg O_c^{t-1} \\ 2C_c^{t-1} - C_c^{t-2} & \text{otherwise} \end{cases} \tag{13}$$

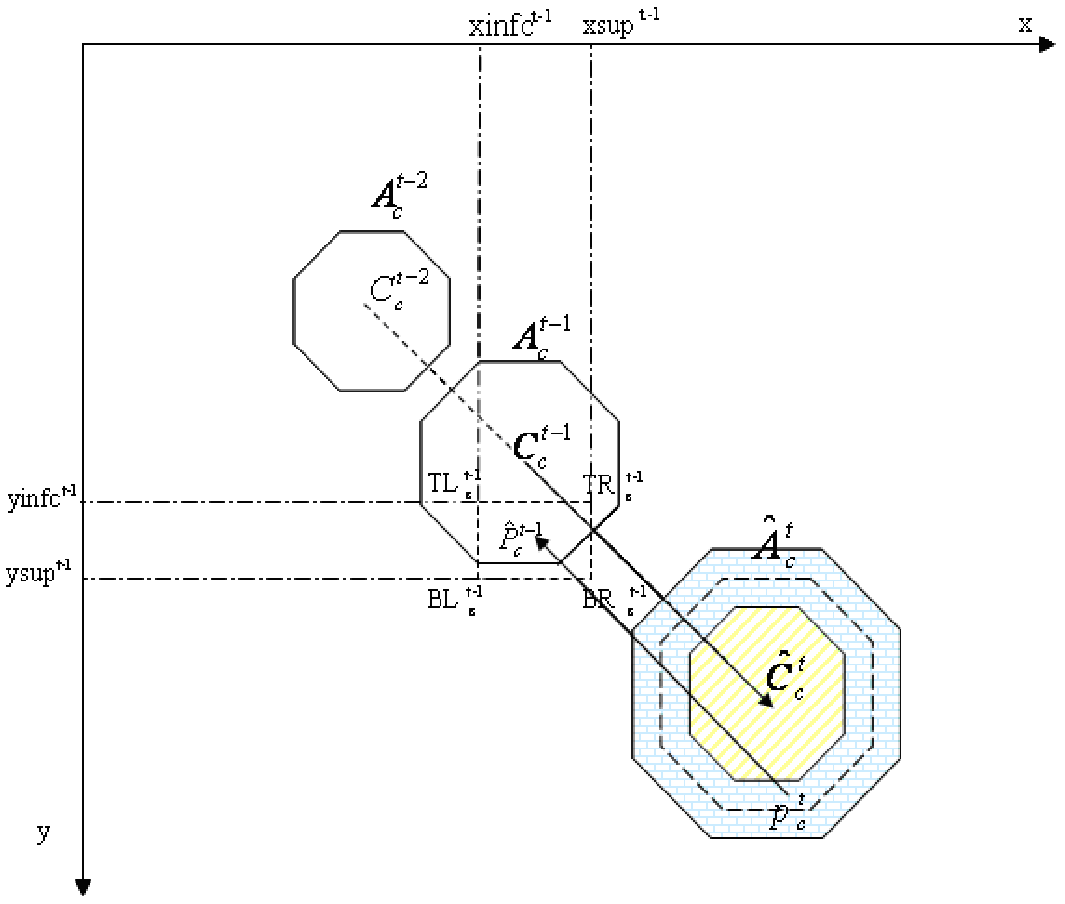


Figure 2: *Geometrical illustration of the tracking process*. Estimates of object's area and mass center for step t are computed from previous values in t-1 and t-2. For each pixel in step t a rectangular region in step t-1 is determined which allows the assignment to the pixel of one of three labels: "certainly belonging to the object" (yellow diagonal-bar-shaded region), "uncertain" (blue brick-shaded region) and "certainly not belonging to the object" (the rest of the image).

When the object is exiting an occlusion, C_c^{t-2} is not reliable enough to be used together with C_c^{t-1} to predict the next movement; therefore, a conservative estimate is given, just the previous measured value. In the rest of cases (the object is visible, is occluded or is entering an occlusion), a constant rate prediction is used. Note, however, that when the object is occluded, the mass center is not measured on the *a posteriori* tracking image, but on the *a priori* one, as we will see later.

It is interesting to notice that the above constant rate prediction can be proved to be equivalent to the one given by a linear Kalman filter for a particular setting of the filter parameters and equations. Let $w_c^{t+1} = Aw_c^t + Bu_c^t + \omega_c^t$ and $d_c^t = Hw_c^t + v_c^t$ be respectively the state and measurement equations of a linear Kalman filter (KF) for predicting the mass center of object *c*. If we set $A=I$, $B=I$, $H=I$, $d_c^t = C_c^t$ as the measurement, $u_c^t = C_c^t - C_c^{t-1}$ as the input and $R^t=0$ as the covariance matrix of the measurement noise v_c^t (which is assumed to be zero), then the *a priori* and *a posteriori* estimates of state w_c^t given by the KF are $2C_c^{t-1} - C_c^{t-2}$ and C_c^t respectively.

The *a priori* estimate of the object area is calculated as follows:

$$\hat{A}_c^t = \begin{cases} \left(A_c^{t-1}\right)^2 / A_c^{t-2} & \text{if } \neg O_c^{t-2} \wedge \neg O_c^{t-1} \\ A_c^{t-1} & \text{otherwise} \end{cases} \tag{14}$$

If the object has been visible in the two previous frames, a constant rate of a scale factor is used to predict the area. It can be proved that this prediction is equivalent to the one given by a (non-linear) extended Kalman filter for a particular setting of the filter parameters and equations. Let $w_c^{t+1} = f\left(w_c^t, u_c^t, \omega_c^t\right)$ and $d_c^t = h\left(w_c^t, v_c^t\right)$ be the state and measurement equations, respectively, of an extended Kalman filter (EKF) for predicting the area of object *c*. If we set $f\left(w_c^t, u_c^t, \omega_c^t\right) = w_c^t u_c^t + \omega_c^t$, $h\left(w_c^t, v_c^t\right) = w_c^t + v_c^t$, $d_c^t = A_c^t$ as the measurement, $u_c^t = A_c^t / A_c^{t-1}$ as the input and $R^t = 0$ as the covariance matrix of the measurement noise v_c^t (which is assumed again to be zero), then the *a priori* and *a posteriori* estimates of state w_c^t given by the EKF are $(A_c^{t-1})^2 / A_c^{t-2}$ and A_c^t respectively. In the rest of cases (the object is occluded or is entering or exiting an occlusion), the area is supposed to remain constant.

From these predictions, a change of coordinates transformation can also be estimated that maps each pixel $P_c^{t-1} = (x_c^{t-1}, y_c^{t-1})$ of the object *c* in step *t-1* (maybe occluded) into its foreseen position in step *t*:

$$\hat{P}_c^t = \hat{C}_c^t + \left(P_c^{t-1} - C_c^{t-1}\right)\sqrt{\hat{A}_c^t / A_c^{t-1}} \tag{15}$$

Actually, we are interested in applying the transformation in the inverse way, i.e. to know which is the expected corresponding position in time *t*-1, $\hat{P}_c^{t-1} = \left(\hat{x}_c^{t-1}, \hat{y}_c^{t-1}\right)$, of a given pixel $P_c^t = (x_c^t, y_c^t)$ in *t*:

$$\hat{P}_c^{t-1} = C_c^{t-1} + \frac{\left(P_c^t - \hat{C}_c^t\right)}{\sqrt{\hat{A}_c^t / A_c^{t-1}}} \tag{16}$$

This is enough to compute the *a priori* tracking image $\hat{T}_c^t(x,y)$ in time *t*, either from the previous *a posteriori* or *a priori* tracking image, depending on the previous occlusion flag:

$$\hat{T}_c^t(x,y) = \begin{cases} T_c^{t-1}\left(\hat{x}_c^{t-1}, \hat{y}_c^{t-1}\right) & \text{if } \neg O_c^{t-1} \\ \hat{T}_c^{t-1}\left(\hat{x}_c^{t-1}, \hat{y}_c^{t-1}\right) & \text{otherwise} \end{cases} \tag{17}$$

where the values of $\hat{x}_c^{t-1}$, $\hat{y}_c^{t-1}$ are clipped whenever is necessary to keep them within the range of valid coordinates.

5.2 First computation of the tracking images

To compute the *a posteriori* tracking image $T_c^t(x,y)$, the pixel class probabilities $p^t(x,y)$ are taking into account only in some image region that is determined from $T_c^{t-1}(x,y)$ or

$\hat{T}_c^{t-1}(x,y)$ (depending on O_c^{t-1}) and the tolerance parameters ε_c^t and δ_c^t. Since the estimates of the translation and scale parameters in the coordinate transformation can be inaccurate, we define a rectangular region of possible positions for each pixel by specifying some tolerances in these estimates. To this end, we use the adaptive parameters ε_c^t and δ_c^t, which must be positive values to be set in accordance with our confidence in the translation and scale estimates respectively (the most confidence the smallest tolerance and vice versa), and which are adjusted according to the following rules:

$$\varepsilon_c^t = \begin{cases} \min\left(\varepsilon_{\max}, \varepsilon_c^{t-1} + \varepsilon_{incr}\right) & \text{if } O_c^{t-2} \vee O_c^{t-1} \\ \varepsilon_{ini} & \text{otherwise} \end{cases} \tag{18}$$

$$\delta_c^t = \begin{cases} \min\left(\delta_{\max}, \delta_c^{t-1} + \delta_{incr}\right) & \text{if } O_c^{t-2} \vee O_c^{t-1} \\ \delta_{ini} & \text{otherwise} \end{cases} \tag{19}$$

where ε_{ini}, δ_{ini} are default values, ε_{max}, δ_{max} are the maximal allowed values and ε_{incr}, δ_{incr} are the respective increases for each successive step under occlusion. Note that the tolerances keep on growing when exiting an occlusion until the object has been visible in the two previous frames; this is needed to detect and track the object again.

Let $C_c^{t-1} = \left(x_{Cc}^{t-1}, y_{Cc}^{t-1}\right)$ and $\hat{C}_c^t = \left(\hat{x}_{Cc}^t, \hat{y}_{Cc}^t\right)$ be respectively the previous mass center and the *a priori* estimate of the current mass center. The four vertices of the rectangular uncertainty region centered at $\hat{P}_c^{t-1}$ are denoted (top-left) $TL_c^{t-1} = (xinf_c^{t-1}, yinf_c^{t-1})$, (top-right) $TR_c^{t-1}=(xsup_c^{t-1}, yinf_c^{t-1})$, (bottom-left) $BL_c^{t-1}=(xinf_c^{t-1}, ysup_c^{t-1})$ and (bottom-right) $BR_c^{t-1} = (xsup_c^{t-1}, ysup_c^{t-1})$, where:

$$xinf_c^{t-1} = \begin{cases} x_{Cc}^{t-1} + \dfrac{\left(x_c^t - \hat{x}_{Cc}^t - \varepsilon_c^t\left|\hat{x}_{Cc}^t - x_{Cc}^{t-1}\right|\right)}{\sqrt{\left(\hat{A}_c^t + \delta_c^t \hat{A}_c^t\right)/A_c^{t-1}}} & \text{if } x_c^t - \hat{x}_{Cc}^t - \varepsilon_c^t\left|\hat{x}_{Cc}^t - x_{Cc}^{t-1}\right| \geq 0 \\ x_{Cc}^{t-1} + \dfrac{\left(x_c^t - \hat{x}_{Cc}^t - \varepsilon_c^t\left|\hat{x}_{Cc}^t - x_{Cc}^{t-1}\right|\right)}{\sqrt{\left(\hat{A}_c^t - \delta_c^t \hat{A}_c^t\right)/A_c^{t-1}}} & \text{otherwise} \end{cases} \tag{20}$$

$$xsup_c^{t-1} = \begin{cases} x_{Cc}^{t-1} + \dfrac{\left(x_c^t - \hat{x}_{Cc}^t + \varepsilon_c^t\left|\hat{x}_{Cc}^t - x_{Cc}^{t-1}\right|\right)}{\sqrt{\left(\hat{A}_c^t - \delta_c^t \hat{A}_c^t\right)/A_c^{t-1}}} & \text{if } x_c^t - \hat{x}_{Cc}^t + \varepsilon_c^t\left|\hat{x}_{Cc}^t - x_{Cc}^{t-1}\right| \geq 0 \\ x_{Cc}^{t-1} + \dfrac{\left(x_c^t - \hat{x}_{Cc}^t + \varepsilon_c^t\left|\hat{x}_{Cc}^t - x_{Cc}^{t-1}\right|\right)}{\sqrt{\left(\hat{A}_c^t + \delta_c^t \hat{A}_c^t\right)/A_c^{t-1}}} & \text{otherwise} \end{cases} \tag{21}$$

$$yinf_c^{t-1} = \begin{cases} y_{Cc}^{t-1} + \dfrac{\left(y_c^t - \hat{y}_{Cc}^t - \varepsilon_c^t \left|\hat{y}_{Cc}^t - y_{Cc}^{t-1}\right|\right)}{\sqrt{\left(\hat{A}_c^t + \delta_c^t \hat{A}_c^t\right) / A_c^{t-1}}} & \text{if } y_c^t - \hat{y}_{Cc}^t - \varepsilon_c^t \left|\hat{y}_{Cc}^t - y_{Cc}^{t-1}\right| \geq 0 \\ y_{Cc}^{t-1} + \dfrac{\left(y_c^t - \hat{y}_{Cc}^t - \varepsilon_c^t \left|\hat{y}_{Cc}^t - y_{Cc}^{t-1}\right|\right)}{\sqrt{\left(\hat{A}_c^t - \delta_c^t \hat{A}_c^t\right) / A_c^{t-1}}} & \text{otherwise} \end{cases} \tag{22}$$

$$ysup_c^{t-1} = \begin{cases} y_{Cc}^{t-1} + \dfrac{\left(y_c^t - \hat{y}_{Cc}^t + \varepsilon_c^t \left|\hat{y}_{Cc}^t - y_{Cc}^{t-1}\right|\right)}{\sqrt{\left(\hat{A}_c^t - \delta_c^t \hat{A}_c^t\right) / A_c^{t-1}}} & \text{if } y_c^t - \hat{y}_{Cc}^t + \varepsilon_c^t \left|\hat{y}_{Cc}^t - y_{Cc}^{t-1}\right| \geq 0 \\ y_{Cc}^{t-1} + \dfrac{\left(y_c^t - \hat{y}_{Cc}^t + \varepsilon_c^t \left|\hat{y}_{Cc}^t - y_{Cc}^{t-1}\right|\right)}{\sqrt{\left(\hat{A}_c^t + \delta_c^t \hat{A}_c^t\right) / A_c^{t-1}}} & \text{otherwise} \end{cases} \tag{23}$$

The values of $xinf_c^{t-1}$, $yinf_c^{t-1}$, $xsup_c^{t-1}$ and $ysup_c^{t-1}$ are clipped whenever is necessary to keep them within the range of valid coordinates. In order to understand eqs. (20) to (23), they must be first compared with eq. (16), which gives the position of the rectangle center (jointly for *x* and *y* coordinates). Then, consider for instance eq. (20), since the other three are simply derived by symmetry; eq. (20) aims at setting the leftmost value of *x* in the uncertainty rectangle; to this end, some small proportion of the estimated center displacement in the *x* coordinate is subtracted in the numerator, and the scale ratio is either enlarged or shrunk in the denominator (by adding or subtracting respectively a small proportion of the new estimated area) depending on which of these two options yields the smallest (leftmost) *x* value; it's easy to check that the last depends on the sign of the numerator.

Now, each pixel $P_c^t = (x_c^t, y_c^t)$ is labeled, with respect to object *c*, as one of three labels ("certainly belonging to the object *c*", "certainly not belonging to the object *c*" or "uncertain") as follows.

If O_c^{t-1} is false then: if all the pixels in the rectangular region delimited by TL_c^{t-1}, TR_c^{t-1}, BL_c^{t-1}, BR_c^{t-1} have a common value of 1 in $T_c^{t-1}(x,y)$, it is assumed that P_c^t is definitely inside and certainly belongs to object *c*; to the contrary, if they have a common value of 0 in $T_c^{t-1}(x,y)$, it is assumed that P_c^t is clearly outside and certainly does not belong to object *c*; otherwise, the rectangular region contains both 1 and 0 values, the pixel P_c^t is initially labeled as "uncertain".

However, if O_c^{t-1} is true, $T_c^{t-1}(x,y)$ will represent a totally or partially occluded object and we cannot rely on it, but on the predicted $\hat{T}_c^{t-1}(x,y)$, which is based on information previous to the occlusion. If all the pixels in the rectangular region delimited by TL_c^{t-1}, TR_c^{t-1}, BL_c^{t-1}, BR_c^{t-1} have a common value of 0 in $\hat{T}_c^{t-1}(x,y)$, it is assumed that P_c^t does not belong to object *c*; otherwise (the rectangular region contains both 1 and 0 values or only 1 values), the pixel P_c^t is labeled as "uncertain".

Only for the uncertain pixels (*x*,*y*) the dynamic probabilities $p^t(x,y)$ will be used. Recall that these probabilities will have been updated previously from the object recognition results in time *t*, $Q^t(x,y)$, also expressed as probabilities. More precisely, we propose the following rule to compute the value of each pixel of the *a posteriori* tracking image for object c in time t:

$$T_c^t(x,y) = \begin{cases} 1 & \begin{array}{l}\text{if } (x,y) \text{ certainly belongs to object } c \\ \textit{or} \text{ it is uncertain and } p_c^t(x,y) \text{ is the} \\ \text{maximum for all } c \text{ between } 1 \text{ and } N+1\end{array} \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

5.3 Post-processing of the tracking images

Sometimes, the tracking images $T_c^t(x,y)$ obtained by applying eq. (24) contain disconnected regions of 1-valued pixels, or, said in other words, more than one connected component T_{ci}^t, $1 \le i \le I$, $I > 1$. This may be produced by a variety of causes, mainly segmentation or recognition errors, but also may be due to possible partial occlusions of the target object by an object of a different class. In addition, a particular problem that leads to object split occurs immediately after a same-class crossing or occlusion: when the target object has just finished crossing another object or region which is recognized to be in the same class (distracter), then the tracking method is misled to follow both the object and the distracter. It is very difficult to devise a general method that can always distinguish between erroneous components due to noise or distracters and correct object components, especially if separated components are allowed to cope with partial occlusions, but some useful heuristics based on properties such as size, movement or shape may be defined that work reasonably well in a majority of cases.

In order to eliminate noisy regions and to circumvent the same-class crossing problem, while handling partial occlusions at the same time, we propose a post-processing step that removes from $T_c^t(x,y)$ some possible artifacts or distracters (setting some initially 1-valued pixels to zero). In fact, this step is only carried out if $T_c^t(x,y)$ contains more than one component. In such a case, we need to choose which components T_{ci}^t to keep (one or more) and which to discard. To this end, three heuristic filters are applied sequentially, whenever two or more components remain before the filter application.

The first filter is aimed at deleting small noisy regions and is solely based on their size. Let A_{ci}^t be the area of the i-th connected component T_{ci}^t and let $\text{Area}(T_c^t)$ be the total area covered by 1-valued pixels in $T_c^t(x,y)$. The i-th component is removed if the ratio $A_{ci}^t / \text{Area}(T_c^t)$ is below a given threshold κ, e.g. $\kappa = 0.15$.

The second filter is aimed at deleting distracters, including those appearing after same-class occlusion, and is based on a comparison between the apparent movement of the remaining components and the previous recent trajectory of the tracked object represented by the movement vector $\vec{m}_c^{t-1}$. Let C_{ci}^t be the mass center of the i-th connected component T_{ci}^t and define an associated movement vector $\vec{z}_{ci}^t = C_{ci}^t - C_c^{t-1}$ for each component. Then,

$$\cos\theta_{ci} = \frac{\langle \vec{z}^{\,t}_{ci}, \vec{m}^{\,t-1}_{c} \rangle}{\left\| \vec{z}^{\,t}_{ci} \right\| \left\| \vec{m}^{\,t-1}_{c} \right\|} \tag{25}$$

is a measure of the alignment between the vectors $\vec{m}^{\,t-1}_{c}$ and $\vec{z}^{\,t}_{ci}$, which is only reliable for our purposes if both vectors have a sufficiently large norm. Otherwise, the angle θ_{ci} can be considered rather random, since may be affected a lot by adding small perturbations on the vectors. Consequently, abrupt trajectory changes (greater than 90 degrees) are penalized if we remove the *i*-th component T^t_{ci} when the condition $\cos\theta_{ci} < 0 \wedge \left\|\vec{z}^{\,t}_{ci}\right\| \geq \lambda \wedge \left\|\vec{m}^{\,t-1}_{c}\right\| \geq \lambda$ holds, where λ is another threshold, e.g. $\lambda = 3$. However, to guarantee that at least one component is kept, the remaining component for which $\vec{z}^{\,t}_{ci}$ is the most collinear vector with respect to $\vec{m}^{\,t-1}_{c}$, i.e. the component *i* such that

$$i = \arg\max \left\{ \frac{\langle \vec{z}^{\,t}_{ci}, \vec{m}^{\,t-1}_{c} \rangle}{\left\| \vec{z}^{\,t}_{ci} \right\|} \right\} \tag{26}$$

is never removed by this second filter.

The third filter is also aimed at deleting distracters and is based on a comparison between the shapes of the components and that of the *a priori* prediction of the target object (represented by the 1-valued region in $\hat{T}^t_c(x,y)$). For each remaining component T^t_{ci}, the *a priori* prediction of the target object is moved from its original center $\hat{C}^t_c$ to the component center C^t_{ci}, thus resulting in a translated copy $\hat{T}^t_{ci}(x,y)$, and the spatial overlap between both shapes is then measured as follows:

$$SO_{ci} = \frac{\text{Area}\left(T^t_{ci} \cap \hat{T}^t_{ci}\right)}{\text{Area}\left(T^t_{ci} \cup \hat{T}^t_{ci}\right)} \tag{27}$$

The components having a spatial overlap $SO_{ci} < 0.243$ (which is the overlap obtained between two circles of the same size when one of the centers is located in the border of the other circle) are deleted in this third filter, unless SO_{ci} is the maximum spatial overlap of the remaining components. This exception guarantees the persistence of at least one component in the final tracking image.

As a result of the post-processing, the pixels of all the components T^t_{ci} removed by any of the three filters are set to zero in the final tracking image $T_c^t(x,y)$.

5.4 Determination of occlusion and geometric measurements

Once both $T_c^t(x,y)$ and $\hat{T}^t_c(x,y)$ have been determined, it is possible to detect the occurrence of an occlusion (i.e. to set the current occlusion flag) in the following way. Let Area(T_c^t) be the measured area of the 1-valued region in the final $T_c^t(x,y)$ and let Area($\hat{T}^t_c$) be the measured area of the 1-valued region in $\hat{T}^t_c(x,y)$. Then,

$$O_c^t = \begin{cases} \text{Area}\left(T_c^t\right)/\text{Area}\left(\hat{T}_c^t\right) < r_1 & \text{if } O_c^{t-1} \\ \text{Area}\left(T_c^t\right)/\text{Area}\left(\hat{T}_c^t\right) < r_2 & \text{otherwise} \end{cases} \tag{28}$$

where $0 < r_1 < r_2 < 1$ (for instance, r_1=0.5, r_2=0.75). Note that the condition for remaining in occlusion mode is harder than the condition for initiating an occlusion. This facilitates the recovery of the object track when exiting an occlusion or when a false occlusion has been detected.

Next, the *a posteriori* estimates of the object mass center and area are selected between those of the *a priori* and *a posteriori* tracking images based on the value of the occlusion flag:

$$C_c^t = \begin{cases} MC\left(T_c^t\right) & \text{if } \neg O_c^t \\ MC\left(\hat{T}_c^t\right) & \text{otherwise} \end{cases} \tag{29}$$

where $MC(T_c^t)$ is the measured mass center of the 1-valued region in the final $T_c^t(x,y)$ and $MC(\hat{T}_c^t)$ is the measured mass center of the 1-valued region in $\hat{T}_c^t(x,y)$, and

$$A_c^t = \begin{cases} \text{Area}\left(T_c^t\right) & \text{if } \neg O_c^t \\ \text{Area}\left(\hat{T}_c^t\right) & \text{otherwise} \end{cases} \tag{30}$$

Finally, the movement weighted average vector $\vec{m}_c^t$ is updated afterwards as follows:

$$\vec{m}_c^t = \begin{cases} \left(\vec{v}_c^t + (t-1)\vec{m}_c^{t-1}\right)/t & \text{if } t < 1/\beta \\ \beta\vec{v}_c^t + (1-\beta)\vec{m}_c^{t-1} & \text{if } t \geq 1/\beta \end{cases} \tag{31}$$

where β is a positive parameter between 0 and 1, e.g. β=0.2, and $\vec{v}_c^t$ is the current movement defined by $\vec{v}_c^t = C_c^t - C_c^{t-1}$. Note that the second row in (31) is a typical moving average computation, while the first row denotes a simple average for the starting steps, and both give the same result for t=1/β.

6. Experimental results

We were interested in testing both PIORT approaches in video sequences including object occlusions and taken with a moving camera. Nevertheless, we also performed a first set of validation experiments in video sequences taken with a still camera. In all tests we defined N=1 objects of interest to track.

All images in the video sequences were segmented independently using the EDISON implementation of the mean-shift segmentation algorithm, code available at http://www.caip.rutgers.edu/riul/research/code.html. The local features extracted for each spot were the RGB colour averages. For object learning, spots selected through ROI (region-of-interest) windows in the training sequence were collected to train a two-layer perceptron using backpropagation and to build the target class histogram. When using the neural net in the test phase, the class probabilities for all the spots in the test sequences were estimated from the net outputs. When using the histogram, the spot class probabilities were estimated according to equation (6). In both cases, the spot class probabilities were replicated for all the pixels in the same spot. For object tracking in the test sequences, ROI windows for the target object were only marked in the first image to initialise the tracking process.

The recognition and tracking results for the test sequences of our PIORT approaches were stored in videos where each frame has a layout of 2 x 3 images with the following contents: the top left is the image segmented by EDISON; the top middle is the image of probabilities given by the static recognition module for the current frame; the top right is the *a priori* prediction of the tracking image; the bottom left is the image of dynamic probabilities; the bottom right is the original image with a graphic overlay that represents the boundaries of the *a posteriori* binary tracking image (the final result for the frame); and the bottom middle is an intermediate image labelled by the tracking module where yellow pixels correspond to pixels labelled as “certainly belonging to the object”, light blue pixels correspond to pixels initially labelled as “uncertain” but with a high dynamic probability, dark blue pixels correspond to pixels labelled as “uncertain” and with a low probability, dark grey pixels are pixels labelled as “certainly not belonging to the object” but with a high probability and the rest are black pixels with both a low probability and a “certainly not belonging to the object” label.

For comparison purposes, tracking of the target objects in the test sequences was also carried out by applying the six following methods, which only need the ROI window mark in the first frame of the test sequence: Template Match by Correlation (TMC), which refers to normalized correlation template matching [30]; Basic Meanshift (BM) [31]; Histogram Ratio Shift (HRS) [32]; Variance Ratio Feature Shift (VRFS) [32]; Peak Difference Feature Shift (PDFS) [32]; and Graph-Cut Based Tracker (GCBT) [33, 30]. These methods have been commented briefly in Section 1.

From the tracking results of all the tested methods, two evaluation metrics were computed for each frame: the **spatial overlap** and the **centroid distance** [46]. The spatial overlap $SO(GT_k,ST_k)$ between the ground truth GT_k and the system track ST_k in a specific frame k is defined as the ratio

$$SO(GT_k,\, ST_k) = \frac{\text{Area}\left(GT_k \cap ST_k\right)}{\text{Area}\left(GT_k \cup ST_k\right)} \tag{32}$$

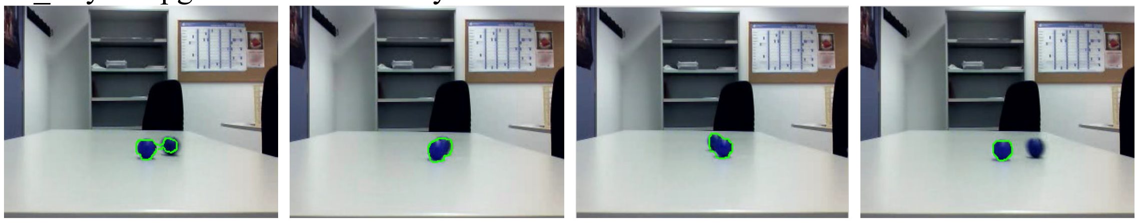
and $Dist(GTC_k,\ STC_k)$ refers to the Euclidean distance between the centroids of the ground truth (GTC_k) and the system track (STC_k) in frame k. Naturally, the larger the overlap and the smaller the distance, the better performance of the system track.

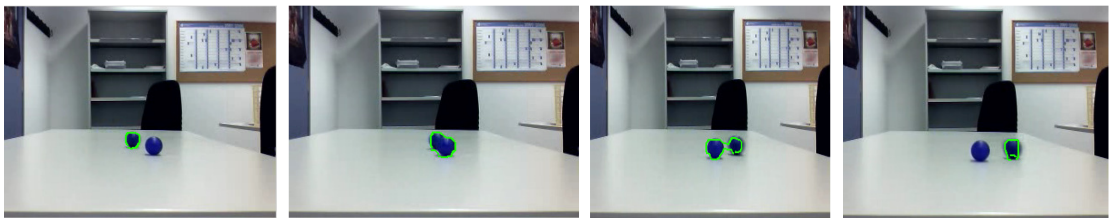
Since the centroid distance can only be computed if both GT_k and ST_k are non-null, a **failure ratio** was measured as the number of frames in which either GT_k or ST_k was null (but not both) divided by the total number of frames. Finally, an **accuracy** measure was computed as the number of good matches divided by the total number of frames, where a good match is either a true negative or a true positive with a spatial overlap above a threshold of 0.243 (which is the overlap obtained between two circles of the same size when one of the centres is located in the border of the other circle).

6.1 Experimental results on video sequences taken with a still camera

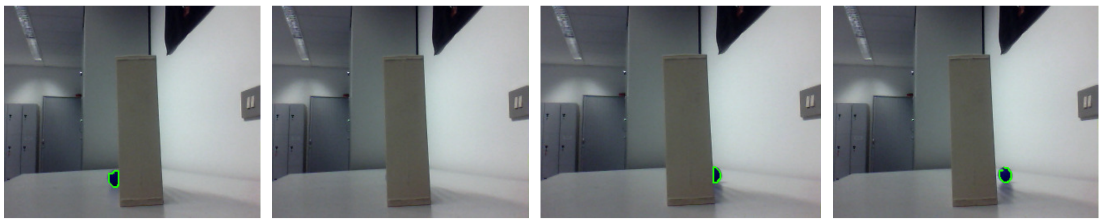
The first set of experiments comprised three test video sequences taken with a still camera that show indoor office scenes where the target to track is a blue ball moving on a table. A similar but different sequence was used for training a neural network to discriminate between blue balls and typical sample regions in the background and for constructing the class histogram of the blue ball (this training sequence is available at http://www-iri.upc.es/people/ralqueza/bluetraining.avi).

In the first test sequence, http://www-iri.upc.es/people/ralqueza/S1S2.avi, two blue balls are moving on the table and one occludes temporally the other one during some frames. Two experiments were performed on this test sequence depending on the initialisation of the tracking. In test S1, the tracking was initialised at the right ball and in test S2, the tracking was initialised at the left ball. The static recognition module considers that both balls belong to the same class. In both tests, the temporal overlapping was correctly managed by our methods since the tracked ball is well relocated after exiting the occlusion. The corresponding videos displaying the results of PIORT methods (in the layout described above) are at http://www-iri.upc.es/people/ralqueza/S1_NN.mpg and S2_NN.mpg for the PIORT-Neural net method and at S1_Bayes.mpg and S2_Bayes.mpg for the PIORT-Bayesian method.


Sequence S1: Blue balls crossed. Tracking initial right ball


Sequence S2: Blue balls crossed. Tracking initial left ball

In the second test sequence (test S3), http://www-iri.upc.es/people/ralqueza/S3.avi, the tracked blue ball is occluded twice by a box during 5 and 12 frames, respectively. Recognition and tracking results for the whole sequence using the PIORT-Neural Net and Bayesian methods are at http://www-iri.upc.es/people/ralqueza/S3_NN.mpg and S3_Bayes.mpg, respectively. The tracking of the blue ball is quite satisfactory since both occlusions are correctly detected and the ball is correctly relocated when exiting the occlusion.


Sequence S3: Blue ball moving occluded by box

In the last test sequence of this group, http://www-iri.upc.es/people/ralqueza/S4.avi, there are again two blue balls and the target moving ball crosses twice, once in front of and once behind, the second ball, which does not move. As the recognition module classifies both balls in the same class, the same-class occlusion is not detected as an occlusion (the two balls are merged into a single blue object), but anyway the target ball is well tracked after the two crossings. The videos displaying the results of the PIORT-Neural Net and Bayesian methods for this sequence are at http://www-iri.upc.es/people/ralqueza/S4_NN.mpg and S4_Bayes.mpg, respectively.

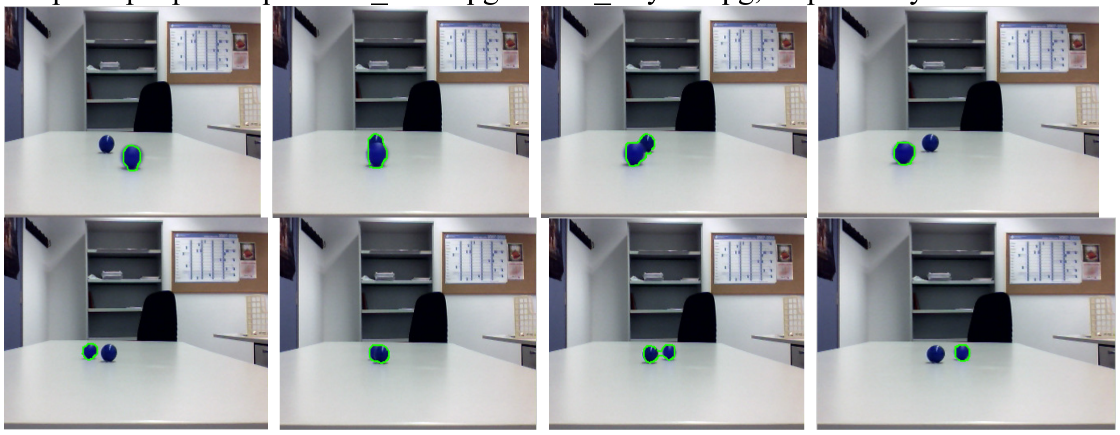

Sequence S4: Blue ball moving around another blue ball

Table 1 presents the results (mean ± std. deviation) of the spatial overlap (SO) and centroid distance (CD) measures together with the failure ratio (FR) and accuracy (Acc) of each tracking method for the four tests S1 to S4, emphasizing the best values for each measure and test in bold. Our PIORT tracking methods worked fine in the four tests obtaining the best values of the four measures (except in the Accuracy measure for test S4, where the HRS method gave a slightly superior performance). All methods performed quite well in S1; only PDFS method performed comparably to PIORT approaches in S2; only PIORT methods worked in S3 while the rest failed; and only BM and HRS methods performed comparably to PIORT approaches in S4.

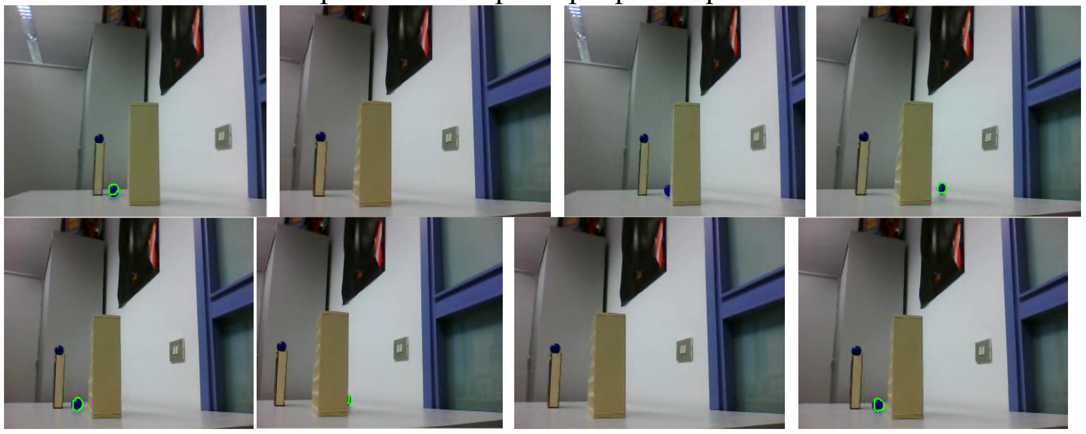
Table 1. Results of ball tracking on video sequences taken with a still camera.

Video Sequence	Tracking method	SO	CD	FR	Acc
S1 Blue balls crossed (Right ball)	TMC	0.56 ± 0.10	5.07 ± 2.07	**0**	0.98
	BM	0.60 ± 0.06	3.19 ± 1.21	**0**	**1.00**
	HRS	0.46 ± 0.11	6.03 ± 2.05	**0**	**1.00**
	VRFS	0.66 ± 0.07	1.15 ± 0.47	**0**	**1.00**
	PDFS	0.63 ± 0.10	2.01 ± 0.94	**0**	**1.00**
	GCBT	0.64 ± 0.18	13.20 ± 52.52	0.05	0.94
	PIORT-Neural Net	**0.84** ± 0.09	1.38 ± 1.39	**0**	**1.00**
	PIORT-Bayesian	0.80 ± 0.07	**0.75** ± 0.76	**0**	**1.00**
S2 Blue balls crossed (Left ball)	TMC	0.22 ± 0.27	44.34 ± 52.24	**0**	0.41
	BM	0.23 ± 0.29	42.51 ± 50.42	**0**	0.36
	HRS	0.25 ± 0.31	44.93 ± 51.96	**0**	0.41
	VRFS	0.28 ± 0.35	42.82 ± 52.62	**0**	0.41
	PDFS	0.50 ± 0.30	36.27 ± 86.95	0.14	0.77
	GCBT	0.20 ± 0.27	70.69 ± 68.80	**0**	0.36
	PIORT-Neural Net	**0.60** ± 0.23	**3.94** ± 4.98	**0**	**0.91**
	PIORT-Bayesian	0.46 ± 0.25	15.04 ± 52.64	0.05	0.73
S3 Blue ball moving occluded by box	TMC	0.01 ± 0.04	173.40 ± 68.71	0.22	0
	BM	0.01 ± 0.07	182.54 ± 68.14	0.22	0
	HRS	0	187.85 ± 67.96	0.25	0
	VRFS	0.02 ± 0.18	140.14 ± 93.44	0.20	0.17
	PDFS	0.13 ± 0.41	131.07 ± 106.1	0.42	0.02
	GCBT	0	237.02 ± 134.6	0.74	0.22
	PIORT-Neural Net	**0.81** ± 0.42	**0.47** ± 0.38	**0**	**1.00**
	PIORT-Bayesian	0.53 ± 0.37	8.39 ± 48.61	0.03	0.95
S4 Blue ball moving around still blue ball	TMC	0.35 ± 0.22	13.10 ± 32.38	**0.01**	0.75
	BM	0.56 ± 0.15	7.39 ± 29.05	**0.01**	0.93
	HRS	0.60 ± 0.13	6.21 ± 29.16	**0.01**	**0.96**
	VRFS	0.10 ± 0.62	74.68 ± 45.00	**0.01**	0.14
	PDFS	0.13 ± 0.43	44.39 ± 36.14	**0.01**	0.17
	GCBT	0.10 ± 0.53	201.60 ± 98.35	0.80	0.18
	PIORT-Neural Net	**0.74** ± 0.21	5.90 ± 29.33	**0.01**	0.94
	PIORT-Bayesian	0.72 ± 0.20	**5.58** ± 29.38	**0.01**	0.94

SO: Spatial Overlap; CD: Centroid Distance; FR: Failure Ratio; Acc: Accuracy

6.2 Experimental results on video sequences taken with a moving camera

The second set of experiments comprised another three test video sequences where the target is a ball, but this time taken with a moving camera. The first of them (test S5) again shows an indoor office scene where a blue ball is moving on a table and is temporally occluded, while other blue objects appear in the scene. This test sequence can be downloaded at http://www-iri.upc.es/people/ralqueza/S5.avi.


Sequence S5: Blue bouncing ball on table

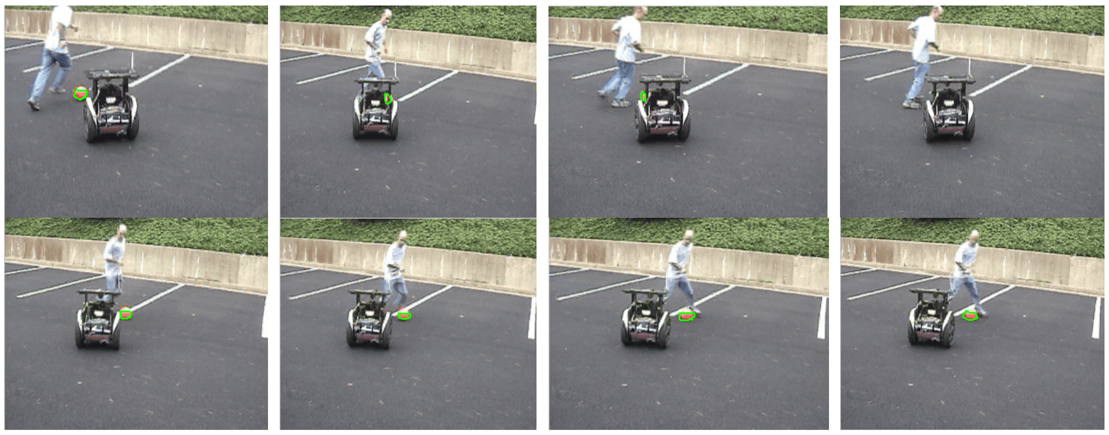
The other two test sequences in this group show outdoor scenes in which a Segway robot tries to follow an orange ball that is being kicked by a person. Both include multiple occlusions of the tracked orange ball and differ in the surface over which the ball runs, which is pavement in the case of test S6 and grass in test S7 (see http://www-iri.upc.es/people/ralqueza/S6.avi and S7.avi, respectively). A similar but different sequence was used for training a neural network to discriminate between orange balls and typical sample regions in the background and for constructing the class histogram of the orange ball (this training sequence is available at http://www-iri.upc.es/people/ralqueza/orangetraining.avi).

Table 2. Results of ball tracking on video sequences taken with a mobile camera.

Video Sequence	Tracking method	SO	CD	FR	Acc
S5 Blue bouncing ball on table	TMC	0.28 ± 0.48	74.65 ± 91.53	0.19	0.43
	BM	0.23 ± 0.52	78.40 ± 90.33	0.19	0.37
	HRS	0.16 ± 0.45	125.88 ± 11.80	0.43	0.30
	VRFS	0.20 ± 0.38	96.72 ± 134.84	0.39	0.60
	PDFS	0.28 ± 0.57	103.60 ± 36.77	0.41	0.59
	GCBT	0.01 ± 0.29	188.79 ± 18.13	0.75	0.21
	PIORT-Neural Net	**0.60** ± 0.40	12.53 ± 59.38	**0.05**	**0.95**
	PIORT-Bayesian	0.59 ± 0.39	**12.46** ± 59.40	**0.05**	**0.95**
S6 Segway - Orange ball on pavement	TMC	0.06 ± 0.40	146.35 ± 81.83	0.03	0.14
	BM	0.09 ± 0.43	110.94 ± 76.70	0.03	0.19
	HRS	0.09 ± 0.38	156.99 ± 103.80	0.41	0.21
	VRFS	0.16 ± 0.68	70.46 ± 49.17	0.03	0.21
	PDFS	0.14 ± 0.59	117.09 ± 81.43	0.03	0.21
	GCBT	0.01 ± 0.34	233.56 ± 62.12	0.93	0.06
	PIORT-Neural Net	**0.72** ± 0.20	**2.67** ± 19.21	**0.01**	**0.98**
	PIORT-Bayesian	0.13 ± 0.73	202.14 ± 99.35	0.81	0.19
S7 Segway - Orange ball on grass	TMC	0.02 ± 0.29	137.93 ± 84.53	0.04	0.04
	BM	0.15 ± 0.27	125.13 ± 116.14	0.34	0.35
	HRS	0.03 ± 0.33	190.63 ± 89.72	0.54	0.08
	VRFS	**0.59** ± 0.21	**7.93** ± 38.85	**0.02**	**0.95**
	PDFS	0.33 ± 0.50	121.46 ± 125.91	0.48	0.51
	GCBT	0.01 ± 0.37	208.39 ± 83.88	0.79	0.04
	PIORT-Neural Net	0.47 ± 0.23	17.02 ± 60.98	0.06	0.88
	PIORT-Bayesian	0.25 ± 0.49	133.43 ± 126.22	0.53	0.42

SO: Spatial Overlap; CD: Centroid Distance; FR: Failure Ratio; Acc: Accuracy

The tracking results videos for the above test sequences are attainable at http://www-iri.upc.es/people/ralqueza/S5_NN.mpg, S5_Bayes.mpg, S6_NN.mpg, S6_Bayes.mpg, S7_NN.mpg and S7_Bayes.mpg.


Sequence S6: Segway - Orange ball on pavement

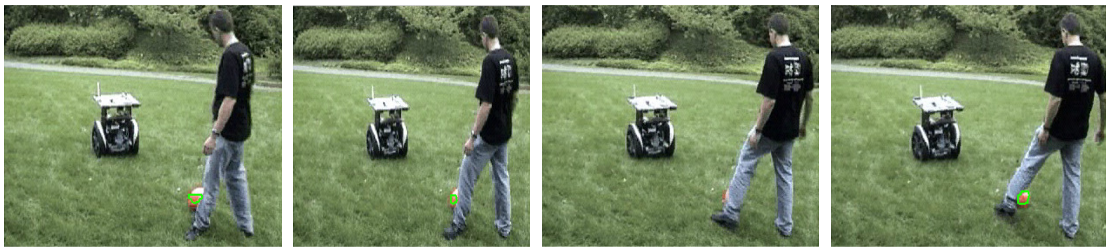
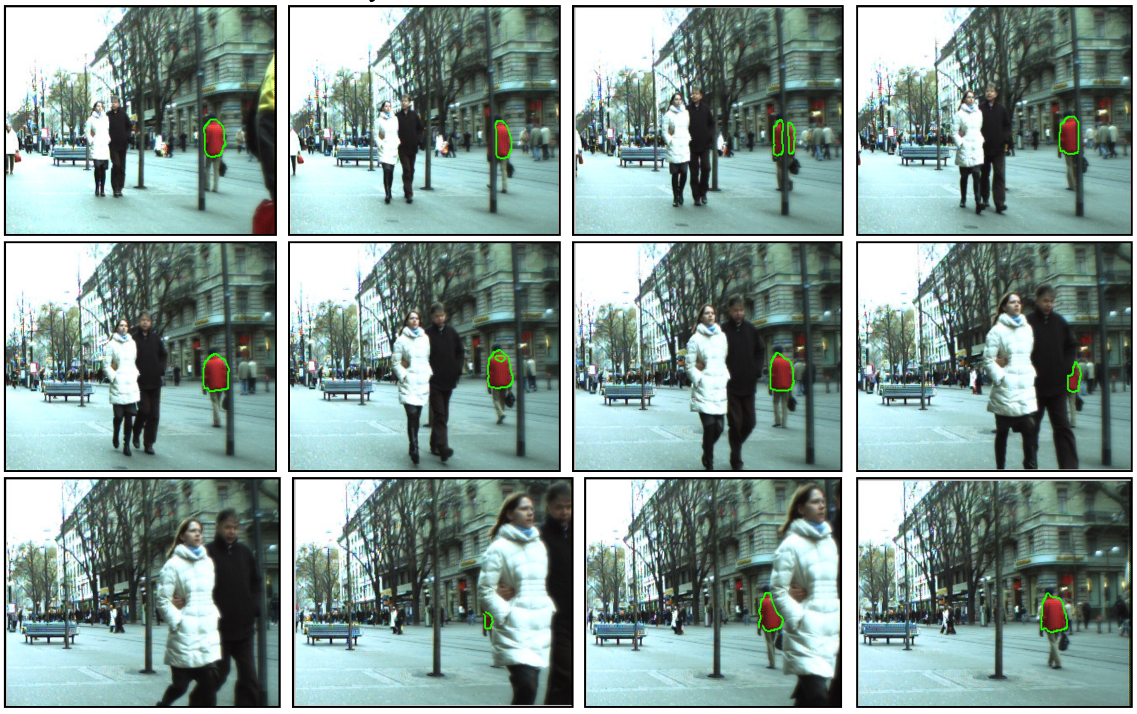
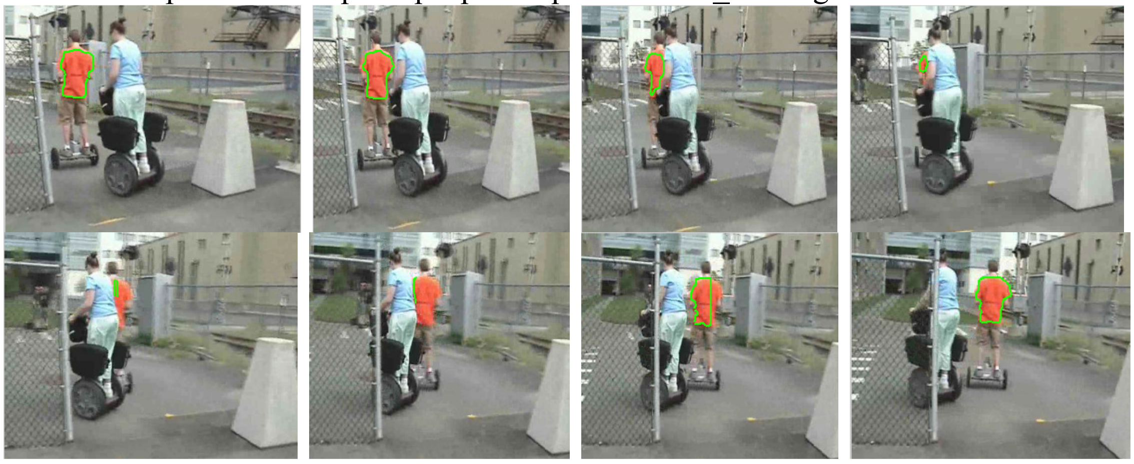

Sequence S7: Segway - Orange ball on grass

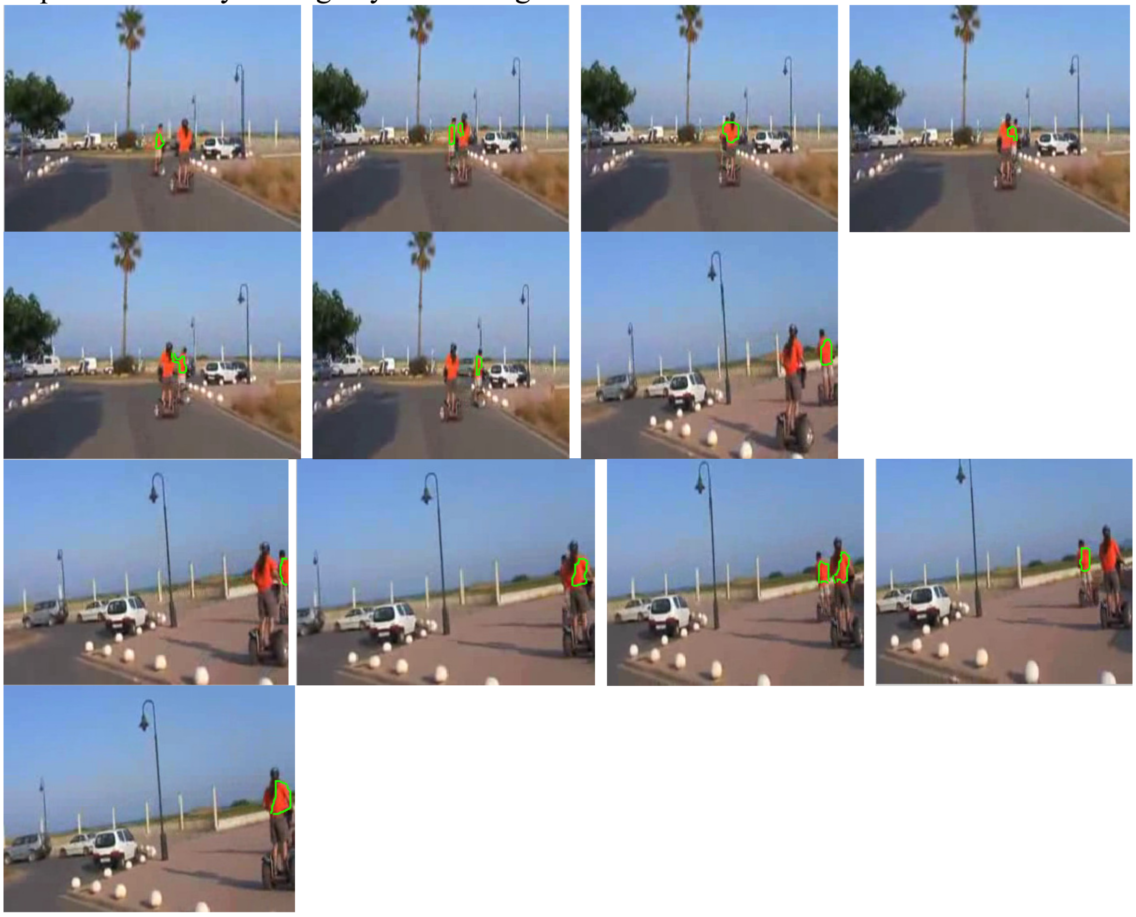
Table 2 presents the results (mean ± std. deviation) of the spatial overlap (SO) and centroid distance (CD) measures together with the failure ratio (FR) and accuracy (Acc) of each tracking method for the three tests S5 to S7, emphasizing the best values for each measure and test in bold. Our PIORT-Neural net method worked fine in the three tests obtaining the best values of spatial overlap and accuracy measures in tests S5 and S6 and yielding results a little bit under the performance of the VRFS method in test S7, in which the VRFS method gave the best values of the four measures. Our PIORT-Bayesian method worked well in test S5 but failed to track the orange ball correctly in tests S6 and S7. Only both PIORT methods performed well in S5; only PIORT-Neural net method worked in S6 while the rest failed; and only VRFS and PIORT-Neural net methods obtained satisfactory results in S7.


Sequence S8: Pedestrian with red jacket

The last set of experiments comprised another three test video sequences, taken with a moving camera in outdoor environments, where the targets are humans, more precisely, some part of their clothing. The first sequence in this group (test S8) is a long sequence taken on a street where the aim is to track a pedestrian wearing a red jacket (see http://www-iri.upc.es/people/ralqueza/S8.avi) and includes total and partial occlusions of the followed person by other walking people and objects on the street. In this case, a short sequence of the scene taken with a moving camera located in a different position (http://www-iri.upc.es/people/ralqueza/redpedestrian_training.avi) was used as training sequence.

The other two test sequences in this group, tests S9 and S10, show outdoor scenes in which humans riding Segway robots and wearing orange T-shirts are followed. In test S9 a single riding guy is followed, whereas in test S10, two men are riding two Segway robots simultaneously and crossing each other. These test sequences are at http://www-iri.upc.es/people/ralqueza/S9.avi and S10.avi and the training sequence associated with them is at http://www-iri.upc.es/people/ralqueza/T-shirt_training.avi.


Sequence S9: Guy on Segway with orange T-shirt


Sequence S10: Men on Segway with orange T-shirt

The tracking results videos for the above test sequences are attainable at http://www-iri.upc.es/people/ralqueza/S8_NN.mpg, S8_Bayes.mpg, S9_NN.mpg, S9_Bayes.mpg, S10_NN.mpg and S10_Bayes.mpg.

Table 3 presents the results of the evaluation measures of each tracking method for the three tests S8 to S10, emphasizing the best values for each measure and test in bold. Both PIORT methods gave the best results, very similar between them, in tests S8 and S9, and PIORT-Neural net method performed clearly the best in test S10. Note that in the pedestrian sequence (S8), an occlusion by people carrying red bags distracted the attention of the PIORT tracking module and caused a momentarily impairment in performance, especially for the centroid distance measure, but the tracker was able to recover correctly the target after that occlusion. In this sequence S8, only the PDFS method performed comparably to PIORT approaches in terms of accuracy and centroid distance, although it achieved a rather lower spatial overlap. In test S9, the HRS, VRFS and PDFS methods obtained similar and reasonably well results, but not as good as those of PIORT methods. Finally, only the PIORT-Neural net method worked well in test S10, where the PIORT-Bayesian method performed poorly because it followed the other Segway-riding man after a crossing between both men.

Table 3. Results of human tracking on video sequences taken with a mobile camera.

Video Sequence	**Tracking method**	**SO**	**CD**	**FR**	**Acc**
S8 Pedestrian with red jacket	TMC	0.44 ± 0.31	25.25 ± 61.10	0.07	0.77
	BM	0.24 ± 0.58	72.08 ± 64.33	0.07	0.34
	HRS	0.35 ± 0.24	13.49 ± 38.27	**0.02**	0.64
	VRFS	0.45 ± 0.32	34.27 ± 81.13	0.12	0.82
	PDFS	0.50 ± 0.20	11.42 ± 45.11	0.03	0.95
	GCBT	0.04 ± 0.32	194.7 ± 105.3	0.77	0.16
	PIORT-Neural Net	**0.79** ± 0.24	11.90 ± 50.87	0.04	**0.96**
	PIORT-Bayesian	0.74 ± 0.24	**11.15** ± 48.14	0.04	0.95
S9 Guy on Segway with orange T-shirt	TMC	0.10 ± 0.53	130.3 ± 69.75	**0.00**	0.15
	BM	0.22 ± 0.13	41.30 ± 58.70	0.01	0.40
	HRS	0.53 ± 0.25	22.83 ± 58.43	0.05	0.86
	VRFS	0.69 ± 0.25	27.69 ± 75.15	0.10	0.90
	PDFS	0.56 ± 0.21	29.19 ± 74.65	0.10	0.90
	GCBT	0.14 ± 0.22	101.6 ± 112.7	0.36	0.19
	PIORT-Neural Net	0.73 ± 0.16	**3.40** ± 14.78	**0.00**	0.97
	PIORT-Bayesian	**0.74** ± 0.13	3.70 ± 14.61	**0.00**	**0.98**
S10 Men on Segway with orange T-shirts	TMC	0.06 ± 0.39	104.3 ± 83.15	**0.03**	0.10
	BM	0.29 ± 0.28	42.10 ± 59.06	**0.03**	0.59
	HRS	0.28 ± 0.30	38.72 ± 65.09	0.06	0.58
	VRFS	0.38 ± 0.34	36.81 ± 64.53	0.06	0.61
	PDFS	0.32 ± 0.36	91.14 ± 119.4	0.35	0.56
	GCBT	0.04 ± 0.31	187.1 ± 103.1	0.72	0.08
	PIORT-Neural Net	**0.73** ± 0.18	**8.37** ± 40.74	**0.03**	**0.96**
	PIORT-Bayesian	0.16 ± 0.58	81.36 ± 62.93	**0.03**	0.22

SO: Spatial Overlap; CD: Centroid Distance; FR: Failure Ratio; Acc: Accuracy

7. Conclusions, discussion and future work

In this paper we have described an updated version of the *probabilistic integrated object recognition and tracking* (PIORT) methodology that we have developed in the latest years, partially reported in [36-39], and presented a collection of experimental results in test video sequences, with the aim of comparing two particular approaches derived from PIORT, based on Bayesian and neural net methods, respectively, with some state-of-the-art tracking methods proposed by other authors.

An improved method for object tracking, capable of dealing with rather long occlusions and same-class object crossing, has been proposed to be included within our probabilistic framework that integrates recognition and tracking of objects in image sequences. PIORT does not use any contour information but the results of an iterative and dynamic probabilistic approach for object recognition. These recognition results are represented at pixel level as probability images and are obtained through the use of a classifier (e.g. a neural network) from region-based features.

The PIORT framework is divided in three parts: a static recognition module, where the classifier is applied to single-frame images, a dynamic recognition module that updates the object probabilities using previous recognition and tracking results, and a tracking decision module, where tracking binary images are determined for each object. This third module combines the recognition probabilities with a model that predicts the object's apparent motion in terms of translation and scale changes, while coping with the problems of occlusion and re-emergence detection. Moreover, the tracking module can deal with object splitting, either due to partial occlusions or same-class object crossing, and, in most cases, is able to select and track only the target object after it crosses or is occluded by another object which is recognized as belonging to the same class, i.e. it is able to re-establish the identity of the target object.

The experimental work reported in this paper has been focused on the case of single object tracking, just because the tracking methods we had available for the comparison only allowed single object tracking. However, as shown in [36], the PIORT system is capable of tracking multiple objects of different classes simultaneously and, as demonstrated in the experiments, it can be applied to video sequences acquired either by a fixed or a moving camera. The size, shape and movement of the target objects can vary softly along the sequence, but the appearance features used by the classifier (up to now, colour features) should remain rather stable for a successful tracking. It must be taken into account that the global performance of the system depends not only on the ability of the tracking method but also on the quality of the object recognition probabilities provided by the trained classifier.

In this regard, false positive detections by the classifier can only be harmful for tracking when they are very close or "touching" the target, otherwise they are filtered by the second and third modules. Even in the first case, the tracking module is sometimes able to distinguish between the target and a false distracter, when the latter is different enough in terms of size, shape or motion trajectory. Concerning false negative errors by the classifier, they can be coped partially by the second module, especially when the apparent motion of the target is slow and hence the previous probabilities (adaptively) weight more than the current ones given by the classifier. Nevertheless, if the classifier fails to detect the target object in just a few consecutive frames the tracker will assume a target occlusion and proceed in occlusion mode, which implies an assumption of constant motion directed by the previous trajectory and a growing uncertainty in the target position. In this case, object tracking can be sometimes recovered if the classifier redetects the target afterwards, depending basically on the real trajectory of the target and the gap duration.

In this paper, we have presented two static recognition methods that can be embedded in the first module of PIORT, giving rise to two different instances of the methodology. Both methods are based on the use of a classifier that is trained from examples and provides posterior class probabilities for each pixel from a set of local features. The first classifier is based on a maximum likelihood Bayesian method in which the conditional probabilities for object classes are obtained from the information of the class histograms (for discretized RGB values) and a uniform conditional probability is assumed for the background. The second classifier is based on a neural net which is trained with the RGB colour averages extracted for each spot of the segmented images.

Even though the characteristics of these two classifiers are quite different, the recognition and tracking results of PIORT using both approaches were excellent and very similar in five of the ten test sequences, which might mean that the good ability of PIORT to track the objects is mostly due to a smart cooperation of the three inner modules and is not very dependent on the specific method used for object recognition. However, in the remaining five test sequences, the tracking method based on a neural net classifier clearly outperformed the one based on a simple Bayesian classifier, which failed in three of these test sequences. Indeed, we observed that updating the histograms at each frame may cause severe drift errors when the tracker begins to fail, which result in a rapid breakdown of the Bayesian classifier performance in subsequent frames. Hence, depending on the particular application, it might be preferable not to update the histograms after training.

The performance of both Bayesian and neural net classifiers also depends somewhat on the quality of the image segmentation process carried out previously. In the case of good segmentations, like the ones we obtained using EDISON for the test sequences, the probability images given by the classifiers are smooth (large areas with same values) and this eases the tracking, whereas in the case of over-segmentations, the probability images may be noisy due to an excess of spots and this may hinder a stable tracking.

In the experimental comparison with other six methods proposed in the literature for object tracking, a PIORT method obtained the best results in nine of the ten test sequences and only a slightly inferior performance with respect to best method in the other one (VRFS). Except for the case of the first test sequence S1, where all methods worked fine, the six alternative methods tested mostly failed to track the target objects correctly in the test sequences, due to the difficult instances of occlusions and object crossings they contain. However, we are aware of the fact that the six alternative methods tested here are not model-based (i.e. they are not trained in advance) to the contrary of PIORT, and thus, it is little surprising that PIORT obtained the best results. The availability (for us) of their implementation was the main reason why we selected them, but we foresee to carry out future experimental comparisons of PIORT against some state-of-the-art model-based tracking methods like those by Cremers [5] and

Lepetit [15] (once we have available an implementation of these methods to run the experiments).

Although further experimental work is needed, the new tracking module included in PIORT has demonstrated by now to be effective under several-frames occlusions produced by an object of a class different to that of the target object. If the occluding and the target objects are recognised as belonging to the same class, then the occlusion is not detected as such, both objects are merged temporarily, but despite this behaviour, the tracking method is able in most cases to recover and track the original target when the same-class object occlusion or crossing ends. However, as observed in some of the test sequences, still there are cases where the behaviour of the tracking decision module of PIORT should be improved, particularly in the step of object re-emergence after occlusion and when other objects of similar appearance are next to the target. The upgrade of this tracking module will be subject of future research.

We think that PIORT approaches for object tracking are especially suitable in noisy environments where segmented images vary so much in successive frames that it is very hard to match the corresponding regions or contours of consecutive images. The empirical results presented are quite satisfactory, despite the numerous mistakes made by the static recognition module, which can be mostly ignored thanks to the integration with the proposed tracking decision module.

A right criticism that can be raised against PIORT is that too many parameters need to be set. Apart from the parameters specific of the classifier in the first (static recognition) module, the dynamic recognition module uses two parameters, which are bounds on the linear adaptive weighting of previous and current probabilities, and the tracking decision module uses up to twelve parameters: six related to the uncertainty in the target position prediction, three for tracking image post-processing filters (one for each filter), two for occlusion mode determination and one more for a weighted average computation of the target movement vector. It is very difficult to get rid of these parameters in our approach, but the default values reported in the previous sections have been tuned carefully to yield a stable satisfactory behaviour of PIORT in all the test sequences. Of course, for new sequences, these default values may not be optimal and some further tuning might improve the performance. A sensitivity analysis for each one of the PIORT parameters would be extremely hard to do and assess, since the system response may depend a lot also on the specific features of the input sequences. By experience, we hypothesize and claim that, in general, small variations on the given default values do not affect importantly the obtained tracking results, but larger ones could do.

As future work, we want to extend the experimental validation of PIORT by applying it to new and more difficult image sequences; in particular, sequences where multiple objects are tracked simultaneously in the scene. And, as commented before, new comparative studies against state-of-the-art model-based tracking methods (e.g. [5, 15]) would be very interesting to do whenever possible.

For the two currently used approaches in the static recognition module, an obvious upgrade is to replace the RGB by the HSI colour space, since the latter seems to be more suited for matching or tracking objects, especially in natural environments with changing illumination. In addition, we are interested in implementing and testing new classifiers in the static recognition module, which could exploit other features completely different to the basic colour features used up to now. For instance, an SVM classifier could be applied to a set of features formed by Gabor filter responses, provided that class probability values were estimated from margin values.

Another possible extension would be to replace in the third module the simple rules used in the *a-priori* predictions of target centres and areas (equivalent to noiseless Kalman filters) by the whole Kalman filter formulation considering noise for both the dynamics and the observations. However, this replacement would increase even more the number of the system parameters and it is not clear that resulted in significant changes in the whole system behaviour.

Acknowledgements

This research was partially supported by Consolider Ingenio 2010, project CSD2007-00018, by the CICYT project DPI 2007-61452 and by the Universitat Rovira i Virgili (URV) through a predoctoral research grant.

References

[1] H. Wang, J. Peng, L. Li, "Runway detection of an unmanned landing aerial vehicle based on vision", IJPRAI 20(8), pp: 1225-1244, 2006.

[2] H-D. Yang, S-W. Lee, S-W. Lee, "Multiple Human Detection and Tracking Based on Weighted Temporal Texture Features", IJPRAI 20(3), pp: 377-392, 2006

[3] J-W. Hsieh, Y-S. Huang, "Multiple-person tracking system for content analysis", IJPRAI 16(4), pp: 447-462 2002.

[4] A. Villanueva, R. Cabeza, S. Porta, "Gaze Tracking System Model Based on Physical Parameters", IJPRAI 21(5), pp: 855-878, 2007

[5] S. Kang, B-W. Hwang, S-W. Lee, " Multiple People Tracking Based on Temporal Color Feature", IJPRAI 17(6), pp: 931-949, 2003

[6] J. Ning, L. Zhang, D. Zhang, C. Wu, "Robust Object Tracking Using Joint Color-Texture Histogram", IJPRAI 23(7), pp: 1245-1263, 2009.

[7] A. Sanfeliu, F. Serratosa, R. Alquézar, "Second-order random graphs for modeling sets of attributed graphs and their application to object learning and recognition", *Int. Journal of Pattern Recognition and Artificial Intelligence* 18 (2004) 375-396.

[8] A. Chatterjee, O. Ray, A. Chatterjee, A- Rakshit, "Development of a real-life EKF based SLAM system for mobile robots employing vision sensing", *Expert Systems with Applications*, (38), pp: 8266–8274, 2011.

[9] Li, Stan Z.; Jain, Anil K. (Eds.) "Handbook of Face Recognition", Springer, 2005.

[10] K. Burçin, N.V. Vasif, "Down syndrome recognition using local binary patterns and statistical evaluation of the system", *Expert Systems with Applications*, (38), pp: 8690–8695, 2011.

[11] C.Y. Tsai, Y.H. Lee, "The parameters effect on performance in ANN for hand gesture recognition system", *Expert Systems with Applications*, (38), pp: 7980–7983, 2011.

[12] G. L. Foresti, "Object recognition and tracking for remote video surveillance", *IEEE Trans. on Circuits and Systems for Video Technology* 9 (1999) 1045-1062.

[13] Francesc Moreno-Noguer, Alberto Sanfeliu, Dimitris Samaras, "Dependent Multiple Cue Integration for Robust Tracking", IEEE Trans. Pattern Anal. Mach. Intell. 30(4), pp: 670-685, 2008.

[14] H. Lee, J. Kim, H. Ko, "Prediction Based Occluded Multitarget Tracking Using Spatio-Temporal Attention", IJPRAI 20(6), pp: 925-938, 2006.

[15] C-J. Chang, J-W Hsieh, Y-S. Chen, W-F. Hu, "Tracking Multiple Moving Objects using a Level-Set Method", IJPRAI 18(2), 2004.

[16] A. Senior, A. Hampapur, Y-L. Tian, L. Brown, S. Pankanti, R. Bolle, "Appearance models for occlusion handling", *Image and Vision Computing* 24 (2006) 1233-1243.

[17] H.T. Nguyen, A.W.M. Smeulders, "Fast occluded object tracking by a robust appearance filter", *IEEE Trans. Pattern Anal. Mach. Intell.* 26 (2004) 1099-1104.

[18] S.K. Zhou, R. Chellappa, B. Moghaddam, "Visual tracking and recognition using appearance-adaptive models in particle filters", *IEEE Trans. Image Process.* 13 (2004) 1491-1506.

[19] A.D. Jepson, D.J. Fleet, T.F. EI-Maraghi, “Robust online appearance models for visual tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.* 25 (2003) 1296-1311.
[20] K. Ito, S. Sakane, “Robust view-based visual tracking with detection of occlusions”, in: *Proc. Int. Conf. Robotics Automation*, 2001, vol. 2, pp. 1207-1213.
[21] K. Hariharakrishnan, D. Schonfeld, “Fast object tracking using adaptive block matching”, *IEEE Trans. Multimedia* 7 (2005) 853-859.
[22] L. Zhu, J.Zhou, J. Song, “Tracking multiple objects through occlusion with online sampling and position”, *Pattern Recognition* 41 (2008) 2447-2460.
[23] J. Pan, B. Hu, "Robust occlusion handling in object tracking", in: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (CVPR 2007), Minneapolis, Minnesota, June 2007.
[24] Z. Tu, X. Chen, A.L. Yuille, S.C. Zhu, “Image parsing: unifying segmentation, detection, and recognition”, in: *Proc. Ninth IEEE Int. Conf. on Computer Vision*, 2003, pp 18- 25.
[25] K-C. Lee, J. Ho, M-H. Yang, D. Kriegman, “Visual tracking and recognition using probabilistic appearance manifolds”, *Computer Vision and Image Understanding* 99 (2005) 303-331.
[26] S.C. Zhu, A. Yuille, “Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation”, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 18 (1996) 884-900.
[27] J. Malik, S. Belongie, T. Leung, J. Shi, “Contour and texture analysis for image segmentation”, *Int. J. Computer Vision*, 43 (2001) 7-27.
[28] Z. Tu, S.C. Zhu, “Image segmentation by data driven Markov chain Monte Carlo”, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 24 (2002) 657-673.
[29] F-S. Chen, C-M. Fu, C-L. Huang, “Hand gesture recognition using a real-time tracking method and hidden Markov models”, *Image and Vision Computing* 21 (2003) 745-758.
[30] L. Maddalena, A. Petrosino, A. Ferone, “Object Motion Detection and Tracking by an Artificial Intelligence Approach”, IJPRAI 22(5), 2008.
[31] N. Amezquita Gomez, R. Alquézar, F. Serratosa, “Object recognition and tracking in video sequences: a new integrated methodology“, in: *Proc. 11th Iberoamerican Congress on Pattern Recognition,* CIARP 2006, LNCS 4225, pp. 481-490.
[32] N. Amézquita Gómez, R. Alquézar, F. Serratosa, “A new method for object tracking based on regions instead of contours”, in: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (CVPR 2007), Minneapolis, Minnesota, June 2007.
[33] N. Amézquita Gómez, R. Alquézar, F. Serratosa, “Dealing with occlusion in a probabilistic object tracking method”, in: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (CVPR 2008), Anchorage, Alaska, June 2008.
[34] R. Alquézar, N. Amézquita Gómez, F. Serratosa, “Tracking deformable objects and dealing with same class object occlusion”, in: *Proc. Fourth Int. Conf. on Computer Vision Theory and Applications* (VISAPP 2009), Lisboa, Portugal.
[35] D. Comaniciu, V. Ramesh, P. Meer, “Kernel-based object tracking”, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 25 (2003) 564-577.
[36] D. Comaniciu, P. Meer, “Mean shift: a robust approach toward feature space analysis”, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 24 (2002) 603-619.
[37] R. Collins, Y. Liu, “On-line selection of discriminative tracking features”, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 27 (2005), 1631-1643.
[38] A. Bugeau, P. Pérez, “Track and cut: simultaneous tracking and segmentation of multiple objects with graph cuts”, in: *Proc. Third Int. Conf. on Computer Vision Theory and Applications* (VISAPP 2008), Funchal, Madeira, Portugal.
[39] Y. Boykov, O. Veksler, R. Zabih, “Fast approximate energy minimization via graph cuts”, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 23 (2001) 1222-1239.
[40] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.
[41] F. Yin, D. Makris, S.A. Velastin, “Performance evaluation of object tracking algorithms”, in: *Proc. 10th IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance* (PETS’2007).
[42] S.M. Khan & M. Shah, “Tracking Multiple Occluding People by Localizing on Multiple Scene Planes”, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 31 (2009) 505-519.
[43] D. Cremers, “Dynamical Statistical Shape Priors for Level Set Based Tracking”, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 28 (2006) 1262-1273.
[44] V. Lepetit & P. Fua, “Keypoint Recognition using Randomized Trees”, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 28 (2006) 1465-1479.
[45] M. de la Gorce, N. Paragios and David Fleet. “Model-Based Hand Tracking with Texture, Shading and Self-occlusions”, *IEEE Conference in Computer Vision and Pattern Recognition*, 2008.
[46] Yan Huang, Irfan A. Essa: Tracking Multiple Objects through Occlusions. *IEEE Conference in Computer Vision and Pattern Recognition 2005*: 1051-1058.
[47] T.Yang, S.Li, Q.Pan, J.Li, “Real-Time multiple objects tracking with occlusion handling in dynamic scenes”, ”, *IEEE Conference in Computer Vision and Pattern Recognition*, 2005, 970 - 975.