# Importance of detection for video surveillance applications

**Javier Varona**
Universitat de les Illes Balears (UIB)
Dept. C. Mat. i Informàtica
Ed. A.Turmeda, crt. Valldemossa km 7.5
Palma de Mallorca, Spain
E-mail: xavi.varona@uib.es

**Jordi Gonzàlez**
Institut de Robòtica i Informàtica Industrial
(UPC-CSIC)
Barcelona, Spain

**Ignasi Rius**
**Juan José Villanueva**
Computer Vision Center and
Universidad Autònoma de Barcelona
Bellaterra, Barcelona, Spain

**Abstract.** Though it is the first step of a real video surveillance application, detection has received less attention than tracking in research on video surveillance. We show, however, that the majority of errors in the tracking task are due to wrong detection. We show this by experimenting with a multi object tracking algorithm based on a Bayesian framework and a particle filter. This algorithm, which we have named iTrack, is specifically designed to work in practical applications by defining a statistical model of the object appearance to build a robust likelihood function. Likewise, we present an extension of a background subtraction algorithm to deal with active cameras. This algorithm is used in the detection task to initialize the tracker by means of a prior density. By defining appropriate performance metrics, the overall system is evaluated to elucidate the importance of detection for video surveillance applications. 
[DOI: 10.1117/1.2965548]

Subject terms: video surveillance; visual tracking; target detection.

Paper 080143R received Feb. 21, 2008; revised manuscript received May 22, 2008; accepted for publication May 23, 2008; published online Aug. 7, 2008.

## 1 Introduction

Monitoring public and private sites by means of human operators presents several problems, such as the need to monitor a great number of cameras at the same time and to minimize the operator's distractions. Therefore, a computer vision system has to be able to assist humans. The main difficulties of an automatic video surveillance system are due to the variety of scenes and acquisition conditions. It is possible to design systems with one or more cameras, which can be static or mobile, with different sensors such as color or infrared cameras. In this paper, we deal with large outdoor scenes using an active color camera.

Typically, an automatic video surveillance system involves the following tasks: detection, tracking, and event recognition. The detection task locates objects in the images. Then, the objects' positions are robustly estimated over time by the tracking task. Lastly, the goal of the event recognition module is to describe what is happening in the scene.

There are two main approaches for object detection in automatic video surveillance applications: temporal differences and background subtraction. Frame differencing performs well in real time, but fails when a tracked object ceases its motion.[1] Background subtraction is based on statistical models in order to build the appearance model of a static scene.[2] Both methods usually require the use of a static camera. Recently, advances in algorithms for robust real-time object detection allow their use in video surveillance applications. These algorithms perform a search in the image to find previously learned objects such as pedestrians.[3] An important advantage of these algorithms is that they are not restricted to a static camera. In view of this, we present an algorithm that can be used with active cameras. This algorithm allows the application of background subtraction techniques to panoramic scenes typical of video surveillance applications.

Referring to the tracking module, there are works based on a combination of different computer vision algorithms that perform properly in real environments.[4,5] The main difficulty of these tracking algorithms is representing objects' trajectories when new objects appear, when they are occluded, or when they disappear. To manage these cases one needs a process of data association, usually based on heuristics. Another possibility is to use a particle filter.[6] Particle filters are a possible implementation of optimal Bayesian estimation.[7] They can manage multimodal densities to represent the state of multiple objects. However, it is necessary to use an adequate state representation to apply these filters to multiobject tracking. It is possible to include all objects and the background in the state estimation.[8] But this approach may require an extremely large number of samples.

Instead, we present a tracking algorithm for the management of multiobject tracking by augmenting the state of each tracked object with a label to identify the object. This scheme is completed with a likelihood function whose definition is directly based on the image values of the objects to be tracked. This model can be updated to allow for changes in the object's appearance. Therefore, the algorithm does not depend on environmental conditions, and it can be used in different application scenarios because it does not require any *a priori* knowledge about either the scene or the appearance and number of agents. It is only necessary to define an appropriate prior density that relates detection and tracking to adapt the application to several scenarios. By means of a proper evaluation of the video surveillance system, we are able to show the relationships between detection and tracking tasks. Specifically, we prove by experi-

0091-3286/2008/$25.00 © 2008 SPIE

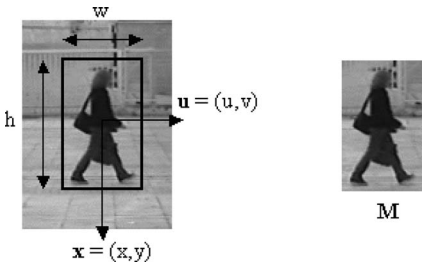



**Fig. 1** State components.

menting how the performance of the tracking algorithm is affected by the presence of detection errors.

In this paper, we first define a visual tracking method suitable for video surveillance applications. This method is based on a Bayesian framework and a particle filter. Subsequently, we present a background subtraction algorithm for active cameras that is used by the detection task to locate the objects of interest in the scene. In addition, we present a proper definition of a prior density, which relates to both detection and tracking. Finally, performance metrics are defined to evaluate the behavior of the complete system. The obtained results are discussed in the last section to demonstrate the importance of detection for obtaining good results in the tracking task.

## 2 Image-Based Tracking: iTrack

In this section, we define an estimation algorithm to track people in video surveillance applications, which we have named iTrack. This algorithm is based on the Bayesian probabilistic framework and implemented by using a particle filter. The algorithm's basic idea is to estimate the state of the object to be tracked by using a likelihood function that is based only on image data. This idea is formalized by defining an appearance model that is continuously updated to take into account the objects' appearance changes. In addition, by using a particle filter, the detection results are easily included in the estimation algorithm by introducing new particles from a prior density. Then, the algorithm can be used in different application environments without significant changes.

Let $\mathbf{s}_t=(\mathbf{x}_t,\mathbf{u}_t,\mathbf{w}_t,\mathbf{M}_t)$ be the state vector for an object, where $\mathbf{x}_t=(x_t,y_t)$ is the position, $\mathbf{u}_t=(u_t,v_t)$ the velocity, $\mathbf{w}_t=(w_t,h_t)$ the size, and $\mathbf{M}_t$ the appearance of the object (see Fig. 1).

Given a sequence of images, $\mathbf{I}_{1:t}=(\mathbf{I}_1,\ldots,\mathbf{I}_t)$, the posterior probability density of the object's state at time $t$ is expressed as

$$p(\mathbf{s}_t|\mathbf{I}_{1:t})=\int p(\mathbf{s}_{1:t}|\mathbf{I}_{1:t})\mathrm{d}\mathbf{s}_{1:t-1}, \tag{1}$$

where $\mathbf{s}_{1:t}$ is the object state history, $\mathbf{s}_{1:t}=(\mathbf{s}_1,\ldots,\mathbf{s}_t)$. Applying the Bayes rule and the Markov condition, we obtain

$$p(\mathbf{s}_t|\mathbf{I}_{1:t})\propto p(\mathbf{I}_t|\mathbf{s}_t)\int p(\mathbf{s}_t|\mathbf{s}_{t-1})p(\mathbf{s}_{t-1}|\mathbf{I}_{1:t-1})\mathrm{d}\mathbf{s}_{t-1}, \tag{2}$$

where $p(\mathbf{I}_t|\mathbf{s}_t)$ is the likelihood function.

The integral in Eq. (2) is referred to as the *temporal prior* or the *prediction*, and $p(\mathbf{s}_t|\mathbf{s}_{t-1})$ is the *motion model*. In order to define the motion model we assume the following independent relations between the state parameters:

$$p(\mathbf{x}_t,\mathbf{u}_t,\mathbf{w}_t,\mathbf{M}_t|\mathbf{x}_{t-1},\mathbf{u}_{t-1},\mathbf{w}_{t-1},\mathbf{M}_{t-1})$$
$$=p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_{t-1})p(\mathbf{u}_t|\mathbf{u}_{t-1})p(\mathbf{w}_t|\mathbf{w}_{t-1})p(\mathbf{M}_t|\mathbf{M}_{t-1}).$$

We use the smooth motion model for the position, velocity, and size parameters, i.e.,

$$p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_{t-1})=\eta(\mathbf{x}_t-(\mathbf{x}_{t-1}+\mathbf{u}_{t-1}),\sigma^{\mathbf{x}}),$$

$$p(\mathbf{u}_t|\mathbf{u}_{t-1})=\eta(\mathbf{u}_t-\mathbf{u}_{t-1},\sigma^{\boldsymbol{u}}),$$

$$p(\mathbf{w}_t|\mathbf{w}_{t-1})=\eta(\mathbf{w}_t-\mathbf{w}_{t-1},\sigma^{\boldsymbol{w}}),$$

where $\eta(\mu,\sigma)$ denotes a Gaussian density with mean $\mu$ and standard deviation $\sigma$. The deviations $\sigma^{\mathbf{x}}$ $\sigma^{\mathbf{u}}$, and $\sigma^{\mathbf{w}}$ are defined empirically. To complete the motion model, it is necessary to define the appearance evolution, $p(\mathbf{M}_t|\mathbf{M}_{t-1})$. Using probabilistic terms, the density of the appearance model is defined as

$$p(\mathbf{M}_t|\mathbf{M}_{t-1})=\delta(\mathbf{M}_t-\mathbf{M}_{t-1}), \tag{3}$$

where $\delta(\cdot)$ is a Dirac delta function. This model was also used for 3-D people-tracking.[9]

### 2.1 *Appearance Model for the Likelihood Function*

To compute the recursive expression (2) we also need a likelihood function, i.e., $p(\mathbf{I}_t|\mathbf{x}_t,\mathbf{u}_t,\mathbf{w}_t,\mathbf{M}_t)$. This function is the probability of observing the image $\mathbf{I}_t$ given the object parameters. First, we observe that the likelihood function is independent of the velocity parameter. The parameters $\mathbf{x}_t$ and $\mathbf{w}_t$ define an image region denoted as $\mathbf{I}^p$. In order to compare this image region with the object appearance model $\mathbf{M}_t$, we apply an affine transformation to the image region:

$$\mathbf{R}=\mathbf{A}\mathbf{I}^p, \tag{4}$$

where $\mathbf{A}$ is an affine matrix transform containing translation and scale parameters. Finally, the complete likelihood function is expressed as

$$p(\mathbf{I}_t|\mathbf{x}_t,\mathbf{w}_t,\mathbf{M}_t)=p(\mathbf{R}|\mathbf{M}_t), \tag{5}$$

$$p(\mathbf{R}|\mathbf{M}_t)=\frac{1}{N}\sum_{i,j\in\mathbf{R}}p_{ij}(R_{ij}|M_{ij,t}), \tag{6}$$

where $N$ is the number of the region's pixels, and $p_{ij}$ is the probability that the value of the pixel $(i,j)$ belongs to the distribution of the pixel's appearance model and is defined as

$$p_{ij}(R_{ij}|M_{ij,t})=\eta(R_{ij}-M_{ij,t},\sigma^M), \tag{7}$$

where $\eta(\cdot)$ is a Gaussian density whose standard deviation $\sigma^M$ allows for small changes in object appearance and acquisition noise. A similar appearance model for dynamic layers is presented in Ref. 10. The main difference is that

model is based on a generalized EM algorithm instead of a particle filter to continuously estimate objects over time. This definition of the likelihood function is robust to outliers, because their presence (due to clutter and occlusions) does not penalize the overall probability measurement.

The expression (3) means that the object appearance does not change over time. Thus, it is necessary to adjust the model after each estimation step for a correct appearance model. Once the new state has been estimated, $p(\mathbf{s}_t|\mathbf{I}_{1:t})$, the appearance model is updated using an adaptive rule for each pixel of the model,

$$\mu_{ij,t} = \mu_{ij,t-1} + \alpha(R_{ij,t} - \mu_{ij,t-1}), \tag{8}$$

where $R_{i,j,t}$ is the appearance value of pixel $(i,j)$ of the region obtained with the new state parameters. To learn the coefficient $\alpha$, we use the temporal adjustment

$$\alpha_t = e^{-t}. \tag{9}$$

We have chosen this approximation because the best estimations are computed during the first frames.

The results on the expected positions and the marginal density for the $x$ position of different test sequences are shown in Fig. 2. In the marginal density for the $x$ position it can be seen the multimodality of the posterior density in the multiple-object tracking case.

## 2.2 *Algorithm*

In order to make multiple-object tracking possible, it is necessary to represent a multimodal density. Using the Condensation algorithm, we can implement the probabilistic model by means of a particle filter.[6] Therefore, the conditional state density, $p(\mathbf{s}_t|\mathbf{I}_{1:t})$, is represented by a sample set $\{\mathbf{s}_t^{(n)}\}$, $n=1,\ldots,N$. In order to represent a multimodal density and to identify each object, we use an augmented state adding a label $l$. The label $l$ associates one specific appearance model to the corresponding samples, allowing the computation of the likelihood function of Eq. (6). Thus, the sample vector is given by

$$\mathbf{s}_t^i = (\mathbf{x}_t^i, \mathbf{u}_t^i, \mathbf{w}_t^i, l) \equiv \mathbf{s}_t^{i,l} = (\mathbf{x}_t^i, \mathbf{u}_t^i, \mathbf{w}_t^i).$$

From the propagated samples $\{\mathbf{s}_t^i\}$, that represent the posterior at time $t$, the state estimation for the object labeled $L$ is computed as the mean of their samples, i.e.,

$$\hat{\mathbf{s}}_{L,t} = \frac{1}{N_L} \sum_{i,l=L} \mathbf{s}_t^{i,l}, \tag{10}$$

where $N_L$ is the number of samples for the object $L$. However, as the estimation progresses over many frames this representation may increasingly bias the posterior density estimates towards objects with dominant likelihood.[11] This occurs because the probability of propagating a mode is proportional to the cumulative weights of the samples that constitute it. In order to avoid single target modes absorbing other target samples, weights are normalized according to

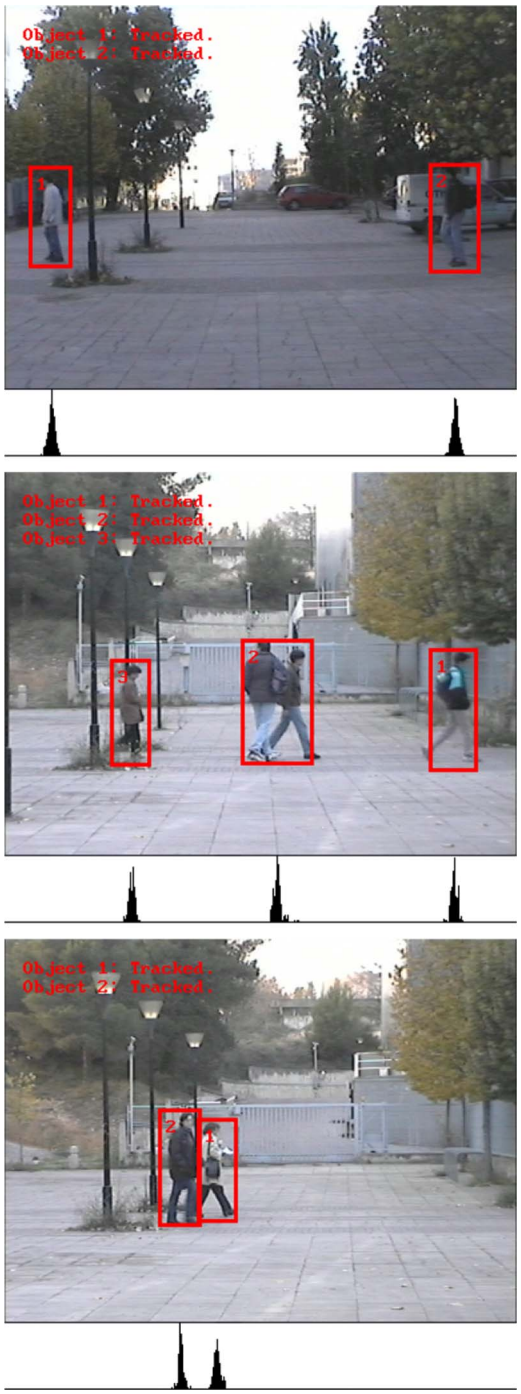



**Fig. 2** Multiple-object tracking results using iTrack. At the bottom of each frame is displayed the horizontal position marginal density.

$$\hat{\pi}_t^{i,l} = \frac{\pi_t^{i,l}}{\sum_{i=1,j=l}^{N} \pi_t^{i,j}} \frac{1}{N_O}, \tag{11}$$

where $N_O$ is the number of objects being tracked. Each weight is normalized according to the total weight of the target's samples. Thus, all targets have the same probability of being propagated.[12] The complete algorithm is described in Table 1.

**Table 1** iTrack algorithm.

The posterior density at time $t-1$ is represented by the sample set, $\{\mathbf{s}_{t-1}^i\}$, where $i=\{1,\ldots,N\}$.

In addition, the prior density $p(\mathbf{s}_t)$ for time $t$ is assumed to be known at this stage.

Generate the $i$'th sample of $N$ that represents the posterior at time $t$ as follows:

**1. Predict:** Generate a random number $\alpha \in [0,1)$ uniformly distributed:

(a) If $\alpha < r$, use the prior $p(\mathbf{s}_t)$ to generate $\mathbf{s}_t^{i-}$.

(b) If $\alpha \geq r$, apply the motion model to the sample $\mathbf{s}_{t-1}^i$: $\mathbf{s}_t^{i-}=p(\mathbf{s}_t|\mathbf{s}_{t-1}=\mathbf{s}_{t-1}^i)$ using the smooth motion model

$$\mathbf{x}_t^{i-}=\mathbf{x}_{t-1}^i+\mathbf{u}_{t-1}^i+\xi_x^i,$$

$$\mathbf{u}_t^{i-}=\mathbf{u}_{t-1}^i+\xi_u^i,$$

$$\mathbf{w}_t^{i-}=\mathbf{w}_{t-1}^i+\xi_w^i.$$

**2. Correct:** Measure and weight the new sample, $\mathbf{s}_t^{i-}$, in terms of image data $\mathbf{I}_t$, using the likelihood function of Eq. (6):

$$\pi_t^{i,l}=p(\mathbf{I}_t|\mathbf{x}_t=\mathbf{x}_t^{i-},\mathbf{w}_t=\mathbf{w}_t^{i-},\mathbf{M}_{t-1}^l).$$

Once the $N$ samples have been generated, normalize the weights applying Eq. (11), and
build the cumulative probabilities:

$$c_t^0=0,$$

$$c_t^i=c_t^{i-1}+\pi_t^i,\ i=1,\ldots,N.$$

Use the values of the cumulative probabilities to generate by sampling the new samples $\{\mathbf{s}_t^i\}$ that
represent the posterior at time $t$.

For each object, estimate the new state by computing the mean of its samples:

$$\hat{\mathbf{s}}_{L,t}=\frac{1}{N_L}\Sigma_{i,i=L}\mathbf{s}_t^i,$$

where $N_L$ is the number of samples for object $L$

Finally, use the new state to actualize the appearance model.

## 3 Detection

The iTrack algorithm requires a prior density, $p(\mathbf{s}_t)$, for the tracking process to be initialized. Subsequently, this prior density is used to initialize new objects appearing in the scene. In this section, we define the prior density by using the results obtained in the detection task. First, we present a background subtraction algorithm for active cameras that is used for locating the objects of interest in the scene. This method is an extension of a robust background subtraction algorithm for active cameras.[13] It uses a Gaussian-mixture-based adaptive background modeling. In this way, it is robust to changes in the scene that are not due to the objects of interest. The problem of this algorithm is that it requires a static scene. To solve this problem it is possible to make a scene set with one image for each acquisition parameter of the active camera. However, that is impractical due to the great number of active parameters of the camera. To address this problem, one could find the minimum set of the camera's parameters for seeing the entire surveillance perimeter and constraint the camera motions to these parameters.[14] A less expensive method is to model the scene like a panorama.[15,16] Therefore, our objective is to use the Mixture-of-Gaussians scene model for active cameras by means of a panoramic representation of the scene.

**Fig. 3** Panorama for parking monitoring.

### 3.1 *Panoramic Scene Model*

In video surveillance applications, for monitoring a wide area with enough image resolution, active cameras are usually used. These cameras scan the entire surveillance perimeter to detect events of interest. Another possibility is to use a static camera with a wide field of view to locate objects and an active camera to track them. However, this approximation needs geometric and kinematic coupling between the two cameras.[17] Therefore, we focus on using an active camera with *pan* and *tilt* degrees of freedom.

First, we explain how to build the panorama by assuming that the camera rotates about its optical center. In order to build the panorama, it is necessary to transform each image into a sphere corresponding to the camera field of view. Next, we convert the spherical panorama into a planar surface to represent the scene model. In addition, we assume that it is possible to know the camera's parameters to make our scene model—in our case, the pan degree of freedom $\theta$, the tilt degree $\psi$, and the camera's field of view in both directions, $\beta$ and $\gamma$ (we assume a fixed focal length; therefore we do not consider the zoom parameter). First, in order to project each point $(x,y)$ of the camera into the sphere, we apply

$$\theta_x=\theta+x\cdot\frac{\beta}{S_x}, \tag{12}$$

$$\psi_x=\psi+y\cdot\frac{\gamma}{S_y}, \tag{13}$$

where $S_x$ and $S_y$ are the horizontal and vertical image sizes, respectively, and $x,y$ are the pixel coordinates with respect to the image center, that is, $x\in[-S_x/2,S_x/2]$ and $y\in[-S_y/2,S_y/2]$. Next, the image axes are matched with the angular values of the transformed pixels by the previous expressions. For example, in Fig. 3 we show a panorama that corresponds to 100 deg for pan values and 20 deg for tilt values. To avoid lens distortions, we only take into account the central region of each image.

This process to create panoramas has two main problems: brightness changes and the appearance of parts of


**Fig. 4** Indexation of a panoramic scene model.

**Table 2** Evaluation of the panoramic scene model.

| Class | Detected objects $A_i$ | False positives $A_i - B_i$ | Index of false alarms $m_i$ |
|---|---|---|---|
| Pedestrians | 82 | 22 | 0.27 |
| Other | 993 | 70 | 0.07 |

objects in motion. The first problem is due to the nonuse of brightness correction functions. The second problem occurs if there are objects in motion when the images are taken to build the panorama. In our approach, we use the scene model based on the mixture-of-Gaussians parameterization for each pixel. As a result, both problems are solved. The process consists in creating the panorama following the previously explained steps and using the mean value of the most probable Gaussian as panorama pixel value.

Once the panoramic scene is modeled, in order to make object detection, the known values of the pan and tilt camera parameters are used to obtain the desired piece of the panoramic scene model. This process is carried out by indexing the scene when it is created. To speed up computation, we use a lookup table with the correspondence between the position of each pixel within the image and its position in the panorama for the different camera's parameters (see Fig. 4).

In Fig. 4 it can be seen that both problems in building panoramas—brightness uniformity and objects in motion—are solved by using the panoramic scene model based on mixtures of Gaussians. For each registered piece of the scene, it is possible to detect the objects in motion. Therefore, they can be deleted for building the panorama, and they do not appear in the final scene model. In addition, by creating the panorama using the mean values of the most probable Gaussian density, the problem of change of brightness between consecutive pieces is eliminated. For these two reasons, our method provides a robust way of creating panoramas in the presence of objects in motion and smooth changes of illumination. This is an improvement over the methods that create panoramas assuming static scenes. However, we want to point out that a similar algorithm, presented in Ref. 18 can be viewed as a generalization of this idea for nonvideo surveillance applications. This background subtraction algorithm has been applied to scene modeling using sequences from hand-held cameras.

In order to evaluate our panoramic scene model, two classes of objects have been defined: pedestrians and other. The "other" class includes cars and algorithm errors like reflections or small movements of the objects in the scene. In order to classify objects, we have modeled the size's characteristics of the bounding box of the detected object, as well as its temporal continuity. Objects detected during $k$ consecutive frames have been associated by proximity. During the $k$ images, the object is classified separately, and later, a voting process is made to give its final class.

As we measure the performance of algorithms, our interest lies in knowing the number of false positives, or index of false alarms, and the number of false negatives, or index of losses. To know the index of losses, it is necessary to monitor all the experiments. However, our objective is to define one evaluation method that does not need human monitoring, i.e., an automatic evaluation. Therefore, we only measure false positives, by storing the images of all objects detected by the algorithm. In this way, at the end of the experiment, it is possible to determine the total number of objects and the number of false alarms for each type of object. Formally, we define the index of false alarms, $m_i$, for each type of object $i$ as

$$m_i = \left(1 - \frac{B_i}{A_i}\right), \tag{14}$$

where $A_i$ is the number of objects detected of class $i$, and $B_i$ the number of objects classified correctly. A good classification gives a value of $m_i$ next to 0.

The results obtained after running the algorithm for several hours in real environments are in Table 2. The algorithm has been implemented on a platform PC and runs at 25 frames/s using a Sony EVI-D31 pan-tilt video camera. Using this active camera, we build the panoramic scene model by setting the appropriate pan and tilt values to cover the entire scene.

It is necessary to point out that although there are errors, the majority of them are not critical. For example, although a pedestrian is not classified correctly in the first $k$ frames, in the following ones this can be remedied. Figure 5 shows some of the objects classified as pedestrians.

**Fig. 5** Detected pedestrians.

### 3.2 *The Prior Density*

As it has been explained, the prior density $p(\mathbf{s}_t)$ is used to initialize the tracking process at the first frame and to initialize new objects as they appear in the scene. We define the prior density by expressing in a probabilistic way the foreground regions segmented according to the previously explained panoramic scene model.

The sample vector at time $t$ is given by $\mathbf{s}_t^{i,l}=(\mathbf{x}_t^i,\mathbf{u}_t^i,\mathbf{w}_t^i)$, where $\mathbf{x}_t^i$ is the position, $\mathbf{u}_t^i$ the velocity, and $\mathbf{w}_t^i$ the size of the object. Then, in order to define $p(\mathbf{s}_t)$ it is necessary to define a prior density for the position $p(\mathbf{x})$, the velocity $p(\mathbf{u})$, and the size $p(\mathbf{w})$, components of the object's state vector.

By means of the detection algorithm, pixels are classified into two categories: foreground and background. By grouping foreground pixels into blobs it is possible to define the prior density of the position component, $\mathbf{x}$, by means of a mixture-of-Gaussians density

$$p(\mathbf{x})=\sum_{j=1}^{B} P(j)p(\mathbf{x}|j), \tag{15}$$

where $B$ is the number of blobs detected [so $P(j)=1/B$] and $p(\mathbf{x}|j)=\eta(\mathbf{b}_j,\Sigma_B)$. Here $\mathbf{b}_j$ is the blob mean position, and $\Sigma_B$ is a constant for all blobs, which is fixed *a priori*. Similarly, the size component $\mathbf{w}$ is formulated in terms of the blob's size. This prior density formulation was used in Ref. 19 as an *importance function* in the combination of low-level and high-level approaches for hand tracking.

In connection with the velocity component, there are three different initialization possibilities. The first possibility is to set the velocity to zero, but then the problem is that the object usually appears in the scene in motion. In that case, this would cause an important difference between the velocity estimation and the real velocity; hence the algorithm would take time in becoming stabilized. The consequence is that the appearance model could be corrupted and the tracking algorithm would not work correctly. The second possibility is, by using an optical flow estimation algorithm, to make an initial estimation of the object's velocity by considering the mean displacement of the blob's pixels. The problem of this approach is that optical flow computation would slow down the whole process. In addition, in the case of the object's occlusion, this estimation would be wrong. The third possibility is to assume the object's temporal continuity, that is, to detect the object in several consecutive frames to ensure that it is not a detection error. In these consecutive frames, detections are associated using only distances between the position components. Once the object is considered as stable, it is possible to make the initial estimation of its velocity.

All these possibilities have been tested, and the last gave the best results. Temporal continuity ensures a good initialization for the object. The problem is that the object takes more time to start being tracked. The main conclusion is that one frame is not enough to initialize correctly the objects to track. Hence, it is necessary to make a more accurate initialization, to detect the object in several consecutive frames to ensure the algorithm's correct performance.

**Fig. 6** In Group objects.

## 4 Performance Evaluation

Nowadays, the evaluation of visual tracking algorithms still constitutes an open problem. This fact was shown at the International Workshops of Performance Evaluation of Tracking and Surveillance (PETS).[20] Two main objectives of PETS are the development of a common evaluation framework and the establishment of a reference set of sequences to allow true comparison between different approaches. From this reference set, a 20-s test sequence has been selected, where four different pedestrians appear. We use this sequence to present the results of our proposed performance evaluation procedure and to show the results of the iTrack algorithm using the target detection algorithm described in the previous section.

In essence, metrics that allow evaluating the performance of a visual tracking algorithm require generating the ground-truth trajectory for each object. However, it is very hard to obtain ground-truth data in video surveillance applications. Another possibility is defining a set of heuristic measures that can be related to the visual tracking application.[21] These heuristic measures are divided into two groups: cardinality and event-based measures. Cardinality measures are used to check whether the number of tracked objects corresponds with the true number of objects within the scene. In this way, it does not assess the maintenance over time of the identification of each object. For example, if the numbers of wrong appearances and disappearances were equal at a particular time instant, the tracker would misbehave without detection by cardinality measures.

Event-based measures allow checking for the consistency of the identification of tracked objects over time. Thus, the continuity of a set of predefined events is annotated. Which events are chosen depends on the application scenario, but they should be generic enough to be easily obtained. An event consists of a label $e$ and the time instant $t$ at which the event has been observed. These events will be the ground-truth data used to compute performance measures, and the basis for comparison with the tracking system results.

It is expected that labels for detected events will usually coincide over time, but not the exact time instant at which they will occur. The following events are considered: *Enter, Exit, Occluded, In Group*, and *Reenter*. A *group* is defined as a set of objects that appear in the image; it may be just one object (see Fig. 6). All events and their causes are described in Table 3.

To report these events, the iTrack algorithm does not require a new definition; it only uses the prior density and the likelihood values as follows:

- *Enter:* The prior density detects the object's first occurrence, and it is maintained during $k$ consecutive

**Table 3** Considered events and their causes.

| Event | Cause |
|---|---|
| Enter | The object appears in the image |
| Exit | The object disappears from the image |
| Occluded | The scene occludes the object |
| In Group | Another object occludes the object |
| Reenter | End of occlusion or rejoining of an object that was tracked singly |

frames to properly initialize the object's appearance model and its velocity.

- *Exit:* The position components of a sample are propagated out of the image limits, and its weight is set to 0. Therefore, this sample does not appear into the posterior density.
- *Occluded:* An occlusion occurs; then the likelihood value decreases significantly, and that can be detected by the system.
- *In Group:* The numerical behavior is identical to that in the Occluded event. The difference is that it is possible to maintain the object's location by tracking the object that occludes it.
- *Reenter:* For short occlusions it is possible to easily identify this event, because several samples survive the occlusion. For large occlusions, it is necessary to use the appearance model to recognize the object again and to distinguish this event from Enter.

To complete the description, we add another event: *Tracked*. Usually, this event is not included in the event table. Alternatively, it is assumed that the Tracked event starts after the Enter or Reenter events and ends when a new event from Table 3 occurs. Finally, to prevent model degeneracy, the appearance model is not updated during the Occluded and In Group events.

Next, we define two event-based measures that can be used for the evaluation of visual tracking algorithms in complex image sequences, where many agents can appear. The main idea is to build up a table of events versus time, which is compared with the table of results obtained by the visual tracking algorithm. The first measure, $C_n$, is based only on the coincidence of observed events. The second measure, $C_l$, is based on computing similarities in the reported object's labels. Both measures reflect the percentage of images where a correct correspondence of events exists. Thus, the tracking can be properly evaluated because the trajectory of each detected object is embedded between events. According to the first measure, $C_n$, one object in a particular event has a valid correspondence when the vision system has also found the same event. In the second measure, $C_l$, a valid correspondence is found only when the label also coincides. In order to verify this second measure, the object is manually labeled $L_i$ at its initial detection, where the index $i$ refers to the object's label. As a result, in the cases that the tracker loses one object during several frames and recovers it afterwards, but fails to identify the recovered object, its corresponding label will be changed and the $C_l$ measurement will be wrong.

**Table 4** Annotated and algorithm-computed events for the test sequence.

| $t$ | Event | $\mathbf{W}^4$ | iTrack |
|---|---|---|---|
| 105 | Enter, $O_1$ | Enter, $L_1=O_1$ | Enter, $L_1=O_1$ |
| 115 | Enter, $O_2$ | Enter, $L_2=O_2$ | Enter, $L_2=O_2$ |
| 188 | Enter, $O_3$ | Enter, $L_3=O_3$ | Enter, $L_3=O_3$ |
| 208 | | Exit, $L_2$ | |
| 239 | | | Exit, $L_2$ |
| 244 | | $L_3=O_2$ | |
| 244 | | Enter, $L_4=O_3$ | |
| 279 | | Exit, $L_1$ | |
| 296 | | Enter, $L_5=O_1$ | |
| 322 | Enter, $O_4$ | Enter, $L_6=O_4$ | Enter, $L_4=O_4$ |
| 410 | Exit, $O_1$ | Exit, $L_5$ | Exit, $L_1$ |
| 416 | | In Group, $(L_3,L_4)$ | |
| 445 | Exit, $O_2$ | Exit, $(L_3, L_4)$ | |
| 450 | Exit, $O_3$ | | Exit, $L_3$ |

Both measures have been used to compare the iTrack algorithm with the $\mathbf{W}^4$ algorithm.[5] In $\mathbf{W}^4$, Collins et al. also use a scene model and an estimation procedure for object tracking. However, $\mathbf{W}^4$ has a previous process of data association in order to compute the correspondence between each object and its estimation filter. Both the annotated and computed events for the test sequence of the PETS database are shown in Table 4. Also, the system's results for several frames of this sequence are shown in Fig. 7.

Interpretation can be carried out by applying the defined measures, $C_n$ and $C_l$, to the computed events of Table 4. Next, we detail the procedure for the first object that appears in the scene, object $O_1$. The $\mathbf{W}^4$ algorithm found a correct correspondence for both measures from $t=105$ to $t=279$, i.e., until the algorithm loses the object. This object appears again at $t=296$, and it is tracked until $t=410$ with a different label, $L_5$. Therefore, $C_n=174+114=288$ and $C_l=174$ for the $\mathbf{W}^4$ algorithm. On the other hand, the iTrack algorithm does not lose this object; therefore $C_n=305$ and $C_l=305$. The results for all objects within the scene are shown in Table 5.

It should be noted that the image sequence is noisy and complex due to the similarity in appearance of both the agents and the background. It can be seen in Table 5 that $\mathbf{W}^4$ is an algorithm that relies on the detection task rather than tracking. As a result, it obtains good measurements for $C_n$, but fails to identify objects correctly and therefore gives poor results in the $C_l$ measurements. Errors reflected in $C_l$

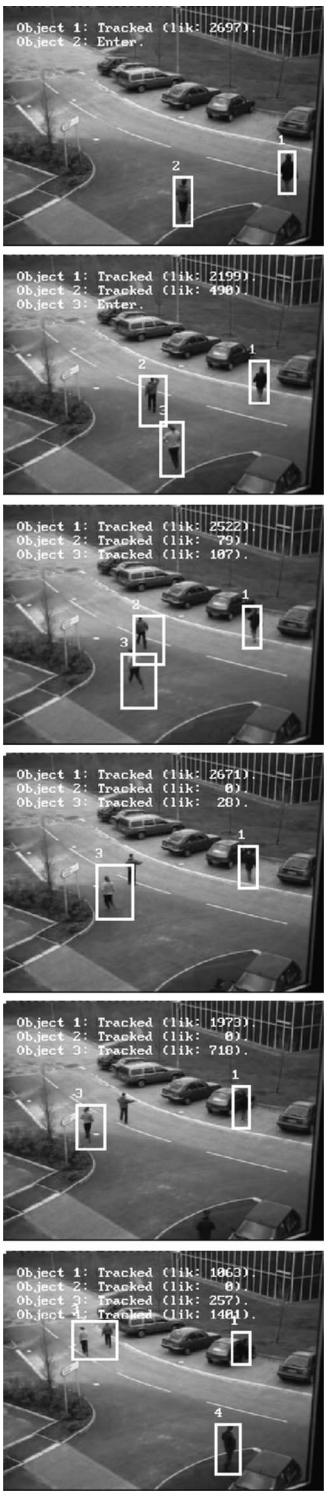



**Fig. 7** Visual tracking results of the iTrack algorithm for the test sequence.

are mainly due to shadows [see Fig. 8(a)] and to the proximity between two objects in the image due to the scene perspective [see Fig. 8(b)].

Best results on $C_l$ are obtained using the iTrack algorithm, since it is more robust because of the use of particle filters and adaptive appearance models. The main drawback of our approach is that the temporal continuity of the prior density can cause a too noisy object not to reappear. For example, this fact occurs for object 2 in the test sequence. Another important problem that remains open is that if the objects' appearance model is not correctly initialized, or it suddenly changes, the object can be lost. Anyway, the results in Table 5 show that in real environments, and for normal conditions, the algorithm's performance is good enough.

**Table 5** Performance measurements for the events of Table 4.

| Object | Number of frames | W$^4$ | | iTrack | |
|---|---|---|---|---|---|
| | | $C_n$ | $C_l$ | $C_n$ | $C_l$ |
| $O_1$ | 305 | 288 | 174 | 305 | 305 |
| $O_2$ | 330 | 265 | 093 | 124 | 124 |
| $O_3$ | 277 | 228 | 056 | 277 | 277 |
| $O_4$ | 178 | 178 | 178 | 178 | 178 |
| Total | 1090 | 959 | 501 | 884 | 884 |
| | 100% | 88% | 46% | 81% | 81% |

## 5 Discussion and Conclusion

From the results obtained in Sec. 4, it can be stated that wrong detections cause an important decrease in the performance of the tracking task. It is important to provide accurate results in the detection task to ensure the correct performance of the overall video surveillance system. Indeed, another possibility is the integration of detection and tracking. For example, in Ref. 22 is presented an online selection of discriminative features to improve the tracking task by maximizing the contrast between the object and its surroundings. However, multiple-object tracking is not considered. Other approaches propagate detections over time[23] or use detection algorithms on the likelihood function.[24] This fact can be easily included when using particle filters by introducing new particles from a prior density based on the detection results at each time instant in a similar fashion to our algorithm iTrack.

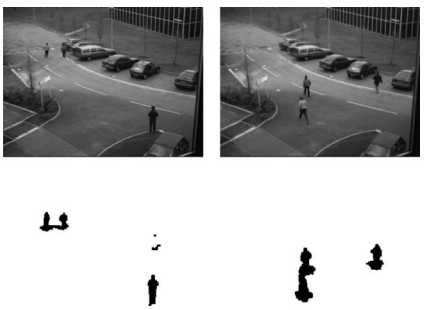


**Fig. 8** Wrong detections. Left: due to shadows. Right: due to the scene perspective.

In this regard, we consider that the most important contribution of our algorithm is the use of an adaptive appearance model that is automatically updated to improve the tracking. Thus, the algorithm can be used in different application environments without significant changes. Nevertheless, if more visual cues are considered, the algorithm can be made more robust.[25] For example, color is the most currently used cue in image-based tracking algorithms.[26]

Another contribution of this work is the definition of two performance measures for the evaluation of a video surveillance system. These measures are computed automatically by the system, and they only require an easy manual annotation to describe the real events of the sequence. We have used these measures to compute the performance of our tracking and object detection algorithms in a complex outdoor scene. Our future work will include the evaluation of our tracking algorithm for indoor scenes using the CAVIAR database.[27] But we consider that our system's evaluation is already sufficient for the paper's main objective, that is, to emphasize the importance of detection in video surveillance applications.

## Acknowledgments

This work is supported by EC grants IST-027110 for the HERMES project and IST-045547 for the VIDI video project, and by the Spanish MEC under projects TIN2006-14606, TIN2007-67896, and CONSOLIDER-INGENIO 2010 (CSD2007-00018). Jordi Gonzàlez and Javier Varona also acknowledge the support of a Juan de la Cierva and a Ramon y Cajal (cofunded by the European Social Fund) Postdoctoral Fellowship from the Spanish MEC, respectively.

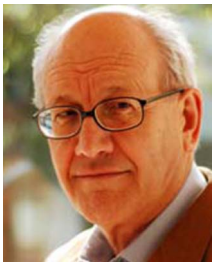
## References

1. A. Lipton, H. Fujiyoshi, and R. Patil, "Moving target classification and tracking from real-time video," in *IEEE Workshop on Applications of Computer Vision (WACV'98)*, pp. 8–14 (1998).
2. T. Horprasert, D. Harwood, and L. Davis, in *4th Asian Conf. on Computer Vision*, Vol. **1**, pp. 983–988 (2000).
3. P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *Int. J. Comput. Vis.* **63**(2), 153–161 (2005).
4. R. Collins, A. Lipton, and T. Kanade, in *ANS 8th Int. Topical Mtg. on Robotics and Remote Systems*, pp. 1–15 (1999).
5. I. Haritaoglu, D. Harwood, and L. Davis, *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 809 (2000).
6. M. Isard and A. Blake, *Int. J. Comput. Vis.* **29**, 5 (1998).
7. Y.-C. Ho, *IEEE Trans. Autom. Control* **9**, 333 (1964).
8. M. Isard and J. MacCormick, in *Proc. Int. Conf. on Computer Vision (ICCV'2001)* (2001).
9. H. Sidenbladh, M. Black, and D. Fleet, in *Proc. Eur. Conf. on Computer Vision (ECCV'2000)* (2000).
10. H. Tao, H. Sawhney, and R. Kumar, *IEEE Trans. Pattern Anal. Mach. Intell.* **24**, 75 (2002).
11. H. Tao, H. Sawhney, and R. Kumar, in *Proc. Vision Algorithms 99, a Workshop Associated with the Int. Conf. on Computer Vision, ICCV'99* (1999).
12. D. Rowe, I. Rius, J. Gonzàlez, and J. Villanueva, in *3rd Int. Conf. on Advances in Pattern Recognition* (Springer), Vol. **2**, pp. 384–393 (2005).
13. C. Stauffer and W. Grimson, *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 747 (2000).
14. Y. Ye, J. K. Tsotsos, E. Harley, and K. Bennet, *Mach. Vision Appl.* **12**, 32 (2000).
15. T. Wada and T. Matsuyama, in *Proc. 13th Int. Conf. on Pattern Recognition* (1996).
16. M. Nicolescu, G. Medioni, and M.-S. Lee, in *Proc. IEEE Workshop on Omnidirectional Vision*, pp. 169–174 (2000).
17. R. Horaud, D. Knossow, and M. Michaelis, *Mach. Vision Appl.* **16**, 331 (2006).
18. Y. Ren, C.-S. Chua, and Y.-K. Ho, *Mach. Vision Appl.* **13**, 332 (2003).
19. M. Isard and A. Blake, in *Proc. Eur. Conf. Computer Vision (ECCV'98)*, pp. 893–908 (1998).
20. *International Workshops of Performance Evaluation of Tracking and Surveillance (PETS)*, http://peipa.essex.ac.uk/ipa/pix/pets/ (2006).
21. S. Pingali and J. Segen, in *Proc. 3rd IEEE Workshop on Applications of Computer Vision*, pp. 33–38 (1996).
22. R. Collins, Y. Liu, and M. Leordeanu, *IEEE Trans. Pattern Anal. Mach. Intell.* **27**, 1631 (2005).
23. R. Verma, C. Schmid, and K. Mikolajczyk, *IEEE Trans. Pattern Anal. Mach. Intell.* **25**, 1215 (2003).
24. M. Han, W. Xu, H. Tao, and Y. Gong, in *Proc. Computer Vision and Pattern Recognition (CVPR'04)*, Vol. **I**, pp. 874–861 (2004).
25. P. Pérez, J. Vermaak, and A. Blake, *Proc. IEEE* **92**, 495 (2004).
26. K. Nummiaro, E. Koller-Meier, and L. Van-Gool, *Image Vis. Comput.* **21**, 99 (2003).
27. R. Fisher, in *Proc. Sixth IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS'04)*, pp. 1–5 (2004).

**Javier Varona** is a Ramon y Cajal Research Fellow of the Spanish Government at the Universitat de les Illes Balears (UIB). He received his doctoral degree in Computer Engineering from the Universitat Autònoma de Barcelona (UAB) and the Computer Vision Center (CVC) in 2001. His PhD thesis was on robust visual tracking. His research interests include visual tracking and human motion analysis. Currently, he is the leader of a research project for building natural interfaces based on computer vision founded by the Spanish Government (TIN2007-67896). He is member of the ACM.

**Jordi Gonzàlez** obtained his PhD degree from the UAB in 2004. At present he is a Juan de la Cierva postdoctoral researcher at the Institut de Robòtica i Informàtica Industrial (UPC-CSIC). The topic of his research is the cognitive evaluation of human behaviors in image sequences. The aim is to generate both linguistic descriptions and virtual environments that explain those observed behaviors. He has also participated as a WP leader in the European projects HERMES and VIDI-Video, and as a member in the euCognition network. He cofounded the Image Sequence Evaluation research group at the CVC in Barcelona.

**Ignasi Rius** obtained his BSc(Hons) in Computer Science Engineering in 2003 from the Universitat Autònoma de Barcelona (UAB). In 2005 he received his MSc, and he is now taking his PhD studies at the Computer Vision Center (CVC) from UAB in Barcelona. His work is focused on the modeling and analysis of human motion for visual tracking and recognition applications. He is an active member of the Image Sequence Evaluation (ISE) research group at the CVC.



**Juan José Villanueva** received a BSc in physics from the University of Barcelona in 1973 and a PhD degree in computer science from the UAB in 1981. Since 1990, he has been a full professor in the Computer Science Department at the UAB. He promoted the Computer Vision Center (CVC) and has been its director since its foundation. He was a cofounder of the Image Sequence Evaluation (ISE) research group at CVC. His research interests are focused on computer vision, in particular human sequence evaluation.