Fast detecting and tracking of moving objects in video scenes

Françoise Dibos, Georges Koepfler, Sylvain Pelletier

Fast detecting and tracking of moving objects in video scenes

F. Dibos[1], G. Koepfler[2] and S. Pelletier[2]

[1]Institut Galillée, LAGA et L2TI,
Avenue J.B Clément, 93430 Villetaneuse, France
dibos@math.univ-paris13.fr

[2]MAP5, UMR CNRS 8145,
45 rue des Saints-Pères, 75270 PARIS cedex 06, France
{georges.koepfler,sylvain.pelletier}@math-info.univ-paris5.fr

December 11, 2006

Abstract

In this article we present a new method for detecting textured moving objects. Based on a known background estimation and a fixed camera, the algorithm is able to detect moving objects and locates them at video rate, moreover this method is used for object tracking purposes.

Our method is multi-step: First, we use level lines to detect pixels of the background which are occluded by moving object. Then, we use an *a contrario* model as general framework to make an automatic clustering. Thus the moving objects are detected as regions and not only as pixels, eventually we correct this region to better fit the moving object.

Experimental results show that the algorithm is very robust to noise and to the quality of the background estimation (*e.g.* ghosts). The algorithm has been successfully tested in video sequences coming from different databases, including indoor and outdoor sequences.

1 Introduction

Finding moving objects in a video sequence obtained with a fixed camera is a classical problem, it is often the first task in order to extract high level information from video scenes, see [16, 12].

Video segmentation remains a challenging problem as there are usually a lot of constraints. In the case of video surveillance an important issue is **real-time computation**. Moreover, detection should be **on-line** in order to get immediate high level information, *e.g.* events detection [12]. As one cannot always suppose the existence of a moving object in the scene, the first task is to **detect moving objects**, for example shadows and highlights should not be considered as moving objects.

Also, for easy manipulation, the algorithm should need only a **few parameters**, easy to estimate. Finally, if a background estimation is used, the method should be **robust** with respect to this estimation, *e.g.* if there are moving objects in the background estimation, the algorithm should not detect these spots as moving objects. The method we will propose in the sequel will answer all of these questions.

Let us give a brief review of methods which have been developed to address the video segmentation problem, we will consider two groups: real-time methods and time consuming methods. Due to the different constraints, results obtained by these methods are of course quite different.

Most real time methods are based on background subtraction techniques [25], these methods infer the moving object by subtracting/comparing the input image from/to a background image, which is initially estimated and which might be updated. Another successful real time technique is the mixture of Gaussians method, see [21, 26]. Each pixel is modeled separately by a mixture of K Gaussians, having different weight, mean and variance. New pixel values are checked against the K existing Gaussians distributions and the parameters are updated with regard to the new values. Then the background is detected as the component in the mixture having the highest weight, and lowest variance.

These methods can be implemented to be real-time and on-line. As pointed out in [1], real time methods are usually pixel-wise independent, they attempt to classify each pixel as either foreground or background. Exceptions are the method of [1], where the rigid structure of the object is used to get a template, or the algorithm in [3], where the output is made of pixel blocks.

Time consuming methods are often region based, *e.g.* in snake models the output is a region, obtained from an initial region which has been adapted to the moving object. For example in [15] a method based on a snake model and comparison with the background is proposed, in [24] optical flow techniques are used. Other interesting references include [6, 7].

The method proposed in this paper starts by answering the following question: Is there a moving object in the current frame? The detection of moving objects gives us an approximate location, to be more precise, our multi-step algorithm ca be summarized as follows: First, given the background image, we compare independently each frame to it, this is based on pixel information and the the topographic maps of the background and the current frame. Next we use this pixel information to get region information by applying an automatic clustering, based on the *a contrario* detection principle. For some applications, we might add a last step in order to refine the clusters and obtain an almost exact segmentation. Another possibility is tracking objects. Notice that the output of our method are well defined regions, still it remains real-time.

Our method is based on the work of Lisani and Morel, [17]: to measure the difference between two images they compare gradient directions and use the general *a contrario* detection principle [9] to find major local changes. An approach close to our method is the work of Veit, Cao and Bouthémy [27].

The plan of the paper follows the steps of our algorithm: Section 2 shows how to obtain the *difference image* D which represents the pixel information obtained by comparing the topographic map of the current image and of the background; Section 3 introduces the *a contrario* detection principle to find moving objects, thus getting regional information. Section 4 presents experimental results and discusses the advantages of multi step methods.

2 Computing the difference image

In this section we will develop our method to obtain a robust difference image D, containing the clusters which will be candidates for the perceptual groupings introduced in the next section. First we discuss how we model changes between the background and the current frame, we then present a symmetric and a non-symmetric way of computing D. In the last paragraphs of this section we analyze the theoretical behavior of our method.

Estimation of the background.

In our method we suppose given the reference background image B. It might be obtained by just taking an image without moving objects. Notice that we choose to not treat the problem of background estimation, this is usually done by tacking into account the segmentation of moving objects and the type of video sequence treated (*e.g.* indoor or outdoor). Examples of background estimation techniques can be found in [4, 25].

Although less reliable than mixture of Gaussians [26], an image without moving objects gives usually a correct background for our robust algorithm.

If the background contains an object which disappears during the sequence this might be detected as a "ghost". The problem of ghost elimination is addressed below.

The occlusion model.

The difference image is the result of the local comparison of the background B and the current image, which we will denote I: we want to detect pixels of the background which are occluded by moving objects, and then label pixels into foreground or background.

We model a video scene I by using three layers: the background B, the moving object $\mathcal{O}$ and the noise n. Our model is not additive: the object $\mathcal{O}$ occludes the background B. Thus we have to detect if we see moving objects *instead of* seeing background. To detect occlusions, we use the same criterion as proposed in [18] for in-painting: a moving object "breaks" the local structure of the background scene by occlusion. But local structure is given by sufficiently contrasted level lines [19], we thus will use the level set representation of an image.

In classical real time methods, the video scene model is the addition of the layers: $I(x) = B(x) + \mathcal{O}(x) + n(x)$ for the foreground and $I(x) = B(x) + n(x)$ for the background. Thus, in the case of background subtraction, the detection is based on a threshold on $|I(x) - B(x)|$. In the case of mixture of Gaussians, the detection depends on the absolute difference of $I(x)$ with the mean (gray level) of each Gaussian.

Level sets and local structure of an image.

Let Ω be the image domain, and suppose $I : \Omega \to [0, 255]$, let $\lambda \in [0, 255]$, then the lower level set of λ is:

$$\chi_\lambda = \{y \in \Omega \mid I(y) \leq \lambda\},$$

The level lines are the boundaries of the sets χ_λ, and they give the topographic map $(\partial\chi_\lambda)_{\lambda\in[0,255]}$ which is not affected by a contrast change.

Now the local structure of the image at pixel x is given by the unique level line passing through x, which exists if x is not in a flat zone. In flat zones there is no real local structure as there are no edges. Thus we classify the pixels into those with local structure, i.e. with large gradient norm, and those without local structure, which are in flat zones. In practice this classification is obtained by using a threshold on the gradient norm of the image.

As the gradient is normal to the level line, we decide that the local structure changes if the direction of the gradient changes, see [17]. Notice that this information is independent of contrast change and only available for pixels with a large enough gradient norm.

Local structure changes.

By computing the normal to the level lines, *i.e.* the gradient, of both, the background B and the current image I, we are able to compare local structure. Now, at a pixel x, structure

appears	for a high gradient in I, but a low gradient in B;
disappears	for a low gradient in I, but a high gradient in B;
changes	for a high gradient in both, I and B, and with a change of the gradient direction;
does not change	for a high gradient in I and B, but with no change of gradient direction;
does not exist	for low gradients in both, the image and the background.

We thus compute the gradient of B and I at x, written $\nabla B(x)$ and $\nabla I(x)$, and we deduce:

$$n_I(x) = |\nabla I|(x), \;\; n_B(x) = |\nabla B|(x) \;\text{ and }\; \text{angle}(x), \;\; \text{the angle between } \nabla I(x) \text{ and } \nabla B(x)\,.$$

We use a first threshold e, which is the minimal gradient norm for the level line to be relevant: if $n_I(x)$ is larger than e there is local structure at x.

Using this threshold, the case "structure appears" could be formalized by the following condition : $[n_I(x) \geq e$ and $n_B(x) < e]$. But this test is not contrast invariant, indeed suppose that $n_B(x)$ is smaller than e, but very close to it, then even a very small change of contrast in the sequence could make $n_I(x) \geq e$ and thus yield structure. Thus we introduce a second threshold E, with $e < E$, which measures the minimal contrast of an object, and we detect new structure only if: $[n_I(x) \geq E$ and $n_B(x) < e]$.

The second case, "structure disappears" is symmetric from the previous one, it correspond to: $[n_B(x) \geq E$ and $n_I(x) < e]$.

Now, if there is structure at pixel x in the background and in the image, a change of structure will be detected, like in [17], by using the direction of the gradient. So we introduce a third threshold ϕ, which, applied to angle, decides whether there occurs a change or not. So the case "structure changes" corresponds to the test $[n_I(x) \geq e$ and $n_B(x) \geq e$ and $\text{angle}(x) > \phi]$, and the case "structure does not change" to pixels where: $[n_I(x) \geq e$ and $n_B(x) \geq e$ and $\text{angle}(x) < \phi]$ is true.

Finally, pixels where no information is available are those verifying $[n_I(x) < E$ and $n_B(x) < e]$ or $[n_B(x) < E$ and $n_I(x) < e]$.

Symmetric difference.

A first way to build the difference image D is to label as foreground those pixels where structure appears, disappears, or changes. Pixels with no structure changes are in the background. Notice that, unlike other models, we do not try to classify all pixels: if the detection cannot be made precisely, we say there is "no information". So we build a difference image D, made of three labels, as follows:

- we put $D(x) =$ **white**, if the pixel is labeled as foreground, that is if

$$[n_I(x) \geq e \text{ and } n_B(x) \geq e \text{ and } \text{angle}(x) > \phi]$$

$$\text{or } [n_I(x) \geq E \text{ and } n_B(x) < e] \text{ or } [n_B(x) \geq E \text{ and } n_I(x) < e]\ ;$$

- we put $D(x) =$ **black**, if the pixel is labeled as background, that is if

$$[n_I(x) \geq e \text{ and } n_B(x) \geq e \text{ and } \text{angle}(x) < \phi]\ ;$$

- we put $D(x) =$ **gray**, if we cannot compare the two images, that is if

$$[n_I(x) < E \text{ and } n_B(x) < e] \text{ or } [n_B(x) < E \text{ and } n_I(x) < e]\ .$$

As shown in Figure 1, bottom left, there are a lot of **gray** pixels in a difference image, but one can notice also that **black** and **white** pixels are placed as well in the background as in the foreground. Due to noise, there exist wrong labeled pixels, but wrong labeled **white** pixels are usually close to **black** ones and do not form clusters. We notice only few wrong **black** pixels. Moreover, a foreground object will yield mainly **gray** pixels in its interior but only very few **black** ones.

Non symmetric difference, robustness to ghosts.

The way to build the difference image presented above is symmetric as we treat the image and the background in the same way. In particular, we label a pixel as foreground if structure appears or disappears at its location. In the case of a perfect background this should be the best way to do. In practical applications there might be traces of moving objects in the background estimation and these spots might get labeled as foreground. This problem is known as the **ghost elimination** problem.

To avoid this wrong detection, we have to change the way we compute the difference image. If structure disappears from the background and is not replaced by another, different, structure, there might be a ghost. Thus, the case "structure disappears" $[n_B(x) \geq E$ and $n_I(x) < e]$ will be labeled as no information, i.e. **gray**, instead of **white**.

Thus we loose information at pixels where a uniform object occludes a non-uniform part of the background. In the case of video surveillance application, with objects which are well textured and not uniform, this loss of information is small. In Figure 2 we show the effect of the non-symmetric difference, when an object is present in the background.

Robustness of the difference image.

In this paragraph, we will show on a data model how our computation of D behaves, this provides evidence why our difference image is a reasonable way to compare images and it explains the results we obtain on real data.

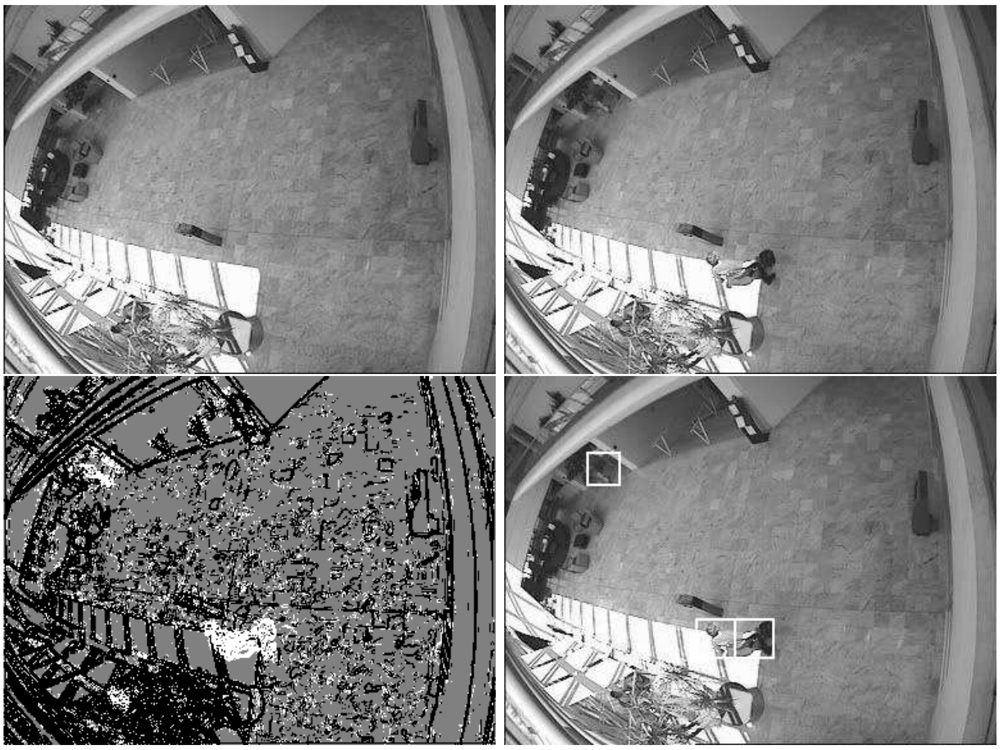
FIGURE 1: Top left, the reference background B, on its left the image I, bottom left the difference image and, bottom right, the maximal meaningful windows.

Suppose a partition of the image domain $\Omega = \bigcup_i \overline{\Omega_i}$, with the Ω_i disjoint open sets and consider the case of a local affine contrast change modeled by:

$$\forall \Omega_i, \exists \alpha_i > 0, \exists \beta_i > 0, \forall x \in \Omega_i \; : \; I(x) = \alpha_i B(x) + \beta_i \tag{1}$$

In this case the gradients of I and B have the same direction, so the difference image will only indicate **white** if there is a change of the gradient norm. We have

Proposition 1 *If I verifies (1) and if $\frac{e}{E} \leq \alpha_i \leq \frac{E}{e}$, then for each x in Ω_i, $D(x) =$* **black** *or* **gray**. *In the case of non-symmetric difference computation, the result is true if $\alpha_i \leq \frac{E}{e}$.*

This result gives us an interpretation of the two parameters e and E: local contrast changes between $\frac{e}{E}$ and $\frac{E}{e}$ are not detected in the (symmetric) difference image.

A model of cast shadows.

To add an artificial shadow or highlight to an original image I_0, like in Figure 3, we compute a new image

$$I_s(x) = (1 - m(x))I_0(x) + m(x)(\alpha I_0 + \beta),$$

with $m(x)$, a smooth function equal to 1 on the shadow and 0 far away. Thus we model

- first, that a cast shadow has not really edges, indeed the transition between the shadow and the original image is smooth, this fact has been used in [14];
- second, that the effect of the shadow is just an affine contrast change.

To create a shadow, the parameter α has to be smaller than 1 and β has to be negative, in Figure 3 we took $\alpha = 0.85$ and $\beta = -50$. For highlights, we have of course the opposite, *e.g.* $\alpha = 1.15$ and $\beta = +50$.

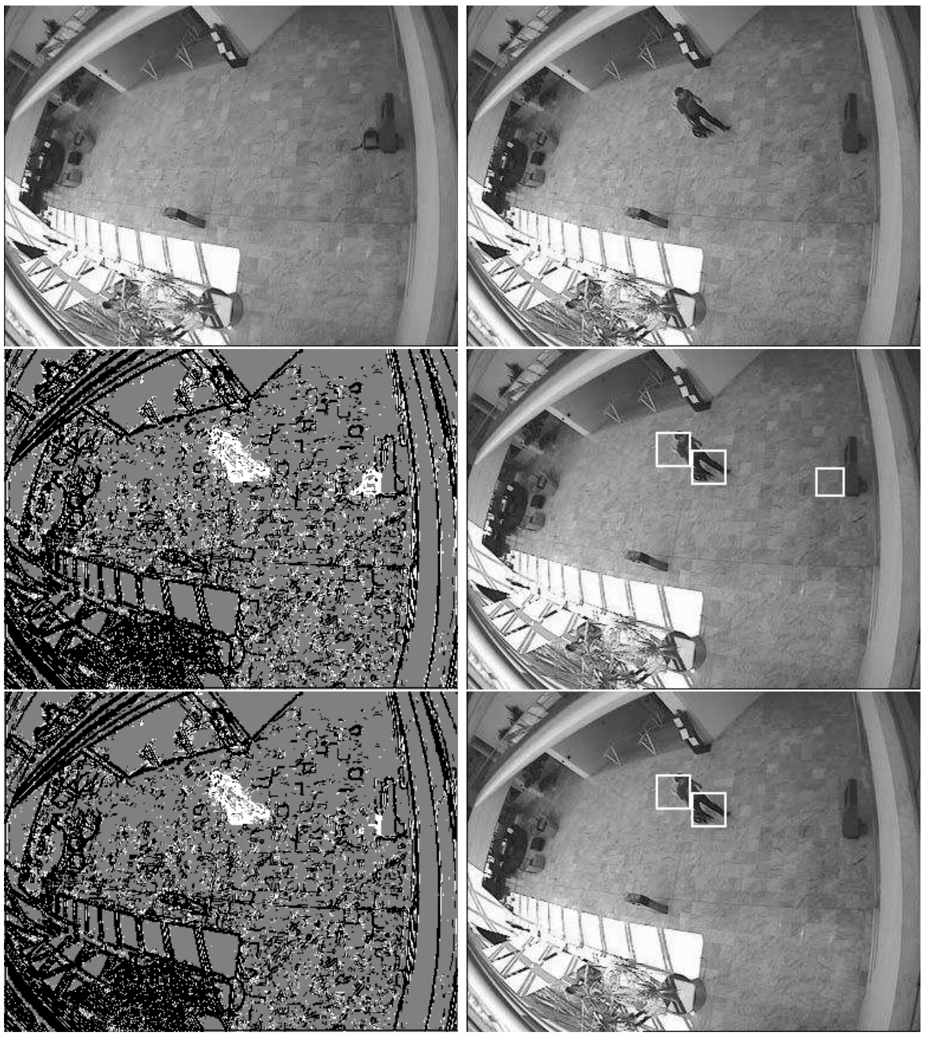

FIGURE 2: Top row: on the left a background with a forgotten object (bag) and on the right an image of the sequence where the bag has been taken away; middle row: the difference image and the segmentation in the symmetric case: the ghost is detected as moving object; bottom row: the difference image and the segmentation in the non symmetric case, the ghost is no longer detected.

As shown by Proposition 1, these shadows or highlights are not detected in the difference image. In Figure 3, a shadow and a highlight are added to image I_0, and the new image I_s is compared to I_0. The corresponding difference image has only **white** pixels on the boundary of the shadow.

3 Finding perceptual groupings in the difference image

In the previous section we explained how to obtain a difference image D which will contain the location of pixels where we are able to say that something has changed between the background and the current frame. Now we develop how the clusters thus obtained are grouped into regions where perceptual changes take place.

Let us start with an experiment: we compare an image without moving objects to the background image, or to another image of the sequence with no moving object either. The resulting difference image has a lot of **white** pixels spread all over, these pixels are due to additive noise with large amplitude, creating distortion in the gradient evaluation as there is no smoothing.

In a difference image obtained by comparing an image with a moving object to the background, we find also **white** pixels almost uniformly distributed over the image, but moreover, we detect clusters of **white**

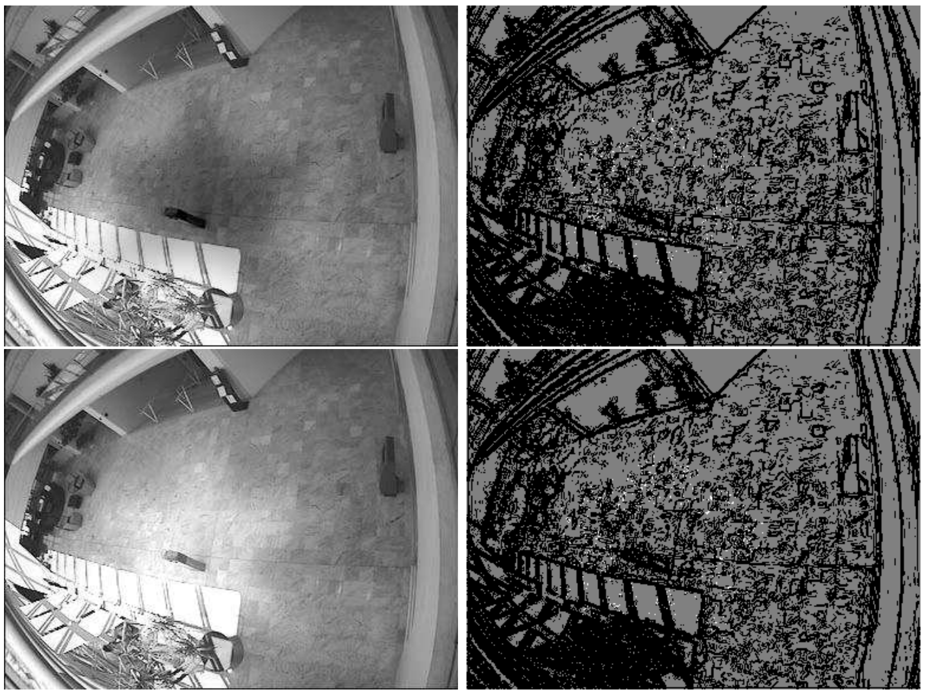

FIGURE 3: Top left, an image with a shadow created as explained in Section 2; bottom left, using the same model a highlight has been added. The right column shows, with the corresponding symmetric difference images, that the comparison is not affected by shadows and highlights.

pixels at the location of the moving objects. This experiment is shown in Figure 4.

We base our detection on the following observation: **white** pixels due to moving objects form clusters whereas those due to additive noise are almost uniformly distributed over the image. Thus we have now to automatically detect and locate clusters of **white** pixels, this will be achieved by using the *a contrario* model developed below.

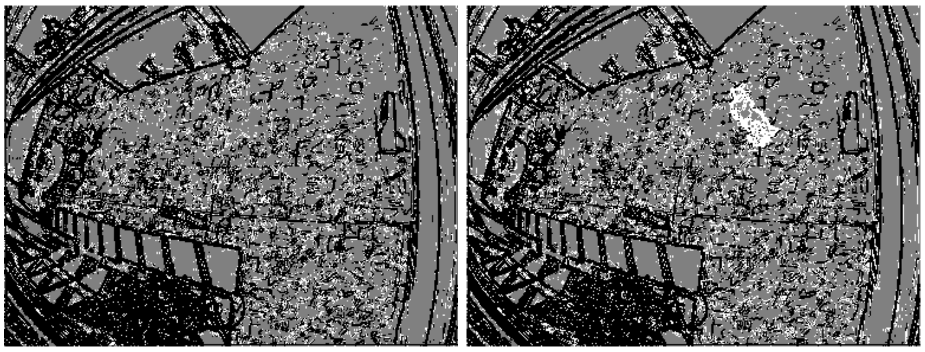

FIGURE 4: A comparison between a difference image without moving objects, on the left, and with one moving object, on the right, shows the same kind of noise and only clusters of **white** pixels at the moving object's location.

Principle of the *a contrario* detection.

The principle of *a contrario* detection is to first define an *a priori* model for the generic case when there is nothing to detect. Detection will only take place when the number of occurrences of such an event in the *a priori* model is low. This kind of method has proven to be very robust to noise, see [10, 9, 8, 20, 27].

We call "observable pixel" a non **gray** pixel, *i.e.* a pixel for which information is available. The events we want to detect are clusters of **white** pixels by taking into account only the observable pixels, *i.e.* **black** or **white**, in D.

The uniform *a priori* model.

In the uniform *a priori* model, we suppose that observable pixels are uniformly independently distributed: **white** with probability p and **black** with probability $1-p$. We obtain an estimate of p by considering the number of **white** pixels among all the observable pixels of the difference image D.

Now, for each window $W \subset D$, with at least n observable pixels, we compute the probability, in the *a priori* model, for W to contain more than k white pixels. Using the tail of the binomial distribution we get

$$\mathbf{B}(p,n,k) = \sum_{i=k}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) p^i (1-p)^{(n-i)}.$$

The next paragraphs describe how the *a priori* model is used to find perceptual groupings, *i.e.* unlikely clusters of **white** pixels.

Detecting perceptual groupings.

To detect perceptual groupings, we introduce the notion of $\epsilon-$meaningful windows from [9]. We consider N sub-windows of the difference image D which are at various positions and are of different size. For computational reasons explained below, we fix a grid of mesh size K and consider only windows W, which fit to this coarse grid, their dimensions will thus be multiples of K.

Definition 1 *For a window $W \subset D$, containing n observable pixels and k white pixels, we define the* **number of false alarms***:*

$$NFA(W) = \mathbf{B}(p,n,k)N\,.$$

*For $\epsilon \leq 1$, we say that a window W is $\epsilon-$***meaningful** *if $NFA(W) < \epsilon$ and $\dfrac{k}{n} > p$.*

Thus, a window W is $\epsilon-$meaningful if the probability to have k white pixels among n observable pixels is less than the adaptive threshold ϵ/N. An interpretation (see [9]) of this adaptive threshold is given by

Proposition 2 *The mathematical expectation of the number of meaningful windows in the uniform a priori model is less than ϵ.*

In other words, the expectation of the number of detections allowed by the noise model, *i.e.* the number of false alarms, is less than ϵ. The detection depends in fact on $\log(\epsilon)$ and we define the **"meaningfulness"** of W to be $S(W) = -\log(NFA(W))$.
This quantity represents the significance of a window: the higher $S(W)$, the more the window is meaningful and the denser is the cluster.

Now, we introduce a threshold $T = -\log(\epsilon)$ and we declare the **perceptual groupings** to be windows with $S(W) > T$. Thus, we detect a moving object as a perceptual grouping if there is a window W with $S(W) > T$.

Notice that the meaningfulness of a window W depends only on k and n, but instead to base our detection directly on a threshold on these two local parameters, we use the number of errors, *i.e.* false alarms, allowed by the *a priori* model on D. For example by tacking $T = 15$, we allow one false alarm each 10^{15} images in our noise model.

Let us remark that for some applications it is sufficient to take $\epsilon = 1$, see [9], *i.e.* $T = 0$. We use higher values for T in order to consider as moving objects only groupings having "high meaningfulness", indeed, as we are handling video sequences we cannot allow an error in each image of the sequence.

Computing the number of false alarms.

The numerical value of $S(W)$ is computed using the Hoefding approximation:

$$S(W) \approx n \left[p_l \log(\frac{p_l}{p}) + (1-p_l) \log(\frac{1-p_l}{1-p}) \right] - log(N).$$

where p_l is the local probability for an observable pixel to be **white**:

$$p_l = \frac{\#\textbf{white} \text{ pixels in } W}{\#\text{observable pixels in } W} = \frac{k}{n}.$$

The Hoefding approximation provides us with a new interpretation of the condition $S(W) > T$. If we compute the Kullback-Leibler distance (see [5]) between p, the global probability/density of an observable pixel to be **white**, and p_l, the same probability restricted to the window W, then W is meaningful if this distance is greater than the (adaptive) threshold $T + log(N)$.

Maximal meaningful windows.

At this point we are able to detect perceptual groupings in the difference image D. By doing this, we obtain a set of windows, each of which contains clusters of **white** pixels and has meaningfulness $S(W) > T$. But a lot of meaningful windows will be placed around the same object so results are redundant. So the problem is now to avoid intersections of meaningful windows. For this, we introduce the notion of maximal meaningful windows:

Definition 2 *We say that a window W is* **maximal meaningful** *if W is meaningful and if for all meaningful windows W', with $W \cap W' \neq \emptyset$, we have:*

- *$S(W) > S(W')$, thus W fits the object better than does W';*

  or

- *there exists W'', such that $W'' \cap W' \neq \emptyset$ and $S(W'') > S(W') > S(W)$, thus W' corresponds to an object located in W''.*

Notice that if both, W and W' are maximal meaningful windows, then $W \cap W' = \emptyset$.

To extract the list of maximal meaningful windows from the initial list of meaningful windows, we proceed as follows:
− first, we look for the most meaningful window W_{max} in the initial list and add it to the new list of maximal meaningful windows;
− second we erase all the windows W which have non empty intersection with W_{max};
we repeat these two steps until the initial list is empty.
This will yield a set of non intersecting windows having maximal meaningfulness.

These non intersecting regions are the first output of our algorithm: if there exist significant windows, the maximal meaningful windows give the approximate location of the moving objects. In our experiments these windows are drawn on the original image.

Set of test windows and influence on the computation time.

The set of test windows has to be chosen in order to fit the moving objects. For computational issues we consider a grid of mesh-size K, and only use rectangles which fit to this grid, thus their dimensions will be multiples of K.

We first compute the number of **black** and **white** pixels in each $K \times K$ block of the image partition, this block information makes it very easy to compute $S(W)$ for each test window W. Moreover, as test windows of the same dimensions might overlap, we can reduce even more the computational effort.

Once we have computed this block information, the number of operations needed to compute the meaningful windows depends on the number of blocks, *i.e.* the number of pixels divided by K^2. Thus it is the computation of the difference image which takes most of the time. The number of operations needed by our algorithm is thus depending on the determination of the gradient on each frame.

Notice that taking small test windows does not give a better segmentation. Indeed, local noise produces small clusters of **white** pixels in the difference image and if these clusters are large with respect to the size of the test windows they might yield meaningful windows.

Almost exact segmentation.

The maximal meaningful windows, as described above, provide the approximate position of an object. This might be sufficient for a lot of applications, such as motion estimation, but it is interesting to get an almost exact segmentation in order to extract moving objects.

To do so, we correct the approximate segmentation in order to increase meaningfulness. Thus we add/subtract blocks of K by K pixels from the initially obtained maximal meaningful windows, in such a way that the meaningfulness of the resulting windows is increased. This gives a preciser position of the detected object. Let us give some details about the procedure.

We denote $\mathcal{B}$ a generic $K \times K$ block of our grid and start with $\mathcal{O} = \cup W$, the union of all maximal meaningful windows. To the disconnected set of pixels $\mathcal{O}$, we apply a split and merge algorithm as follows:

- we delete all blocks $\mathcal{B} \subset \mathcal{O}$, which are on the boundary of $\mathcal{O}$, and such that $S(\mathcal{O} \setminus \mathcal{B}) > S(\mathcal{O})$,
- we merge all blocks $\mathcal{B} \not\subset \mathcal{O}$, which are on the boundary of $\mathcal{O}$, such that $S(\mathcal{O} \cup \mathcal{B}) > S(\mathcal{O})$, and $\mathcal{B}$ has a higher density of observable pixels than the difference image D,

Object tracking in a sequence

In this section we explain how our algorithm, which is fast because frame-by-frame, can nevertheless be used for some object tracking purposes.

The algorithm, as presented above, does only detect regions in a given frame. These regions correspond to moving objects with respect to the background image. At this point there is no space connectivity: two adjacent rectangles might be part of the same object or not, and there is no time connectivity: the treatment is frame by frame thus no links exist between frames. If we want to track moving objects we have to connect, or not, adjacent regions in space and time.

Thus, we first apply a connected region labeling algorithm ([2, 13]). For each frame we consider a binary table where the entry is "1" if the block is part of a region detected as moving object, else we have "0". Now the algorithm scans this image twice, block-by-block, from top to bottom and left to right, in order to identify connected regions. In our implementation we have chosen 4 connectivity on the discrete grid.

For each of these connected regions, we compute its barycenter. Using this information, we look for the region in the preceding frame which is closest.

Of course, this method gives only an approximate solution for object tracking. Errors do occur, *e.g.* two persons crossing [16]. Nevertheless, in a lot of applications, the results are very satisfactory and the computational effort is very reasonable, see Figure 10, and 11.

4 Concluding remarks and experimental results

Advantages of multi-step methods.

The first advantage of our multi-step method is **robustness to the parameters**, indeed, if we change the parameters in the difference image computation, we do not destroy the clusters, some pixels might change their "color" but the groups, as such, remain. See Figure 5.

As already pointed out in the introduction, the output is a **region** and not pixel level information, thus the algorithm **explicitly detects** moving objects and locates them. We obtain a quite precise segmentation of objects and are able to track moving objects through the sequence. Yet, as treatment of our algorithm is frame-by-frame we can work real time and on-line.

Choice of parameters.

In this section we will demonstrate that the way the parameters involved in our algorithm are chosen is quite natural and straightforward.

The parameter e depends on the quantification process. A complete study of the influence of quantification error in the gradient direction is presented in [11]. For example, for gray values coded by integers, *e.g.* $\{0, \dots, 255\}$, the error on the gradient direction is about $\frac{1}{|\nabla I|}$. So tacking $e = 4$ means allowing an error of about $\frac{\pi}{12}$. This, together with $\phi = \frac{\pi}{4}$ is thus a natural choice.

The parameter E depends on how a change of contrast will be detected in the difference image, see proposition 1. If the variation of contrast not to be detected is between 2/3 and 3/2, then, as $e = 4$, we obtain $E = 6$.

The threshold on ϕ which means "no change in structure" or "non detection", can be obtained as follows. If we suppose the direction of the gradient to be uniformly distributed, the probability of "non detection" and the pixel being marked **black**, is given by ϕ/π. By taking $\phi = \pi/4$ we obtain a ratio of 0.25 for non detection, this corresponds to experimental evidence we observed.

The size of the rectangles should correspond to the approximate size of the moving object, this is known in most applications (cars, persons which are close or far, ...). The mesh size K should correspond to a third, or a quarter, of the size of the rectangle.

Thus, mainly the meaningfulness threshold T has a real influence on the final segmentation and cannot be computed automatically. It depends on the number of false alarms we can allow. In figure 6, we show the value of the maximal meaningful windows, and their meaningfulness values: the windows on the moving object have the highest meaningfulness (more than 100), the ones at the boundary of moving have high meaningfulness (between 30 and 10), and the bad placed ones have the lowest meaningfulness (less than 10). This figure demonstrates that the parameter T is very robust.

Experimental results.

For the experiments we present here, we used the "Person leaving bag by wall" movie from the CAVIAR database [23], the "Scenario 1 Sequence 4 Camera C" movie from the CREDS database [12] and the "Apron" movie from the ETISEO dataset [22].

The parameters are set to $e = 4$, $E = 6$, and $\phi = \pi/4$, the significance threshold T is set to 15. The test windows are of size 30×30, 20×20, and $K = 10$ for the CAVIAR and CREDS data, we take 60×60, 40×40, and $K = 20$ for the ETISEO sequence as the images are double of the size.

In Figures 7, 8, 9, 10 and 11 we present only some pictures out of the sequences, in order to illustrate the type of results we obtain, this shows that:
− there is no detection in the noise, this ensures robustness;
− moving objects are well detected, with an almost exact position;
− we use the non-symmetric difference image, but in these examples the symmetric and non-symmetric method give the same results;
− tracking results allow to understand the scene;
− the same set of parameters is adapted to both sequences, although they present different illumination conditions.

Conclusion.

We have introduced a general multi-step framework for video segmentation: we build a "difference image", find perceptual groupings, correct this approximate segmentation and add a tracking algorithm.

We use a new way to compare the image and the background based on level lines, and provide a stable way to handle this information: *a contrario* detection for perceptual groupings. We have shown the advantages of our method by using a model for cast shadows and highlights, and provide a way to deal with the ghost elimination problem.

Significance of blocks is used to refine the approximate segmentation, and the result is used in a tracking algorithm.

This general framework can be easily adapted to color images, by using the color information to build difference images. Another useful generalization is to deal with several background estimations, computing several difference images, and take as definition of the number of false alarm of a windows W the minimum of the NFA among the difference images. Such generalizations will be investigated.

The algorithm has been tested in video coming with different databases, including indoor and outdoor sequence, with the same parameters. Experimental results show that the algorithm is very robust to noise and to the background estimation.

References

[1] P. Aguiar and J. Moura. Figure-ground segmentation from occlusion. *IEEE Trans. on Image Processing*, 2005.

[2] D. Ballard and C. Brown. *Computer Vision.* Prentice-Hall, 1982.

[3] A. Bevilacqua, L. Di Stefano, and A. Lanza. Coarse-to-fine strategy for robust and efficient change detectors. *Advanced Video and Signal-Based Surveillance, 2005.*, 2005.

[4] A. Bevilacqua, L. Di Stefano, and A. Lanza. An effective multi-stage background generation algorithm. *Advanced Video and Signal-Based Surveillance, 2005.*, 2005.

[5] T. M. Cover and J. A. Thomas. *Elements of information theory.* Wiley Series in Telecommunications. John Wiley & Sons Inc., New York, 1991. A Wiley-Interscience Publication.

[6] D. Cremers. A multiphase level set framework for motion segmentation. In L.D. Griffin and M. Lillholm, editors, *Proceedings of the 4th Int. Conf. on Scale-Space Theories in Computer Vision*, volume 2695 of *Lecture Notes on Computer Science*, pages 599–614. Springer-Verlag, 2003.

[7] R. Deriche and N. Paragios. Geodesic active regions for motion estimation and tracking. *Proceedings of the Int. Conf. in Computer Vision*, pages 224–240, 1999.

[8] A. Desolneux, L. Moisan, and J.-M. Morel. Edge detection by helmholtz principle. *Journal of Mathematical Imaging and Vision*, vol 14:3:pp 271–284, 2001.

[9] A. Desolneux, L. Moisan, and J.-M. Morel. Maximal meaningful events and applications to image analysis. *Ann. Statist.*, 31(6):1822–1851, 2003.

[10] A. Desolneux, L. Moisan, and J.-M. Morel. Meaningful alignments. *International Journal of Computer Vision*, Volume 40, Issue 1:Pages 7 – 23, October 2000.

[11] A. Desolneux, L. Moisan, J.-M. Morel, and S. Ladjal. Dequantizing image orientation. *IEEE Transactions on Image Processing*, vol. 11:10:pp. 1129–1140, 2002.

[12] Call for Real-Time Event Detection Solutions. Url: http://www-dsp.elet.polimi.it/avss2005/MainFrame/Challenge.htm.

[13] B. Horn. *Robot Vision.* MIT Press, 1986.

[14] O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *Motion02*, pages 22–27, 2002.

[15] S. Jehan-Besson, M. Barlaud, and G. Aubert. Detection and tracking of moving objects using a new level set based method. *International Conference on Pattern Recognition*, 2000.

[16] P.M. Jorges, A.J. Abrantes, and J.S. Marques. Tracking with bayesian networks: extension to arbitrary topologies. *Image Processing, 2005.*, 2005.

[17] J.L. Lisani and J.M. Morel. Detection of major changes in satellite images. *Image Processing, 2003.*, 1:I– 941–4 vol.1, 2003.

[18] S. Masnou. *Thèse - etc...* PhD thesis, Ceremade, University Paris 9–Dauphine, 2005.

[19] P. Monasse. *Contrast Invariant Representation of Digital Images and Application to Registration.* PhD thesis, Ceremade, University Paris 9–Dauphine, 2000. Available at `http://pascal.monasse.free.fr`.

[20] P. Musé, F. Sur, F. Cao, and Y. Gousseau. Unsupervised thresholds for shape matching. *International conference on Image Processing*, 2003.

[21] P.W. Power and J.A. Schoonees. Understanding background mixture models for foreground segmentation. *Imaging and Vision Computing New Zealand.*, 2002.

[22] Etiseo program. `URL:http://www.etiseo.net/`.

[23] EC Funded CAVIAR project/IST 2001 37540. Url: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

[24] F. Ranchin and F. Dibos. Moving objects segmentation using optical flow. *Cahiers du CEREMADE 2004-41, Mathematics and Image Analysis 2004*, 2004.

[25] A.I. Reveal. Background subtraction techniques.

[26] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[27] T. Veit, F. Cao, and P. Bouthémy. Probabilistic parameter-free motion detection. *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 715–721, 2004.

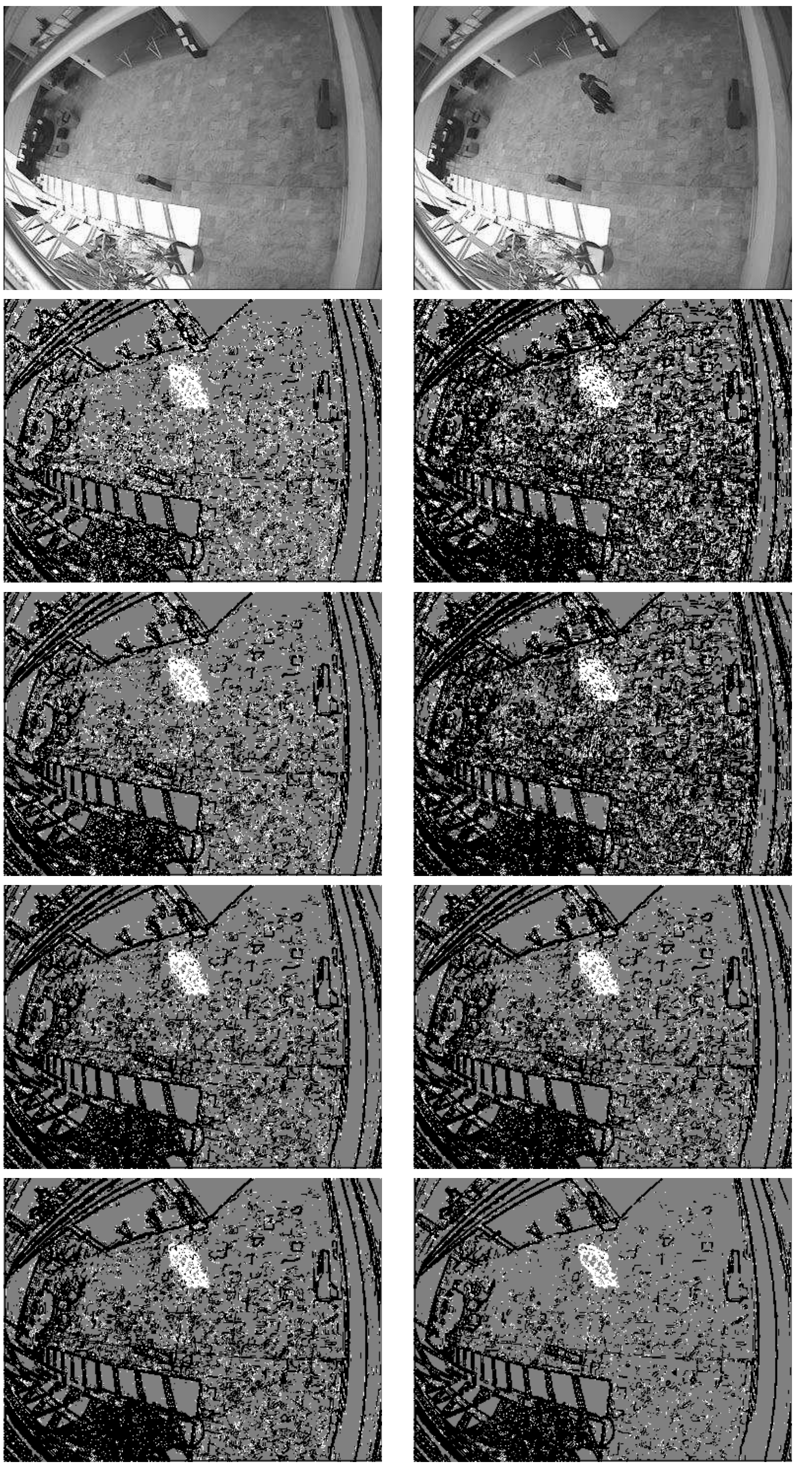

Figure 5: Top row, left the reference background and right the image 870. Below the difference image, first column, second to fifth row $e = 4$, $E = 6$ and $\phi = 20$, 30, 45 and 60. Second column, second to fifth row, ϕ is fixed to 45, we have (e, E) equal to $(2, 4)$, $(2, 6)$, $(4, 6)$ and $(6, 8)$. We notice that the cluster (one person moving) of white pixels is stable.

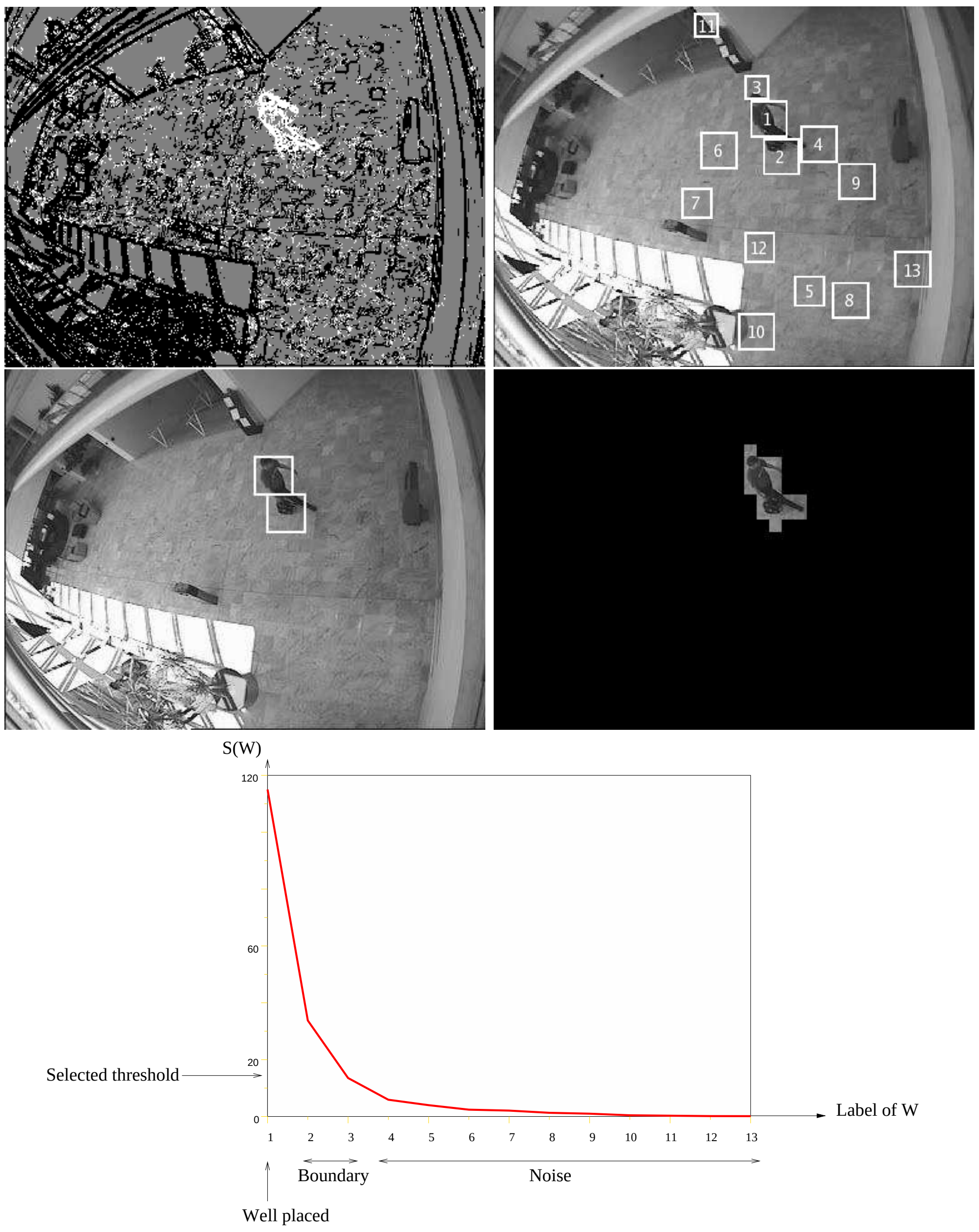

FIGURE 6: Top row, the difference image and the maximal meaningful windows labeled by decreasing $S(W)$ value; middle row, segmentation results, approximate and exact, with significance threshold $T = 15$; bottom row, graph of $W \mapsto S(W)$: windows with the highest meaningfulness are well-placed, T is a robust parameter.

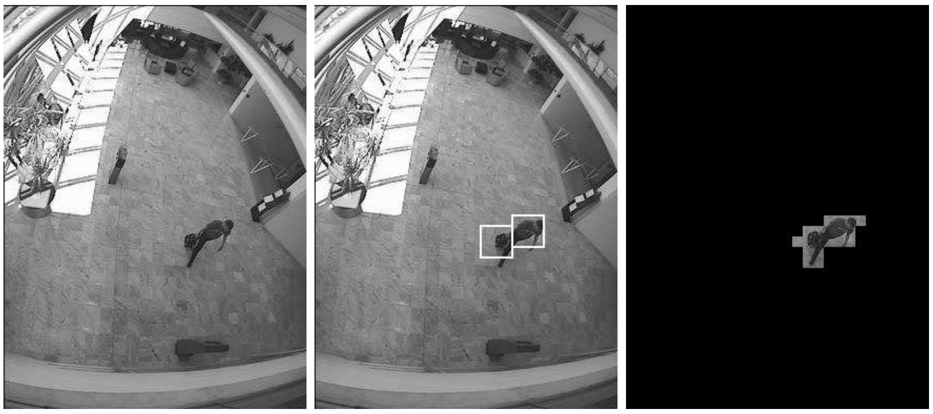

FIGURE 7: The original image, the maximal meaningful windows, the almost exact segmentation.

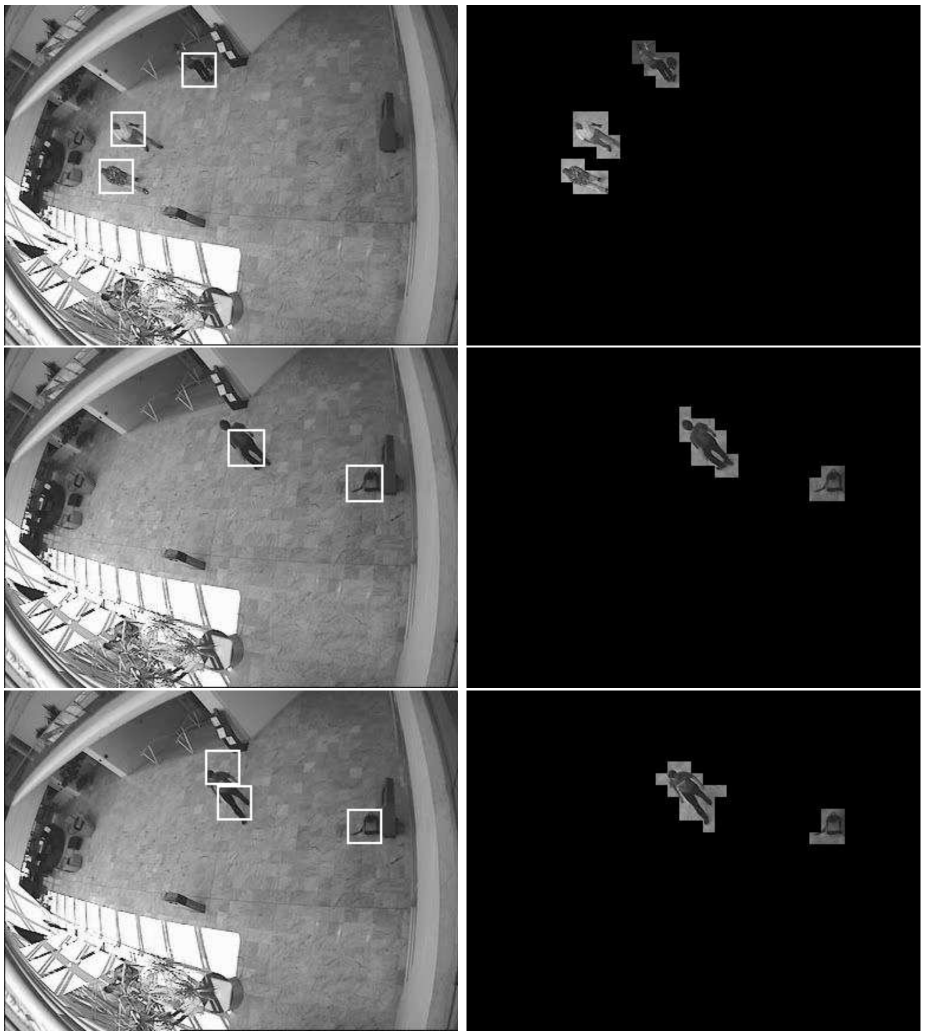

FIGURE 8: More results : left, maximal meaningful windows; right, segmentation results.

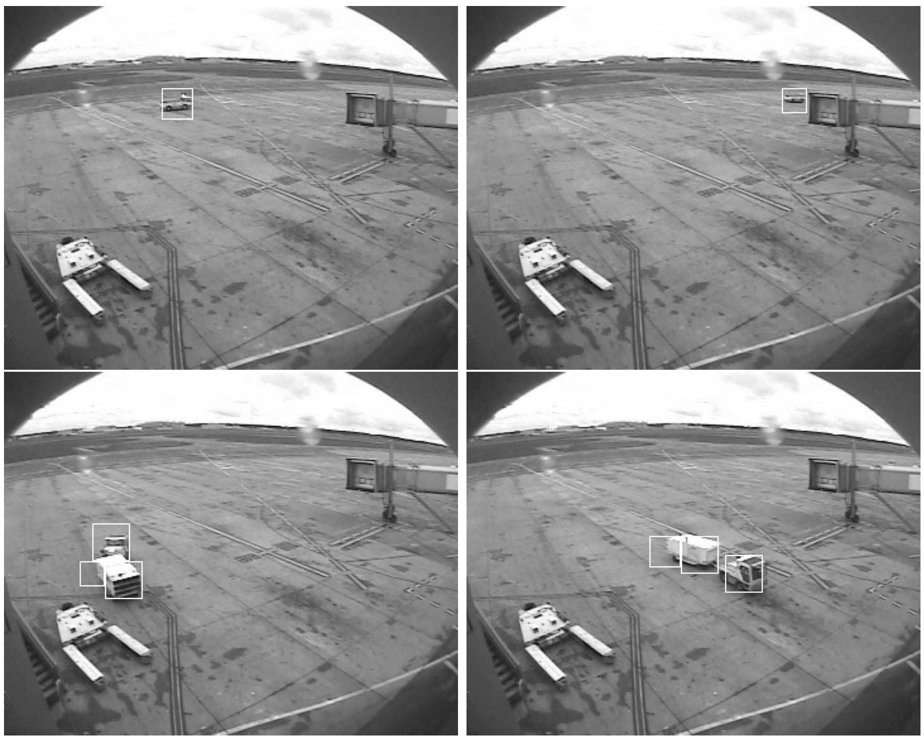

FIGURE 9: ETISEO: detecting moving objects in the vicinity of a plane

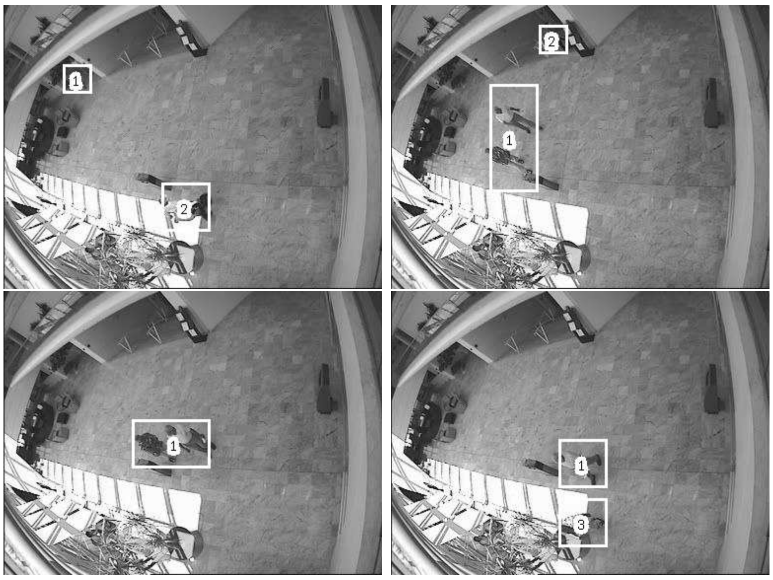

FIGURE 10: Tracking results on CAVIAR Database: top left, in the center a man (label 2) and a group of two persons (label 2) entering the scene; top right, the man is leaving the room, the group has moved to the center; bottom left, the man has disappeared, the group stays in the center; bottom right, the group splits (label 1 ,3).

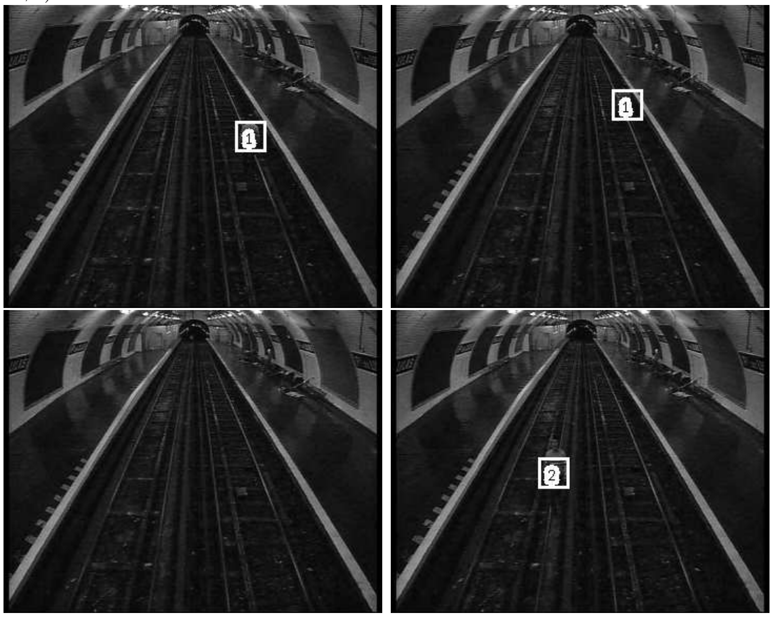

FIGURE 11: Tracking results on CREDS Database: top row, a man (label 1) is walking on the railway; bottom row, the man left the scene through the back and returns now label 2.