Performance evaluation of object detection algorithms for video surveillance

Jacinto Nascimento⋆, *Member, IEEE*
jan@isr.ist.utl.pt

Jorge Marques
jsm@isr.ist.utl.pt

IST/ISR, Torre Norte, Av. Rovisco Pais, 1049-001, Lisboa Portugal

EDICS: 4-SEGM

Abstract

In this paper we propose novel methods to evaluate the performance of object detection algorithms in video sequences. This procedure allows us to highlight characteristics (e.g., region splitting or merging) which are specific of the method being used. The proposed framework compares the output of the algorithm with the ground truth and measures the differences according to objective metrics. In this way it is possible to perform a fair comparison among different methods, evaluating their strengths and weaknesses and allowing the user to perform a reliable choice of the best method for a specific application. We apply this methodology to segmentation algorithms recently proposed and describe their performance. These methods were evaluated in order to assess how well they can detect moving regions in an outdoor scene in fixed-camera situations.

Index Terms

Surveillance Systems, Performance Evaluation, Metrics, Ground Truth, Segmentation, Multiple Interpretations.

I. Introduction

VIDEO surveillance systems rely on the ability to detect moving objects in the video stream which is a relevant information extraction step in a wide range of computer vision applications. Each image is segmented by automatic image analysis techniques. This should be done in a reliable and effective way in order to cope with unconstrained environments, non stationary background and different object motion patterns. Furthermore, different types of objects are manually considered e.g., persons, vehicles or groups of people.

Many algorithms have been proposed for object detection in video surveillance applications. They rely on different assumptions e.g., statistical models of the background [1]–[3], minimization of Gaussian differences [4], minimum and maximum values [5], adaptivity [6,7] or a combination of frame differences and statistical background models [8]. However, few information is available on the performance of these algorithms for different operating conditions.

Two approaches have been recently considered to characterize the performance of video segmentation algorithms: pixel-based methods, template based methods and object-based methods. Pixel based methods assume that we wish to detect all the active pixels in a given image. Object detection is therefore formulated as a set of independent pixel detection problems. This is a classic binary detection problem provided that we know the ground truth (ideal segmented image). The algorithms can therefore be evaluated by standard measures used in Communication theory e.g., misdetection rate, false alarm rate and receiver operating characteristic (ROC) [9].

This work was supported by FCT under the project LTT and by EU project CAVIAR (IST-2001-37540).

Corresponding Author: Jacinto Nascimento, (**email:**jan@isr.ist.utl.pt), **Complete Address**: Instituto Superior Técnico-Instituto de Sistemas e Robótica (IST/ISR), Av. Rovisco Pais, Torre Norte, 6^o piso, 1049-001, Lisboa, PORTUGAL **Phone:** +351-21-8418270, **Fax:** +351-21-8418291

Several proposals have been made to improve the computation of the ROC in video segmentation problems e.g., using a perturbation detection rate analysis [10] or an equilibrium analysis [11]. The usefulness of pixel-based methods for surveillance applications is questionable since we are not interested in the detection of point targets but object regions instead. The computation of the ROC can also be performed using rectangular regions selected by the user, with and without moving objects [12]. This improves the evaluation strategy since the statistics are based on templates instead of isolated pixels.

A third class of methods is based on an object evaluation. Most of the works aim to characterize color, shape and path fidelity by proposing figures of merit for each of these issues [13]–[15] or area based performance evaluation as in [16]. This approach is instrumental to measure the performance of image segmentation methods for video coding and synthesis but it is not usually used in surveillance applications.

These approaches have three major drawbacks. First object detection is not a classic binary detection problem. Several types of errors should be considered (not just misdetection and false alarms). For example *what should we do if a moving object is split into several active regions ? or if two objects are merged into a single region ?* Second some methods are based on the selection of isolated pixels or rectangular regions with and without persons. This is an unrealistic assumption since practical algorithms have to segment the image into background and foreground and do not have to classify rectangular regions selected by the user. Third, it is not possible to define a unique ground truth. Many images admit several valid segmentations. If the image analysis algorithm produces a valid segmentation its output should be considered as correct.

In this paper we propose objective metrics to evaluate the performance of object detection methods by comparing the output of the video detector with the ground truth obtained by manual edition. Several types of errors are considered: splits of foreground regions; merges of foreground regions; simultaneous split and merge of foreground regions; false alarms, and detection failures. False alarms occur when false objects are detected. The detection failures are caused by missing regions which have not been detected.

In this paper five segmentation algorithms are considered as examples and evaluated. We also consider multiple interpretations in the case of ambiguous situations e.g., when it is not clear if two objects overlap and should be considered as a group or if they are separated apart.

The first algorithm is denoted as basic background subtraction (BBS) algorithm. It computes the absolute difference between the current image and a static background image and compares each pixel to a threshold. All the connected components are computed and they are considered as active regions if their area exceeds a given threshold. This is perhaps the simplest object detection algorithm one can imagine. The second method is the detection algorithm used in the $W4$ system [17]. Three features are used to characterize each pixel of the background image: minimum intensity, maximum intensity and maximum absolute difference in consecutive frames. The third method assumes that each pixel of the background is a realization of a random variable with Gaussian distribution (SGM - Single Gaussian Model) [1]. The mean and covariance of the Gaussian distribution are independently estimated for each pixel. The fourth algorithm represents the distribution of the background pixels with a mixture of Gaussians [2]. Some modes correspond to the background and some are associated with active regions (MGM - Multiple Gaussian Model). The last method is the one proposed in [18] and denoted as *Lehigh Omnidirectional*

Tracking System ($LOTS$). It is tailored to detect small non cooperative targets such as snipers. Some of these algorithms are described in a special issue of IEEE transactions on PAMI (August 2001), which describes a state of art methods for automatic surveillance systems.

In this work we provide segmentation results of these algorithms on the PETS2001 sequences, using the proposed framework. The main features of the proposed method are the following. Given the correct segmentation of the video sequence we detect several types of errors $i)$ splits of foreground regions, $ii)$ merges of foreground regions, $iii)$ simultaneously split and merge of foreground regions, $iv)$ false alarms (detection of false objects) and $v)$ the detection failures (missing active regions). We then compute statistics for each type of error.

The structure of the paper is as follows. Section 2 briefly reviews previous work. Section 3 describes the segmentation algorithms used in this paper. Section 4 describes the proposed framework. Experimental tests are discussed in Section 5 and Section 6 presents the conclusions.

II. Related Work

Surveillance and monitoring systems often require on line segmentation of all moving objects in a video sequence. Segmentation is a key step since it influences the performance of the other modules, e.g., object tracking, classification or recognition. For instance, if object classification is required, an accurate detection is needed to obtain a correct classification of the object.

Background subtraction is a simple approach to detect moving objects in video sequences. The basic idea is to subtract the current frame from a background image and to classify each pixel as foreground or background by comparing the difference with a threshold [19]. Morphological operations followed by a connected component analysis are used to compute all active regions in the image. In practice, several difficulties arise: the background image is corrupted by noise due to camera movements and fluttering objects (e.g., trees waving), illumination changes, clouds, shadows. To deal with these difficulties several methods have been proposed (see [20]).

Some works use a deterministic background model e.g., by characterizing the admissible interval for each pixel of the background image as well as the maximum rate of change in consecutive images or the median of largest inter-frames absolute difference [5,17]. Most works however rely on statistical models of the background, assuming that each pixel is a random variable with a probability distribution estimated from the video stream. For example, the Pfinder system ("Person Finder") uses a Gaussian model to describe each pixel of the background image [1]. A more general approach consists of using a mixture of Gaussians to represent each pixel. This allows the representation of multi modal distributions which occur in natural scene (e.g., in the case of fluttering trees) [2].

Another set of algorithms is based on spatio-temporal segmentation of the video signal. These methods try to detect moving regions taking into account not only the temporal evolution of the pixel intensities and color but also their spatial properties. Segmentation is performed in a 3D region of image-time space, considering the temporal evolution of neighbor pixels. This can be done in several ways e.g., by using spatio-temporal entropy, combined with morphological operations [21]. This approach leads to an improvement of the systems performance, compared with traditional frame difference methods. Other approaches are based on the 3D structure tensor defined from the pixels spatial and temporal derivatives, in a given time interval [22]. In this case, detection is based on the Mahalanobis distance, assuming a Gaussian distribution for the derivatives. This approach has been implemented in real time and tested with PETS 2005 data set. Other alternatives have also been considered e.g., the use of a region growing method in 3D space-time [23].

A significant research effort has been done to cope with shadows and with nonstationary backgrounds. Two types of changes have to be considered: show changes (e.g., due to the sun motion) and rapid changes (e.g., due to clouds, rain or abrupt changes in static objects). Adaptive models and thresholds have been used to deal with slow background changes [18]. These techniques recursively update the background parameters and thresholds in order to track the evolution of the parameters in nonstationary operating conditions. To cope with abrupt changes, multiple model techniques have been proposed [18] as well as predictive stochastic models (e.g., AR, ARMA [24,25]).

Another difficulty is the presence of ghosts [26], i.e., false active regions due to statics objects belonging to the background image (e.g., cars) which suddenly start to move. This problem has been addressed by combining background subtraction with frame differencing or by high level operations [27],[28].

III. Segmentation Algorithms

This section describes object detection algorithms used in this work: BBS, $W4$, SGM, MGM and $LOTS$. The BBS, SGM, MGM algorithms use color while $W4$ and $LOTS$ use gray scale images. In the BBS algorithm, the moving objects are detected by computing the difference between the current frame and the background image. A thresholding operation is performed to classify each pixel as foreground region if

$$|I^t(x,y) - \boldsymbol{\mu}^t(x,y)| > T, \tag{1}$$

where $I^t(x,y)$ is a 3×1 vector being the intensity of the pixel in the current frame and $\boldsymbol{\mu}^t(x,y)$ is the mean intensity (background) of the pixel, T is a constant.

Ideally, pixels associated with the same object should have the same label. This can be accomplished by performing a connected component analysis (e.g., using 8 - connectivity criterion). This step is usually performed after a morphological filtering (dilation and erosion) to eliminate isolated pixels and small regions.

The second algorithm is denoted here as $W4$ since it is used in the $W4$ system to compute moving objects [17]. This algorithm is designed for grayscale images. The background model is built using a training sequence without persons or vehicles. Three values are estimated for each pixel using the training sequence: minimum intensity (Min), maximum intensity (Max), and the maximum intensity difference between consecutive frames (D). Foreground objects are computed in four steps: $i)$ thresholding, $ii)$ noise cleaning by erosion, $iii)$ fast binary component analysis and $iv)$ elimination of small regions.

We have modified the thresholding step of this algorithm since often leads to a significant level of miss classifications. We classify a pixel $I(x,y)$ as a foreground pixel iff

$$|I^t(x,y) < \mathrm{Min}(x,y)| \;\vee\; |I^t(x,y) > \mathrm{Max}(x,y)|) \;\wedge |I^t(x,y) - I^{t-1}(x,y)| > D(x,y) \tag{2}$$

Figs. 1, 2 show an example comparing both approaches. Fig. 1 shows the original image with two active regions. Figs. 2(a),(b) display the output of the thresholding step performed as in [17] and using (2).

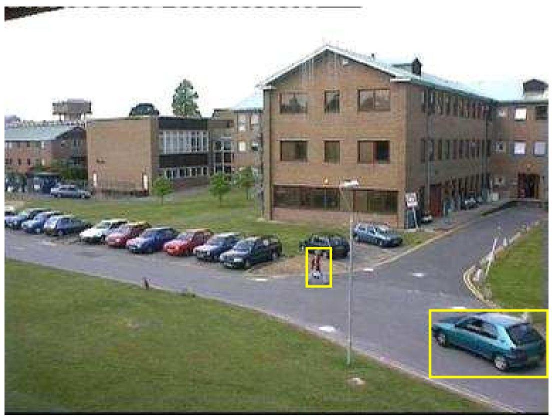


Fig. 1. Two regions (in bounding boxes) of an image.

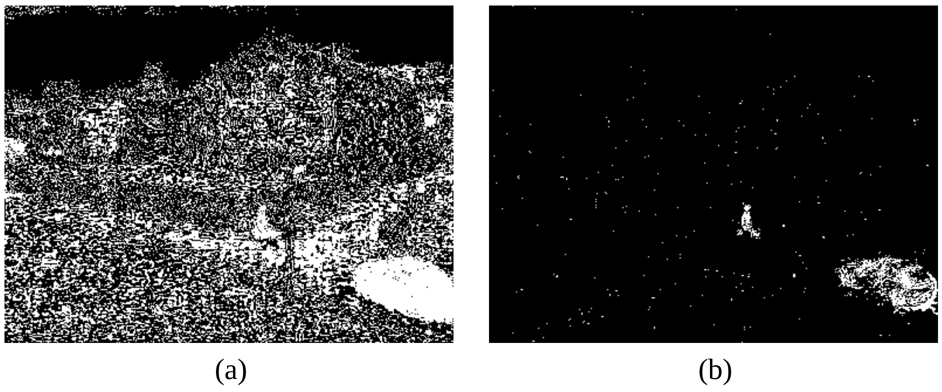


Fig. 2. Thresholding results: (a) using the approach as in [17] and (b) using (2).

The third algorithm considered in this study is the SGM (Single Gaussian Model) algorithm. In this method, the information is collected in a vector $[Y, U, V]^T$, which defines the intensity and color of each pixel. We assume that the scene changes slowly. The mean $\boldsymbol{\mu}(x, y)$ and covariance $\Sigma(x, y)$ of each pixel can be recursively updated as follows

$$\boldsymbol{\mu}^t(x, y) = (1 - \alpha)\boldsymbol{\mu}^{t-1}(x, y) + \alpha I^t(x, y), \tag{3}$$

$$\Sigma^t(x, y) = (1 - \alpha)\Sigma^{t-1}(x, y) + \alpha(I^t(x, y) - \boldsymbol{\mu}^t(x, y))(I^t(x, y) - \boldsymbol{\mu}^t(x, y))^T \tag{4}$$

where $I(x, y)$ is the pixel of the current frame in YUV color space, α is a constant.

After updating the background, the SGM performs a binary classification of the pixels into foreground or background and tries to cluster foreground pixels into blobs. Pixels in the current frame are compared with the background by measuring the log likelihood in color space. Thus, individual pixels are assigned either to the background region or a foreground region

$$l(x, y) = -\frac{1}{2}(I^t(x, y) - \boldsymbol{\mu}^t(x, y))^T(\Sigma^{-1})^t(I^t(x, y) - \boldsymbol{\mu}^t(x, y)) - \frac{1}{2}\ln|\Sigma^t| - \frac{m}{2}\ln(2\pi) \tag{5}$$

where $I^t(x, y)$ is a vector $(Y, U, V)^T$ defined for each pixel in the current image, $\boldsymbol{\mu}^t(x, y)$ is the pixel vector in the background image B.

If a small likelihood is computed using (5), the pixel is classified as active. Otherwise, it is classified as background.

The fourth algorithm (MGM) models each pixel $I(\mathrm{x}) = I(x,y)$ as a mixture of $N (N = 3)$ Gaussians distributions, i.e.

$$p(I(\mathrm{x})) = \sum_{k=1}^{N} \omega_k \mathcal{N}(I(\mathrm{x}), \boldsymbol{\mu}_k(\mathrm{x}), \Sigma_k(\mathrm{x})), \tag{6}$$

where $\mathcal{N}(I(\mathrm{x}), \boldsymbol{\mu}_k(\mathrm{x}), \Sigma_k(\mathrm{x}))$ is a multivariate normal distribution and ω_k is the weight of kth normal,

$$\mathcal{N}(I(\mathrm{x}), \boldsymbol{\mu}_k(\mathrm{x}), \Sigma_k(\mathrm{x})) = c \ \exp\Big\{-\frac{1}{2}\Big(I(\mathrm{x}) - \boldsymbol{\mu}_k(\mathrm{x})\Big)^T \Sigma_k^{-1}(\mathrm{x})\Big(I(\mathrm{x}) - \boldsymbol{\mu}_k(\mathrm{x})\Big)\Big\}. \tag{7}$$

with $c = \frac{1}{(2\pi)^{n/2}|\Sigma_k|^{\frac{1}{2}}}$. Note that each pixel $I(\mathrm{x})$ is a 3×1 vector with three component colors (red, green and blue), i.e., $I(\mathrm{x}) = [I(\mathrm{x})^R I(\mathrm{x})^G I(\mathrm{x})^B]^T$. To avoid an excessive computational cost, the covariance matrix is assumed to be diagonal [2].

The mixture model is dynamically updated. Each pixel is updated as follows: $i)$ The algorithm checks if each incoming pixel value x can be ascribed to a given mode of the mixture, this is the match operation. $ii)$ If the pixel value occurs inside the confidence interval with ± 2.5 standard deviation, a match event is verified. The parameters of the corresponding distributions (matched distributions) for that pixel are updated according to

$$\boldsymbol{\mu}_k^t(\mathrm{x}) = (1 - \lambda_k^t)\boldsymbol{\mu}_k^{t-1}(\mathrm{x}) + \lambda_k^t I^t(\mathrm{x}) \tag{8}$$

$$\Sigma_k^t(\mathrm{x}) = (1 - \lambda_k^t)\Sigma_k^{t-1}(\mathrm{x}) + \lambda_k^t (I^t(\mathrm{x}) - \boldsymbol{\mu}_k^t(\mathrm{x}))(I^t(\mathrm{x}) - \boldsymbol{\mu}_k^t(\mathrm{x}))^T \tag{9}$$

where

$$\lambda_k^t = \alpha \mathcal{N}(I^t(\mathrm{x}), \boldsymbol{\mu}_k^{t-1}(\mathrm{x}), \Sigma_k^{t-1}(\mathrm{x})) \tag{10}$$

The weights are updated by

$$\omega_k^t = (1 - \alpha)\omega_k^{t-1} + \alpha(M_k^t), \qquad \text{with} \quad M_k^t = \begin{cases} 1 & \text{matched models} \\ \\ 0 & \text{remaining models} \end{cases} \tag{11}$$

α is the learning rate. The non match components of the mixture are not modified. If none of the existing components match the pixel value, the least probable distribution is replaced by a normal distribution with mean equal to the current value, a large covariance and small weight. $iii)$ The next step is to order the distributions in the descending order of ω/σ. This criterion favours distributions which have more weight (most supporting evidence) and less variance (less uncertainty). $iv)$ Finally the algorithm models each pixel as the sum of the corresponding updated distributions. The first B Gaussian modes are used to represent the background, while the remaining modes are considered as foreground distributions. B is chosen as follows: B is the smallest integer such that

$$\sum_{k=1}^{B} \omega_k > T \tag{12}$$

where T is a threshold that accounts for a certain quantity of data that should belong to the background.

The fifth algorithm [18] is tailored for the detection of non cooperative targets (e.g., snipers) under non stationary environments. The algorithm uses two gray level background images B_1, B_2. This allows the algorithm to cope with intensity variations due to noise or fluttering objects, moving in the scene. The background images are initialized using a set of T consecutive frames, without active objects

$$B_1(x, y) = \min\{I^t(x, y), t = 1, \ldots, T\} \tag{13}$$

$$B_2(x, y) = \max\{I^t(x, y), t = 1, \ldots, T\} \tag{14}$$

where $t \in \{1, 2, \ldots, T\}$ denotes the time instant.

In this method, targets are detected by using two thresholds (T_L, T_H) followed by a *quasi-connected components* (QCC) analysis. These thresholds are initialized using the difference between the background images

$$T_L(x, y) = |B_1(x, y) - B_2(x, y)\ | + c_U \tag{15}$$

$$T_H(x, y) = T_L(x, y) + c_S \tag{16}$$

where, c_U and $c_S \in [0, 255]$ are constants specified by the user.

We compute the difference between each pixel and the closest background image. If the difference exceeds a low threshold T_L, i.e.,

$$\min_i |I^t(x, y) - B_i^t(x, y)| > T_L(x, y) \tag{17}$$

the pixel is considered as active. A target is a set of connected active pixels such that a subset of them verifies

$$\min_i |I^t(x, y) - B_i^t(x, y)| > T_H(x, y) \tag{18}$$

where $T_H(x, y)$ ia a high threshold. The low and high thresholds $T_L^t(x, y), T_H^t(x, y)$ as well as the background images, $B_i^t(x, y), i = 1, 2$ are recursively updated in a fully automatic way (see [18] for details).

IV. Proposed Framework

In order to evaluate the performance of object detection algorithms we propose a framework which is based on the following principles:

- A set sequences is selected for testing and all the moving objects are detected using an automatic procedure and manually corrected if necessary to obtain the ground truth. This is performed one frame per second.
- The output of the automatic detector is compared with the ground truth.
- The errors are detected and classified in one of the following classes: correct detections, detections failures, splits, merges, split/merges and false alarms.
- A set of statistics (mean, standard deviation) are computed for each type of error.

To perform the first step we made a user friendly interface which allows the user to define the foreground regions in the test sequence in a semi-automatic way. Fig. 3 shows the interface used to generate the ground truth. A set of frames is extracted from the test sequence (one per second). An automatic object detection algorithm is then used to provide a tentative segmentation of the test images. Finally, the automatic segmentation is corrected by the user, by merging, splitting, removing or creating active regions. Typically the boundary of the object is detected with a two pixel accuracy. Multiple segmentations of the video data are generated every time there is an ambiguous situation i.e., two close regions which are almost overlapping. This problem is discussed in section IV-D.

In the case depicted in the Fig. 3, there are four active regions: a car, a lorry and two groups of persons. The segmentation algorithm also detects regions due to lighting changes, leading to a number of false alarms (four). The user can easily edit the image by adding, removing, checking the operations, thus providing a correct segmentation. In Fig. 3 we can see an example where the user progressively removes the regions which do not belong to the object of interest. The final segmentation is shown at the bottom images.

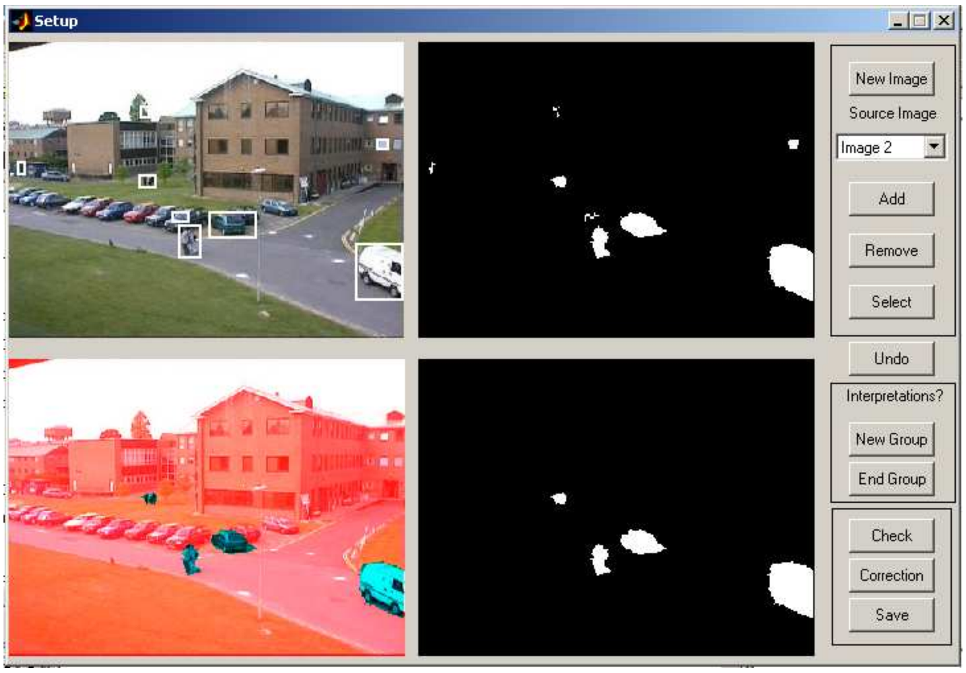


Fig. 3. User interface used to create the ground truth from the automatic segmentation of the video images.

The test images are used to evaluate the performance of object detection algorithms. In order to compare the output of the algorithm with the ground truth segmentation, a region matching procedure is adopted which allows to establish a correspondence between the detected objects and the ground truth. Several cases are considered:

1) **Correct Detection (CD)** or **1-1 match**: the detected region matches one and only one region.
2) **False Alarm (FA)**: the detected region has no correspondence.
3) **Detection Failure (DF)**: the ground truth region has no correspondence.
4) **Merge Region (M)**: the detected region is associated to several ground truth regions.
5) **Split Region (S)**: the ground truth region is associated to several detected regions.
6) **Split-Merge Region (SM)**: when the conditions pointed in 4, 5 are simultaneously satisfied.

A. Region Matching

Object matching is performed by computing a binary correspondence matrix $\mathcal{C}^t$ which defines the correspondence between the active regions in a pair of images. Let us assume that we have N ground truth regions $\tilde{R}_i$ and M detected regions R_j. Under these conditions $\mathcal{C}^t$ is a $N \times M$ matrix, defined as follows

$$\mathcal{C}^t(i,j) = \begin{cases} 1 \quad \text{if} \quad \dfrac{\sharp(\tilde{R}_i \cap R_j)}{\sharp(\tilde{R}_i \cup R_j)} > T \\ \\ 0 \quad \text{if} \quad \dfrac{\sharp(\tilde{R}_i \cap R_j)}{\sharp(\tilde{R}_i \cup R_j)} < T \end{cases} \qquad \forall_{i \in \{1,\dots,N\}, j \in \{1,\dots,M\}} \tag{19}$$

where T is the threshold which accounts for the overlap requirement. It is also useful to add the number of ones in each line or column, defining two auxiliary vectors

$$L(i) = \sum_{j=1}^{M} \mathcal{C}(i,j) \qquad i \in \{1,\dots,N\} \tag{20}$$

$$C(j) = \sum_{i=1}^{N} \mathcal{C}(i,j) \qquad j \in \{1,\dots,M\} \tag{21}$$

When we associate ground truth regions with detected regions six cases can occur: zero-to-one, one-to-zero, one-to-one, many-to-one, one-to-many, many-to-many associations. These correspond to false alarm, misdetection, correct detection, merge, split and split-merge.

Detected regions R_j are classified according to the following rules

$$\begin{array}{ll} \text{CD} & \exists_i : L(i) = C(j) = 1 \wedge \mathcal{C}(i,j) = 1 \\ \text{M} & \exists_i : C(j) > 1 \wedge \mathcal{C}(i,j) = 1 \\ \text{S} & \exists_i : L(i) > 1 \wedge \mathcal{C}(i,j) = 1 \\ \text{SM} & \exists_i : L(i) > 1 \wedge C(j) > 1 \wedge \mathcal{C}(i,j) = 1 \\ \text{FA} & \exists_i : C(j) = 0 \end{array} \tag{22}$$

Detection failures (DF) associated to the ground truth region $\tilde{R}_i$ occurs if $L(i) = 0$.

The two last situations (FA, DF) in (22) occur whenever empty columns or lines in matrix $\mathcal{C}$ are observed.

Fig. 4 illustrates the six situations considered in this analysis, by showing synthetic examples. Two images are shown in each case, corresponding to the ground truth (left) and detected regions (right). It also depicts the correspondence matrix $\mathcal{C}$. For each case, the left image ($\tilde{I}$) contains the regions defined by the user (ground truth), the right image (I) contains the regions detected by the segmentation algorithm. Each region is represented by an white area containing a visual label. Fig. 4 (a) shows an ideal situation, in which each ground truth region matches only one detected region (correct detection). In Fig. 4 (b) the “square-region” has no correspondence with the detected regions, thus it corresponds to a detection failure. In Fig. 4 (c) the algorithm detects regions which have no correspondence to the I image, thus indicating a false alarm occurrence. In Fig. 4 (d) shows a merge of two regions since two different regions (“square” and “dot” regions in I) have the same correspondence to the “square region” in I. The remaining examples in this figure are self explaining, illustrating the split (e) and split-merge (f) situations.

B. Region Overlap

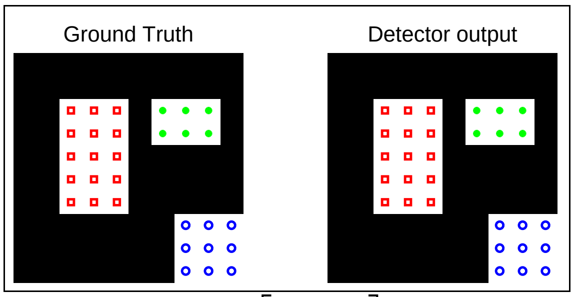
The region based measures described herein depends on an overlap requirement T (see (19)) between the region of the ground truth and the detected region. Without this requirement, this means that a single pixel overlap is enough for establishing a match between a detected region and a region in the ground truth segmentation, which does not make sense.

A match is determined to occur if the overlap is at least as big as the Overlap Requirement. The bigger the overlap requirement, the more the boxes are required to overlap hence performance usually declines as the requirement reaches 100%. In this work we use a overlap requirement of $T = 10\%$.

Fig. 5 illustrates the association matrices in two different cases considering an overlap requirement of $T = 20\%$. It can be seen that in Fig. 5(a) the region in the ground truth ("circle" region) is not represented by any detected region since the overlap is below the overlap requirement, leading to a detection failure. If we increase the overlap between these two regions (see Fig. 5(b)) we see that now we have a correct detection (second line, second column of $\mathcal{C}$). Finally it is illustrated a situation where two detection failures (in Fig. 5 (c)) become a split (in Fig. 5 (d)) if we increase the overlap among these regions.

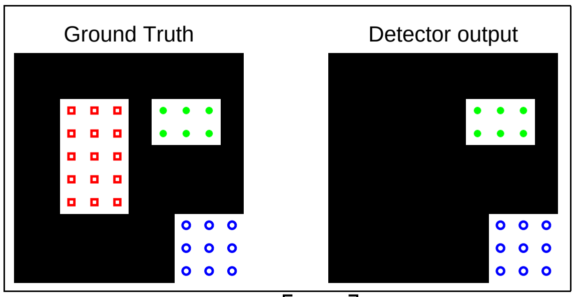
C. Area Matching

The match between pairs of the two regions (ground truth/ automatically detected) is also considered to measure the performance of the algorithms. The higher is the percentage of the match size, the better are the active regions produced by the algorithm. This is done for all the correctly detected regions. The match metric is defined by $\mathcal{M}(i) = \frac{\sharp(\tilde{R}_i \cap R_j)}{\sharp(\tilde{R}_i \cup R_j)}$, where j is the index of the corresponding detected region. The metric $\mathcal{M}$ is the area of the overlap normalized by the total area of the object. The average of $\mathcal{M}(i)$ in a video sequence will be used to characterize the performance of the detector.



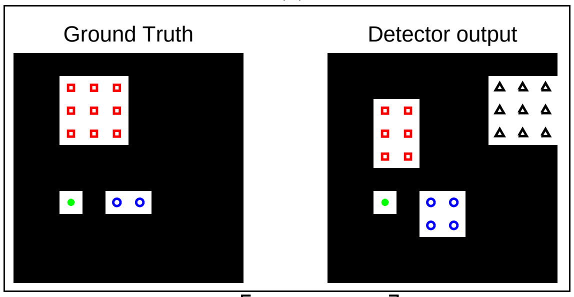
$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(a)



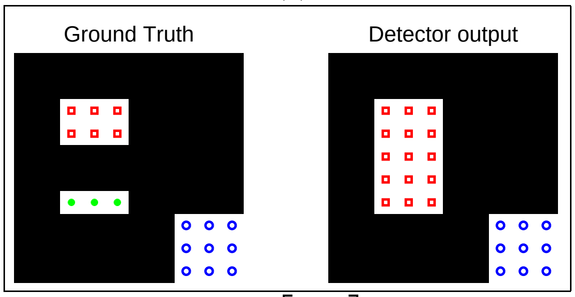
$$\mathcal{C} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(b)



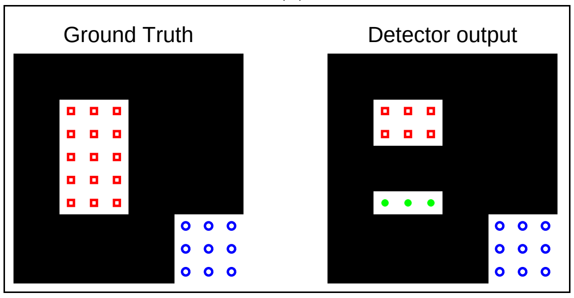
$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

(c)



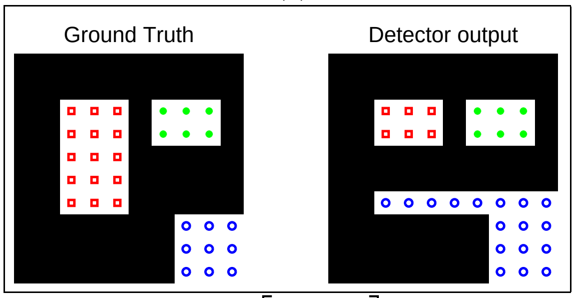
$$\mathcal{C} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(d)



$$\mathcal{C} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

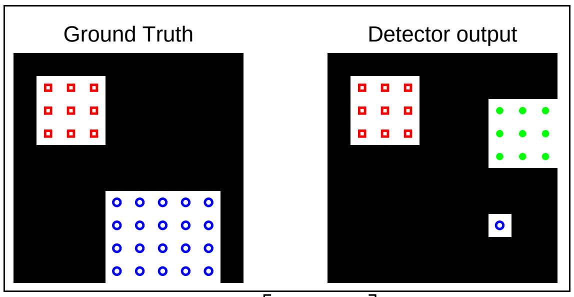
(e)



$$\mathcal{C} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

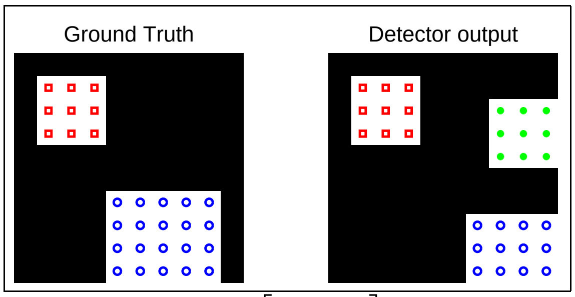
(f)

Fig. 4. Different matching cases: (a) Correct detection; (b) Detection Failure; (c) False alarm; (d) Merge; (e) Split; (f) Split Merge.



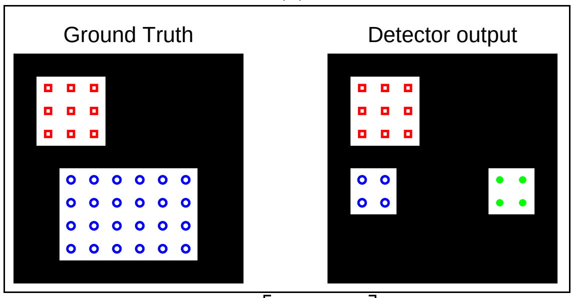
$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

(a)



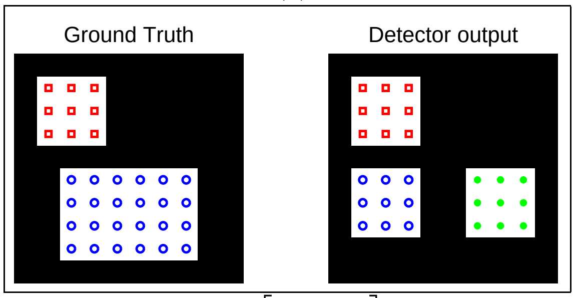
$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

(b)



$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

(c)



$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

(d)

Fig. 5. Matching cases with an overlap requirement of $T = 20\%$. Detection failure (overlap $<$ T) (a) Correct detection (overlap $>$ T) (b); two detection failures (overlap $<$ T) (c) and split (overlap $>$ T) (d).

D. Multiple Interpretations

Sometimes the segmentation procedure is subjective, since each active region may contain several objects and it is not always easy to determine if it is a single connected region or several disjoint regions. For instance, Fig. 6 (a) shows an input image and a manual segmentation. Three active regions were considered: person, lorry and group of people. Fig. 6 (b) shows the segmentation results provided by the SGM algorithm. This algorithm splits the group into three individuals which can also be considered as a valid solution since there is very little overlap. This segmentation should be considered as an alternative ground truth. All these situations should not penalize the performance of the algorithm. On the contrary, situations such as the ones depicted in Fig. 7 should be considered as errors. Fig. 7 (a) shows the ground truth and in Fig. 7 (b) the segmentation provided by the $W4$ algorithm. In this situation the algorithm makes a wrong split of the vehicle.

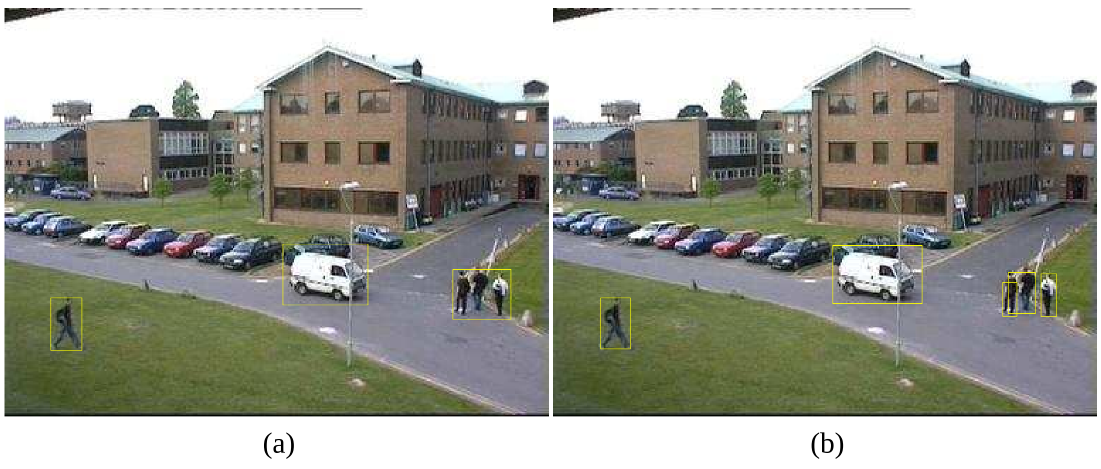


Fig. 6. Correct split example: (a) supervised segmentation, (b) SGM segmentation.

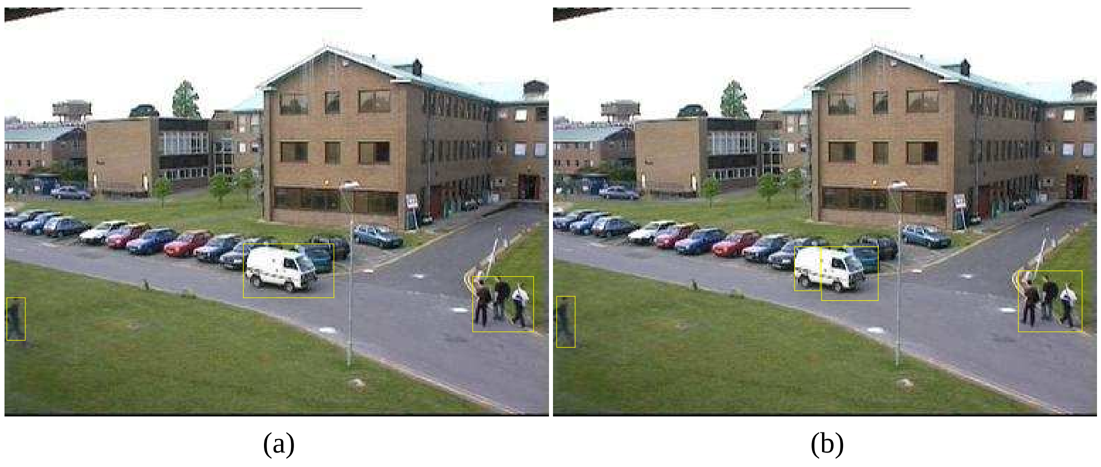


Fig. 7. Wrong split example: (a) supervised segmentation, (b) $W4$ segmentation.

Since we do not know how the algorithm behaves in terms of merging or splitting, every possible combinations within elements, belonging to a group, must be taken into account. For instance, another ambiguous situation is depicted in Fig. 8, where it is shown the segmentation results of the SGM method. Here, we see that the same algorithm provides different segmentations (both can be considered as correct) on the same group in different instants. This suggests the use of multiple interpretations for the segmentation. To accomplish this the evaluation setup takes into account all possible merges of single regions belonging to the same group whenever multiple interpretations should be considered in a group, i.e., when there is a small overlap among the group members.

The number of merges depends on the relative position of single regions. Fig. 9 shows two examples of different merged regions groups with three objects ABC (each one representing a person in the group). In the first example (Fig. 9 (a)) four interpretations are considered: all the objects are separated, they are all merged in a single active region or AB (BC) are linked and the other is isolated. In the second example an addition interpretation is added since A can be linked with C.

Instead of asking the user to identify all the possible merges in an ambiguous situation, an algorithm is used to generate all the valid interpretations in two steps. First we assign all the possible labels sequences to the group regions. If the same label is assigned to two different regions, these regions are considered as merged. Equation (23)(a) shows the labelling matrix M for the example of Fig. 9 (a). Each row corresponds to a different labelling assignment. The element M_{ij} denotes the label of the jth region in the ith labelling configuration. The second step checks if the merged regions are close to each other and if there is another region in the middle. The invalid labelling configuration are removed from the matrix M. The output of this step for the example of Fig. 9 (a) is in equation (23)(b). The labelling sequence 121 is discarded since region 2 is between region 1 and 3. Therefore, regions 1, 3 cannot be merged. In the case of the Fig. 9 (b) all the configurations are possible ($M = M_{FINAL}$). A detailed description of the labelling method is included in appendix VII-A.

Figs. 10,11 illustrate the generation of the valid interpretations. Fig. 10 (a) shows the input frame, Fig. 10 (b) shows the hand segmented image, where the user specifies all the objects (three objects must be provided separately in the group of persons) and Fig. 10 (c) illustrates the output of the SGM. Fig. 11 shows all possible merges of individual regions. All of them are considered as correct. Remain to know which segmentation should be selected to appraise the performance. In this paper we choose the best segmentation, which is the one that provides the highest number of correct detections. In the present example the segmentation illustrated in Fig. 11 (g) is selected. In this way we overcome the segmentation ambiguities that may appear without penalizing the algorithm. This is the most complex situation which occurs in the video sequences used in this paper.

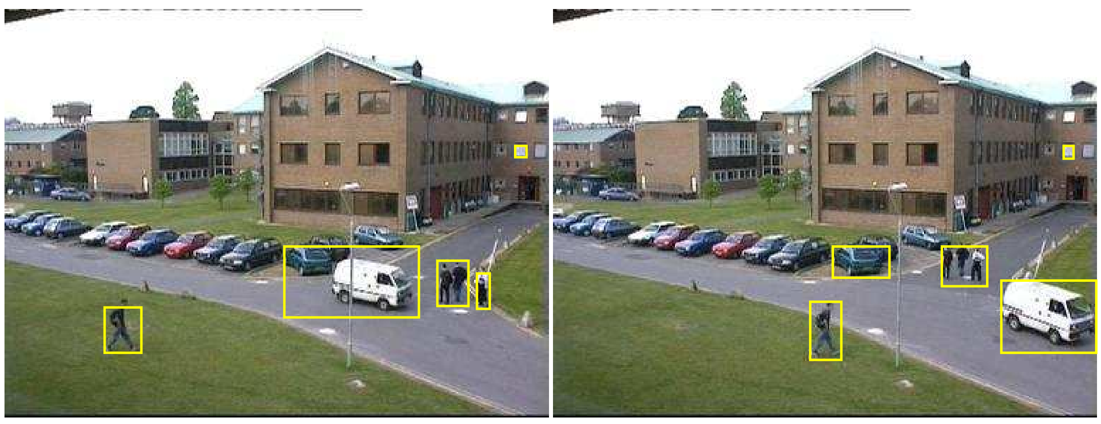


Fig. 8. Two different segmentations, provided by SGM method on the same group taken at different time instants.

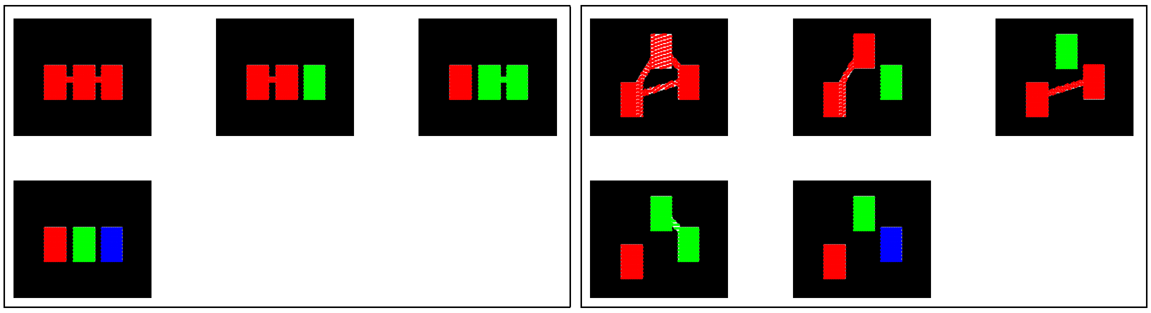

(a) (b)

Fig. 9. Regions linking procedure with three objects A B C (from left to right). The same number of foreground regions may have different interpretations: three possible configurations (a), or four configurations (b). Each color represent a different region.

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \qquad M_{FINAL} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ & & \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \tag{23}$$
$$(a) \qquad\qquad\qquad (b)$$

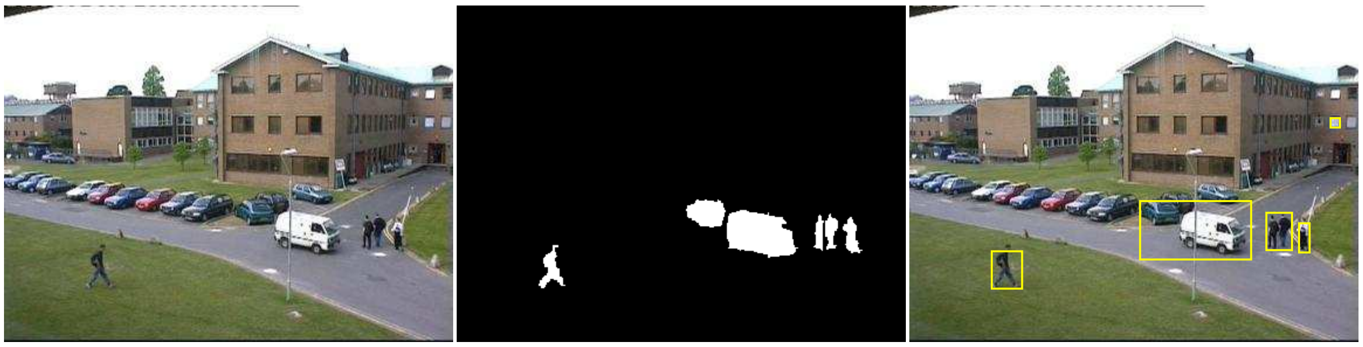

(a) (b) (c)

Fig. 10. Input frame (a), segmented image by the user (b), output of SGM (c).

V. Tests on PETS2001 dataset

This section presents the evaluation of several object detection algorithms using PETS2001 dataset. The training and test sequences of PETS2001 were used for this study. The training sequence has 3064 and the test sequence has 2688 frames. In both sequences, the first 100 images were used to build the background model for each algorithm.

The resolution is half-resolution PAL standard (288×384 pixels, 25 frames per second). The algorithms were evaluated using one frame per second. The ground truth was generated by an automatic segmentation of the video signal followed by a manual correction using a graphical editor described in section IV. The outputs of the algorithms were then compared with the ground truth. Most algorithms require the specification of the smallest area of an object. An area of 25 pixels was chosen since it allows to detect all objects of interest in the sequences.

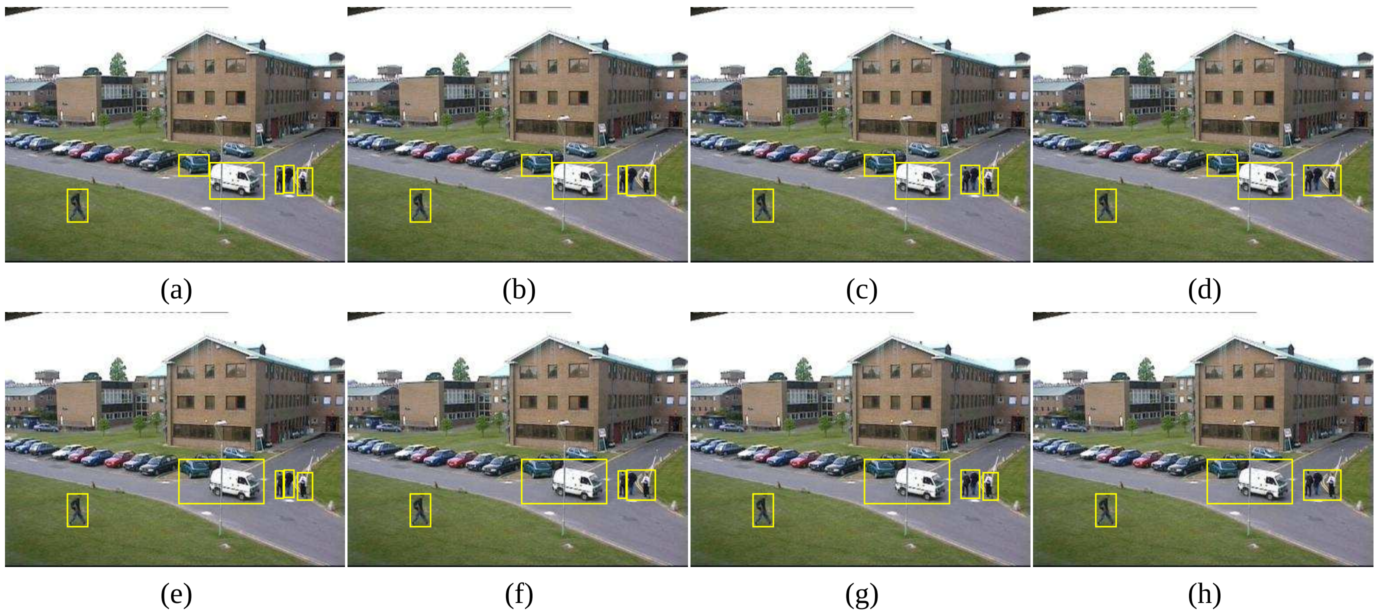


Fig. 11. Multiple interpretations given by the application. The segmentation illustrated in (g) is selected for the current frame.

A. Choice of the Model Parameters

The segmentation algorithms described herein depend on a set of parameters, which are mainly the thresholds and the learning rate α. In this scenario, we must figure out which are the best values for the most significant parameters for each algorithm. This was done using ROC curves which display the performance of each algorithm as a function of the parameters. The Receiver Operation Characteristic (ROC) have been extensively used in communications [9]. It is assumed that all the parameters are constant but one. In this case we have kept the learning rate α constant and varied the thresholds in the attempt to obtain the best threshold value T. We repeated this procedure for several values of α. This requires a considerable number of tests, but in this way it is possible to achieve a proper configuration for the algorithm parameters. These tests were made for a training sequence of the PETS2001 data set. Once the parameters are set, we use these values in a different sequence.

To ROC curves describe the evolution of the false alarms (FA) and detection failures (DF) as T varies. An ideal curve would be close to the origin, and the area under the curve would be close to zero. To obtain these two values, we compute these measures (for each value of T) by applying the region matching trough the sequence. The final values are computed as the mean values of FA and DF.

Fig. 12 shows the receiver operating curves (ROC) for all the algorithms. It is observed that the performance of BBS algorithm is independent of α. We can also see that this algorithm is sensitive with respect to the threshold, since there is a large variation of FA and DF for small changes of T, this can be viewed as a lack of smoothness of the ROC curve ($T = 0.2$ is the best value). There is a large number of false alarms in the training sequence due to the presence of a static object (car) which suddenly starts to move. The background image should be modified when the car starts to move. However, the image analysis algorithms are not able to cope with this situation since they only consider slow adaptations of the background. A ghost region is therefore detected in the place where the car was (a false alarm).

The second row of the Fig. 12 shows the ROC curves of the SGM method, for three values of α $(0.01, 0.05, 0.15)$. This method is more robust than the BBS algorithm with respect to the threshold. We see that for $-400 < T < -150$, and $\alpha = 0.01, \alpha = 0.05$ we get similar FA rates and a small variation of DF. We chose $\alpha = 0.05, T = -400$.

The third row show the results of the MGM method. The best performances are obtained for $\alpha < 0.05$ (first and second column). The best value of the α parameter is $\alpha = 0.008$. In fact, we observe the best performances for $\alpha \leq 0.01$. We notice that the algorithm strongly depends on the value of T, since for small variations of T there are significant changes of FA and DF. The ROC curve suggest that it is acceptable to choose $T > 0.9$.

The fourth row shows the results of the $LOTS$ algorithm for a variation of the sensitivity from 10% to 110%. As discussed in [29] we use a small α parameter. For the sake of computational burden, $LOTS$ does not update the background image in every single frame. This algorithm decreases the background update rate which takes place in periods of N frames. For instance an effective integration factor $\alpha = 0.0003$ is achieved by adding approximately $\frac{1}{13}$ of the current frame to the background in every 256^{th} frame, or $\frac{1}{6.5}$ in every 512^{th} frame. Remark that $B^t = B^{t-1} + \alpha D^t$, with $D^t = I^t - B^t$. In our case we have used intervals of 1024 (Fig. 12 (j)) 256 (Fig. 12 (k)) 128 (Fig. 12 (l)), being the best results achieved in the first case. The latter two cases Fig.(12) (k), (l) present a right shift in relation to (j), meaning that in these cases one obtains a large number of false alarms.

From this study we conclude that the best ROC curves are the curves associated with $LOTS$ and SGM since they have the smallest area under the curve.

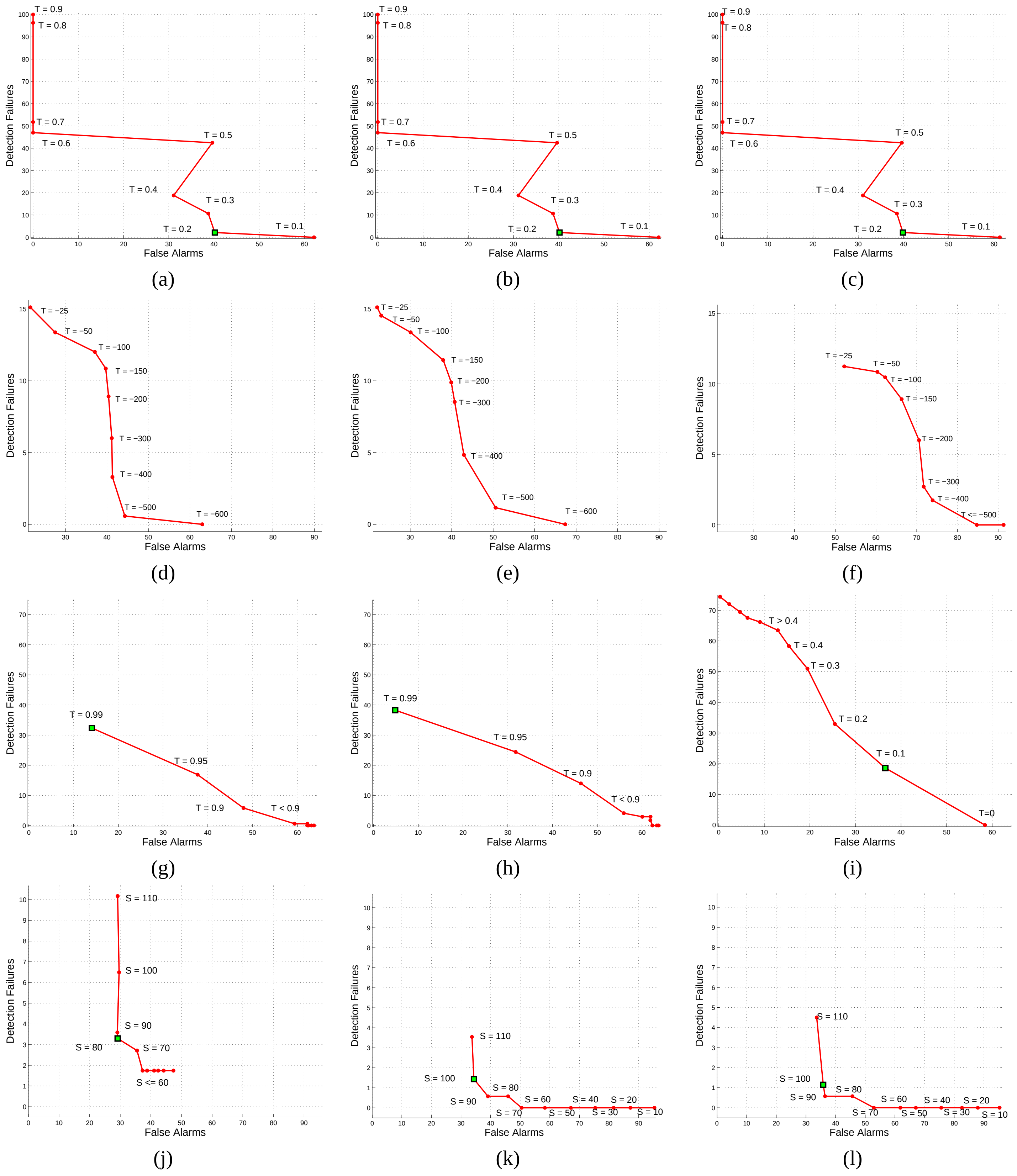


Fig. 12. Receiver Operation Characteristic for different values of α: BBS (first row: (a) $\alpha = 0.05$, (b) $\alpha = 0.1$, (c) $\alpha = 0.15$), SGM (second row: (d) $\alpha = 0.01$, (e) $\alpha = 0.05$, (f) $\alpha = 0.15$), MGM (third row: (g) $\alpha = 0.008$, (h) $\alpha = 0.01$, (i) $\alpha = 0.05$, $LOTS$ (fourth row with background update at every: (j) 1024^{th} frame, (k) 256^{th} frame, (l) 128^{th} frame.

B. Performance Evaluation

Table I (a),(b) shows the results obtained in the test sequence using the parameters selected in the previous study. The percentage of correct detections, detection failures, splits, merges and split-merges were obtained by normalizing the number of each type of event by the total number of moving objects in the image. Their sum is 100%. The percentage of false alarms is defined by normalizing the number of false alarms by the total number of detected objects. It is therefore a number in the range $0 - 100\%$.

Each algorithm is characterized in terms of correct detections, detection failures, number of splits, merges and split/merges false alarms as well as matching area.

Two types of ground truth were used. They correspond to different interpretations of static objects. If a moving object stops and remains still it is considered an active region in the first case (Table I (a)) and it is integrated in the background after one minute in the second case (Table I (b)). For example, if a car stops in front of the camera it will always be an active region in the first case. In the second case it will be ignored after one minute.

Let us consider the first case. The results are shown in Table I (a). In terms of correct detections, the best results are achieved by the $LOTS$ (91.2%) algorithm followed by SGM (86.8%).

Concerning the detection failures, the $LOTS$ (8.5%) followed by $W4$ (9.6%) outperforms all the others. The worst results are obtained by MGM (13.1%). This is somewhat surprising since MGM method, based on the use of multiple Gaussians per pixel, performs worse than the SGM method based on a single Gaussian. We will discuss this issue bellow. The $W4$ has the highest percentage of splits and the BBS, MGM methods tend to split the regions as well. The performance of the methods in terms of region merging is excellent: very few merges are observed in the segmented data. However, some methods tend to produce split/merges errors (e.g., $W4$, SGM and BBS). The $LOTS$ and MGM algorithm have the best score in terms of split/merge errors.

Let us now consider the false alarms (false positives). The $LOTS$ (0.6%) is the best and the MGM and BBS are the worst. The $LOTS$, $W4$ and SGM methods are much better than the others in terms of false alarms.

The $LOTS$ has the best tradeoff between CD and FA. Although the $W4$ produces many splits (splits can often be overcome in tracking applications since the region matching algorithms are able to track the active regions though they are split). The $LOTS$ algorithm has the best performance if all the errors are equally important.

In terms of matching area the $LOTS$ exhibit the best value in both situations.

In this study, the performance of the MGM method, based on mixtures of Gaussians is unexpectedly low. During the experiments we have observed the following: $i)$ when the object undergoes a slow motion and stops, the algorithm ceases to detect the object after a small period of time; $ii)$ when an object enters the scene it is not well detected during a few frames since the Gaussian modes have to adapt to this case.

This situation justify the percentage of the splits in both Tables. In fact, when a moving object stops, the MGM starts to split the region until it disappears, becoming part of the background. Objects entering into the scene will cause some detection failures (during the first frames) and splits, when the MGM method starts to separate the foreground region from the background.

Comparing the results in Table I (a) and (b) we can see that the performance of the MGM is improved. The detection failures are reduced, meaning that the stopped car is correctly integrated in the background. This produces an increase of correct detections by the same amount. However, we stress that the percentage of false alarms also increases. This means that the removal of the false positives is not stable. In fact some frames contain, as small active regions, the object which stops in the scene. In regard to the other methods, it is already expected that the false alarms percentage suffers an increase, since these algorithms remain with false positives throughout the sequence.

The computational complexity of all methods was studied to judge the performance of the five algorithms. Details about the number of operations in each method is provided in the Appendix VII-B.

%	*BBS*	*W4*	*SGM*	*MGM*	*LOTS*
Correct Detections	84.3	81.6	86.8	85.0	**91.2**
Detection Failures	12.2	9.6	11.5	13.1	**8.5**
Splits	2.9	5.4	**0.2**	1.9	0.3
Merges	**0**	1.0	**0**	**0**	**0**
Split/Merges	0.6	1.8	1.5	**0**	**0**
False Alarms	22.5	8.5	11.3	24.3	**0.6**
Matching Area	64.7	50.4	61.9	61.3	**78.8**

(a)

%	*BBS*	*W4*	*SGM*	*MGM*	*LOTS*
Correct Detections	83.5	84.0	86.4	85.4	**91.0**
Detection Failures	12.4	**8.5**	11.7	12.0	8.8
Splits	3.3	4.3	**0.2**	2.6	0.3
Merges	**0**	0.8	**0**	**0**	**0**
Split/Merges	0.8	1.8	1.7	**0**	**0**
False Alarms	27.0	15.2	17.0	28.2	**7.2**
Matching Area	61.3	53.6	61.8	65.6	**78.1**

(b)

TABLE I
PERFORMANCE OF FIVE OBJECT DETECTION ALGORITHMS.

VI. CONCLUSIONS

This paper proposes a framework for the evaluation of object detection algorithms in surveillance applications. The proposed method is based on the comparison of the detector output with a ground truth segmented sequence sampled at 1 frame per second. The difference between both segmentations is evaluated and the segmentation errors are classified into detection failures, false alarms, splits, merges and split/merges. To cope with ambiguous situations in which we do not know if two or more objects belong to a single active region or to several regions, we consider multiple interpretations of the ambiguous frames. These interpretations are controlled by the user through a graphical interface.

The proposed method provides a statistical characterization of the object detection algorithm by measuring the percentage of each type of error. The user can thus select the best algorithm for a specific application taking into account the influence of each type of error in the performance of the overall system. For example, in object tracking detection failures are worse than splits. We should therefore select a method with less detection failures, even if it has more splits than another method.

Five algorithms were considered in this paper to illustrate the proposed evaluation method. These algorithms are: Basic Background Subtraction (BBS), $W4$, Single Gaussian Model (SGM), Multiple Gaussian Model (MGM), Lehigh Omnidirectional Tracking System ($LOTS$). The best results were achieved by the $LOTS$ and SGM algorithm.

Acknowledgement: We are very grateful to the three anonymous reviewers for their useful comments and suggestions. We also thank R. Oliveira and P. Ribeiro for kindly provide the code of $LOTS$ detector.

VII. Appendix

A. Merge Regions Algorithm

The pseudo code of the region labelling algorithm is given in Algorithms 1, 2.

Algorithm 1 describes the synopsis of the first step, i.e., generation of the labels configurations. When the same label is assigned to two different regions, this means that these regions are considered as merged. Algorithm 2 describes the synopsis of the second step, which checks and eliminates label sequences which contain invalid merges. Every time the same label is assigned to a pair of regions we define a strip connecting the mass center of the two regions and check if the strip is intersected by any other region. If so, the labelling sequence is considered as invalid.

In these algorithms N denotes the number of objects, $label$ is a labelling sequence, M is the matrix of all label configurations, M_{FINAL} is a matrix which contains the information (final label configurations) needed to create the merges.

Algorithm 1 Main

```
1: $\mathrm{N} \leftarrow \mathrm{Num};$
2: $\mathrm{M(1)} \leftarrow 1;$
3: for $\mathrm{t} = 2$ to $\mathrm{N}$ do
4:     $\mathrm{AUX} \leftarrow [\,];$
5:     for $\mathrm{i} = 1$ to $\mathrm{size(M, 1)}$ do
6:         $\mathrm{label} \leftarrow \max(\mathrm{M(i, :)}) + 1;$
7:         $\mathrm{AUX} \leftarrow [\mathrm{AUX};\ [\mathrm{repmat(M(i, :), label, 1)}\ (1 : \mathrm{label})^{\mathrm{T}}]\ ];$
8:     end for
9:     $\mathrm{M} \leftarrow \mathrm{AUX};$
10: end for
11: $\mathrm{M_{FINAL}} \leftarrow \mathrm{FinalConfiguration(M)};$
```

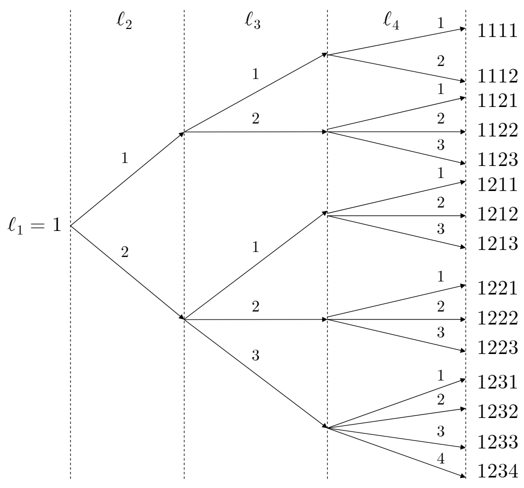


Fig. 13. Generation of the label sequences for the example in the Fig. 14.

To illustrate the purposes of algorithms 1 and 2 we will consider the example illustrated in the figure 14, where each rectangle in the image represents an active region.

Algorithm 1 computes the leaves of the graph shown in the Fig. 13 with all label sequences.

Algorithm 2 $\mathrm{M_{FINAL}}$ = FinalConfiguration (M)

```
1: $\mathrm{M_{FINAL}} \leftarrow [\,]$;
2: for $\mathrm{i} = 1$ to $\mathrm{lenght(M)}$ do
3:     Compute the centroids of the objects to be linked in $\mathrm{M(i,:)}$;
4:     Link the centroids with strip lines;
5:     if the strip lines do not intersect another object region then
6:         $\mathrm{M_{FINAL}} \leftarrow [\mathrm{M_{FINAL}^T}\ \mathrm{M(i,:)^T}]^\mathrm{T}$;
7:     end if
8: end for
```

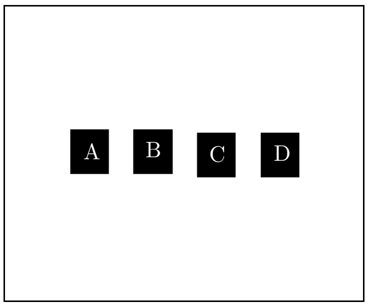


Fig. 14. Four rectangles A,B,C,D representing active regions in the image.

Algorithm 2 checks each sequence taking into account the relative position of the objects in the image. For example, configurations 1212,1213 are considered as invalid since object A cannot be merged with C (see Fig. 14). Equations (24)(a) and (b) show the output of the first and the second step respectively. All the labelling sequences considered as valid (the content of the matrix M_{FINAL}) provides the resulting images shown in Fig. 15.

$$\underset{(a)}{M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 3 \\ 1 & 2 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 3 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 2 & 2 & 3 \\ 1 & 2 & 3 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 2 & 3 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}} \qquad \underset{(b)}{M_{FINAL} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \\ & & & \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 3 \\ & & & \\ & & & \\ & & & \\ & & & \\ 1 & 2 & 2 & 2 \\ 1 & 2 & 2 & 3 \\ & & & \\ & & & \\ 1 & 2 & 3 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}} \tag{24}$$

B. Computational Complexity

Computational complexity was also studied to judge the performance of the five algorithms. Next, we provide comparative data on computational complexity using the “Big-O” analysis.

Let us define the following variables:

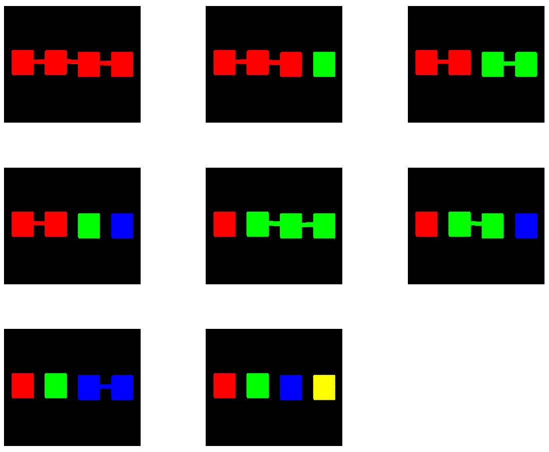


Fig. 15. Valid merges generated from the example in the Fig. 14.

- N, number of images in the sequence,
- L, C, number of lines and columns of the image,
- R, number of regions detected in the image,
- N_g, number of Gaussians.

The BBS, $W4$, SGM, MGM and $LOTS$ methods share several common operations namely: $i)$ morphological operations, for noise cleaning, $ii)$ computation of the areas of the regions and $iii)$ labelling assignment.

The complexity of these three operations is

$$K = \underbrace{(2 \times (\ell \times c) - 1) \times (L \times C)}_{\text{morphological op.}} + \underbrace{(L \times C) + R}_{\text{region areas op.}} + \underbrace{R \times (L \times C)}_{\text{Labels op.}} \tag{25}$$

where ℓ, c are the kernel dimensions ($\ell \times c = 9$, 8 - connectivity is used), L, C are the image dimensions and R is the number of detected regions. The first term, $2 \times (\ell \times c) - 1$, is the number of products and summations required for the convolution of each pixel in the image. The second term, $(L \times C) + R$, is the number of differences taken to compute the areas of the regions in the image. Finally, the $R \times (L \times C)$ term is the number of operations to label all the regions in the image.

BBS Algorithm

The complexity of the BBS is

$$\mathcal{O}\Big\{\Big(\underbrace{11 \times (L \times C)}_{\text{threshold op.}} + K\Big) \times N\Big\} \tag{26}$$

where $11 \times (L \times C)$ is the number of operations required to perform the thresholding step (see (1)) which involves $3 \times (L \times C)$ differences and $8 \times (L \times C)$ logical operations.

W4 Algorithm

The complexity of this method is

$$\mathcal{O}\Big\{\Big(\underbrace{2 \times [2p^3 + (L \times C) \times (p + (p-1))]}_{\text{rgb2gray op.}} + \underbrace{9 \times (L \times C)}_{\text{threshold op.}} + K + K_{W4}\Big) \times N\Big\} \tag{27}$$

where the first term is related to the conversion of the images to grayscale level, $p = 3$ (RGB space). The second one is concerned with the threshold operation (see (2)) which requires $9 \times (L \times C)$ operations (8 logical and 1 difference operations). The term K_{W4} corresponds to the background subtraction and morphological operations inside the bounding boxes of the foreground regions

$$K_{W4} = R \times \Big(\underbrace{9 \times (L_r \times C_r)}_{\text{Threshold op.}} + \underbrace{(2 \times (\ell \times c) - 1) \times (L_r \times C_r)}_{\text{morphological op.}}\Big) + \underbrace{(L \times C) + R}_{\text{region areas op.}} + \underbrace{R \times (L \times C)}_{\text{Labels op.}} \tag{28}$$

where L_r, C_r are the dimensions of the bounding boxes, assuming that the bounding boxes of the active regions have the same length and width.

SGM Algorithm

The complexity of the SGM method is

$$\mathcal{O}\Big\{\Big(\underbrace{p \times [2p \times (L \times C)]}_{\text{rgb2yuv op.}} + \underbrace{28 \times (L \times C)}_{\text{likelihood op.}} + \underbrace{(L \times C)}_{\text{threshold op.}} + K\Big) \times N\Big\} \tag{29}$$

The first term is related to the conversion of the images to YUV color space (in (29) $p = 3$). The second term is the number of operations required to compute the likelihood measure (see (5)). The third term is related to the threshold operation to classify the pixel as foreground if the likelihood is greater than a threshold, or classified as background otherwise.

MGM Algorithm

The number of operations of the MGM method is

$$\mathcal{O}\Big\{\Big(\underbrace{N_g(136 \times (L \times C))}_{\text{mixture modelling}} + \underbrace{2 \times (2N_g - 1) \times (L \times C)}_{\text{norm. and mixture op.}} + K\Big) \times N\Big\} \tag{30}$$

The first term depends on the number of Gaussians N_g. This term is related to the following operations: $i)$ matching operation - $70 \times (L \times C)$, $ii)$ weight update - $3 \times (L \times C)$ (see (11)), $iii)$ background update - $3 \times 8 \times (L \times C)$ (see (8)), $iv)$ covariance update for all color components - $3 \times 13 \times (L \times C)$ (see (9)). The second term accounts for: $i)$ weight normalization - $(2N_g - 1)(L \times C)$ and $ii)$ $(2N_g - 1) \times (L \times C)$ computation of the Gaussian mixture for all pixels.

LOTS Algorithm

The complexity of the $LOTS$ method is

$$\begin{aligned}\mathcal{O}\Big\{\Big(&\underbrace{[2p^3 + (L \times C) \times (p + (p - 1))]}_{\text{rgb2gray op.}} \\ &+ \underbrace{11 \times (L \times C) + (2 \times (L_b \times C_b) - 1) \times n_b + (2 \times (\ell \times c) - 1) \times (L_{rsize} \times C_{rsize}) + (L_{rsize} \times C_{rsize})}_{\text{QCC op.}} \\ &+ K\Big) \times N\Big\}\end{aligned} \tag{31}$$

The first term is related to the conversion of the images and it is similar with the first term in (27). The second term is related to the QCC algorithm. A number of $11 \times (L \times C)$ operations are needed to compute (17,18).

BBS	$1 + 30 \times (L \times C)$	3.3×10^6
LOTS	$55 + (35 + \frac{145}{64}) \times (L \times C)$	4.1×10^6
W4	$760 + 40 \times (L \times C)$	4.4×10^6
SGM	$1 + 66 \times (L \times C)$	7.2×10^6
MGM	$1 + 437 \times (L \times C)$	48.3×10^6

TABLE II
THE SECOND COLUMN GIVES THE SIMPLFIED EXPRESSION FOR EQUATIONS (26, 27, 29, 30, 31). THE SECOND COLUMN GIVES THE NUMBER OF TOTAL OPERATIONS.

The QCC analysis is computed in a low resolution image P_H, P_L. This is accomplished by converting each block of $L_b \times C_b$ pixels (in high resolution images) into a new element of the new matrices (P_H, P_L). Each element of P_H, P_L contains the active pixels of each block in the respective images. This task requires $(2 \times (L_b \times C_b) - 1) \times n_b$ operations (second term of QCC in (31)) where ($L_b \times C_b$) is the size of each block and n_b is the number of blocks in the image. A morphological operation (4-connectivity is used) over P_H is performed, taking $(2 \times (\ell \times c) - 1) \times (L_{_{rsize}} \times C_{_{rsize}})$ operations where ($L_{_{rsize}} \times C_{_{rsize}}$) is the dimension of the resized images. The targets candidates are obtained by comparing P_H and P_L. This task takes $(L_{_{rsize}} \times C_{_{rsize}})$ operations (fourth term in QCC).

For example, the complexity of the five algorithms is shown in table II assuming the following conditions for each frame

- the kernel dimensions, $\ell \times c = 9$,
- the block dimensions, $L_b \times C_b = 8 \times 8$, i.e., $(L_{_{rsize}} \times C_{_{rsize}}) = \frac{L \times C}{64}$ (for LOTS method),
- the number of Gaussians, $N_g = 3$ (for MGM method),
- a single region is detected with an area of 25 pixels, ($R = 1$, Ł$_r \times C_r = 25$),
- the image dimension is $(L \times C) = 288 \times 384$.

From the table, it is concluded that the four algorithms (BBS,$LOTS$,$W4$,SGM) have a similar computational complexity whilst MGM is more complex requiring a higher computational cost.

References

[1] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, “Pfinder: Real-time tracking of the human body,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 780–785, July 1997.

[2] C. Stauffer, W. Eric, and L. Grimson, “Learning patterns of activity using real-time tracking,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 747–757, August 2000.

[3] S. J. McKenna and S. Gong, “Tracking colour objects using adaptive mixture models,” *Image Vision Computing*, vol. 17, pp. 225–231, 1999.

[4] N. Ohta, “A statistical approach to background suppression for surveillance systems,” in *Proceedings of IEEE Int. Conference on Computer Vision*, 2001, pp. 481–486.

[5] I. Haritaoglu, D. Harwood, and L. S. Davis, “W^4: Who? when? where? what? a real time system for detecting and tracking people,” in *IEEE International Conference on Automatic Face and Gesture Recognition*, April 1998, pp. 222–227.

[6] M. Seki, H. Fujiwara, and K. Sumi, “A robust background subtraction method for changing background,” in *Proceedings of IEEE Workshop on Applications of Computer Vision*, 2000, pp. 207–213.

[7] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russel, “Towards robust automatic traffic scene analysis in real-time,” in *Proceedings of Int. Conference on Pattern Recognition*, 1994, pp. 126–131.

[8] R. Collins, A. Lipton, and T. Kanade, “A system for video surveillance and monitoring,” in *Proc. American Nuclear Society (ANS) Eighth Int. Topical Meeting on Robotic and Remote Systems*, Pittsburgh, PA, April 1999, pp. 25–29.

[9] H. V. Trees, *Detection, Estimation, and Modulation Theory*. John Wiley and Sons, 2001.

[10] T. H. Chalidabhongse, K. Kim, D. Harwood, and L. Davis, “A perturbation method for evaluating background subtraction algorithms,” in *Proc. Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS 2003)*, Nice, France, October 2003.

[11] X. Gao, T.E.Boult, F. Coetzee, and V. Ramesh, “Error analysis of background adaption,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2000, pp. 503–510.

[12] F. Oberti, A. Teschioni, and C. S. Regazzoni, “Roc curves for performance evaluation of video sequences processing systems for surveillance applications,” in *IEEE Int. Conf. on Image Processing*, vol. 2, 1999, pp. 949–953.

[13] J. Black, T. Ellis, and P. Rosin, “A novel method for video tracking performance evaluation,” in *Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, Nice, France, 2003, pp. 125–132.

[14] P. Correia and F. Pereira, “Objective evaluation of relative segmentation quality,” in *Int. Conference on Image Processing*, 2000, pp. 308–311.

[15] C. E. Erdem, B. Sankur, and A. M.Tekalp, “Performance measures for video object segmentation and tracking,” *IEEE Trans. Image Processing*, vol. 13, no. 7, pp. 937–951, 2004.

[16] V. Y. Mariano, J. Min, J.-H. Park, R. Kasturi, D. Mihalcik, H. Li, D. Doermann, and T. Drayer, “Performance evaluation of object detection algorithms,” in *Proceedings of 16th Int. Conf. on Pattern Recognition (ICPR02)*, vol. 3, 2002, pp. 965–969.

[17] I. Haritaoglu, D. Harwood, and L. S. Davis, “W^4: real-time surveillance of people and their activities,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 809–830, August 2000.

[18] T. Boult, R. Micheals, X. Gao, and M. Eckmann, “Into the woods: Visual surveillance of non-cooperative camouflaged targets in complex outdoor settings,” in *Proceedings of the IEEE*, October 2001, pp. 1382–1402.

[19] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice Hall, 2002.

[20] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, “Detecting moving objects, ghosts and shadows in video streams,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 10, pp. 1337–1342, 2003.

[21] Y.-F. Ma and H.-J. Zhang, “Detecting motion object by spatio-temporal entropy,” in *IEEE Int. Conf. on Multimedia and Expo*, Tokyo, Japan, August 2001.

[22] R. Souvenir, J. Wright, and R. Pless, “Spatio-temporal detection and isolation: Results on the PETS2005 datasets,” in *Proceedings of the IEEE Workshop on Performance Evaluation in Tracking and Surveillance*, 2005.

[23] H. Sun, T. Feng, and T. Tan, “Spatio-temporal segmentation for video surveillance,” in *IEEE Int. Conf. on Pattern Recognition*, vol. 1, Barcelona, Spain, September, pp. 843–846.

[24] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh, “Background modeling and subtraction of dynamic scenes,” in *Proceedings of the ninth IEEE Int. Conf. on Computer Vision*, 2003, pp. 1305–1312.

[25] J. Zhong and S. Sclaroff., “Segmenting foreground objects from a dynamic, textured background via a robust kalman filter,” in *Proceedings of the ninth IEEE Int. Conf. on Computer Vision*, 2003, pp. 44–50.

[26] N. T. Siebel and S. J. Maybank, “Real-time tracking of pedestrians and vehicles,” in *Proc. of IEEE workshop on Performance Evaluation of tracking and surveillance*, 2001.

[27] R. Cucchiara, C. Grana, and A. Prati, “Detecting moving objects and their shadows: an evaluation with the PETS2002 dataset,” in *Proceedings of Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2002) in conj. with ECCV 2002*, Pittsburgh, PA, May 2002, pp. 18–25.

[28] Collins, Lipton, Kanade, Fujiyoshi, Duggins, Tsin, Tolliver, Enomoto, and Hasegawa, “A system for video surveillance and monitoring: Vsam final report,” Robotics Institute, Carnegie Mellon University, Tech. Rep. Technical report CMU-RI-TR-00-12, May 2000.

[29] T. Boult, R. Micheals, X. Gao, W. Y. P. Lewis, C. Power, and A. Erkan, “Frame-rate omnidirectional surveillance and tracking of camouflaged and occluded targets,” in *Second IEEE International Workshop on Visual Surveillance*, 1999, pp. 48–55.