Optical Engineering

SPIEDigitalLibrary.org/oe

Adaptive background modeling in multicamera system for real-time object detection

Massimo Camplani
Luis Salgado

SPIE

Adaptive background modeling in multicamera system for real-time object detection

Massimo Camplani
Luis Salgado
E.T.S.I Telecomunicación
Universidad Politécnica de Madrid
Grupo de Tratamiento de Imágenes
Avenida Complutense, 30
Madrid, 28040 Spain
E-mail: mac@gti.ssr.upm.es;
L.Salgado@gti.ssr.upm.es

Abstract. We present an adaptive and efficient background modeling strategy for real-time object detection in multicamera systems. The proposed approach is an innovative multiparameter adaptation strategy of the mixture of Gaussian (MoG) background modeling algorithm. This approach is able to efficiently adjust the computational requirements of the tasks to the available processing power and to the activity of the scene. The innovative approach allows one to adapt the MoG without a significant loss in the detection accuracy while contemporarily adhering to the real-time constraints. The adaptation strategy works at the local level by modifying, independently, the MoG parameters of each task, and then, whenever the results of the local strategy are not satisfactory, a global adaptation strategy starts that aims at balancing the workload among the tasks. Our approach has been tested on three different data sets, including several image sizes, heterogeneous environments (indoor and outdoor scenarios), and different real-time constraints. The results show that the proposed adaptive system is well suited for multicamera applications thanks to this efficiency and adaptability; it guarantees real-time highly accurate detections. [DOI: 10.1117/1.3662422]

Subject terms: multicamera; object detection; real-time processing; background modeling; mixture of Gaussians.

Paper 110382RR received Apr. 14, 2011; revised manuscript received Sep. 1, 2011; accepted for publication Oct. 31, 2011; published online Dec. 7, 2011.

1 Introduction

The growing computer-processing capabilities and the decreasing prices of high-quality cameras enable the use of multicamera systems in several computer-vision applications and broaden their market into new promising areas. In fact, multicamera systems are very attractive because they can avoid problems encountered with monocamera systems, such as occlusions and limited field of view; moreover, they allow are to extract 3-D information from the scene. Thus, 3-D scene information extraction and reconstruction through multicamera systems are currently used in several practical applications. In particular, they are well suited for video surveillance applications for object tracking and positioning in 3-D environments. Usually, independent 2-D tracking algorithms are applied to each camera node and then this information is fused by a 3-D tracking module. In Ref. 1, the 2-D tracking is performed combining Gabor filters and templates of human silhouettes, whereas the 3-D tracking is obtained using geometrical correspondence on the detected objects. In Ref. 2, the mixture of Gaussian (MoG) algorithm is used to detect moving objects in each view and then they are projected into the floor plane and combined with the projection of other views. In this way, it is possible to identify candidate moving objects that are tracked with Kalman filters.

Another interesting application is related to immersive videoconferencing environments, where the acquired images of conference participants are rendered in a shared virtual 3-D environment. Examples of this application can be found in Refs. 3 and 4. Similar applications are constituted by remote and collaborative environments where the users can interact with 3-D virtual objects; an example can be found in Ref. 5. In general, in these works, the objects of interest (user's body) are separated from the background scene and projected in a 3-D virtual world.

Novel video game platforms based on controllers that are sensitive to user motion and speed are becoming more and more popular. In fact, multicamera systems can be used to extract information about movements and position of the user, such as the one presented in Ref. 6. The main idea is to separate the user silhouette from the background and convert its pose in a game command.

Another important application for multicamera systems is sport game monitoring, where the information from different cameras is fused in order to track all the objects in the scene and collect information and statistics of players and teams. Example of these kind of applications can be found in Ref. 7, where particle filters are used to track objects in each single view and an unscented Kalman filter is used to combine the information provided by the particle filters and to calculate the trajectory of the final players. Another example is Ref. 8, where the images provided by several cameras (from 8 to 15 cameras) are processed by a hybrid system that combines graphic processing units (GPUs) and classical CPUs: Mixture of Gaussian algorithms have been used to segment the football players that are subsequently classified and tracked.

Starting from the continuously increasing interest for these systems, the demand of new and more efficient solutions is posing challenging issues to the research community. New systems and architectures are required in order to efficiently manage the huge amount of data coming from

0091-3286/2011/$25.00

different sensors: networks of smart cameras,[9] peer-to-peer network of computers or centralized architectures[10] are the most used solutions.

Moreover, new and optimized approaches for video processing are needed in order to merge and fuse the information acquired by the cameras. In particular, these approaches are mainly based on two fundamental steps: 2-D information extraction (from each data stream) and 2-D information combination for 3-D environment characterization. Independent of the implementation of the second step, the first one usually includes modules such as background modeling, object detection, and segmentation that are able to separate the objects of interest from the overall scene. The accuracy of the object-detection phase is very important and deeply affects the performance of high-level tasks, such as tracking and object recognition, because it has also been highlighted in Ref. 11.

Therefore, to successfully accomplish complex tasks in multicamera applications, the object-detection modules must guarantee a good trade-off between detection accuracy and real-time performance. This is the challenge that has been addressed in the work described in this paper.

Object detection is one of the basics of video processing and analysis and is generally performed by estimating a model of the background of the scene, and then, any deviation from the background model is considered as a moving object. Different background modeling techniques have been presented in the literature.[12] One of the most popular ones is the Gaussian mixture model presented by Stauffer and Grimson.[13] The MoG is very attractive because it allows one to accurately estimate the background model in the presence of varying backgrounds and gradual illumination changes. MoG aims at estimating a background model of the scene where each pixel is independently modeled as a mixture of Gaussian distributions. This pixelwise approach is computationally demanding because, for each pixel, all the parameters of the distributions must be stored and updated.

The use of the MoG in real-time applications is a challenging issue and can represent a bottleneck in video-processing applications; for this reason, several variants of this algorithm have been presented in literature (see Sec. 2.1 for more details and the survey presented in Ref. 14). Nevertheless, these approaches are strictly related to a specific application, showing a low degree of adaptability because they are based on the modification of one characteristic of the algorithm, for example, by modifying the number of Gaussians distributions[15–17] or by reducing the number of operations considering a smaller images[18,19] or using a different feature space.[20]

Moreover, MoG and its derivations are widely used in multicamera applications as first stage of video processing (see, for example, Refs. 2,5,8, and 21). However, their effective applicability under real-time constraints is quite limited (especially for centralized servers architectures) and, above all, the impact on the system performance and on the overall processing time has not been clearly investigated and considered in the development of these systems. Therefore, being the object-detection accuracy a fundamental issue,[11] in our view, it is very important the investigation of innovative approaches are able to adapt MoG through multiparametric variation strategies for the operation in real-time multicamera applications.

To address these problems, our research activity has been focused on the optimization of the object-detection stages in a multicamera environment, based on a centralized server architecture, for real-time applications: we present an adaptive and efficient background-modeling strategy based on MoG. The proposed approach is based on a multiparametric modification of the MoG algorithm: this strategy is able to adapt the computational requirements of the object detection tasks as a function of the scene activity and the available processing resources in order to respect the real-time constraints of the desired application without a significant loss of detection accuracy. Algorithm accuracy should be understood here not only as the capability to accurately detect new objects in the scene, but also as the ability to keep a continuously updated and accurate model of the quasi-static background. In our view, not to process one or more frames (frame skipping) to fulfill the real-time constraints is the worst solution in terms of performance because no information about the moving object(s) is extracted and, at the same time, the background model is not correctly updated and cumulative errors could affect the algorithm performance.

Another advantage of the proposed parameters adaptation policy is that it follows a multilevel approach. It works at local (camera) level by adapting the MoG parameters independently for each processing task. If the changes done at local level are not sufficient, then a global-level adaptation strategy balances the processing load among all the tasks.

The paper is structured as follows: in Sec. 2, the theoretical background of the MoG algorithm is given and, with the aim of clarifing the added value of the proposed technique with respect to existing approaches, an overview of the state of the art is presented. In Sec. 3 the proposed novel adaptive strategy is described in detail; in Sec. 4, the multicamera system is presented. Results are shown in Sec. 5, and conclusions are drawn in Sec. 6.

2 Background Modeling and Object Detection with Mixture of Gaussians

Object detection can be obtained estimating a representation of the background of the scene, and typically, any deviation form the background model is considered as a moving object. Different background modeling techniques have been presented in the literature,[12] being one of the most popular ones the Gaussian mixture model introduced by Stauffer and Grimson.[13] This strategy, the MoG algorithm, is very attractive because it allows one to accurately estimate the background model even for varying backgrounds. Even more, its pixel-based operation provides high flexibility to implement strategies to adapt the accuracy of the model to the detection needs and available computational resources. Therefore, we propose using it as the core of the object-detection approach. We first introduce the basic concepts and operation details of the MoG background subtraction strategy to further focus on state-of-the-art proposals for its real-time application that motivate the need for our work.

2.1 *Mixture of Gaussian Background Subtraction*

The MoG algorithm is composed by two fundamental steps. The first step is the estimation of whether or not a pixel belongs to the background model. In the second step, the model parameters are recursively updated. The probability to find a pixel at time t of intensity $\mathbf{X}$ is modeled as

$$P(\mathbf{X}_t) = \sum_{i=1}^{K} \omega_{i,t} \cdot \eta(\mathbf{X}_t, \mu_{i,t}, \Sigma_{i,t}), \tag{1}$$

where K is the number of Gaussians, $\omega_{i,t}$ is the weight associated at time t to the i'th Gaussian with mean $\mu_{i,t}$, and covariance matrix $\Sigma_{i,t}$. If the RGB planes are considered independent, then form of the covariance matrix can be considered as $\Sigma_{i,t} = \sigma_{i,t}^2 \mathbf{I}$. The weight of a Gaussian measures the accuracy with which it models the value of the corresponding pixel.

In the first step, for each incoming pixel, whether or not it belongs to the background model is estimated. The multimodal background model is considered to be formed by the distributions that have a high ratio between their weight and variance values ($r_i = \omega_i/\sigma_i$). A high value for r_i means that the distribution has modeled the pixel in the past (ω_i is high) very well and that it has shown low variability (low σ_i) with values close to the mean (μ_i) of the Gaussian. The first B distributions that exceed a certain threshold T are used for the background model,

$$B = \arg\min_b \left(\sum_{i=1}^{b} \omega_{i,t} \geq T \right). \tag{2}$$

T is a measure of the minimum portion of the data that should be accounted for by the background. For small values of T, a background modeled by few distributions is obtained; the limit is a unimodal Gaussian distribution. If T is high, then a multimodal background model is obtained that can include more than one color in it. A pixel belongs to one of the K distributions if Eq. (3) is satisfied,

$$\sqrt{(\mathbf{X}_{t+1} - \mu_{i,t})^T \Sigma_{i,t}^{-1} (\mathbf{X}_{t+1} - \mu_{i,t})} < \lambda \sigma_{i,t}. \tag{3}$$

A pixel is considered as belonging to one distribution if its value is within λ standard deviations of the distribution; in this way, a per-pixel/per-distribution threshold is defined.[13] A common choice (proposed in Ref. 13) is to set λ equal to 2.5, thus discarding only the pixel values that are very far from the mean of the distribution. It has been shown in Ref. 13 that the MoG algorithm performance does not present a high sensitivity to λ, and for this reason, in our implementation we fix its value to 2.5. If the pixel belongs to one of the background distributions, then it is classified as a background pixel; otherwise, it is classified as a foreground pixel. If a match is found, the parameters of the matching Gaussian are updated with the following equations:

$$\omega_{i,t+1} = \omega_{i,t}(1 - \alpha) + \alpha, \tag{4}$$

$$\rho = \alpha \cdot \eta(\mathbf{X}_{t+1}, \mu_{i,t}, \Sigma_{i,t}), \tag{5}$$

$$\mu_{i,t+1} = \mu_{i,t}(1 - \rho) + \rho \mathbf{X}_{t+1}, \tag{6}$$

$$\sigma_{i,t+1}^2 = \sigma_{i,t}^2(1 - \rho) + \rho(\mathbf{X}_{t+1} - \mu_{i,t+1})(\mathbf{X}_{t+1} - \mu_{i,t+1})^T, \tag{7}$$

where α is the so-called learning rate. The learning rate α determines the speed of adaptation to changes in the scene (i.e., illumination) and the speed of the incorporation of foreground objects to the background. It indicates the influence that the last image has on the Gaussian distribution parameters. For the unmatched Gaussians, all their parameters remain unchanged except the weight,

$$\omega_{i,t+1} = \omega_{i,t}(1 - \alpha). \tag{8}$$

If no match is found, then the least probable distribution with the lowest ratio r is substituted by a new one with low weight, high variance, and a mean equal to the pixel value. When all the parameters have been updated, the weights are normalized such that

$$\sum_{i=1}^{K} \omega_{i,t+1} = 1. \tag{9}$$

The original version of the algorithm of Stauffer and Grimson[13] has been widely studied and modified to overcome several problems, such as sudden change of illumination, shadows of moving objects, moving background objects, real-time constraints, and memory requirements. A recent and complete survey of the proposed algorithms can be found in Ref. 14.

In our experiments, we use the original algorithm except for Eq. (5). In order to reduce the processing time, we modified it with the following:

$$\rho = \frac{\alpha}{\omega_{i,t}}. \tag{10}$$

The choice of the learning parameter α and ρ has been widely explored in order to solve different problems of the original algorithm (see again Ref. 14). In particular, Eq. (10) was proposed in Ref. 22 to speed up the computation of ρ. In fact, Eq. (5) requires the calculation of a more complex term. As explained in Ref. 22, the update of Eq. (10) is faster than the update of Eq. (5) and it represents a good approximation of it for the matching distributions.

2.2 *Real-Time Application*

As previously mentioned, the pixelwise approach of the algorithm is demanding in terms of processing power and memory occupancy because it requires a continuous update of the parameters of each distribution. In literature, several optimized versions of the algorithm have been proposed to reduce the processing time. However, these approaches are focused only on the modification of one parameter of the MoG, thus presenting a low degree of adaptability.

One common solution to speed up processing is to dynamically adapt the number of distributions used to model the pixels. The main idea of these approaches is to reduce K in the image area where there is a static background. Examples of these techniques can be found in Refs. 15 and 16, where a threshold on ω is applied to decide what distribution must be removed. Another similar approach is the one presented in Ref. 17 that selects the number of Gaussians using the Dirichlet prior. Although this approach is efficient in terms of MoG algorithm speed up, it is limited in situations where a significant part of the images is wrapped by new objects. Moreover, some errors can be introduced because K is modified for each pixel individually, without considering any information related to the moving object position. For this reason, in the proposed strategy K is modified, also taking into account the region occupied by the foreground objects (for more details, see Sec. 3).

Because the update step of the MoG is a time-critical phase of the entire algorithm, some authors propose not to update the background model for each processed frame. In particular, in Ref. 23 the update is completed only if in the image there are significant illumination changes. In Ref. 24 the update frequency is tuned independently for each pixel, in particular pixels that have been recently classified as foreground are updated faster than pixels that belong to the background. In the proposed strategy, we partially use the latter approach considering also the region occupied by the foreground objects (for more details, see Sec. 3).

Other works reduce the computational time by applying the algorithm on a subset of the image pixels. In Ref. 18, a hierarchical approach has been implemented where the image is subdivided into blocks and MoG is applied to the random pixels of each block: when a foreground pixel is found in a block, it is recursively subdivided and new pixels of the block are analyzed. In Ref. 19, MoG has been applied at subblock level and then the obtained foreground objects are refined to a silhouette detector. Other authors suggest applying the algorithm only in regions of interest that are typically smaller compared to the entire image and, in this way, it is possible to reduce the computational time of the MoG. In Ref. 25, the region of interest has been identified by an experiential sampling technique system that has to be continuously reinitialized in order to detect new objects in the scene. In Ref. 26, the regions of interest have been detected via frame differencing.

A recent and interesting approach is based on the adaptation of the MoG algorithm to the mosaic-bayer space Ref. 20. In this way, it is possible to obtain a MoG based on color information that operates in a reduced space (approximately reduced 66%). Even if this approach reduces the processing time of the MoG, it is not flexible, does not present any adaptation to the content of the scene, and is only applicable to cameras that provide a bayerlike data stream.

Another work[27] proposes to switch from the MoG algorithm to a faster algorithm based on a background model obtained by frame averaging. One critical issue of this algorithm is related to the memory occupancy of both models. Moreover, the switching policy has not been adequately investigated.

In many applications, the real-time constraints are accomplished by implementing adhoc versions of the MoG on dedicated hardware platforms like FPGA and DSP.[28] Recently, a great attention has been paid to MoG algorithms based on Compute Unified Device Architecture (CUDA).[29,30] These types of solutions are very efficient, but they cannot be easily adapted to consumer applications, where conventional cameras are used and specific and expensive hardware is avoided. An increase of the real-time performance of these approaches is obtained with the use of more powerful hardware rather than being achieved through improved versions of MoG.

From our point of view, all these optimization strategies are very application specific and show a low degree of adaptability because speed up is achieved by the modification of only one characteristic of MoG. Furthermore, their direct applicability to multicamera systems is very limited, and therefore, more efficient and adaptive strategies are required.

3 Adaptive Background Modeling

As previously mentioned MoG has been widely used for background modeling and objects detection because it presents several advantages, such as the possibility to successfully model varying backgrounds and gradual illumination changes. However, it is a computational demanding algorithm in terms of memory usage and processing power. For this reasons, in case of multicamera applications based on the MoG algorithm, the respect of real-time constraints is a challenging issue.

The proposed multilevel adaptation strategy aims at the dynamic modification of the MoG parameters to reduce the computational requirements while keeping an acceptable reduction of the detection accuracy. In a first stage, it allows efficiently tuning, individually, the MoG parameters that are used to process each camera data stream. This adaptation at camera level is identified as the local-level adaptation (LLA). In a second stage, if LLA does not guarantee meeting the real-time constraints, a global-level adaptation (GLA) strategy is enabled to balance the processing load among all the data streams.

The main idea of this approach is to modify the parameters of the MoG algorithm independently for each pixel by using a less precise but faster algorithm in the image regions where there are no foreground objects. On the contrary, in the regions with foreground objects, the MoG is modified to use a more precise but slower algorithm. The parameters of the MoG algorithm, which are adaptively modified for each pixel, are the number of Gaussian distributions per pixel (K), the frequency with which the parameters are updated (F_{u}), and the size of the processed area: the algorithm can work either at block-size level (B_{mode} option enabled) rather than at pixel level (B_{mode} option disabled).

In the following sections, the multilevel adaptation strategy for background modeling is described: the main blocks are first introduced, followed by the details of the background modeling parameters variation policy, and finally, the multilevel adaptation strategy modeled as a finite state machine (FSM) is described.

3.1 *Main Blocks of the Adaptive Background Modeling Strategy*

In Fig. 1, the block diagram of the proposed adaptive strategy is shown. It is composed of three main blocks: the MoG block represents the background modeling algorithm; the 2-D tracking module is in charge of tracking all the detected objects through a Kalman filter; finally, the multilevel adaptation block is the module that manages how the parameters of the MoG algorithm have to be modified for each pixel.

3.1.1 *Mixture of Gaussian*

This block is in charge of applying the MoG algorithm to each incoming frame I_t according to the parameters set by the multilevel adaptation block. It generates the foreground (FG_t) and background (BG_t) images, which are binary masks (with a size equal to that of I_t) that indicate the foreground and background pixels in the image. Moreover, this block updates the background model. As previously mentioned, each pixel is modeled independently using different parameters (set by the multilevel adaptation block) for the MoG algorithm; for this reason, the parameters K, F_{u}, B_{mode}, and the region of interest (ROI) information represented in Fig. 1 must be considered as matrices (with a size equal to that of I_t) containing for each pixel the corresponding parameter value.

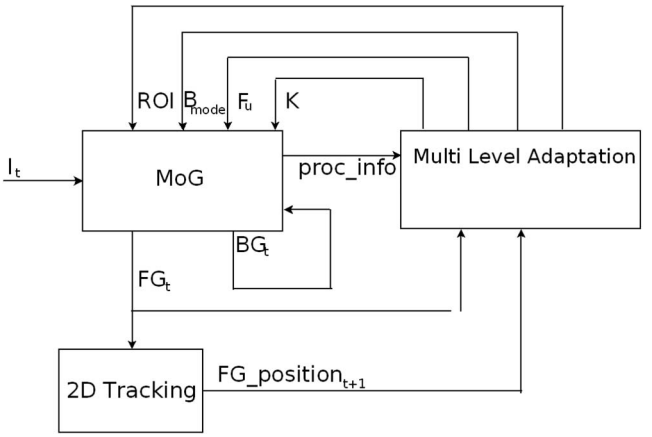

Fig. 1 Block diagram of the adaptive background modeling strategy.

3.1.2 *Two-dimensional tracking*

The 2-D tracking module is in charge of tracking each object detected by the MoG block and present in FG_t. For each detected object, one Kalman filter is used to track its trajectory. In particular, a linear Kalman filter, which uses the centroid and the size of the object bounding box as measured state variables, has been employed (for more details about tracking an object with Kalman filtering, see Ref. 31). Kalman filtering is used to track the detected objects and to estimate the size and position of their bounding boxes in the next frame. Thus, the $FG\_position_{t+1}$ presented in Fig. 1 is a vector containing the predicted state variables (position and size) relative to each object bounding box.

3.1.3 *Multilevel adaptation—finite state machine*

The proposed multilevel adaptation strategy is in charge of modifying the MoG parameters for each image pixel to find a trade-off between processing speed and detection accuracy. The multilevel strategy has been modeled with a FSM, which analyzes the foreground masks FG_t, the information provided by the 2-D tracking block $FG\_position_{t+1}$ and the information related to the processing time required by the MoG block (proc_info) to tune the MoG parameters (K, F_u, and B_{mode}). It is worth noting that, for the sake of simplicity, in Fig. 1 we connect the FSM with only one MoG module, but it is actually connected with all the MoG modules that process the video streams coming from the different cameras in the multicamera system.

As previously mentioned, the FSM aims to reduce the computational requirements of the MoG algorithm using a less precise but faster algorithm for those image regions where it is not expected to have a moving object. The predicted moving objects position, provided in $FG\_position_{t+1}$, is used to identify ROI that likely corresponds to moving objects; consequently, the FSM tunes the MoG parameters for each image pixel according to whether or not it belongs to the ROI.

3.2 *Background Modeling Parameters Variation*

The update phase in the MoG algorithm is a time-consuming operation that has a great impact on the MoG algorithm performance. For this reason, the approach presented in this paper proposes a default adaptation policy that aims to reduce the update frequency (F_u) for those pixels that belong to the background and present a stable value. On the contrary, the pixel update phase is accomplished for each frame only on the pixels that belong to the detected ROIs. Moreover, the FSM modifies the learning rate as a function of the F_u value; in fact, low update rates will require a larger value of α to improve the influence of the last image on the Gaussian parameters. As will be shown in Sec. 5, the adaptation of (F_u) guarantees an excellent trade-off between detection accuracy and algorithm speed up. The second parameter that is modified is K, the number of Gaussians, for each pixel in the image. In fact, pixels that belong to the static background can be modeled with few distributions. On the contrary, the pixels that have been recently detected as part of the foreground will be modeled with more distributions. The FSM can also activate the B_{mode} to further reduce the computational requirements on image areas that do not belong to the detected ROIs; however, this approach also reduces the spatial accuracy of the detection algorithm.

FSM monitors continuously proc_info of the MoG blocks and, if required, can modify the parameters in order to speed up the algorithm. As will be shown in detail in Sec. 4 the acquired images are stored temporarily in a buffer until they have been processed by the object detectors. Therefore, increasing (decreasing) buffer occupancy corresponds to a MoG processing that is slower (faster) than the acquisition rate. The adaptation strategy aims at preventing overflow conditions and, consequently, frame skipping. We define B_{max} as the buffer size, B_{ovfl} and B_{ocp} as the maximum allowed buffer occupancy (smaller than B_{max}), and the occupancy level during normal working condition (usually set to a small value). The proposed strategy modifies the MoG parameters to keep buffer occupancy close to B_{ocp} and prevent it from exceeding B_{ovfl}.

As previously explained, the FSM tries to speed up the algorithm in the regions that belong to a stable background and to maintain a more accurate, but slower algorithm, in the ROI regions. It is clear that, in cases where there are no objects in the scene and the detected foreground pixels are mainly due to noise and small changes in the images, it is convenient to use a very fast algorithm. These situations are identified analyzing the overall activity in the scene.

3.2.1 *Low-Pixel Activity Threshold (T_{LPA})*

The scene activity is measured as a function of pixels that belong to the foreground. The threshold T_{LPA} is used to evaluate the presence of significant objects in the scene: when the pixel activity level (Act_l) is smaller than T_{LPA}, the current frame is said to be characterized by a low-pixel activity (LPA). On the contrary, when $Act_l > T_{LPA}$, this means that moving objects have been detected and the current frame is said to be characterized by a high-pixel activity (HPA). In the case of LPA, it is useful to reduce the computational requirements using a less accurate but faster algorithm. On the contrary, when HPA is detected, a more accurate but slower algorithm is required, particularly for those areas where the foreground objects are located.

Act_l is evaluated as the sum of the area of the detected objects, normalized by the image size (IM_{size}),

$$Act_l = \frac{\sum_{i=1}^{N_{det}} A_i h(A_i)}{IM_{size}}, \tag{11}$$

where N_{det} is the number of the detected objects, A_i is the area of the i'th object and $h(A_i)$ is a function [described in Eq. (12)] introduced to remove from the pixel activity statistics isolated foreground pixels (very small areas) and objects that are not compact enough,

$$h(A_i) = \begin{cases} 1 & \text{if } A_i > A_{\min} \wedge A_i/A_{\text{bb}} > A_{\text{ratio}}, \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where $A_{\min}$ and A_{ratio} are respectively the minimum accepted value of object area and the minimum accepted ratio between object area and the area of the corresponding bounding box (A_{bb}).

3.3 *Multilevel Adaptation Strategy*

In this section, we present a detailed description of how the proposed strategy works at different levels and in what cases the parameters are modified. The proposed multilevel adaptation strategy is based on a local-level adaptation (LLA) and a global-level adaptation (GLA).

In the LLA, each video data stream is monitored independently from the other video data streams. Information about scene activity and buffer occupancy level is acquired, and consequently, the MoG parameters are adapted.

The GLA is activated only in critical situations: its behavior will be explained with the following example. We refer to MoG_{opt} as the optimal algorithm configuration that works without any parameters adaptation (maximum F_{u}, maximum K and B_{mode} disabled). On the contrary MoG_{min} is the fastest algorithm that does not compromise the detection accuracy (minimum F_{u}, minimum K, and B_{mode} enabled). Let us suppose that the MoG module that processes the i'th video data stream is working at MoG_{min}. However, its measured buffer occupancy is close to B^i_{ovfl}. In this case, it is not possible to further modify the algorithm parameters because this would dramatically decrease the background model quality and, therefore, the detection accuracy: then, a global strategy is required to adapt the parameters taking jointly into account the status of all video stream processing tasks.

In this case, GLA selects the processing task that is using more processing resources and forces it to reduce its computational cost (i.e., enabling B_{mode} option in the algorithm). As a consequence, the i'th processing task can use these newly available processing resources and consequently reduce the buffer occupancy. It is worth noting that, if this first modification is not sufficient to reduce the buffer occupancy of the i'th video data stream, the GLA strategy can be applied iteratively to the others processing tasks.

In Secs. 3.1.1 and 3.1.2, a detailed description of the state transitions of the FSM is given and then, for each state, the adaptation mechanism is described.

3.3.1 *Finite State Machine–State Transition*

The multilevel adaptation policy has been modeled as a FSM; its states and the transitions are shown in Fig. 2. The first state is AQ_START, where the acquisition and processing modules are initialized. When the first frame (event *first_frame*) is acquired and processed, the FSM changes state and it goes to the LPA_STATE.

The LPA_STATE is characterized by LPA frames: this is a typical case where a static background is present in the scene and there are no new objects to detect; therefore, it is possible to decrease the computational requirements of the process independently of the buffer occupancy level. If the incoming frames have been identified as HPA, the next state will be HPA_STATE.

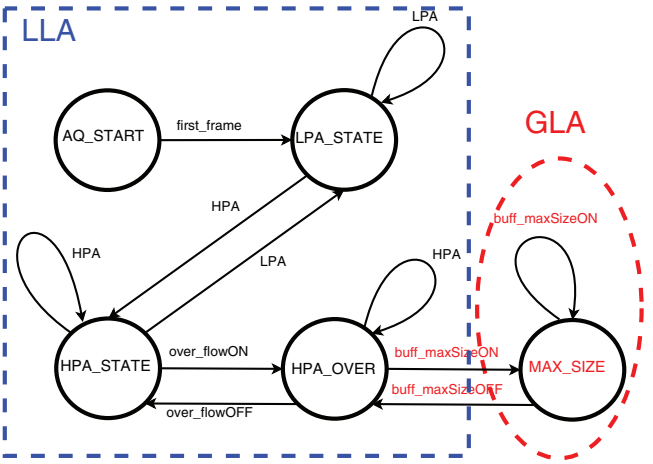

Fig. 2 FSM state transitions.

The FSM during normal condition operation stays in the HPA_STATE. This state is characterized by frames where there are significant detections, high precision in the detection algorithm is required, and the algorithm parameters are adapted with the target to keep buffer occupancy close to B_{ocp}. In case the parameters adaptation strategy that is applied to ensure highly accurate detections does not allow to keep the buffer occupancy close to B_{ocp}, for example if there are too many detected objects, buffer occupancy will start to increase. If it increases up to B_{ovfl} an over_flowON event occurs and the FSM changes state to HPA_OVER. On the other side, if the incoming frames have been identified as LPA the next state will be LPA_STATE.

The HPA_OVER is characterized by HPA frames and buffer occupancy greater than B_{ovfl}. In this case, moving objects have been accurately detected but still, to avoid frame skipping, it is further necessary to reduce the computational requirements of the algorithm. Once that buffer occupancy is below B_{ovfl} (over_flowOFF event), FSM changes state to HPA_STATE.

All these states are part of the LLA strategy (enclosed in the dashed square in Fig. 2). As previously mentioned, when local parameter adaptation is not sufficient to keep buffer occupancy below B_{ovfl}, it is necessary to start the GLA. In particular, the choice $B_{\text{ovfl}} < B_{\text{max}}$ gives a margin in which it is still possible to modify the adaptation strategy in order to further decrease the processing requirements of the tasks and then avoid frame skipping and buffer overflow. The event buff_maxSizeON occurs when the FSM is in the HPA_OVER state and the buffer occupancy is still increasing up to the buffer maximum size. In this case, the FSM changes state to MAX_SIZE, which is the only state that belongs to the GLA. Let us identify the detection process that has reached this state as the i'th process.

When the FSM is in the MAX_SIZE state, the processing task that is using more processing resources apart from the i'th process (let us call it the j'th process) is selected and forced to reduce its computational cost. It must be noted that the GLA stops the normal execution of the FSM for the j'th process and modifies its parameters regardless of its

current state. When the buffer occupancy of the i'th process has decreased (buff_maxSizeOFF event), the state machine moves again to HPA_OVER states and the FSM of the j'th process is restored to its previous state. In case the results are not sufficient, the FSM remains in this state and another process is forced to reduce its computational requirements. It must be noted that for the sake of simplicity of Fig. 2, we have reported the LLA states (enclosed in the square) only for one data stream. The same FSM model is used to manage parameters adaptation for all the video data streams in the multicamera system.

3.3.2 *Finite State machine-parameters modification policy*

The parameters of the MoG algorithm are modified according to the current state of the FSM. In this section, a detailed description of how they are modified in each state of the FSM is given. The range of parameter values for each state is set according to the analysis of the results obtained in the detection-accuracy experiments carried out modifying individually the different MoG parameters (see Sec. 5.1).

In the LPA_STATE, F_u is first decreased down to a minimum value F_{min} (typically, one every four frames). If the FSM remains in this state and $F_u = F_{min}$, then K is decreased down to a minimum value of 3. Finally, if the FSM still stays in this state and the buffer occupancy keeps on increasing, then the B_{mode} option is enabled.

In the HPA_STATE, the algorithm parameters are tuned in order to obtain a more accurate detection while keeping the buffer occupancy close to B_{ocp}. Initially for the whole image, the B_{mode} option is disabled first; then, K is gradually increased up to K_{max} and, finally, F_u is increased. During this parameter modification, if the buffer occupancy evolution shows up a risk for the real-time performance of the algorithm, then the highest detection quality is kept only for those image areas that include moving objects (see ROI in Fig. 1). For the rest of the image areas, first F_u and then K are decreased. If necessary, the B_{mode} option is finally enabled.

In the HPA_OVER the parameters are modified to reduce the computational requirements even for pixels corresponding to moving objects. Therefore, for all the pixels, first K is decreased up to a minimum value of 2. Then, F_u is decreased down to F_{min}. Finally, if still the FSM state is HPA_OVER , B_{mode} is enabled.

When GLA is enabled, which means that FSM is in the MAX_SIZE state, the process that is using more computational power is selected and forced to reduce its computational requirements. First of all, the parameters are modified in the regions that do not contain moving objects as for the case of HPA_STATE; then if this approach is not sufficient, the same parameters modification strategy of the HPA_OVER state is applied.

4 Multicamera System Software Architecture

The proposed adaptive strategy has been implemented in a multicamera system composed by a central node connected with several cameras. We have developed a modular and scalable software architecture for on-line multicamera video processing (see Ref. 32 for more details). The software architecture is scalable because it can efficiently manage multiple data streams, adding small overhead and saving useful computational resources. It is modular because it is composed by functional blocks that operate independently in different phases of the overall processing chain. The architecture is hardware independent: it can be easily ported on different host machines and is not related to any particular model or type of camera. It is also flexible from the point of view of the data; in fact, all the modules are automatically adapted to different image formats or sizes. Moreover, the architecture is designed to operate in off-the-shelf machines with commercial operating systems (no real-time operating systems are required) and without the use of any dedicated hardware. The software architecture is fully developed in C++ and the management of image data and the processing tasks is based on OpenCV libraries[33] (version 2.1). The architecture is composed of two main modules: a processing unit (PU) and a controller unit (CU).

4.1 *Processing Unit*

The PU is a scalable module that can be connected with several cameras; it is in charge to manage the incoming data streams and process them, and it is constituted by three functional blocks as shown in Fig. 3: the acquisition module (AM), the image buffer manager (IBM), and the processing module (PM). The AM continuously acquires images that are temporally stored in the image buffer. The PM reads the acquired images from the buffer and processes them by applying the MoG algorithm. Basically AM is in charge of continuously acquiring the images and delivering them to the IBM. The IBM is the module one that manages the buffer (of fixed size). It takes care of the buffer accesses and manages undesired situations, such as buffer overflows. These modules operate independently, and this aspect confers a great modularity to the global software architecture.

The AM allows acquiring images from the devices and delivering them to the IBM. It initializes the acquisition tasks and then starts a continuous process that is activated when a new image is available. When a new image is acquired, it is labeled with a time stamp; the data are converted to an OpenCV data structure, and then the images are delivered to the IBM.

The image buffer is used to compensate the different rates with which the flow of data is acquired by the AM and processed by the PM. In fact, whereas the acquisition rate remains constant, the processing rate can present deviations from a fixed value. This aspect is particularly relevant in our application because it is based on general purpose processors, where the available computational resources must be shared among different applications. The IBM manages the accesses to the buffer; in fact, the buffer access is thread safe because a semaphore system guarantees that only one process can write

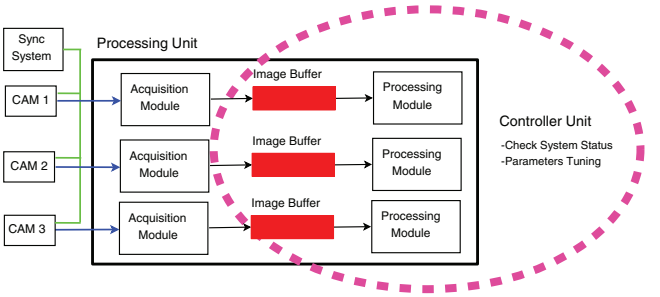

Fig. 3 Scheme of the multicamera system.

on the buffer (the AM) and one process can read an image from it (the PM). A specific policy to handle buffer overflow is defined in this module: it is possible to eliminate the oldest frame or to skip the newest one. The IBM communicates to the central unit the presence of new frames and the occurrence of the overflow condition.

The PM has one main task: continuously reading images from the buffer and processing them. It is the most customizable module of the entire software architecture and can be constituted by any processing task, including visualization or image storage. In this application, the PM implements the MoG algorithm. It communicates with the controller unit, which is in charge of modifying the parameters of the MoG as a function of the overall system state

4.2 *Controller Unit*

The controller unit initializes, monitors, and closes all the processes. It starts and stops the acquisition, gets the camera settings (i.e., frame size, gain, etc.), and using this information, it initializes the PU. It is also in charge of correctly closing all modules and shutting down the system. During the acquisition, it gathers the information about all the modules of the system, such as acquisition frame rate, processing time, the state of the image buffers, and the scene activity. The controller unit is also in charge of dynamically adapting the background subtraction parameters following the adaptation strategy presented in Sec. 4.1.

5 Results

We have tested the proposed adaptation strategy in a multicamera system composed of three cameras and one workstation as a central processing node. Camera models are JAI CB-080 GE. It is a Giga Ethernet camera model that offers a maximum resolution of 1032 × 778 pixels with a maximum frame rate of 30 fps. The data provided by the camera are 8-bit Bayer format. The workstation is equipped with two quad core processors of the Intel Xeon family, 16 GB of RAM, and fast HD (15 krpm) in RAID-0 configuration.

It is important to highlight that this kind of multicamera application requires a high accuracy of camera-acquisition synchronization to guarantee a correct fusion of the 2-D information extracted by the object-detection modules. For this reason, we have developed an external triggering system that allows minimizing the synchronization error. In particular, the cameras have been connected in a master–slave configuration, where the timing circuit of the master camera is used to generate the trigger of the system. Image acquisition for all cameras, master included, is governed by the generated trigger signal. The symmetry of the connection circuit guarantees that all the cameras are tightly synchronized with a synchronization error on the order of microseconds. This error value guarantees a very accurate synchronization in our indoor application, where the cameras typically acquire images at 25 fps (40 ms) and the shutter time of the camera is generally set at 10 ms. By considering these settings, it is clear that a synchronization error of few microseconds does not affect the overall performance of the system much; in fact, even if the cameras start the image acquisition (open shutter) with a misalignment of microseconds, they are in practice acquiring information about the same scene.

The presented results have been obtained using, as benchmark, three different data sets: the first one is composed by sequences acquired by our multicamera system (we call this database D1); the second one is composed by sequences from the Performance Evaluation of Tracking and Surveillance 2006 data set, that is freely available on the Internet[34] (henceforth we refer to this database as D2); and the third one (D3) is composed by three sequences, provided with a hand-labeled ground truth, of the database presented in Ref. 35, which is freely available on the Internet.[36] D2 has been selected because it provides medium-high resolution images of a multicamera system in an indoor environment—characteristics that are typical of a multicamera video surveillance application. On the contrary, D3 is composed of three sequences that present different scenarios with respect to D1 and D2; in fact, we select two outdoor scenarios (called, respectively, Campus and Fountain) and one indoor scenario with challenging lighting conditions (Lobby sequence). Moreover, each sequence of D3 is provided with a set of 20 hand-labeled ground-truth frames that we have extended to 60 ground-truth frames; in this way, it has been possible to provide an objective evaluation of the performance of the proposed approach.

As a measure of algorithm performance, we compare the following values: False positive (FP), that is the fraction of the background pixels that are marked as foreground; false negative (FN), the fraction of foreground pixels that are marked as background; and the total error (TE), the total number of misclassified pixels (normalized with respect to the image size). Moreover, we consider also the similarity measure S introduced in Ref. 35, which fuses into one metric the FP and FN information, and is defined as follows:

$$S(A, B) = \frac{A \cap B}{A \cup B}, \tag{13}$$

where A is a detected region and B the corresponding ground-truth region. The similarity measure S is a nonlinear measure that is very close to 1 if A and B are very similar and close to 0 when the two regions are completely different.

The use of several databases and the application of two different performance metrics helps one to undertake a thorough evaluation of the proposed approach under different conditions: variable illumination (indoor/outdoor environments), different moving object sizes (cars, humans) that lead to different activity levels, and challenging background conditions (strong moving backgrounds). Moreover, we individually test different aspects of our approach to demonstrate its strengths and capabilities. In Sec. 5.2, we investigate how the modification of individual MoG parameters affects the original MoG algorithm performance, and how the proposed adaptive strategy based on the tracked ROIs achieves a very good trade-off between detection error and algorithm speed up. Section 5.2 presents the advantages of introducing LLA in real-time scenarios; in particular, it is shown how the system is able to adapt the MoG to various processing loads obtaining detections with different accuracy levels. The avoidance of frame skipping, guaranteed by the LLA, improves the results obtained using only the ROI adaptation approaches. Finally, in Sec. 5.3, the results obtained by using GLA in a multicamera environment are presented.

5.1 *Region-of-Interest Parameter Adaptation*

The results obtained with the ROI adaptive strategy proposed in this paper have been compared to those obtained using the MoG algorithm without any parameter adaptation (MoG_{opt})

and with the MoG algorithm varying individually K, F_u, and B_{mode} on all the image areas. In this section, we use two data sets: D1 and D3. The disadvantage of using D1 is that no hand-labeled ground truth is available. For this reason, the foreground detected objects obtained with MoG_{opt} have been used as ground truth to test the detection accuracy of the other algorithms. It is worth noting that although this comparison cannot give an absolute measure of the algorithm performance because the reference is itself affected by errors, it does provide a relative measure of detection accuracy. Moreover, we use the analysis of these results to motivate the parameter adaptation policy, as presented in Sec. 3, in terms of their adequate values and the order in which they must be modified. D3 is a subset of the videos presented in Ref. 35, which has been widely used in literature. As previously mentioned, it has been selected because it provides detection mask ground truth and contains examples of different environments. The *Campus* and *Fountain* videos correspond to outdoor scenarios that pose challenging detection conditions, such as nonstatic backgrounds (waving trees and moving water). The third video, *Lobby*, is an indoor scenario where sudden illumination changes occur (light switched off and on). D3 is composed of low-resolution videos (160×128 pixels).

The objective of the first set of tests is to evaluate the impact introduced using the proposed adaptation strategy but modifying only one parameter at time. For this reason, this simulation is performed processing one frame at time without real-time constraints, thus, guaranteeing that no errors associated to possible frame-skipping conditions affect the statistics. In fact, the frame-skipping condition would affect the error statistics, producing a substantial contribution to the error for each discarded frame (i.e., considering an empty foreground mask frame produces the highest FN). Moreover, frame skipping would progressively degrade the background model quality. An estimation of the processing speed of each algorithm is given by the average frame rate (AFR). It is worth noting that for each sequence, the same MoG parameters (such as α, T, weight initialization, etc.) and the same postprocessing routines (basically, morphological closing operations) have been used for all the approaches.

Table 1 reports the performance of the MoG algorithm on dataset D1 varying F_u (reported in the first column). The first row ($F_u = 1$) represents MoG_{opt}, and the last one reports the results of the ROI-based parameter adaptation varying

Table 1 Detection accuracy and AFR obtained varying F_u. Data set *D1*.

F_u	TE	FN	FP	S	AFR
1					13.73
2	1.08	25.9	0.13	0.72	18.90
4	1.34	31.15	0.20	0.63	23.72
6	1.45	33.15	0.24	0.62	25.60
8	1.51	34.33	0.26	0.62	26.58
10	1.53	35.07	0.25	0.61	27.17
varF_u	0.66	12.68	0.20	0.85	24

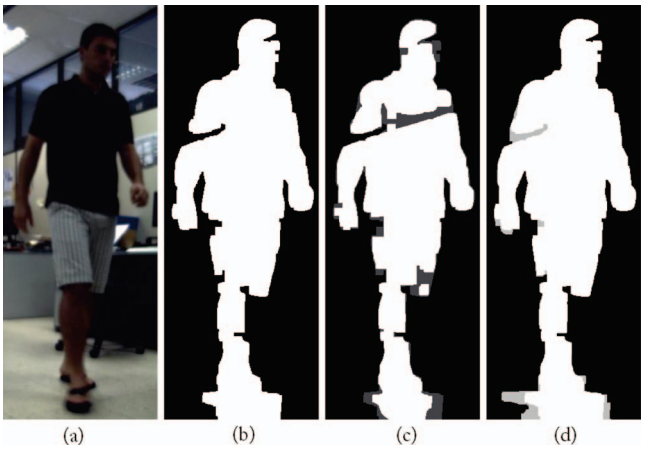

Fig. 4 (a) Data set D1 frame 497 detail, (b) object detected with MoG_{opt}, (c) object detected with $F_u = 4$, and (d) object detected with $varF_u$.

F_u (varF_u). As can be noted, varF_u guarantees the lowest TE because it allows one to considerably reduce the FN rate and then to increase the detection accuracy. The FP rate is low (0.2, as in the case of $F_u = 4$), and the achieved AFR (24 fps) is satisfactory for many real-time applications. For example, the multicamera application used for the immersive 3-D system,[5] presented in Sec. 1, has been successfully tested with a frame rate between 15 and 20 fps, whereas sport game monitoring applications vary from 14 fps in Ref. 7 to 25 fps.[37] As can be noted, the obtained value of S is the highest, confirming the trend identified by the other indicators.

In Fig. 4, detection results using different configurations of F_u are presented. To highlight the differences between the foreground masks obtained with MoG_{opt} ($F_u = 1$ and the others approaches, FP pixels are painted in light gray, FN pixels in dark gray and correctly classified ones in white. (This color labeling is used also for Figs. 5, 6, and 9). As can be noted in Fig. 4(d), the precision accuracy obtained with varF_u is higher than that obtained with MoG_{opt} [Fig. 4(b)] and clearly over performs the case of $F_u = 4$ [Fig. 4(c)], which presents a high level of FN. Moreover, varF_u guarantees the same AFR as $F_u = 4$ (see Table 1). As can be noted from Fig. 4(c), the accuracy of the silhouette detected with varF_u is very high, even helping to remove some detection errors, such as, for example, the discontinuities in the object's left part (that are marked as FP).

Table 2 reports the performance of the MoG algorithm with data set D1 for different values of K (reported in the first

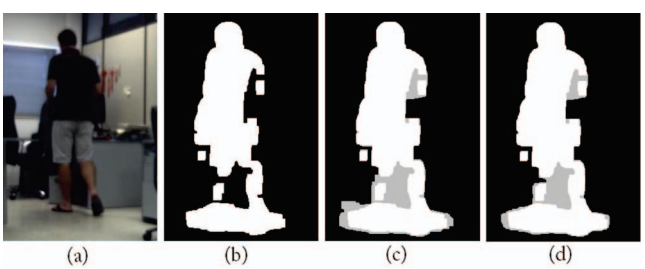

Fig. 5 (a) Data set D1 frame 283 detail, (b) object detected with MoG_{opt}, (c) object detected with $K = 2$, and (d) object detected with varK.

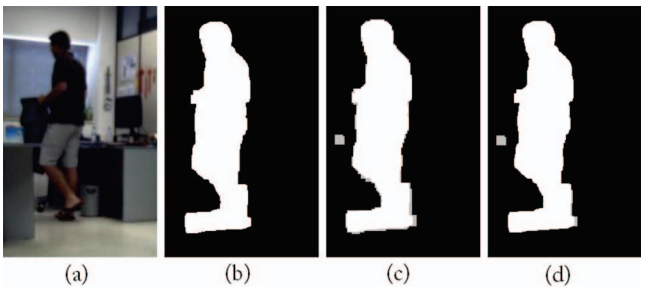

Fig. 6 (a) Data set D1 frame 303 detail, (b) object detected with MoG_{opt}, (c) object detected with B_{mode}, and (d) object detected with $varB_{mode}$.

column). The first row ($K = 5$) corresponds to MoG_{opt}, and the last one reports the results of the ROI adaptive strategy varying only K (see Sec. 3.2).

The variation of K according to the ROI information (var$K$ in Table 2) allows a good increase of the AFR, similar to the one obtained with $K = 2$, but improving its TE and S values. Figure 5(a) shows a detail of frame 283. As can be noted, the precision accuracy obtained with var$K$ [Fig. 5(d)] is similar to the one obtained with MoG_{opt} [Fig. 5(b)]. Moreover, the detected object with var$K$ fills the small hole near the right arm present in the reference-detected object. The filled space between the legs does not have to be considered a relevant error because it does not affect the object silhouette and makes it more compact as required in video-surveillance applications.

Table 3 reports the performance of the MoG algorithm with B_{mode} enabled (2×2 pixel blocks) for the D1 data set. The first row (B_{mode} disabled) represents MoG_{opt}, and the last row ($VarB_{mode}$) reports the results of the ROI parameter-adaptation strategy varying only B_{mode}. It is worth noting that $VarB_{mode}$ not only obtains a similar AFR value to that with B_{mode} enabled for the whole image (second row), but also significantly reduces FN, thus resulting in much more accurate detected object silhouettes, as shown also in Fig. 6. The precision accuracy obtained with $VarB_{mode}$ [Fig. 6(d)] is similar to the result obtained with MoG_{opt} [Fig. 6(b)]. Moreover, the detected object silhouette in Fig. 6(d) does not show the blocking effects that appear when B_{mode} is enabled for all the images [Fig. 6(c)]. In fact, the silhouette obtained with B_{mode} enabled contains several FP pixels (shown in light gray) which form the blocking effects.

Table 2 Detection accuracy and AFR obtained varying K. Data set D1.

K	TE	FN	FP	S	AFR
5					13.73
4	0.12	0.97	0.09	0.94	15.06
3	0.51	2.32	0.44	0.78	16.27
2	0.82	4.75	0.67	0.70	17.32
1	2.80	70.35	0.23	0.22	18.15
var$K$	0.76	6.68	0.53	0.72	17.18

Table 3 Detection accuracy and AFR obtained varying B_{mode}. Data set D1.

B_{mode}	TE	FN	FP	S	AFR
B_{mode} disabled					13.73
B_{mode} enabled	0.85	13.7	0.36	0.72	29.5
$VarB_{mode}$	0.92	4.69	0.77	0.69	28.3

To complete the evaluation of the error introduced by the proposed ROI parameter-adaptation strategy with respect to the MoG_{opt} algorithm, we test the different ROI approaches (var$K$, varF_u and varB_{mode}) on data set D3 (for which an extended hand-labeled detection ground truth is available). For each sequence, the same MoG parameters (such as α, T, weight initialization, etc.) and the same postprocessing routines (basically, morphological closing operations) are used. In Table 4, the results of the performance of the different ROI approaches and the MoG_{opt} with respect to the available ground truth are reported. For the *Campus* sequence, the TE values obtained with ROI parameter-adaptation strategies are low and close to the values obtained by MoG_{opt}; this clearly confirms previous results where the relative error introduced by the ROI approach with respect to the MoG_{opt} was low. Also, the comparison of the similarity measure S confirms that the foreground regions obtained with the ROI adaptation strategies have a similar quality to those generated by MoG_{opt}. The AFR column shows that, also in this case, the ROI approaches lead to a higher AFR. However, in this case the increment obtained with var$K$ and var$F$ is smaller than the one obtained in the previous experiments (see Tables 1 and 2). This lower AFR gain is due to the particular nature of the *Campus* sequence. It presents a highly moving background (weaving trees) that result in the detection of

Table 4 Detection accuracy and AFR of MoG_{opt} and ROI parameter adaptation. Data set D3.

Sequence	Mode	TE	FN	FP	S	AFR
Campus	MoG_{opt}	1.45	6.68	1.31	0.57	324.98
	var$K$	1.62	5.71	1.51	0.56	344.82
	varF_u	2.38	3.20	2.36	0.46	357.14
	varB_{mode}	1.33	9.59	1.11	0.52	623.61
Fountain	MoG_{opt}	1.86	16.13	1.35	0.64	501.26
	var$K$	1.91	10.76	1.59	0.63	599.09
	varF_u	1.93	10.50	1.62	0.62	838.39
	varB_{mode}	2.26	10.62	1.96	0.58	906.14
Lobby	MoG_{opt}	1.40	20.52	0.95	0.54	494.66
	var$K$	3.39	32.13	2.70	0.48	595.41
	varF_u	2.32	30.61	1.65	0.49	854.30
	varB_{mode}	21.74	10.94	22.00	0.30	692.38

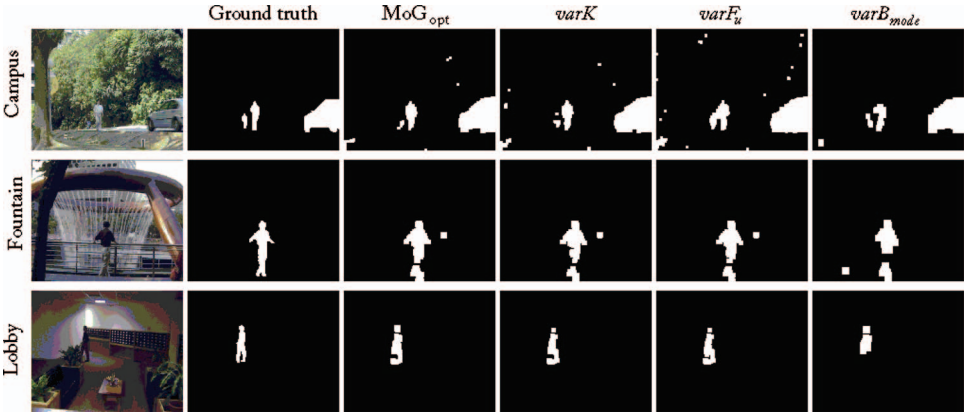

Fig. 7 Foreground masks results for *Campus, Fountain, and Lobby videos.*

several objects, simultaneously. Consequently, the number of image pixels for which it is possible to modify the MoG parameters is reduced, thus limiting the computational improvement of these adaptive approaches. Also in the case of the *Fountain* sequence, the TE and *S* values obtained with the ROI approaches are very close to the one obtained by MoG_{opt}; an increase of FP is observed, but it is balanced by a reduction of FN. Higher AFR values are also achieved with respect to the MoG_{opt} algorithm. The same results have been observed when analyzing the *Lobby* sequence, except for the $\text{var}B_{\text{mode}}$. In fact, by using this approach, a high TE ($\approx$ %20) is caused by the FP. These data are confirmed also by a low value for *S* (0.3). This effect is caused by the fast illumination changes in the *Lobby* sequence that affect, particulary, the $\text{var}B_{\text{mode}}$ approach.

In Fig. 7, example frames of the sequences and the corresponding foreground masks are presented. In the row, from top to bottom, the frames relative to *Campus*, *Fountain*, and *Lobby* are shown; from left to right, the original sequence frame, the ground-truth foreground mask, and the resulting masks obtained with MoG_{opt}, var*K*, $\text{var}F_{\text{u}}$, and $\text{var}B_{\text{mode}}$ are presented. As expected, foreground masks obtained with the ROI adaptive strategies are very similar to the MoG_{opt} ones except for $\text{var}B_{\text{mode}}$ and, particularly, for the *Lobby* sequence: the lowest part of the body is not correctly detected. Although some noisy detections are also present in *Campus*, due to the strongly moving background, they can be easily eliminated by fitering the objects with a very small area. On the other hand, it is relevant the robustness of the adaptive approaches in the presence of softly moving backgrounds, as the one in the *Fountain* sequence: variations produced by the moving water are adequately modeled as background, keeping one of the main strengths of MoG background modeling strategies: the capability to model quasi-static backgrounds.

These results demonstrate that the tracked ROI-based parameter adaptation guarantees accurate detection and a considerable speed up of the algorithm performances with respect to parameter modifications applied to the whole image. However, a more flexible strategy is still necessary to modify the MoG parameters as a function of the available processing resources and fulfill the real-time constraints.

5.2 *Local-Level Adaptation*

In this section, the functionalities of LLA (see Sec. 3) are presented with different examples of the three data sets. In particular, we investigate how the proposed system automatically adapts the MoG parameters to keep real-time performance while guaranteeing high detection accuracy. Moreover, we test LLA with sequences at different frame rates to show the effect on the detection accuracy. It is worth noting that LLA improves the detection performance with respect to simple ROI adaptation approaches because frame skipping is avoided.

The first example shows the results for data set D1 when the video data stream has to be processed at 20 fps: the performance of the ROI adaptation strategies is compared to the proposed LLA approach. In Fig. 8, the pixels activity evolution (black curve) for one camera is reported, together with the evolution of the buffer occupancy (light gray curve). Dashed lines represent the activity threshold value to move from LPA to HPA, the target buffer occupancy for the HPA_STATE (B_{ocp}) and the buffer occupancy threshold to switch to the HPA_OVER state (B_{ovfl}).

This example shows how the FSM moves across all the states of LLA. Let us analyze, in detail, the curves in Fig. 8. At the beginning of the acquisition, there are no detected objects. Therefore, the FSM is in the LPA_STATE and the buffer occupancy is close to zero. After 10 s, a moving object enters in the scene and the pixel activity quickly exceeds T_{LPA}. As a consequence, the FSM moves to the HPA_STATE and the parameters are tuned in order to keep the buffer occupancy close to B_{ocp} and to guarantee a good detection accuracy. However, the pixel activity is high and the parameters adaptation strat-

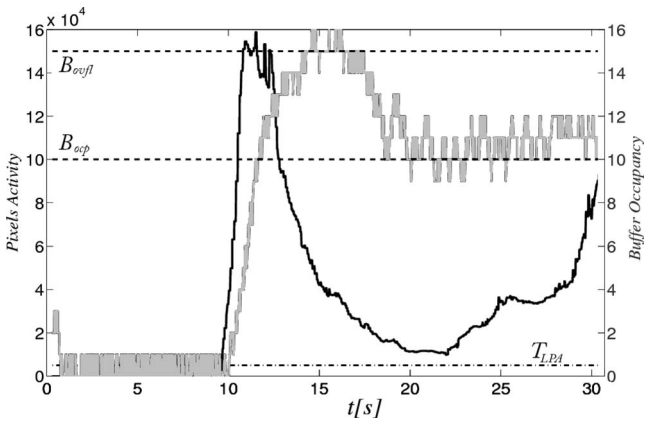

Fig. 8 Pixel activity and buffer occupancy evolution. Data set D1.

Table 5 Detection accuracy obtained with LLA. Dataset D1.

MODE	TE	FN	FP	S
var$K$	1.91	20.49	0.46	0.69
varF_u	2.79	65.20	0.03	0.2
varB_{mode}	2.42	1.66	2.44	0.68
LLA	0.84	8.72	0.54	0.72

egy associated to the HPA_STATE is not sufficient to keep the buffer occupancy close to B_{ocp}. In fact, the buffer occupancy rapidly reaches the upper bound (B_{ovfl}), thus forcing the FSM to move to the HPA_OVER. In this state, algorithm parameters are tuned significantly to force reducing the buffer occupancy to prevent frame skipping. The result is that buffer occupancy slowly decreases and remains close to B_{ocp}.

Detailed results of the detection accuracy are shown in Table 5 for the different ROI-based adaptation and LLA strategies. We must keep in mind that, in this simulation, we are considering a real-time application. Therefore, the results of the ROI adaptation strategies are expected to be significantly worse than those in Sec. 5.1 because they are affected by frame skipping. In particular, when a frame is discarded and not processed, we compare the previous available foreground mask to the reference ground truth. It is clear that the higher the number of discarded frames there are; the greater the error introduced is. On the contrary, the LLA algorithm, through the combined modification of different algorithm parameters, adapts the required processing resources to the real-time constraints, thus avoiding frame skipping and keeping a very good detection accuracy. As can be noted, LLA guarantees the lowest value of TE and the highest value of S.

In Fig. 9, a foreground object detected by the adaptive algorithm when the FSM is in the HPA_STATE is shown. In particular Fig. 9(a) reports the details of the acquired original frame, Fig. 9(b) shows the foreground object detected with MoG_{opt}, and Fig. 9(c) shows the foreground object detected with the adaptive strategy, which reduces the computational requirements down to the required values to keep the real-time operation. As can be noted, also in this case, when full parameter adaptation is required, highly accurate detection results are obtained.

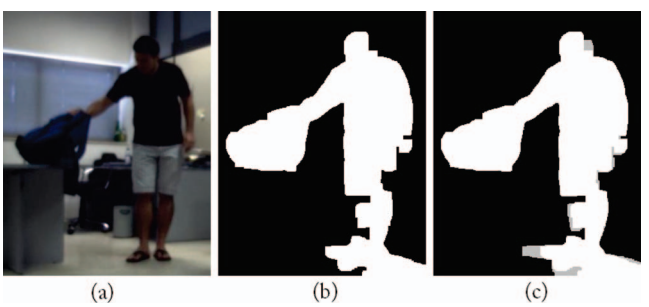

Fig. 9 (a) Data set D1 frame 421 detail, (b) object detected with MoG_{opt}, and (c) object detected with the adaptive strategy.

In the second example, D2 is used. This example shows how the LLA adequately adapts the MoG algorithm to different real-time conditions: in one case, the system is fed with a sequence at 21 fps and, in the other case, the sequence is provided at 31 fps. Results are reported in Fig. 10 where the evolution of the pixels activity (in black) and the buffer occupancy (in light gray) are shown for both real-time conditions. As can be noted, for the sequence at 21 fps [Fig. 10(a)] the activity at the beginning of the sequence is low and the buffer occupancy is kept low. A sudden change in the pixel activity (which appears between 40 and 50 s) causes an increase of the computational time and, consequently, an increase of the buffer occupancy. The proposed strategy adapts the MoG parameters dynamically to smoothly reach the target value of B_{ocp}, providing accurate detection, and it controls the computational requirements so that buffer occupancy is maintained close to this B_{ocp} value. On the contrary, in the case of the sequence provided at 31 fps [Fig. 10(b)], the initial pixel activity is high enough to lead the buffer occupancy to B_{ocp}. It is worth noting that scales for the pixel activity and buffer occupancy axes for this plot are larger than the one that corresponds to the sequence at 21 fps [Fig. 10(a)]. The proposed strategy perfectly adapts the parameters to keep the buffer occupancy close to this value. Between 30 and 40 s, the activity increases sharply. At the beginning, parameters are adapted to try to compensate for this increase (smooth increase of buffer occupancy). However, the continuous increase in activity leads buffer occupancy to reach B_{ovfl}, thus making the FSM to move from the HPA_STATE to the HPA_OVER. The adaptation policy defined for this state, imposing further adaptations of the parameters to prevent from critical situations such as frame skipping, works perfectly, buffer occupancy is decreased down to B_{ocup}, detection accuracy is increased, and the defined adaptation strategy allows keeping the computational requirements bounded and controlled.

Detection accuracy results with respect to the MoG_{opt} algorithm using the proposed LLA are reported in Table 6. As expected, better results are obtained for the sequence with less real-time constraints (21 fps). It shows a smaller TE value and higher similarity measure S value. However, the average accuracy of the detections is kept high even when harder real-time restrictions appear (31 fps).

For the third example, we use the *Fountain* sequence of the D3 data set, which provides ground-truth detections. The use of the ground-truth detections as reference allows providing objective measurements of the detection accuracy obtained with the proposed LLA under real-time constraints. As previously mentioned, the image size of the videos in the D3 data set is very small and using them directly to fed the processing system does not allow one to generate realistic real-time con-

Table 6 Detection accuracy obtained with LLA at different frame rates. Data set D2.

Frame Rate	TE	FN	FP	S
LLA (21 fps)	1.4731	4.8648	1.3340	0.67
LLA (31 fps)	1.5054	6.4745	1.3017	0.64

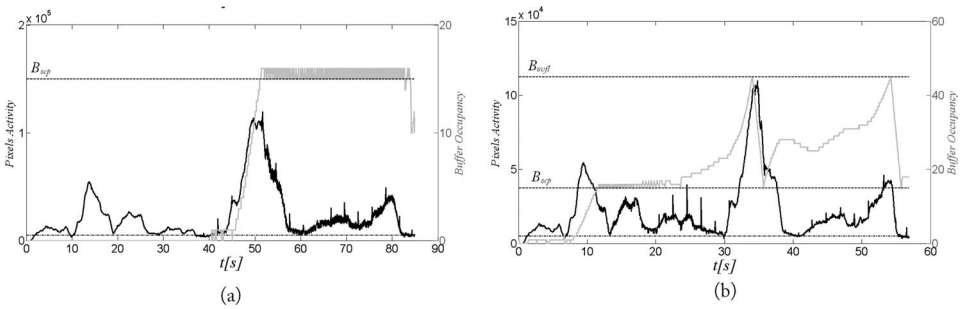

Fig. 10 Data set D2. Pixel activity and buffer occupancy evolution: (a) Sequence at 21 fps. (b) Sequence at 31 fps.

straints. For this reason, we create a new sequence joining 30 low-resolution original videos arranged as a matrix with five rows and six columns. The images of this new sequence are 960×640 pixels, and they are provided to the system at 20 fps.

Figure 11(a) reports the evolution of the pixels activity (in black) and the buffer occupancy (in light green) for the sequence at a frame rate of 20 fps. As can be noted, the buffer occupancy trend is correlated with the pixel-activity values. This means that the FSM moves from the LPA_STATE to the HPA_STATE, hence trying to guarantee a good detection accuracy, and parameters adaptation generates smooth buffer occupancy profiles. It is particularly relevant that the fast adaptation of the system to LPA states: the lack of activity does not require highly accurate background models, thus leaving buffer room for smooth adaptation through locally fast parameters tuning—improving the detection accuracy—when HPA occurs. In Fig. 11(b) a zoom of the two curves in (a) is shown; additionally, the S values obtained using the MoG_{opt} strategy (black diamonds-tested without real time constraints) and the proposed *LLA* approach (gray squares - tested with real time constraints) are displayed. As can be observed, the *S* values obtained with LLA, keeping the real-time constraints, are very close to those that are theoretically optimum MoG_{opt} (that cannot fulfill real-time constraints). These results show that, even if during the processing, the LLA modifies the parameters of the background modeling strategy to reduce the accuracy (i.e., using few Gaussians per pixel when the activity level is low), it demonstrates to rapidly modify the parameters when required (i.e., new objects entering in the scene) to ensure a high detection accuracy. In Table 7, are reported the performance of LLA and MoG_{opt} (this strategy tested without real-time constraints) with respect to the ground truth detections. The values of TE and *S* observed for LLA are very close to the MoG_{opt}. The proposed strategy demonstrates an excellent capability to adapt to the available resources while keeping an adequate trade-off between detection quality and computational efficiency.

5.3 *Global-level adaptation*

In this section, the functionalities of GLA are demonstrated and the results with data sets D1 and D2 are presented.

In a first example, D1 is used, in particular, the three cameras of the system are acquiring images at a frame rate of 18 fps. As introduced previously, GLA is required when LLA is not sufficient to reduce the buffer occupancy in critical situations. In Fig. 12 the buffer occupancy for the three cameras is shown: as it can be noticed the buffer occupancy of camera 0 (light gray line) exceeds B_{ovfl}. The FSM is in the HPA_OVER state, and the algorithm parameters are sequentially tuned to try to reduce the buffer occupancy. However, buffer occupancy keeps increasing even when algorithm parameters have reached their minimum values. In these situations, GLA is activated to reduce the buffer occupancy and to avoid frame skipping: the most consuming processes are selected and forced to reduce their processing requirements. In this example, GLA first modifies parameters of camera 1 and then modifies those of camera 2 so that the new available processing resources are used for camera 0, thus reducing its buffer occupancy. As it can be noticed in Fig. 12 the buffer occupancy of camera 1(black line) and camera 2 (dark gray line) are reduced too.

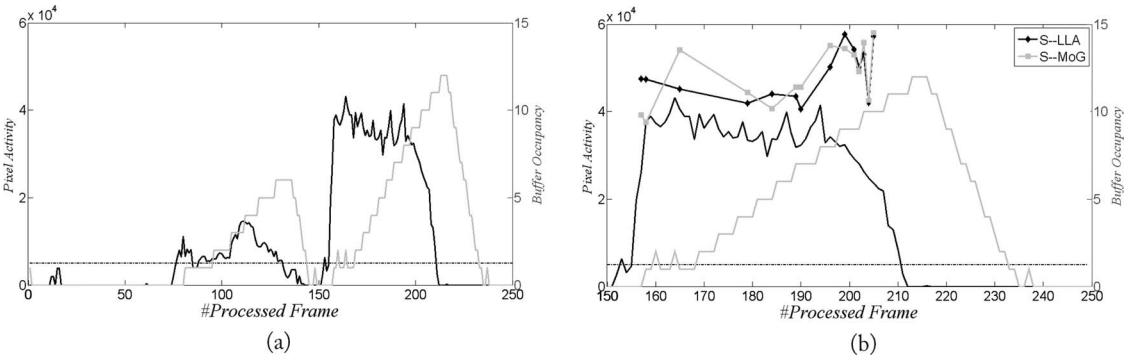

Fig. 11 Data set D3. (a) Pixel activity and buffer occupancy evolution and (b) Pixel activity and buffer occupancy evolution with S measure for LLA and MoG_{opt}.

Table 7 Detection accuracy obtained with LLA and with MoG_{opt}. Data set D3.

Mode	TE	FN	FP	*S*
MoG_{opt}	1.63	14.6	1.17	0.64
LLA (21 fps)	1.98	14.6	1.54	0.58

Figure 13 shows how the detection accuracy of cameras 1 and 2 is reduced to avoid the frame-skipping condition of camera 0. Figures 13(a)–(c) show the detected object in the three cameras before starting the GLA strategy. Parameter adaptation for camera 0 has reached minimum values (in HPA_OVER), introducing some distortion—blocking–in the detected objects silhouette, as shown in Fig. 13(a). Silhouettes corresponding to the same object detected in cameras 1 and 2 show higher accuracy as the adapted parameters correspond to the HPA_STATE. To prevent from frame skipping in camera 0, GLA is applied. As defined in the proposed strategy, parameters for cameras 1 and 2 are modified to force reduction of their computational requirements. Consequently, the accuracy of the detected object in camera 1 [Fig. 13(e)] and camera 2 [Fig. 13(f)] is slightly affected and the effect of the subblock processing can be noted. However, this approach allows reducing the global computational requirements of the application and avoids frame skipping in camera 0. It is worth noting that the detection accuracy of camera 0 remains the same before [Fig. 13(a)] and after [Fig. 13(d)] the application of GLA. In fact, the parameters of MoG for camera 0 cannot be further modified during GLA because the MoG algorithm is already in its minimal configuration (MoG_{min}).

In the second example, we show how the GLA strategy affects the detection accuracy on the D2 data set considering a two-camera system. In Figs. 14–16, the results of the adaptive strategy applied to this data set are reported. In particular, the evolution of the detection accuracy during the GLA phase is shown. Figure 14 shows the detected objects in the two views; in this case, the FSM is in HPA_STATE, only LLA is applied to control the algorithm parameters, and then highly accurate detections are obtained. However, later on in the sequence, the FSM reaches the MAX_SIZE state and the

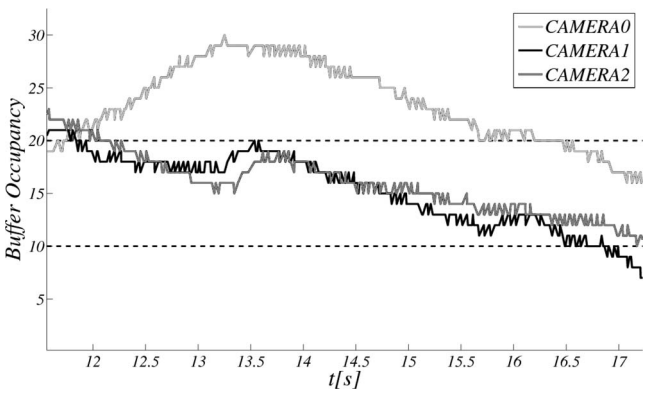

Fig. 12 Data set D1 with a three-camera system: Buffer occupancy evolution for camera 0 (light gray), camera 1 (black), and camera 2 (dark gray).

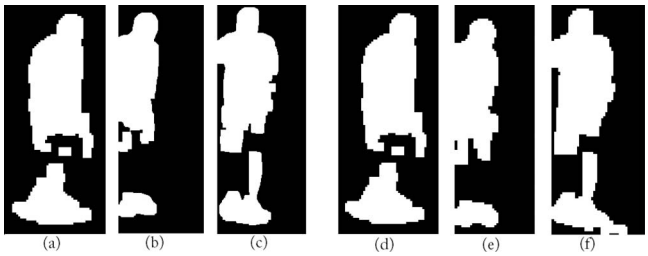

Fig. 13 Data set D1 with a three-camera system. Detail of a detected object: (a–c) before GLA; (e–f) after GLA.

GLA strategy is applied: B_{mode} is enabled, thus reducing the computational requirements and avoiding frame skipping but slightly affecting to the detection accuracy (see, in Fig. 15 the subblocks effect for some silhouettes). When the buffer occupancy level has reached acceptable values, the GLA strategy is disabled and parameters are adapted to provide higher quality detections (see Fig. 16).

6 Conclusions

Background modeling is a fundamental task in many computer-vision applications and becomes a time-critical task in a multicamera system, where a huge amount of data must be processed. In this paper, we have presented an innovative and efficient adaptive strategy for background modeling for real-time object detection in multicamera systems. The proposed approach is particularly attractive because it can dynamically adapt the algorithm parameters in order to respect the real-time constraints of the application and without a significant loss in detection accuracy. In particular, the proposed parameter adaptation policy is a multilevel strategy. It works at the local (camera) level by adapting the background model parameters independently for each processing task. If the changes done at local level are not sufficient, then the global-level adaptation strategy balances the processing load among all the tasks.

The proposed approach is highly innovative because it is more flexible and customizable than state-of-the-art solutions, which are too application specific, offers reduced adaptation capabilities, and whose applicability to multicamera systems is very limited. It has been successfully tested with different sequences, and the results show that the proposed strategy guarantees an excellent trade-off between detection accuracy and real-time processing capabilities in a multicamera environment. In particular, three different data sets have been considered in the evaluation process, including different image sizes, different environments (indoor and outdoor scenarios), and various real-time scenarios.

The proposed approach is currently being successfully used for object detection and tracking purposes for video surveillance in a multicamera system based on a centralized server. Its modular and adaptive structure makes it particularly attractive to seamlessly integrate additional features, such as shadows removal, consistent labeling for multiple objects tracking, and 3-D positioning. In this way, a complete adaptive 3-D positioning system for real-time video-surveillance applications is being built, where the different tasks are tuned according to the available processing resources to keep the real-time constraints.

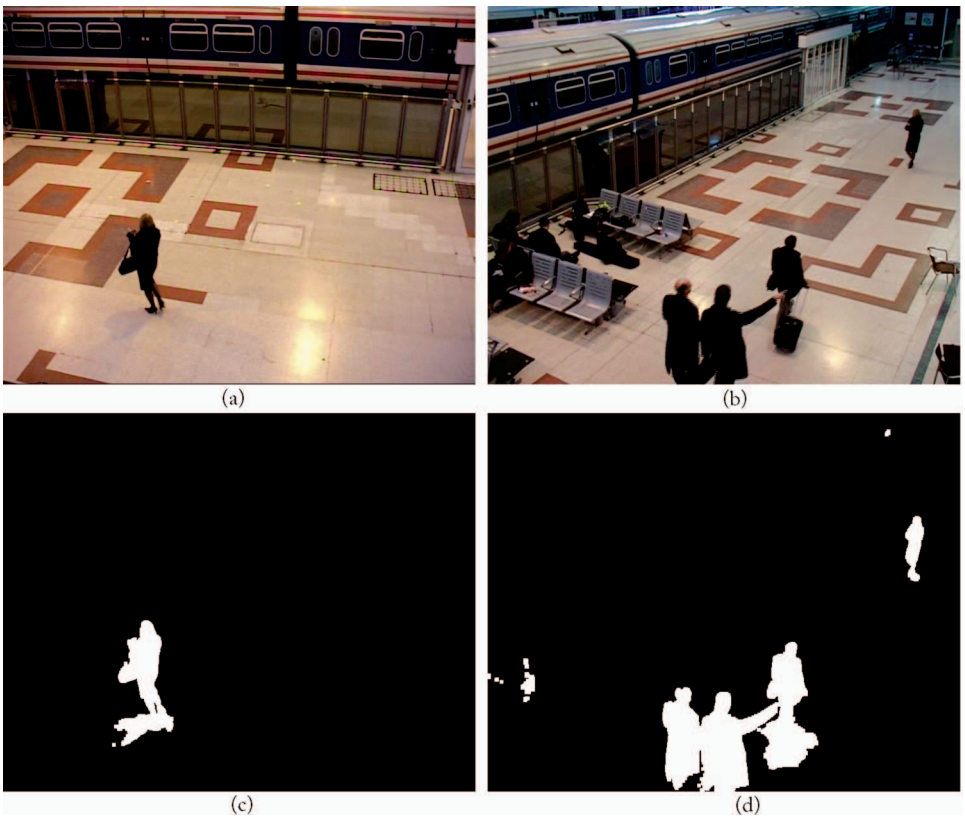

Fig. 14 Data set D2 with a two-camera system. Frames 121 corresponding to (a) camera 3, (b) camera 4; (c), (d) detected objects for both images during LLA.

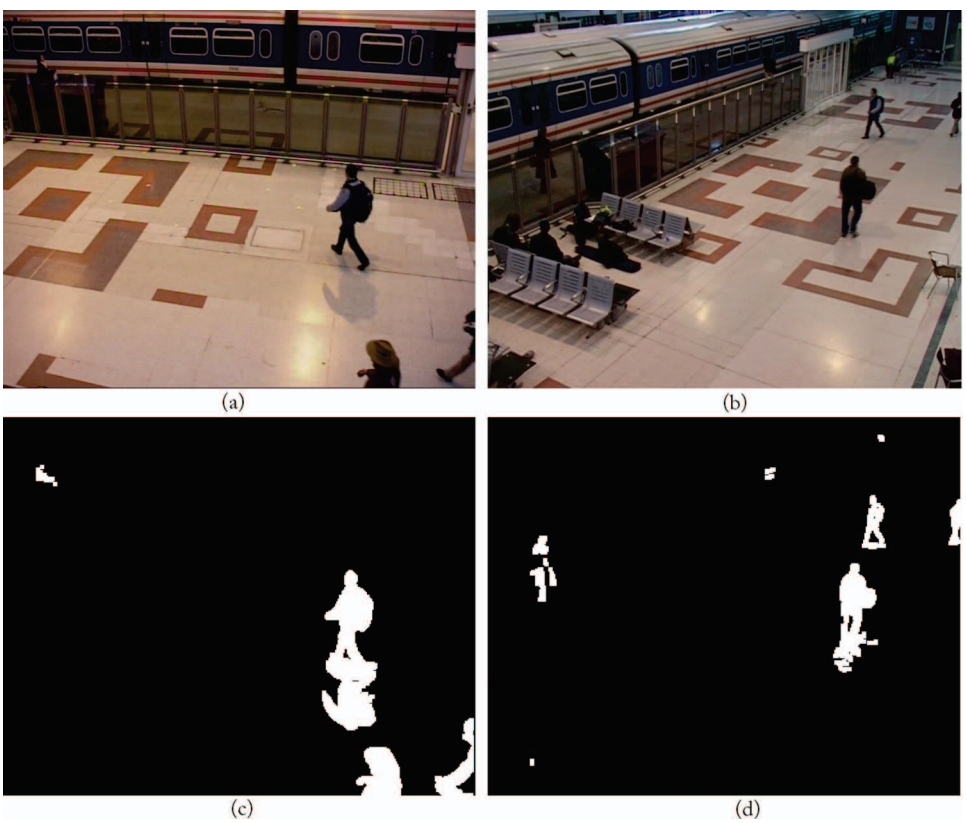

Fig. 15 Data set D2 with a two-camera system: (a), (b) Frames 991 and (c), (d) detected objects for both images during GLA.

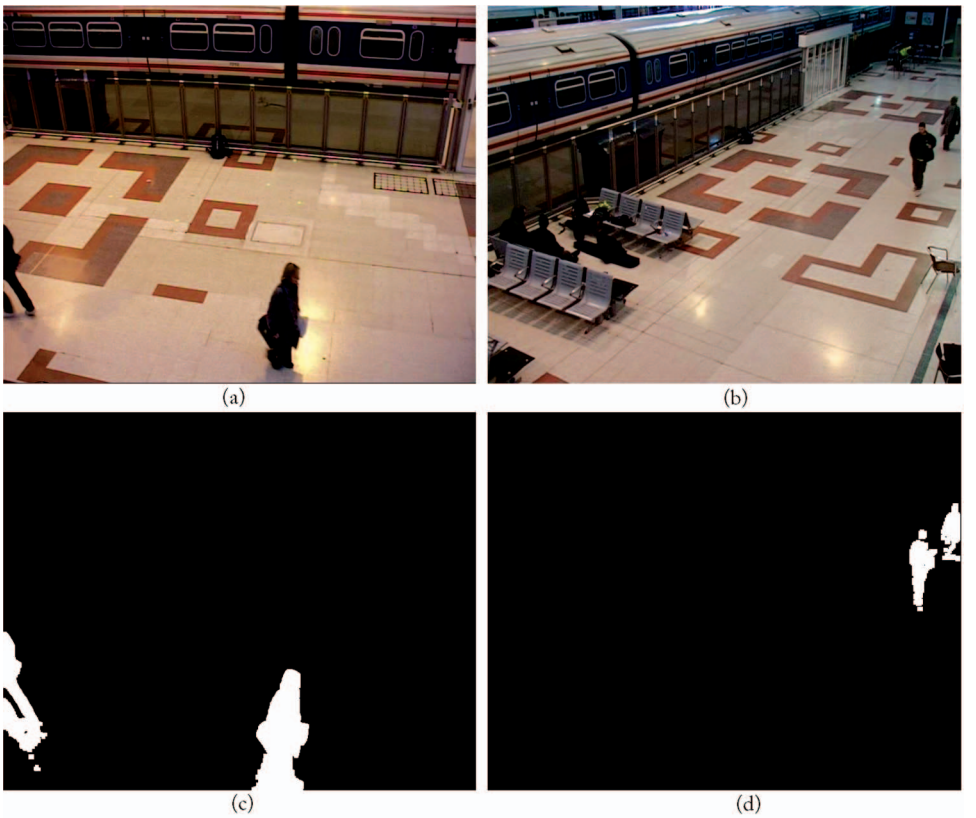

Fig. 16 Data set D2 with a two-camera system: (a), (b) Frames 2299 and (c), (d) detected objects for both images when back to LLA.

Acknowledgments

This work has been supported by the Ministerio de Ciencia e Innovación of the Spanish Government under Projects No. TEC2007-67764 (SmartVision) and No. TEC2010-20412 (Enhanced 3DTV). Massimo Camplani would like to acknowledge the European Union and the Universidad Politécnica de Madrid for supporting his reserach activities through the Marie Curie-Cofund research grant.

References

1. R. Mohedano, C. del Bianco, F. Jaureguizar, L. Salgado, and N. Garcia, "Robust 3D people tracking and positioning system in a semi-overlapped multicamera environment," in *Proc. of 15th IEEE Int. Conf. on Image Processing*, pp. 2656–2659 (2008).
2. T. T. Santos and C. H. Morimoto, "Multiple camera people detection and tracking using support integration," *Pattern Recogn. Lett.*, **32**, 47–55 (2011).
3. H. H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M. E. Goss, W. B. Culbertson, and T. Malzbender, "Understanding performance in coliseum, an immersive videoconferencing system," *ACM Trans. Multimedia Comput. Commun. Appl.* **1**, 190–210 (2005).
4. O. Schreer, I. Feldmann, N. Atzpadin, P. Eisert, P. Kauff, and H. Belt, "3D presence—a system concept for multi-user and multi-party immersive 3D videoconferencing," in *Proc. of 5th Eur. Conf. on Visual Media Production*, pp. 1–8 (2008).
5. B. Petit, J.-D. Lesage, E. Boyer, J.-S. Franco, and B. Raffin, "Remote and collaborative 3D interactions," in *Proc. of 3DTV Conf.: The True Vision—Capture, Transmission and Display of 3D Video*, pp. 1–4 (2009).
6. Y. Chai, D. Jang, K. Chang, and T. Kim, "Vision-based real-time game interface," in *Proc. of Int. IEEE Consumer Electronics Society Innovations Conf. Games*, pp. 43–46 (2009).
7. J. M. del Rincón, E. Herrero-Jaraba, J. R. Gómez, C. O.-U. nuela, C. Medrano, and M. A. M. nés Laborda, "Multicamera sport player tracking with bayesian estimation of measurements," *Opt. Eng.*, **48**, 047201 (2009).
8. M. Montañés Laborda, E. Torres Moreno, J. Martínez del Rincón, and J. Herrero Jaraba, "Real-time GPU color-based segmentation of football players," *J. Real-Time Image Process.*, 1–13 (2011).
9. C. H. Lin, M. Wolf, X. Koutsoukos, S. Neema, and J. Sztipanovits, "System and software architectures of distributed smart cameras," *ACM Trans. Embed. Comput. Syst.* **9**, 1–30 (2010).
10. N. Bellotto, E. Sommerlade, B. Benfold, C. Bibby, I. Reid, D. Roth, C. Fernández, L. V. Gool, and J. Gonzàlez, "A distributed camera system for multi-resolution surveillance," in *Proc. of 3rd ACM/IEEE Int. Conf. on Distributed Smart Cameras*, pp. 1–8 (2009).
11. J. Varona, J. Gonzàlez, I. Rius, and J. J. Villanueva, "Importance of detection for video surveillance applications," *Opt. Eng.*, **47**, 087201 (2008).
12. M. Cristani, M. Farenzena, D. Bloisi, and V. Murino, "Background subtraction for automated multisensor surveillance: a comprehensive review," *EURASIP J. Adv. Signal Process.*, **2010**, 1–24 (2010).
13. C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on*, Vol. 2, pp. 246–252 (1999).
14. T. Bouwmans, F. El Baf, and B. Vachon, "Background modeling using mixture of gaussians for foreground detection—a survey," *Recent Patents Comput. Sci.*, **1**, 219–237 (2008).
15. A. Shimada, D. Arita, and R. Taniguchi, "Dynamic control of adaptive mixture-of-Gaussians background model," presented at *IEEE Int. Conf. on Video and Signal Based Surveillance* (2006).
16. C. Cuevas, N. García, and L. Salgado, "A new strategy based on adaptive mixture of Gaussians for real-time moving objects segmentation," *Real-Time Image Process*. **6811**, 681111 (2008).
17. Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recogn. Lett.* **27**, 773–780 (2006).
18. J. Park, A. Tabb, and A. Kak, "Hierarchical data structure for real-time background subtraction," in *Proc of IEEE Int. Conf. on Image Processing*, pp. 1849 –1852 (2006).
19. D.-Y. Lee, J.-K. Ahn, and C.-S. Kim, "Fast background subtraction algorithm using two-level sampling and silhouette detection," in *Proc of IEEE International Conference on Image Processing*, pp. 3177 –3180 (2009).
20. J. K. Suhr, H. G. Jung, G. Li, and J. Kim, "Mixture of Gaussians-based background subtraction for bayer-pattern image sequences," *IEEE trans. Circuits Syst. Video Technol.*, **21**, 365–370 (2011).
21. T. Yang, Y. Zhang, M. Li, D. Shao, and X. Zhang, "A multicamera network system for markerless 3d human body voxel reconstruction," *Fifth International Conference on Image and Graphics*, 706–711 (2009).

22. W. P. Power and J. A. Schoonees, "Understanding background mixture models for foreground segmentation," *Imaging Vision Comput.* (2002).
23. F. Porikli and O. Tuzel, "Human body tracking by adaptive background models and mean-shift analysis," presented at *IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance*, (2003).
24. D. R. Magee, "Tracking multiple vehicles using foreground, background and motion models," *Image Vision Comput.*, **22**, 143–155 (2004).
25. P. Atrey, V. Kumar, A. Kumar, and M. Kankanhalli, "Experiential sampling based foreground/background segmentation for video surveillance," *IEEE Int. Conf. on Multimedia and Expo*, pp. 1809–1812 (2006).
26. M. Al Najjar, S. Ghosh, and M. Bayoumi, "A hybrid adaptive scheme based on selective gaussian modeling for real-time object detection," in *Proc. of IEEE Int. Sympo. on Circuits and Systems*, pp. 936–939 (2009).
27. R. Krishna, K. McCusker, and N. O'Connor, "Optimising resource allocation for background modeling using algorithm switching," in *Proc. of 2nd ACM/IEEE Int. Conf. on Distributed Smart Cameras*, pp. 1–7 (2008).
28. K. Appiah, A. Hunter, J. Owens, P. Aiken, and K. Lewis, "Autonomous real-time surveillance system with distributed IP cameras," *Third ACM/IEEE International Conference on Distributed Smart Cameras. ICDSC 2009.*, 1–8 (2009).
29. M. Poremba, Y. Xie, and M. Wolf, "Accelerating adaptive background subtraction with gpu and cbea architecture," in *Proc. of IEEE Workshop on Signal Processing Systems*, pp. 305–310 (2010).
30. P. Carr, "GPU accelerated multimodal background subtraction," in *Proc. of Digital Image Computing: Techniques and Applications*, pp. 279–286 (2008).
31. A. Yilmaz, O. Javed, and M. Shah, "Object tracking: a survey," *ACM Comput. Surv.* **38**, 13 (2006).
32. M. Camplani and L. Salgado, "Scalable software architecture for online multicamera video processing," in *Proc. of SPIE* **7871**, 787106 (2011).
33. G. Bradski, The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).
34. PETS 2006 Benchmark Data, March 30, 2011.
35. L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian, "Statistical modeling of complex backgrounds for foreground object detection," *IEEE Trans. Image Process.*, **13**, 1459–1472 (2004).
36. Complex Background Modeling Database, July 1, 2011.
37. M. Leo, T. D'Orazio, and M. Trivedi, "A multi camera system for soccer player performance evaluation," in *Proc. of 3rd ACM/IEEE Int. Conf. on Distributed Smart Cameras*, pp. 1–8 (2009).

Massimo Camplani received his BS in electronic engineering with honors in 2004 and MS in electronic engineering in 2006 with honors, both from the Università degli Studi di Cagliari, Italy. In 2010 he received PhD in Electronic and Computer Engineering at the Università degli Studi di Cagliari. Since 2010, he has been a member of the Grupo de Tratamiento de Imágenes (Image Processing Group) of the Universidad Politécnica de Madrid, Madrid, Spain. In April 2011, he was awarded the Marie Curie-COFUND Grant. His research interests are in the area of computer vision.

Luis Salgado received his Ingeniero de Telecomunicación degree with high honors in 1990 and a PhD in communications (summa cum laude) in 1998, both from the E.T.S. Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Madrid, Spain. Since 1990, he has been a member of the Image Processing Group (GTI) of the Universidad Politécnica de Madrid. He was a PhD scholar of the Spanish National Research Plan from 1991 to 1994, and a research assistant from 1995 to 1996. Since 1996, he has been a member of the faculty at the Universidad Politécnica de Madrid, first as a teaching assistant and, currently, as an associate professor (tenure in 2001) of signal theory and communications in the Department of Signals, Systems, and Communications. He is associate editor of the *Journal of Real-Time Image Processing*, has been member of the Scientific and Program Committees of several international conferences, and has been auditor and evaluator of European research programs since 2002. He has participated in many national and international research projects, and his professional interests include video analysis, processing, and coding.