Machine Vision and Applications (2009) 20:395–409
DOI 10.1007/s00138-008-0134-2

ORIGINAL PAPER

Non-parametric statistical background modeling for efficient foreground region detection

Alireza Tavakkoli · Mircea Nicolescu · George Bebis · Monica Nicolescu

Received: 26 September 2006 / Revised: 14 September 2007 / Accepted: 25 February 2008 / Published online: 30 April 2008

Abstract Most methods for foreground region detection in videos are challenged by the presence of quasi-stationary backgrounds—flickering monitors, waving tree branches, moving water surfaces or rain. Additional difficulties are caused by camera shake or by the presence of moving objects in every image. The contribution of this paper is to propose a scene-independent and non-parametric modeling technique which covers most of the above scenarios. First, an adaptive statistical method, called adaptive kernel density estimation (AKDE), is proposed as a base-line system that addresses the scene dependence issue. After investigating its performance we introduce a novel general statistical technique, called recursive modeling (RM). The RM overcomes the weaknesses of the AKDE in modeling slow changes in the background. The performance of the RM is evaluated asymptotically and compared with the base-line system (AKDE). A wide range of quantitative and qualitative experiments is performed to compare the proposed RM with the base-line system and existing algorithms. Finally, a comparison of various background modeling systems is presented as well as a discussion on the suitability of each technique for different scenarios.

A. Tavakkoli (✉) · M. Nicolescu · G. Bebis
Computer Vision Lab., University of Nevada, Reno, USA
e-mail: tavakkol@cse.unr.edu

M. Nicolescu
e-mail: mircea@cse.unr.edu

G. Bebis
e-mail: bebis@cse.unr.edu

M. Nicolescu
Robotics Laboratory, University of Nevada, Reno, USA
e-mail: monica@cse.unr.edu

Keywords Non-parametric density estimation · Recursive modeling · Background subtraction · Background modeling · Quasi-stationary backgrounds

1 Introduction

Typically, in most visual surveillance systems static cameras are used. However, because of inherent changes in the background, such as fluctuations in monitors and fluorescent lights, waving flags and trees, water surfaces, etc. the background may not be completely stationary. Furthermore, the background may not appear completely empty in any image across the video sequence, thus making the background modeling even more problematic. These difficult situations are illustrated in Fig. 1. We refer to these backgrounds as quasi-stationary.

1.1 Related work

In the presence of quasi-stationary backgrounds a single background frame is not enough to accurately detect foreground regions. Pless et al. [21] evaluated different models for dynamic backgrounds. Depending on the complexity of the problem the background models employ expected pixel features (i.e. colors) [3–5,23], consistent motion [20,33], or fusion of color/contrast and motion [1]. They also may employ pixel-wise information [32] or regional models of features [31,7,15]. To improve robustness to noise, spatial [19] or spatio-temporal [14] features may be used.

In [32] a single 3-D Gaussian model for each pixel is built and the mean and covariance of the model are learned in each frame. This system models the noise and uses a background subtraction technique to detect those pixels whose probabilities are smaller than a heuristically selected threshold.

Fig. 1 Examples of challenges in quasi-stationary backgrounds: **a** Fluctuating monitors. **b** Rain/snow. **c** Waving tree branches. **d** Non-empty background

However, the system failed to label a pixel as foreground or background when it has more than one modality due to fluctuations in its values, such as in a fluctuating monitor.

Kalman filtering [12,9,10] is also used to update the model and linear prediction using Wiener filtering is presented in [31]. These background models were also unable to represent multi-modal situations.

Indupalli et al. in [8] applied a histogram based method and a clustering technique to segment the background of the video. They also used the HSV color space in their pixel-wise system. However, their system requires that its parameters be selected manually. Also, this method fails if the scene does not have an empty background or is crowded.

In [30], Totozafiny et al. proposed a background segmentation system for road surveillance applications. Their technique generates the background model using a background reference frame and a mixture of Gaussians. This method is not adaptive to gradual and local changes in the illumination of the scene since it generates the model only once. The system is not scene independent and its parameters should be updated from application to application.

A mixture of Gaussians modeling technique was proposed in [25,24,6] to address the multi-modality of the underlying background. In this technique background pixels are modeled by a mixture of Gaussians. During the training stage, parameters and weights of the Gaussians are trained and used in the background subtraction where the probability of each pixel is generated using the mixture of Gaussians. The pixel is labeled as foreground or background based on its probability.

There are several shortcomings for mixture learning methods. First, the number of Gaussians needs to be specified. Second, this method does not explicitly handle spatial dependencies. Even with the use of incremental expectation maximization, the parameter estimation and its convergence is noticeably slow where the Gaussians adapt to a new cluster. The convergence speed can be improved by sacrificing memory as proposed in [16] and [17], limiting its applications when mixture modeling is pixel-based and over long temporal windows.

A recursive filter formulation is proposed by Lee in [13] to speed up the convergence. However, the problem of specifying the number of Gaussians as well as the adaptation in later stages still exists. This model does not account for situations in which the number of Gaussians changes due to occlusion or uncovered parts of the background.

In [5], El Gammal et al. proposed a non-parametric kernel density estimation method (KDE) for pixel-wise background modeling without making any assumption about its probability distribution. Therefore, this method can easily deal with multi-modality in background pixel distributions without specifying the number of modes in the background. However, there are several issues to be addressed using non-parametric kernel density estimation.

These methods are memory and time consuming since for each pixel in each frame the system has to compute the average of all kernels centered at each training sample. The size of temporal window used as the background model needs to be specified. Too small a window increases speed, while it does not incorporate enough history for the pixel, resulting in a less accurate model. The adaptation will be problematic by using small window sizes. Increasing the window size improves the model accuracy but at the cost of higher memory requirements and slower convergence. In order to adapt the model a sliding window is used in [18]. However, the model convergence is problematic in situations where the illumination suddenly changes.

In order to update the background for scene changes such as moved objects, parked vehicles or opened/closed doors, Kim et al. in [11] proposed a layered modeling technique. This technique needs an additional model called *cache* and assumes that the background modeling is performed over a long period of time. It should also be used as a post-processing stage after the background is modeled.

Another approach to model variations in the background is to represent these changes as different states corresponding to different environments—such as lights on/off, night/day, sunny/cloudy. For this purpose hidden Markov models (HMM) have been used in [22] and [26]. However, these techniques suffer from slow model training speed and are sensitive to model selection and initialization.

1.2 Motivation and contributions

Because of the aforementioned issues in detecting foreground regions in videos with quasi-stationary backgrounds existing systems addresses some of these problems in a specific or a combination of scenarios. Our focus here is to find a common

ground that would cover a general scenario for background modeling. Contributions of this study can be summarized as follows:

- Finding an appropriate approach to the problem of detecting foreground regions in videos with quasi-stationary background. This approach should address the multi-modality of the background as well as scene-independence. Our proposed solution is based on a non-parametric framework that addresses the issues in the literature. This base-line system, called adaptive kernel density estimation (AKDE), outperforms the existing methods in the literature [27,28].
- Investigating the efficiency of the base-line system and deriving a more universal framework upon this system. The proposed general method is called recursive modeling (RM). This technique addresses the issue of robust background training in slowly changing backgrounds, non-empty backgrounds, and backgrounds with steady, irregular global motion (hand-held camera).

The AKDE. The theory behind the AKDE algorithm is to estimate the probability of each pixel being background based on a number of samples in its history. One advantage of the AKDE method over existing kernel density estimation modeling is in using a different threshold for each pixel, instead of a single threshold for all pixels in the scene. These thresholds are independently trained over a number of video frames.

By training the thresholds the system becomes scene independent and there is no need to heuristically select and tune threshold values in different scenes. By employing these localized thresholds the system works efficiently on different video scenes and is more robust to local changes in the same scene. The proposed AKDE method exploits the dependency between pixel color components as well, thus leading to a more accurate background model.

The RM. The RM method is a recursive counterpart for the AKDE technique which uses pixel intensity/color values in new frames to update the background model at that pixel location. Since the update process is performed continuously, the background model converges to the actual one as more frames emerge and are processed. This gives the RM its ability to detect foreground regions when the background changes occur slowly and do not fit in a small temporal window. In videos without a set of empty background frames the proposed RM technique has the ability to generate a clear background model because pixels belonging to the actual background provide more support for the background model.

In order to make the background model converge to the actual one and recover from the expired model faster the proposed RM method uses a schedule for learning. It should be noticed that a non-parametric recursive modeling scheme has not been investigated in the literature.

The rest of the paper is structured as follows. Section 2 presents the base-line AKDE method and evaluates its performance and efficiency. This system and a benchmark data provide a standard set of comparison tests. Section 3 present the proposed RM technique and an evaluation of its performance with regard to the standard assessment presented for the base-line system. In Sect. 5, a comprehensive comparison between these two methods and other techniques is conducted and the situations in which each of the proposed methods is superior are presented. Finally, Sect. 6 concludes the paper and gives future directions of the research.

2 Adaptive kernel density estimation (AKDE)

In this section we present a novel technique for background modeling based on adaptive non-parametric kernel density estimation (AKDE) [28].

2.1 The algorithm

Figure 2 shows the pseudo-code for the AKDE algorithm, consisting of three major stages: training, classification and update. In the training stage the background model is generated, and for each pixel its model values are used to estimate the probability of that pixel to be background in new frames. The proposed method detects foreground regions by solving a classification problem. However, it should be noted that we only have samples of the background class before any foreground object appears in the scene.

The only parameter in kernel density estimation is the kernel bandwidth. In theory, as the number of training samples grows without a bound the estimated density converges to the actual underlying density regardless of the kernel bandwidth value [2].

2.2 Non-parametric density estimation

In the proposed AKDE method a non-parametric model for each pixel is generated and its classifier is trained. It uses the history of pixel values as training samples and estimates the probability of each pixel being background in new frames as the classification criterion. In the classification stage each pixel is classified as foreground or background based on its estimated probability, computed by:

$$P_t(\mathbf{x_t}) = \frac{1}{N 2\pi |\mathbf{\Sigma}|^{1/2}} \sum_{i=1}^{N} \mathrm{e}^{\left[-\frac{1}{2}(\mathbf{x}_t-\mathbf{x}_i)^T \mathbf{\Sigma}^{-1}(\mathbf{x}_t-\mathbf{x}_i)\right]} \quad (1)$$

where $\mathbf{x}_t$ is the pixel feature vector at time t and $\mathbf{x}_i$ are its values in the training sequence. $\mathbf{\Sigma}$ is a positive definite

Fig. 2 The proposed AKDE modeling algorithm

```
N: size of training buffer
For each frame at time t
  1. Training stage
   for each pixel (u,v)
    - Calculate kernel covariance Σ(u,v) and threshold th(u,v).
  2. Classification stage
   for each pixel (u,v)
    - Compute median of estimated probability in its neighborhood: Med(u,v)
    - if Med(u,v) ≤ th(u,v)
              then FG_t(u,v) = 1  % (Foreground)
              else FG_t(u,v) = 0  % (Backgrounds)
  3. Update stage
    - if size(FG) ≥ 0.5 Image_Size
        then                                          % Global sudden change detected
           for each pixel (u,v)                       % Replace oldest frame (OF)
              OF(u,v)← I_t(u,v)                       % with current frame
        else                                          % Gradual change
           for each pixel (u,v)                       % Replace background pixels in OF
              OF(u,v|FG_t(u,v) = 0) ← I_t(u,v|FG_t(u,v) = 0) % with those in current frame
```

symmetric matrix which is the kernel bandwidth matrix and N is the number of frames used to train the background model. In order to capture dependencies between features for each pixel, $\mathbf{\Sigma}$ has to be a full (non-diagonal) matrix.

Since in the AKDE method no assumptions are made on the covariance matrix $\mathbf{\Sigma}$, any features for each pixel can be used. Because color is the easiest and most reliable feature to extract we use chrominance values for each pixel. That is, given color values in RGB space we determine red (c_r) and green (c_g) chrominance values by:

$$c_\mathrm{r} = \frac{R}{R+G+B} \qquad c_\mathrm{g} = \frac{G}{R+G+B} \tag{2}$$

Therefore the feature vector for each pixel at a given time t is defined by:

$$\mathbf{x}_t = [\, c_\mathrm{r}(t), c_\mathrm{g}(t) \,]^T \tag{3}$$

Due to limited memory and computational power, we need to store a rather short term memory of the background frames as training samples. This makes the non-parametric kernel density estimation dependent on the choice of its kernel bandwidth. In order to achieve an accurate and automatic background model, which is adaptive to the spatial information in the scene, the kernel bandwidth matrix needs to be trained.

The effect of using a full covariance matrix can be observed in Fig. 3. By using a full covariance matrix ($\mathbf{\Sigma}$) in Eq. (1) we do not impose an assumption of feature independence on our estimation. If we assumed that features for each pixel are independent then a simplified version of Eq. (1) could be used, where the covariance matrix is diagonal. However, as it can be seen from Fig. 3 by using chrominance features, the independence assumption is not valid and the full covariance matrix results in a more accurate density estimation, as opposed to the diagonal covariance matrix proposed in [5].

2.3 Training stage

For each pixel the training samples are vectors $\mathbf{X}_N = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, where N is the number of training frames. In our experiments we chose $N = 300$ for most of the scenes. The successive deviation of the above vectors is a matrix Δ_X whose columns are:

$$\left[\mathbf{x}_i - \mathbf{x}_{i-1}\right]^T \quad \text{with } i = 2, 3, \ldots, N \tag{4}$$

For each pixel, the kernel bandwidth matrix is defined such that it represents the temporal scatter of training samples. Thus the kernel bandwidth is:

$$\Sigma = \mathrm{cov}(\Delta_X) \tag{5}$$

From Eqs. (4) and (5) it can be seen that for pixels with more feature changes through time, such as flickering pixels, the kernel bandwidth matrix has larger elements, while for pixels that do not change much its elements are smaller. Also notice that the kernel bandwidth is drawn from the training samples without any assumption on features and their underlying probability density function. The estimated probability density function by using this adaptive kernel bandwidth is accurate, even with a small number of background training frames. Finally, since the kernel bandwidth matrix is computed using successive deviations in Eq. (4) it accounts for temporal dependencies in pixel feature vectors.

In the traditional foreground detection techniques, usually the foreground regions are detected by comparing the value or model of each pixel with its value or model in the background. If this deviation is larger than a heuristically selected threshold it is labeled as a foreground region. If we estimate the probability of each pixel in all of the background frames, given that all pixels are background, their probabilities should have large values, close to 1. But because of noise and inherent background changes, pixels do not take a single value and their probabilities become smaller. The

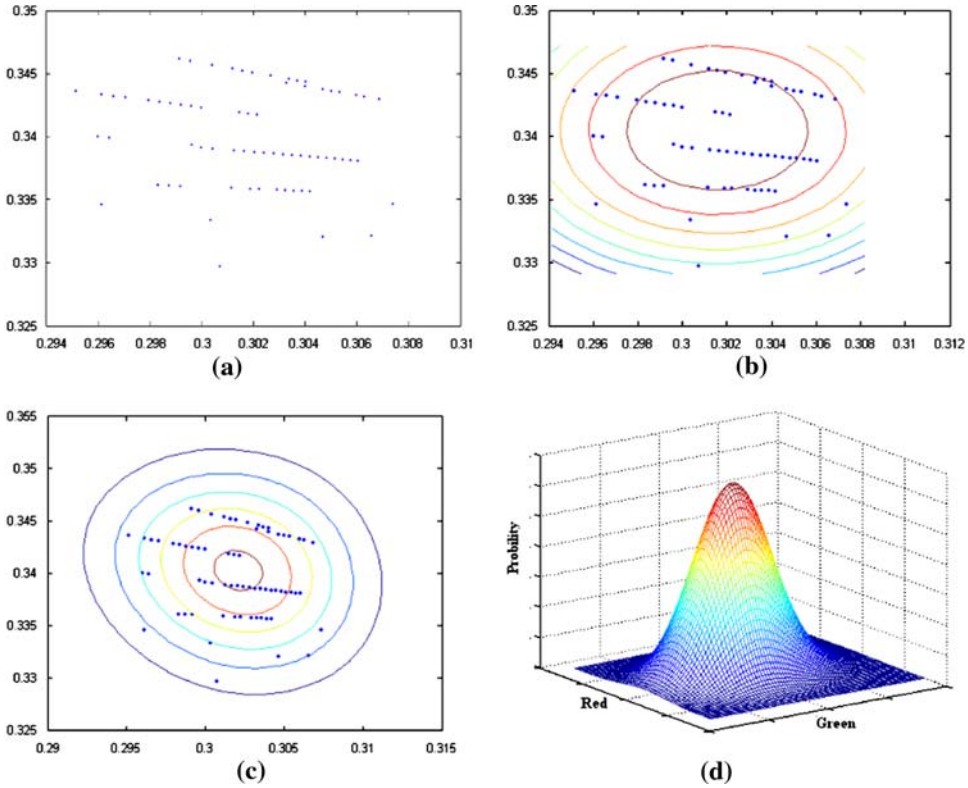

Fig. 3 Diagonal vs full kernel matrix: **a** pixel chrominance scatter plot (c_r, c_g) for one pixel over time. **b** Probability constant contours using diagonal covariance matrix. **c** Probability constant contours using full covariance matrix. **d** Probability density using full covariance matrix

probability of a pixel to be background is related to the amount of change that its features undergo in time. Therefore a single global threshold does not work well because pixels in different locations undergo different amounts of change.

These threshold values need to be trained for each pixel during the training stage to build an accurate and automatic classifier. For each pixel its threshold value (th) is selected such that its classifier results in 5% false reject rate. That is, 95% of the time the pixel is correctly classified as belonging to background:

$$\sum_{\substack{i=1 \\ P(Bg|x_i) \le th}}^{N} P(Bg|x_i) \le 0.05 \sum_{i=1}^{N} P(Bg|x_i) \tag{6}$$

This can be seen in Fig. 4, where (a) shows an arbitrary frame of a sequence containing a water surface and (b) shows the trained threshold map for this frame. Darker pixels in Fig. 4b represent smaller threshold values and lighter pixels correspond to larger threshold values. The thresholds in areas that tend to change more, such as the water surface, are lower than those in areas with less amount of change, such as the sky. Since the probability density function is normalized, for pixels which undergo more changes the estimated probability density function is wider. As a result, in order to keep 5% false reject rate, smaller threshold values are needed. Note

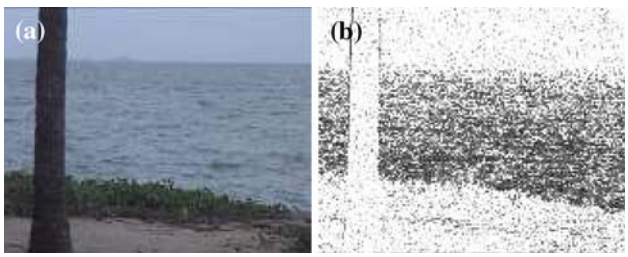

Fig. 4 Adaptive threshold map: **a** an arbitrary frame. **b** Threshold map

that the threshold map is noisy, since for efficiency purposes only 150 frames are used.

2.4 Classification stage

In the training stage, for each pixel its kernel bandwidth matrix Σ and its classification decision criterion th were determined. The probability of each pixel in the new frame is then estimated using Eq. (1). If we directly apply the trained threshold of each pixel to its estimated probability, due to impulse noise isolated pixels may still be erroneously classified.

One of the properties of this type of noise is that, if strong noise affects a pixel, it is less likely to affect its neighbors with the same strength. If a pixel in a region belonging to the background produces a fairly small probability because of noise, its neighboring pixels are expected to produce larger probabilities.

Fig. 5 Enforcing spatial consistency: **a** original frame **b** Detected foreground regions by applying thresholds directly on the estimated probability. **c** Detected foreground regions by applying threshold on median of probabilities in a neighborhood

Fig. 6 Our proposed RM algorithm

1. Initialization: Δ, α_0, β, κ and th
2. For each frame at time t
   For each pixel
   2.1. **Training stage**
   - Update $\alpha_t = \frac{1-\alpha_0}{h(t)} + \alpha_0$ and Δ
   - Update $\theta_t^B(x) = (1-\beta_t)\theta_{t-1}^B(x) + \alpha_t \cdot H_\Delta(x - x_t)$
   - If $\theta_t^B \leq th$ then update $\theta_t^F(x) = (1-\beta_t)\theta_{t-1}^F(x) + \alpha_t \cdot H_\Delta(x - x_t)$

   2.2. **Classification stage**
   - If $ln\left(\text{med}(\theta_t^B)/\text{med}(\theta_t^F)\right) \leq \kappa$ then label pixel as foreground.

   2.3. **Update stage**
   - Update κ and th

Notice that this impulsive noise is introduced to the system as a byproduct of the probability density estimation. As known, median filtering is a suitable tool to remove this type of noise. Applying the median filtering directly to the images suppresses the impulsive noise in the frames but does not significantly affect the noise introduced by the process. In order to remove the process noise we apply the median of estimated probabilities in a region around a pixel. After estimating the probability of each pixel in the new frame, the median of probabilities in its 8-connected neighborhood is compared with its threshold to make the classification decision:

$$\text{Label}_t = \begin{cases} \text{Foreground} & \text{if median } (\text{Prob}_t) \leq th \\ \text{Background} & \text{otherwise} \end{cases} \quad (7)$$

Figure 5 shows the effect of enforcing spatial consistency using the median of probabilities in foreground region detection. As it can be seen, by applying the threshold on the median of estimated probabilities of pixels in a neighborhood, most of the noise can be suppressed, while maintaining the image quality.

2.5 Update stage

In the proposed AKDE method we use two different types of adaptation. To make the system adaptive to *gradual changes* in illumination, we replace the pixels in the oldest background frame with those belonging to the current background mask. In order to detect *sudden changes* in the illumination, the area of the foreground objects are checked. Once a *sudden change* is detected (detected foreground region is very large), the classification stage of the algorithm is suspended and new frames replace all frames in the background training buffer.

Because the training stage of the algorithm is time consuming, the updating stage is actually performed every few frames, depending on the rate of changes and the processing power. In the current implementation the updating stage is performed every 100–150 frames.

3 Recursive modeling (RM)

In this section, we describe our novel recursive method. The formulation is discussed in one dimension and its extension to higher dimensions is straightforward. We explain how dependencies between pixel features in higher dimensions can be captured, resulting in more accurate models.

3.1 The algorithm

The proposed method, in pseudo-code, is shown in Fig. 6. θ_t^{B} is the background model and θ_t^{F} is the foreground model for each pixel. Let x_t be the the intensity value of a pixel at time t. The non-parametric estimation of the background model that accurately follows its multi-modal distribution can be reformulated in terms of recursive filtering [29]:

$$\hat{\theta}_t^{\text{B}}(x) = [1 - \beta_t] \cdot \theta_{t-1}^{\text{B}}(x) + \alpha_t \cdot H_\Delta\,(x - x_t) \quad (8)$$

$$\sum_{x=0}^{255} \theta_t^{\text{B}}(x) = 1 \quad (9)$$

where $x \in [0, 255]$ and θ_t^{B} is the background pixel model at time t, normalized according to (9). $\hat{\theta}_t^{\text{B}}$ is updated by the local kernel $H(\cdot)$ with bandwidth Δ centered at x_t. Parameters α_t and β_t are the learning rate and forgetting

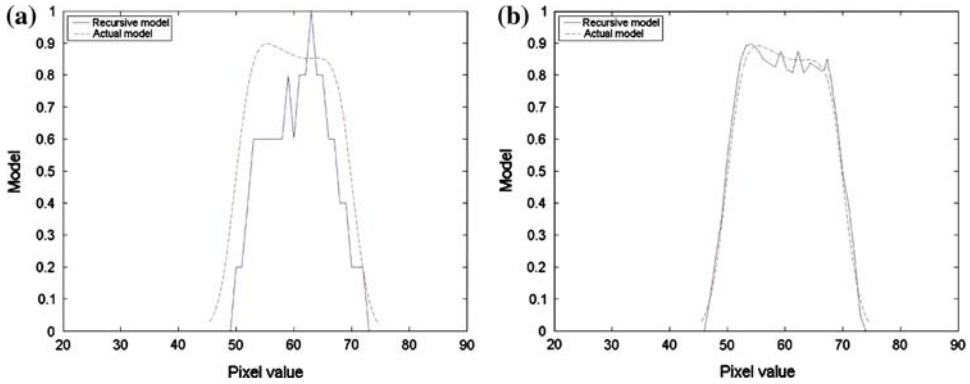

Fig. 7 Recursive modeling: model after **a** 10 frames. **b** 200 frames

rate schedules, respectively. The kernel H should satisfy the following conditions:

$$\sum_x H_\Delta(x) = 1$$
$$\sum_x x \times H_\Delta(x) = 0 \tag{10}$$

These conditions should be satisfied to ensure that the kernel is normalized, symmetric and positive definite in case of multivariate kernels. In our implementation of the RM method we use a Gaussian kernel which satisfies the above conditions. Note that in this context there is no need to specify the number of modalities of the background representation at each pixel.

Figure 7 shows the updating process using our proposed recursive modeling technique. It can be seen that the trained model (solid line) converges to the actual one (dashed line) as new samples are introduced. The actual model is the probability density function of a randomly generated sample population and the trained model is generated by using the recursive formula presented in (8).

In existing non-parametric kernel density estimation methods the learning rate α is selected to be constant and has small values. This makes the pixel model convergence slow and keeps its history in the recent temporal window of size $L = 1/\alpha$. The window size in non-parametric models is important as the system has to cover possible fluctuations in the background model. That is, pixel intensity changes may not be periodic or regular and consequently may not fit in a small temporal window. In such cases larger windows are needed, resulting in higher memory and computational requirements. Another issue in non-parametric density estimation techniques is that the window size is fixed and the same for all pixels in the scene. However, some pixels may have fewer fluctuations and therefore need smaller windows to be accurately modeled, while others may need a much longer history to cover their changes.

3.2 Scheduled learning

In order to speed up the modeling convergence and recovery we use a schedule for learning the background model at each pixel based on its history. This schedule makes the adaptive learning process converge faster, without compromising the stability and memory requirements of the system. The learning rate changes according to the schedule:

$$\alpha_t = \frac{1 - \alpha_0}{h(t)} + \alpha_0 \tag{11}$$

where α_t is the learning rate at time t and α_0 is a small target rate which is:

$$\alpha_0 = 1/256 \times \sigma_\theta \tag{12}$$

where σ_θ is the model variance. The function $h(t)$ is a monotonically increasing function:

$$h(t) = t - t_0 + 1 \tag{13}$$

where t_0 is the time at which a sudden global change is detected. At early stages the learning occurs faster ($\alpha_t = 1$) and monotonically decreases to converge to the target rate ($\alpha_t \to \alpha_0$). When a global change is detected $h(t)$ resets to 1. Later in Sect. 5 we discuss the effect of this schedule on improving the convergence and recovery speed.

The forgetting rate schedule is used to account for removing the values that have occurred a long time ago and no longer exist in the background. In the current implementation we assume that the forgetting rate is a portion of the learning rate $\beta_t = l \cdot \alpha_t$, where $l = 0.5$.

3.3 Training stage

Before new objects appear in the scene, at each pixel all the intensity values have the same probability of being foreground. However, in each new frame the background models are updated according to Eq. (8), resulting in larger model values (θ^{B}) at the pixel intensity value x_t. In essence the value

of background pixel model at each intensity x is:

$$\theta_t^{\mathrm{B}}(x) = P(Bg|x) \quad x \in [0, 255] \tag{14}$$

In order to achieve better detection accuracy, we introduce the foreground model. Later in the classification stage the foreground model is compared to the background model. For all $x \in [0, 255]$, the foreground model is defined by:

$$\hat{\theta}_t^{\mathrm{F}}(x) = [1 - \beta_t^{\mathrm{F}}] \cdot \theta_{t-1}^{\mathrm{F}}(x) + \alpha_t^{\mathrm{F}} \cdot H_{\Delta}(x - x_t) \tag{15}$$

$$\sum_{x=0}^{255} \theta_t^{\mathrm{F}}(x) = 1 \tag{16}$$

Once the background model is updated for each pixel, it is compared to the threshold th. If its value is less than this threshold the foreground model for that pixel value is updated according to (15) and (16).

3.4 Classification stage

For each pixel at time t we use a function θ_t^{B} for the background model and θ_t^{F} for the foreground. The domain of these functions is $[0, 255]^N$, where N is the dimensionality of the pixel feature vector. For simplicity assume the one dimensional case again, where θ_t is the background/foreground model whose domain is [0, 255]. From Eq. (15), each model ranges between 0 and 1 and its value shows the amount of evidence accumulated in the updating process (i.e., the estimated probability). For each new intensity value x_t we have the evidence of each model as $\theta_t^{\mathrm{B}}(x_t)$ and $\theta_t^{\mathrm{F}}(x_t)$. The classification uses a *maximum* a posteriori criterion to label the pixel as foreground:

$$\ln\left(\frac{\theta_t^{\mathrm{B}}}{\theta_t^{\mathrm{F}}}\right) \leq \kappa \tag{17}$$

3.5 Updating stage

In many applications with dynamic or quasi-stationary backgrounds, we need adaptive classification criteria. Because not all pixels in the scene follow the same changes, the decision threshold κ should be adaptive and independent for each pixel and has to be driven from the history of that pixel. Figure 4 explains this issue. The argument is similar to issue the of adaptive, localized threshold map discussed in Sect. 2.2.

From the algorithm shown in Fig. 6 it can be observed that there are two set of thresholds th and κ. Thresholds th for each pixel should adapt to a value where:

$$\sum_{\substack{x \\ \theta_t^{\mathrm{B}}(x) \geq th}} \theta_t^{\mathrm{B}}(x) \geq 0.95 \tag{18}$$

For the other set of thresholds κ, we similarly use a measure of changes in the intensity at each pixel position. Therefore the threshold κ is inversely proportional to the background model variance:

$$\kappa \approx \ln\left\{\left[\sum_{x=0}^{255}\left(\theta_t^{\mathrm{B}}(x) - \mathrm{mean}[\theta^{\mathrm{B}}(x)]\right)^2\right]^{-1}\right\} \tag{19}$$

This ensures that for pixels with more changes, smaller threshold values are chosen for classification, while for those pixels with fewer changes larger thresholds are employed. It should be mentioned that in the current implementation of the algorithm, the thresholds are updated every 30 frames.

3.6 Incorporating color information

In the above section, we described the recursive learning scheme in 1-D where the background and foreground models are updated using the intensity value of pixels at each frame. To extend the modeling in higher dimensions and incorporate color information, one may consider each pixel as a 3-D feature vector in $[0, 255]^3$. The kernel H in this space is a multivariate kernel H_{Σ}. In this case, instead of using a diagonal matrix H_{Σ} a full multivariate kernel can be used. The kernel bandwidth matrix Σ is a symmetric positive definite 3×3 matrix. Given N pixels, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, labeled as background, their successive deviation matrix is a matrix Δ_X whose columns are:

$$\left[\mathbf{x}_i - \mathbf{x}_{i-1}\right]^T \quad \text{with } i = 2, 3, \ldots, N \tag{20}$$

The bandwidth matrix is defined such that it represents the temporal scatter of training samples:

$$\Sigma = \mathrm{cov}(\Delta_X) \tag{21}$$

However, in the current implementation only red and green chrominance values are used. Also in order to decrease the memory requirements of the system we assumed that the two chrominance values are independent. Making this assumption results in a significant decrease in memory requirements while the accuracy of the model does not decay drastically. The red/green chrominance values are quantized into 256 discrete values.

4 Performance evaluation

In this section, we evaluate the performance of each of the proposed methods separately. The evaluation is conducted in terms of the number of system parameters, their impact on the output of the system, memory requirements and accuracy of the results.

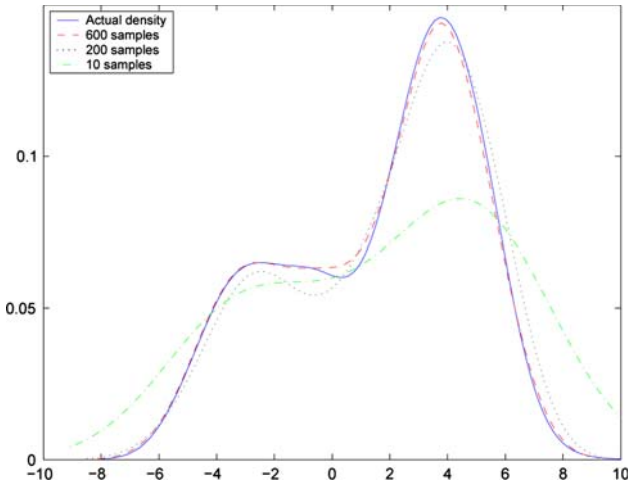

Fig. 8 Effect of the number of training samples on the estimated density function

4.1 Adaptive kernel density estimation method (AKDE)

In this section, we present the system details and analyze its performance.

Parameters. One important parameter in this method is the number of training samples used to estimate the probability density. Other parameters such as the threshold and the kernel bandwidth matrix are trained during the training stage. In Fig. 8, the actual probability function of a randomly distributed population is shown by the solid line. The estimated probability density function converges to the underlying density by increasing the number of training samples. However, there is a trade-off between the number of training samples, memory requirements and convergence speed of the algorithm.

Memory requirements. The system needs to store all the training samples in order to estimate the probability of a new sample. If only pixel intensity values are to be employed for each pixel, n values should be stored. Given that these values range is between 0 and 255, each intensity value is stored in 1 byte, resulting in n bytes per-pixel memory requirement. Also the system needs to store the kernel bandwidth and the thresholds for each pixel, which result in two floating numbers. Considering that each floating number can be stored in 4 bytes, 8 bytes per pixel are needed to store the kernel bandwidth and the threshold. This results in $n + 8$ bytes memory requirement per pixel. Similarly, the per-pixel memory requirements using chrominance values are $8n + 20$.

From the above discussion we can conclude that the asymptotic memory requirement for the system is $O(n)$. That is, if the number of training samples reaches infinity the memory requirements of the system grow linearly.

Computational cost. If we only use pixel intensity values for n training samples per pixel we need two additions and two multiplications for each training sample. This results in $2n$ addition and $2n$ multiplication operations. Given the optimal implementation of the exponential function using look-up tables, its cost is equal to a memory indexing. This can be assimilated to a single addition operation. The per-pixel computational cost of the AKDE method is $5 \times n$. If chrominance values are used the computational cost will be $13 \times n$ per pixel.

Given the optimal implementation of the exponential function and multiplication operations, the asymptotic per-pixel computational cost is $O(n)$. Note that this is the optimal asymptotic computational cost per pixel. The actual frame rate of the current implementation of the AKDE method is about 5–10 fps.

4.2 Recursive modeling method (RM)

In this section, we analyze the performance of our recursive modeling method.

Parameters. In the RM method there are five parameters: the learning and forgetting rate α and β, thresholds th and κ, and the bandwidth Σ. As described earlier, these parameters are trained and estimated from the data to generate an accurate and robust model. The reason that the RM technique is robust is in using most of the information in the data set and not being limited on the number of training samples. With all parameters being updated, the system performance does not depend on heuristically (and scene dependent) values for these parameters.

Memory requirements. If pixel intensity is used in the RM technique the model becomes a 1-D function representing the probability mass function of the pixel. The pixel intensity values range is from 0 to 255 making the memory requirements of the RM equal to 256×4 bytes per pixel. Using chrominance values, the model is 2-D and needs $256^2 \times 4$ bytes in memory.

The current implementation of the RM method uses a simple assumption of independence between color features which results in 8×256 bytes memory requirements [29]. Color components are not independent. However, assuming that they are independent helps decreasing memory needs drastically while the accuracy does not decrease significantly. In conclusion the asymptotic memory requirement of the RM algorithm is constant $O(1)$.

Computational cost. If we only use pixel intensity values for pixels we need 256 addition and 2×256 multiplication operations. Similarly, if we use 2-D chrominance values as pixel features and using the independence assumption discussed earlier, the system requires only 2×256 addition and 4×256 multiplication operations to update the model.

The asymptotic computation cost for this system is constant, $O(1)$, since the updating process merely consists of adding two 1-D functions. Note that this technique does

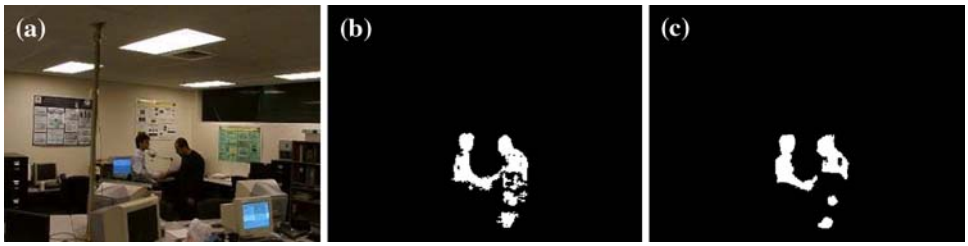

Fig. 9 Rapidly fluctuating background: **a** *Handshake* video sequence. **b** Detected foreground regions using AKDE. **c** Detected foreground regions using RM

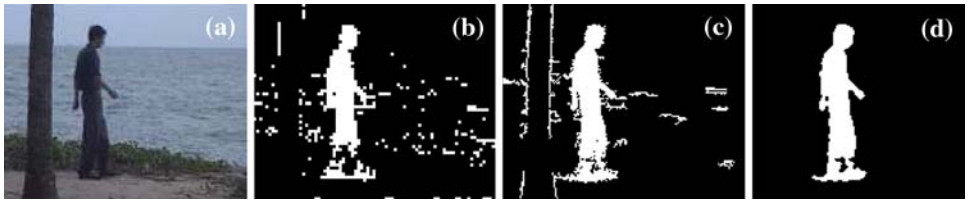

Fig. 10 Slowly changing background: **a** *Water* video sequence. Detected foreground region using: **b** MoG, **c** AKDE and **d** RM

not need to compute the exponential function and it acts as an incremental process, updating the model at each frame using the kernel and the previous model. The algorithm is inherently fast and an efficient implementation runs in real-time reaching frame rates of 15–30 fps.

5 Comparison

In this section, we compare the performance of proposed techniques using several real video sequences that pose significant challenges. Also their performances are compared with the mixture of Gaussians method [25], the spatio-temporal modeling presented in [14] and the simple KDE method [5]. We use different scenarios to test the performance of the proposed techniques and discuss where each method is more suitable.

Rapidly fluctuating backgrounds. As described above, for videos with rapidly changing background, the AKDE method has a better performance in terms of memory requirements and speed. Our experiments showed that for videos where possible fluctuations in the background occur in about 10 s, the AKDE technique needs less memory and works faster compared to the RM method. Figure 9 shows the detection results of the AKDE and RM algorithms on the *Handshake* video sequence. As it can be seen from this figure, capturing dependencies between chrominance features results in a more accurate foreground region (in Fig. 9b), showing that AKDE performs better than the RM. Note that this is a low contrast video sequence and the color of foreground objects is close to the background in some regions. Also in both methods fluctuations in monitors are completely modeled as a part of background and not detected as foreground regions.

Slowly changing backgrounds. For videos with slowly changing backgrounds or backgrounds in which changes are not periodic, the AKDE method needs more training frames to generate a good model for the background. This increases the system memory requirements and drastically decreases its speed. In these situations the RM technique is a very good alternative, since its performance is independent of the number of training frames. Figure 10a shows an arbitrary frame of the *Water* video sequence. In this figure the detection results of both AKDE and RM methods are presented. This example is particularly difficult because waves do not follow a regular motion pattern and their motion is slow. Figure 10b shows the result of the MoG [25]. As it can be seen from Fig. 10c, using the AKDE method without any post-processing results in many false positives. Figure 10d shows the detection results of the RM method, which outperforms both AKDE and MoG in the presence of slowly changing backgrounds.

Hand-held camera. In situations when the camera is not completely stationary, such as the case of a hand-held camera, the AKDE method is not suitable. In these situations there is a consistent, slow and irregular global motion in the scene, which can not be modeled by a limited size sliding window of training frames. In such cases the RM method is highly preferable.

Figure 11 shows the modeling error of the RM method in the *Room* video sequence. In Fig. 11a an arbitrary frame of this video is shown. Figure 11b compares the modeling error using different techniques. As it can be seen, the modeling error using a constant window size in the AKDE (the dotted line) is between 20 and 40%, and it does not decrease with time. This shows that the system using the AKDE method with a constant sized sliding window never converges to the actual model. The dashed line shows the modeling error using the RM method with a constant learning rate, and the solid line shows the modeling error of the RM with scheduled learning. We conclude that the model generated by the RM technique eventually converges to the actual background model and its error goes to zero. Figure 11c and d show misclassified regions using the AKDE method after 2 and 247 frames respectively and Fig. 11e and f show the false positives using the RM method after 2 and 247 frames into the video. As it

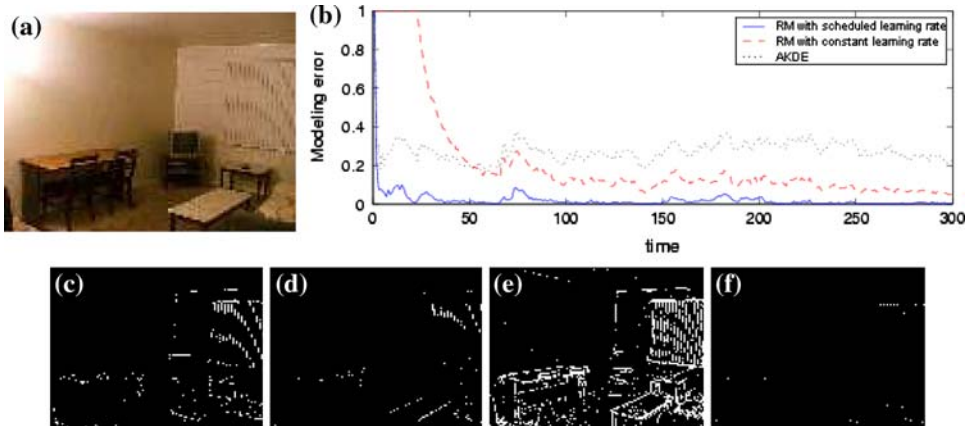

Fig. 11 Hand-held camera: **a** *Room* video sequence. **b** Modeling error in a hand-held camera situation using different methods. **c** False positives after two frames using the AKDE method. **d** False positives after 247 frames using the AKDE method. **e** False positives after two frames using the RM method. **f** False positives after 247 frames using the RM method

Fig. 12 Non-empty background: **a** *Mall* video sequence. **b** Background model after five frames using the RM method. **c** Background model after 95 frames using the RM method

can be seen, the amount of false positives decrease with time as the system accumulates most changes observed in the history of the scene using the RM method, but for the AKDE it does not converge to zero.

Non-empty backgrounds. In situations where the background of the video is not empty (there is no clear background at any time in the video sequence), the AKDE method fails to accurately detect the foreground regions. In these situations the RM technique has to be used to generate an accurate empty background model.

Figure 12 shows the background model in the *Mall* video sequence in which the background is never empty. In this situation the AKDE method fails unless a post-processing on the detected foreground regions is performed to generate models for uncovered parts of the background. This system considers the foreground objects present in the background training window as a part of background. When those objects move their empty position is detected as a foreground region. In the RM method however, the background model is updated at every frame from the beginning of the video. When an object moves the new pixel information is used to update the background model to the new one. Figure 12b shows the background model after five frames from the beginning of the video and Fig. 12c shows the model after 95 frames.

In this scenario consistent background regions are temporarily occluded by transient moving objects. Therefore the background itself contributes more consistent information to the model. As a result, the model converges to the empty background. This can be observed from Fig. 12.

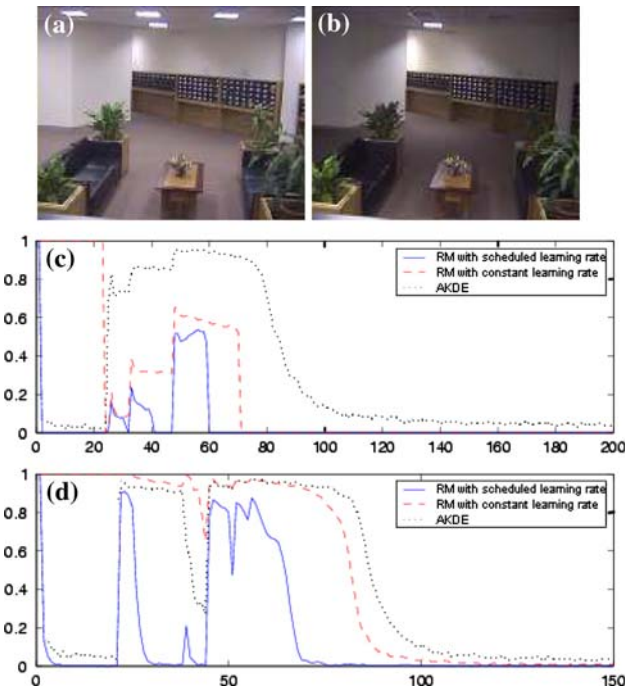

Fig. 13 Convergence speed

Convergence speed. An important issue in the recursive learning is the convergence speed of the system (how fast the model converges to the actual background). Figure 13 illustrates the convergence speed of our approach with scheduled learning, compared to constant learning and kernel density estimation with constant window size.

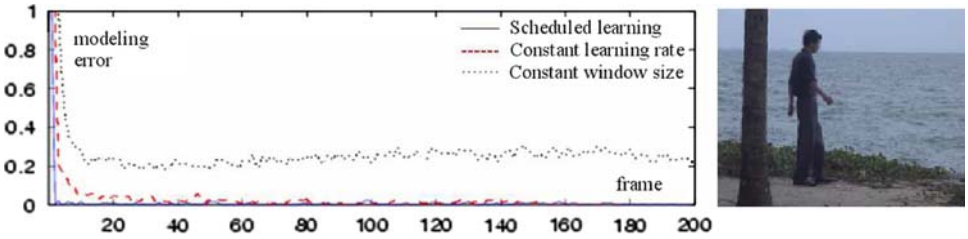

Fig. 14 Sudden global changes in the background: **a** *Lobby* video sequence with lights on. **b** Lights off. **c** Recovery speed comparison in lights turned off scenario. **d** Recovery speed comparison in lights turned on scenario

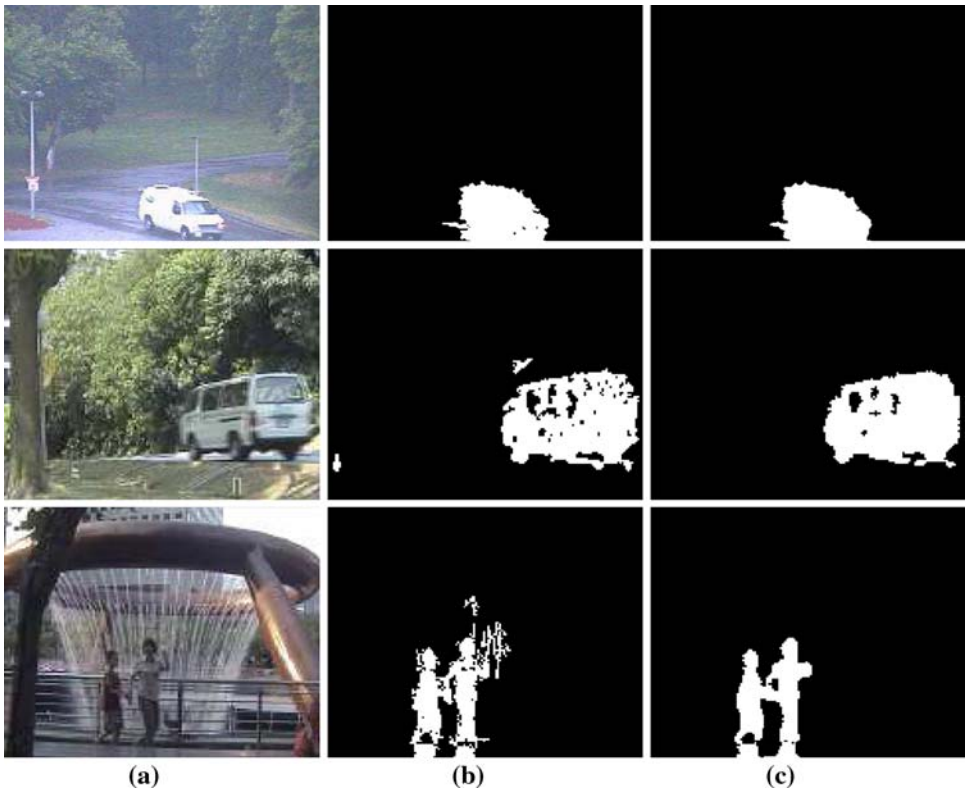

Fig. 15 Other difficult examples: **a** original frame. **b** Detected foreground region using AKDE. **c** Detected foreground regions using RM

Sudden global changes in the background. In situations where the video background suddenly changes—such as lights on/off—the proposed RM technique with scheduled learning recovers faster than the AKDE method. Generally, with the same speed and memory requirements, the RM method results in faster convergence and lower model error than existing techniques.

Figure 14 shows the comparison of the recovery speed from an expired background model to the new one. Figure 14a depicts an indoor scene with lights on and Fig. 14b shows the scene with the lights off. In our example (Fig. 14c) lights go from on to off through three global but sudden changes occurring at frames 23, 31 and 47. As shown, the scheduled learning RM method (solid curve) recovers the background model after these changes faster than non-scheduled RM and the AKDE with constant window size. The constant, large learning rate recovers more slowly (dashed curve) while the AKDE technique (dotted curve) is not able to recover even after 150 frames. A similar situation with lights going from off to on through three global, sudden illumination changes is shown in Fig. 14d. It needs to be mentioned that the mixture learning algorithms are even slower in convergence and recovery. A typical mixture learning technique proposed in [25] needs at least 1,000 frames to converge.

Other difficult examples. Figure 15 shows three video sequences with challenging backgrounds. In column (a) the original frames are shown, while column (b) and (c) show the results of the AKDE and the RM methods, respectively. Heavy rain, waving tree branches, and the water fountain shown in this figure (from top to bottom) pose significant difficulties in detecting accurate foreground regions.

Quantitative evaluation. Performance of our proposed methods, RM and AKDE, is evaluated quantitatively on randomly selected samples from different video sequences, taken from [14].

The similarity measure between two regions $\mathcal{A}$ (detected foreground regions) and $\mathcal{B}$ (ground truth) is defined by $\mathcal{S}(\mathcal{A}, \mathcal{B}) = \frac{\mathcal{A} \cap \mathcal{B}}{\mathcal{A} \cup \mathcal{B}}$. This measure increases monotonically

Table 1 Quantitative evaluation and comparison. The sequences are *Meeting Room, Lobby, Campus, Side Walk, Water* and *Fountain*, from left to right from [14]

Method	Videos						
	MR	LB	CAM	SW	WAT	FT	Avg.$\mathcal{S}(\mathcal{A}, \mathcal{B})$
AKDE	0.74	0.66	0.55	0.52	0.84	0.51	0.64
RM	0.92	0.87	0.75	0.72	0.89	0.87	0.84
Spatio-Temp [14]	0.91	0.71	0.69	0.57	0.85	0.67	0.74
MoG [25]	0.44	0.42	0.48	0.36	0.54	0.66	0.49

with the similarity between detected masks and the ground truth, ranging between 0 and 1. By using this measure we report the performance of the AKDE method, the RM method, the spatio-temporal technique presented in [14] and the mixture of Gaussians (MoG) in [25]. By comparing the average of the similarity measure over different video sequences in Table 1, we can see that the RM method outperforms other techniques. This shows that the RM method works consistently well on a wide range of video sequences. Also, note that both AKDE and RM are automatic, without the need for fine-tuning a large number of parameters for each scene, as opposed to other existing methods.

However from this table one might argue that AKDE does not perform better than the method presented in [14]. The reason is that in [14] the authors used a morphological post-processing stage to refine their detected foreground regions, while the results shown for AKDE are the raw detected regions. We performed a morphological post-processing on the results obtained by the AKDE, and the average similarity measure increased to 0.74.

Computation time. In this section, we present a comparison of the speed of the RM and the AKDE on the *Handshake* video sequence. The frame size for the experiments is 120×160 in *RGB* color format. The systems are implemented on a 4.8 GHz Pentium 4 Processor. We used $N = 300$ frames for the initial background training process for the AKDE. Table 2 shows the computation time of the system for the AKDE and the RM method. As seen, the RM method is a fast technique with frame rate of at least 15 fps.

Comparison summary. Table 3 summarizes this study and provides a comparison between different traditional methods for background modeling proposed in the literature and our proposed methods. The comparison includes the number of parameters, classification type, memory requirements, computation cost and parameter selection. Table 4 shows different scenarios and illustrates which method appears to be particularly suitable for foreground region detection.

Table 2 Computation time

Method	Detection time per frame (s)	Speed (fps)
AKDE	0.186	5.2
RM	0.0625	15.38

6 Conclusions and future work

In this paper, we have presented two novel techniques for background modeling based on non-parametric density estimation and recursive modeling. The advantage of our adaptive kernel density estimation method (AKDE) over existing techniques is that instead of a global threshold for all pixels in the video scene, different and adaptive thresholds are used for each pixel. By training these thresholds the system works robustly on different video scenes without changing or tuning any parameter. Since each pixel is classified by using adaptive thresholds and exploiting its color dependency, the background model is more accurate.

Our novel recursive modeling method (RM) updates the model on-line when a new frame becomes available, instead of processing a set of video frames to generate the background model. Since the model is not generated by a finite set of samples it eventually converges to the actual background model. This method is superior and more robust than other techniques for situations in which background changes are slow and not periodic.

In particular the RM method outperforms other non-parametric techniques when a set of empty background frames is not available (such as the *Mall* video sequence) as well as in the case of a hand-held camera. The recovery and convergence speed of the RM method in cases when the global illumination suddenly changes are better than those of other non-parametric techniques.

From studying the performance of each of the proposed methods in terms of memory requirements and computational cost it can be observed that the AKDE method is more efficient than the RM technique when background changes are fast. On the other hand when changes occur very slowly, or when there are no empty background frames, the RM works better than AKDE because of its recursive nature.

A future research direction is to perform the foreground/background segmentation without establishing a probabilistic

Table 3 Comparison between the proposed methods and traditional techniques

Criteria	AKDE	RM	KDE [5]	Spatio-Temp [14]	MoG [25]	Wallflower [31]
No. of parameters	3	3	3	9	5	8
Scene-independent	Yes	Yes	No	No	No	No
Post proc.	No	No	No	Yes	No	No
Classifier	Bayes	MAP	Bayes	Bayes	Bayes	K-means
Memory req.[a]	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(n)$
Comp. cost[a]	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(n)$

n number of training frames or training features used per pixel
[a] Per-pixel memory requirements or computational cost

Table 4 Scenarios where each method appears to be particularly suitable

Scenario	AKDE	RM	KDE [5]	[14]	MoG [25]	Wallflower [31]
Low contrast video	S[a]	NS[b]	S	NS	NS	NS
Close *Bg/Fg* colors	S	NS	NS	NS	NS	NS
Slowly changing background	NS	S	NS	S	S	S
Rapidly changing background	S	S	S	S	NS	S
Sudden global changes	NS	S	NS	S	S	NS
Non-empty backgrounds	NS	S	NS	S	S	S
Hand-held camera	NS	S	NS	NS	NS	NS

[a] Suitable
[b] Not suitable

model for the background or the foreground. This new approach would aim to establish the decision boundaries between background and foreground classes for each pixel based on support vector classification methods.

Acknowledgments This work was supported by the NSF-EPSCoR Ring True III award EPS0447416 and by the Office of Naval Research award N00014-06-1-0611.

References

1. Criminisi, C., Gross, G., Blake, A., Kolmogorov, V.: Bilayer segmentation of live video. In: Proceedings of the CVPR pp. 17–22 (2006)
2. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, New York (2001)
3. Elgammal, A., Duraiswami, R., Davis, L.S.: Efficient computation of kernel density estimation using fast Gauss transform with application for segmentation and tracking. In: Proceedings of the 2nd IEEE International Workshop on Statistical and Computational Theories of Vision (2001)
4. Elgammal, A., Duraiswami, R., Davis, L.S.: Efficient nonparametric adaptive color modeling using fast Gauss transform. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2001)
5. Elgammal, A., Duraiswami, R., Harwood, D., Davis, L.: Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. Proc. IEEE **90**, 1151–1163 (2002)
6. Friedman, N., Russell, S.: Image segmentation in video sequences: a probabilistic approach. In: Annual Conference on Uncertainty in Artificial Intelligence, pp. 175–181 (1997)
7. Hsu, Y., Nagel, H., Rekers, G.: New likelihood test methods for change detection in image sequences. Comput. Vis. Graph. Image Process. **26**, 73–106 (1984)
8. Indupalli, S., Ali, M., Boufama, B.: A novel clustering-based method for adaptive background segmentation. In: Proceedings of the Computer and Robot Vision, pp. 37–43 (2006)
9. Karman, K.P., von Brandt, A.: Moving object recognition using an adaptive background memory. In: Proc. Time-varying Image Processing and Moving Object Recognition (1990)
10. Karman, K.P., von Brandt, A.: Moving object segmentation based on adaptive reference images. In: Signal Processing: Theories and Applications, pp. 951–954 (1990)
11. Kim, K., Harwood, D., Davis, L.S.: Background updating for visual surveillance. Proc. Int. Symp. Vis. Comput. **1**, 337–346 (2005)
12. Koller, D., Weber, J., Huang, T., Malik, J., Ogasawara, G., Rao, B., Russel, S.: Towards Robust Automatic Traffic Scene Analysis in Real-Time. In Proceedings of ICPR **1**, 126–131 (1994)
13. Lee, D.S.: Effective Gaussian mixture learning for video background subtraction. IEEE Trans. PAMI **27**(5), 827–832 (2005)
14. Li, L., Huang, W., Gu, I., Tian, Q.: Statistical modeling of complex backgrounds for foreground object detection. IEEE Trans. Image Process. **13**(11), 1459–1472 (2004)
15. Matsuyama, T., Ohya, T., Habe, H.: Background subtraction for non-stationary scenes. In: Proceedings of Asian Conference on Computer Vision, pp. 662–667 (2000)
16. McKenna, S., Raja, Y., Gong, S.: Object tracking using adaptive color mixture models. Proc. Asian Conf. Comput. Vis. **1**, 615–622 (1998)
17. McKenna, S., Raja, Y., Gong, S.: Tracking colour objects using adaptive mixture models. Image Vis. Comput. **17**, 223–229 (1999)
18. Mittal, A., Paragios, N.: Motion-based background subtraction using adaptive kernel density estimation. Proc. CVPR **2**, 302–309 (2004)
19. Paragios, N., Ramesh, V.: A MRF-based approach for real-time subway monitoring. IEEE Trans. PAMI **1**, 1030–1040 (2001)
20. Pless, R., Brodsky, T., Aloimonos, Y.: Detecting independent motion: the statistics of temporal continuity. IEEE Trans. PAMI **22**(8), 68–73 (2000)
21. Pless, R., Larson, J., Siebers, S., Westover, B.: Evaluation of local models of dynamic backgrounds. Proc. CVPR **2**, 73–78 (2003)

22. Rittscher, J., Kato, J., Joga, S., Blake, A.: A probabilistic background model for tracking. Proc. 6th Eur. Conf. Comput. Vision **1843**, 336–350 (2000)
23. Sheikh, Y., Shah, M.: Bayesian object detection in dynamic scenes. Proc. CVPR **1**, 74–79 (2005)
24. Stauffer, C., Grimson, W.: Adaptive background mixture models for real-time tracking. Proc. CVPR **2**, 246–252 (1999)
25. Stauffer, C., Grimson, W.: Learning patterns of activity using real-time tracking. IEEE Trans. PAMI **22**(8), 747–757 (2000)
26. Stenger, B., Ramesh, V., Paragios, N., Coetzee, F., Buhmann, J.: A probabilistic background model for tracking. In Proceedings of ICCV, pp. 294–301 (2001)
27. Tavakkoli, A., Nicolescu, M., Bebis, G.: Automatic robust background modeling using multivariate non-parametric kernel density estimation for visual surveillance. In: Proceedings of the International Symposium on Visual Computing, LNSC, vol. 3804, pp. 363–370 (2005)
28. Tavakkoli, A., Nicolescu, M., Bebis, G.: Automatic statistical object detection for visual surveillance. In: Proceedings of IEEE Southwest Symposium on Image Analysis and Interpretation, pp. 144–148 (2006)
29. Tavakkoli, A., Nicolescu, M., Bebis, G.: Robust recursive learning for foreground region detection in videos with quasi-stationary backgrounds. In proceedings of 18th International Conference on Pattern Recognition (2006)
30. Totozafiny, T., Patrouix, O., Luthon, F., Coutllier, J.: Dynamic background segmentation for remote reference image updating within motion detection JPG2000. In: Proceedings of the International Symposium on Industrial Electronics, pp. 505–510 (2006)
31. Toyama, K., Krumm, J., Brumitt, B., Meyers, B.: Wallflower: principles and practice of background maintenance. Proc. ICCV **1**, 255–261 (1999)
32. Wern, C., Azarbayejani, A., Darrel, T., Petland, A.: Pfinder: real-time tracking of human body. IEEE Trans. PAMI **19**(7), 780–785 (1997)
33. Wixson, L.: Detecting salient motion by accumulating directionary-consistent flow. IEEE Trans. Pattern Anal. Mach. Intell. **22**, 774–780 (2000)

Author biographies

Alireza Tavakkoli is a Ph.D. candidate and research assistant in the Department of Computer Science and Engineering at the University of Nevada, Reno. He received a M.Sc. degree in Computer Science from the University of Nevada, Reno in 2006. He obtained his first M.Sc. degree in Digital Electronics and B.Sc. degree in Electronics from Sharif University of Technology, Iran in 2004 and 2001, respectively. His research interests include intent/action recognition, visual surveillance, computer vision and image processing. In 2006, he received a two year fellowship in Cognitive Information Processing from the Nevada NSF-EPSCoR fund. He is one of the recipients of the Outstanding International Graduate Student Award from the University of Nevada, Reno in 2007. He is a student member of the IEEE Computer Society and IEEE Circuits and Systems Society.

Mircea Nicolescu received the BS degree from the Polytechnic University Bucharest, Romania in 1995, the MS degree from the University of Southern California in 1999, and the PhD degree from the University of Southern California in 2003, all in Computer Science. He is currently an assistant professor of Computer Science at the University of Nevada, Reno, and co-director of the Computer Vision Laboratory. His research interests include visual motion analysis, perceptual organization, vision-based surveillance and activity recognition. In 1999 and 2003 he received the USC Academic Achievements Award, and in 2002 the Best Student Paper Award at the International Conference on Pattern Recognition in Quebec City, Canada. He is a member of the IEEE Computer Society.

George Bebis received the B.S. degree in mathematics and M.S. degree in computer science from the University of Crete, Greece in 1987 and 1991, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Central Florida, Orlando, in 1996. Currently, he is an Associate Professor with the Department of Computer Science and Engineering at the University of Nevada, Reno (UNR) and director of the UNR Computer Vision Laboratory (CVL). His research interests include computer vision, image processing, pattern recognition, machine learning, and evolutionary computing. His research is currently funded by NSF, NASA, ONR, and Ford Motor Company. Dr. Bebis is an associate editor of the Machine Vision and Applications Journal, and serves on the Editorial Board of the Pattern Recognition Journal and the International Journal on Artificial Intelligence Tools. He has served on the program committees of various national and international conferences, and has organized and chaired several conference sessions. In 2002, he received the Lemelson Award for Innovation and Entrepreneurship. He is a member of the IEEE and the IAPR Educational Committee.

Monica Nicolescu is an Assistant Professor of Computer Science with the Computer Science and Engineering Department at the University of Nevada, Reno and is the Director of the UNR Robotics Research Lab. Prof. Nicolescu earned her Ph.D. degree in Computer Science at the University of Southern California (2003) at the Center for Robotics and Embedded Systems. She obtained her M.S. in Computer Science at the University of Southern California (1999) and B.S. in Computer Science at the Polytechnic University Bucharest (Romania, 1995). She is a recipient of the NSF Early Development Career Award and a USC Academic Achievements Award. Prof. Nicolescu's research interests are in the areas of human-robot interaction, robot control and learning, and multi-robot systems.