11

Background Learning with Support Vectors: Efficient Foreground Detection and Tracking for Automated Visual Surveillance

Alireza Tavakkoli
Univ. of Houston-Victoria, USA

Mircea Nicolescu
CV Lab., Univ. of Nevada, Reno, USA

Junxian Wang
Microsoft Research, USA

George Bebis
CV Lab., Univ. of Nevada, Reno, USA

11.1 Introduction

With the increase in the availability and computational power of digital imaging devices, it is natural to think about integrating artificial intelligence models with processing of digital images and videos used for visual surveillance applications to improve their performance. However, for such applications to be efficient, there is a need for addressing a number of significant challenges. Accounting for the presence of regions that do not belong to objects of interest, global ambient illumination variations in the environment over long periods of time, and the real-time constraints inherent to visual surveillance applications are among such obstacles. This chapter demonstrates two main categories of mathematical and computational techniques developed to help improve the accuracy and efficiency of automated visual surveillance systems.

First, a statistical modeling approach based on non-parametric density estimation is presented with the goal of accurately detecting foreground regions in videos with quasi-stationary background. Second, we show that pixel models may alternatively be learned analytically to help with the issue of unknown probabilistic distribution of background pixels. In order to train pixel models analytically, Support Vector Machines are utilized to learn the single-class of pixel models – namely the background. This can be achieved by training Support Vector Data Descriptions (SVDD) for each pixel and by utilizing Support Vector Regression (SVR). We demonstrate the applicability of each approach in different surveillance applications - i.e. videos with non-empty backgrounds, indoor and outdoor environments, videos with sudden global illumination changes, etc.- to highlight their strengths and accuracy in delivering robust and reliable foreground regions. Such robust and accurate foreground region detection mechanisms help improve the end results of automated visual surveillance applications. The chapter concludes the contributions made within the proposed unified framework for visual surveillance and presents a road-map for future investigations.

11.2 Literature Review

Detecting foreground regions in videos is an important task in high-level video processing applications. One of the major issues in detecting foreground regions is that 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.

In the presence of these types of backgrounds, referred to as quasi-stationary, a single background frame is not enough to accurately detect moving regions. Therefore the background of the video has to be modeled in order to detect foreground regions - e.g. newly introduced objects to the scene, while allowing for quasi-stationary backgrounds.

There is also a great amount of diversity in scenarios where the background modeling techniques are used to detect foreground regions. Applications vary from indoors scenes to outdoors, from completely stationary to dynamic backgrounds, from high quality videos to low contrast scenes and so on. Therefore, a single system that addresses all possible situations while being time and memory efficient has yet to be devised.

11.2.1 Statistical Background Modeling

In the presence of quasi-stationary backgrounds, a single background frame is not enough to accurately detect foreground regions. Pless *et al.* [16] evaluated different models for dynamic backgrounds. Depending on the complexity of the problem the background models employ expected pixel features (i.e. colors) [17], consistent motion [15], [35], or fusion of color/contrast and motion [4]. They also may employ pixel-wise information [36] or regional models of features [30]. To improve robustness to noise, spatial [14] or spatio-temporal [12] features may be used.

In [36] a single 3-D Gaussian model for each pixel is built and the mean and covariance of the model are learned in each frame. 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.

A mixture of Gaussians modeling technique was proposed in [20], and [19] 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.

A recursive filter formulation is proposed by Lee in [11] 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 [7], Elgammal 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.

In order to update the background for scene changes such as moved objects, parked vehicles or opened/closed doors, Kim et al. in [9] 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.

Recently, we investigated two statistical methods for background modeling, based on adaptive kernel density estimation (AKDE) [24], [22], and recursive modeling (RM) [25], [23]. These techniques will be further investigated and discussed in this chapter.

There is a major drawback inherent to statistical modeling methods including the AKDE and the RM techniques. The accuracy of these methods is limited to the accuracy of the estimated probability density function for the background pixels. In this paper we present a non-statistical method that addresses this difficulty.

Furthermore, there is an additional issue with all statistical foreground detection techniques including the AKDE and the RM methods. In all statistical methods the assumption is that there are two classes, namely foreground and background, and that the model is trained on background samples which are present during a short period of time (the AKDE) or from the beginning of the video (the RM). Note that until a foreground object appears in the scene, there is no information about the foreground class. This problem is addressed by using thresholds in the classification stage to label pixels as foreground or background.

11.2.2 Analytical Learning of Background Pixel Models

To overcome the aforementioned disadvantage of statistical learning tools a non-statistical background modeling technique is proposed in [26], based on Support Vector data description modeling (SVDDM). This novel technique in describing one class of known data samples, is called Support Vector Data Description Modeling [26]. The backbone of the proposed method is a theory based on describing a data set using their Support Vectors [29], [28]. The SVDDM uses Support Vectors to generate a description for the known data class; e.g. the background. These Support Vectors along with the classifier information for each pixel are stored and used in the classification stage to label pixels in new frames as foreground/background. The performance of this system is studied and its experimental results on real video sequences are compared with other existing techniques in the literature.

11.2.3 Target Detection and Tracking

Target behavior analysis depends heavily on the reliability of target detection and tracking which can provide important information about the location of targets and their temporal correspondences over time. Both target detection and tracking have been investigated widely over the last two decades with the majority of approaches employing detection alone, tracking alone, or hybrid schemes such *"detect-then-track"* where detection and tracking work sequentially and independently of each other [21].

Tracking methods can be divided into two main categories. In the first category, the state sequence of a target is iteratively predicted and updated using prior information from past measurements and likelihood information from current measurements, respectively. Various filters have been employed to predict the state sequence of a target including Kalman filters [2] and extended Kalman filters for linear predictions, as well as unscented Kalman filters [2] for non-linear predictions. The most general class of filters, however, includes particle filters [10], also called bootstrap filters [8], which are based on Monte Carlo integration methods. Methods belonging in the second category use various target characteristics, such as color or gray-level information, shape, and motion information. These methods perform tracking by building the unique correspondence relationship in the appearance of the target from frame to frame [3].

In tracking alone methods, the initial location of a target is usually specified manually. The majority of methods employing detection along with tracking use a *detect-then-track* approach where the target is detected in the first frame and then turned over to the tracker in subsequent frames. The main problem with these methods is that they aim to resolve detection and tracking sequentially and independently of each other. An important issue considered in this work is improving the performance of target detection by feeding temporal information from tracking back to the detection stage. In this context, we propose a *detect-and-track* scheme where detection and tracking are addressed simultaneously in a unified framework (i.e., detection results trigger tracking, and tracking re-enforces detection). One approach to deal with this problem is by using a Bayesian decision framework which combines prior probability information provided by tracking with likelihood information provided by frame-based detection [33]. However, the performance of target detection depends heavily on the threshold used to distinguish between foreground and background objects. Another approach is propagating the probabilities of detection parameters (e.g., at several scales and poses) over time using condensation and factored sampling [32].

11.3 Non-Parametric Statistical Estimation of Pixel Distribution for Background Modeling

In this section we present a non-parametric statistical learning approach for modeling background pixel probability distribution [27]. Our focus here is to find a common ground that would cover a general scenario for background modeling. This solution is based on a non-parametric framework. This base-line system is called Adaptive Kernel Density Estimation (AKDE) [27]. To enhance the base-line statistical modeling technique, we derive a universal modeling tool. The proposed general method is called Recursive Modeling (RM) [25]. This technique addresses the issue of robust background training in slowly changing backgrounds, non-empty backgrounds, and backgrounds with irregular global motion (e.g. hand-held cameras).

Algorithm 11.1 - The proposed AKDE modeling algorithm

```
for each frame at time t do
  // Training Stage
  for each pixel:=uv do
    Σ [u,v]←CalcCovariance(frame_t)
    th[u,v]←CalcThreshold(frame_t)
  end for
  // Classification Stage:
  for each pixel:=uv do
    Median [u,v]←CalcMedian(frame_t, [u,v],[w×w])
    if Median[u,v]≤th[u,v] then
      FG_t[u,v]← 1 // Foreground Detected
    else
      FG_t[u,v]← 0 // Background Detected
    end if
  end for
  // Update Stage:
  if Size(FG)≥Size(frame) then
    for each pixel:=[u,v] do
      OldestFrame_t[u,v]←frame[u,v]
    end for
  else
    for each pixel:=[u,v] do
      OldestFrame_t[u,v|FG[u,v]==0]←frame[u,v|FG[u,v]==0]
    end for
  end if
end for
```

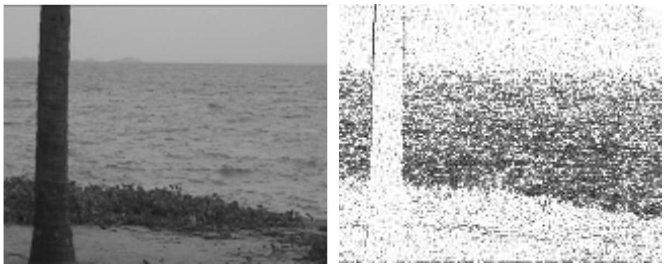
11.3.1 The AKDE Algorithm

Algorithm 11.1 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. In new frames, pixel model values are used to estimate the probability that the pixel belongs to the background model. Since we only have samples of the background class before any foreground object appears in the scene, there should be a mechanism to label low probability values to foreground models.

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 [6]. In the AKDE method a non-parametric model for each pixel is generated and its classifier is trained. The training stage employs the history of pixel values. The algorithm then 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:

$$P_t(\mathbf{x_t}) = \frac{1}{N2\pi|\mathbf{\Sigma}|^{1/2}} \sum_{i=1}^{N} e^{\left[-\frac{1}{2}(\mathbf{x}_t-\mathbf{x}_i)^T\mathbf{\Sigma}^{-1}(\mathbf{x}_t-\mathbf{x}_i)\right]} \tag{11.1}$$

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 symmetric matrix which is the kernel bandwidth matrix and N is



(a) (b)

FIGURE 11.1 Adaptive threshold map: (a) An arbitrary frame. (b) Threshold map.

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.

Due to limited memory and computational power, a rather short term memory of the background frames may be stored 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.

For each pixel the training samples are vectors $\mathbf{X}_N = \{\mathbf{x}_i : i = 1 \cdots N\}$, where N is the number of training frames. The successive deviation of the above vectors is a matrix Δ_X whose columns are $\left[\mathbf{x}_i - \mathbf{x}_{i-1}\right]^T$. For each pixel, the kernel bandwidth matrix is defined such that it represents the temporal scatter of training samples [27].

Note that for pixels that change more frequently the kernel bandwidth matrix has larger elements, while for pixels that do not change much its elements are smaller. Moreover, since the kernel bandwidth matrix is computed using successive deviations, it accounts for temporal dependencies in pixel feature vectors.

To allow for the pixel probability estimation adaptation to the different amount of change, the classifier threshold values need to be trained for each pixel during the training stage. For each pixel a 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 model [27].

This adaptive classifier threshold training can be seen in Fig. 11.1, 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. 11.1(b) represent smaller threshold values and lighter pixels correspond to larger threshold values. The thresholds in areas that tend to change more (the water surface) are lower than those in areas with less amount of change (the sky).

In the classification stage, each pixel background probability in new frames is estimated using equation (11.1) to label the pixel. If we directly apply the trained threshold of each pixel to its estimated probability, due to impulse (salt and pepper) noise, isolated pixels may 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.

Median filtering is known to be a suitable tool to remove this type of noise. 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 label each pixel as background or foreground. Fig. 11.2 shows the effect of enforcing spatial consistency

(a) (b) (c)

FIGURE 11.2 Original frame (a), and detected foreground regions by applying thresholds directly on the estimated probability (b), and on median of probabilities in a neighborhood (c) (©(2009) Springer).

using the median of probabilities in foreground region detection.

11.3.2 Recursive Modeling (RM)

One of the main disadvantages of the kerned density estimation approaches to background modeling is the number of background frames needed to train probabilistic models – i.e. N from equation (11.1).

The Recursive Modeling (RM) technique, in pseudo-code, is shown in Algorithm 11.2. θ_t^B is the background model and θ_t^F is the foreground model for each pixel. Let x_t be the intensity value (or the chromaticity vector) 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 [25]:

$$\hat{\theta}_t^B(x) = [1 - \beta_t] \cdot \theta_{t-1}^B(x) + \alpha_t \cdot H_\Delta(x - x_t) \tag{11.2}$$

where $x \in [0, 255]$ and θ_t^B is the non-normalized background pixel model at time t. The background model represents the probability of each pixel belonging to the background and must be normalized accordingly; i.e. $\sum_{x=0}^{255} \theta_t^B(x) = 1$.

$\hat{\theta}_t^B$, before normalization, 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 rate schedules, respectively. The kernel H should satisfy the following conditions: $\sum_x H_\Delta(x) = 1$ and $\sum_x x \times H_\Delta(x) = 0$.

Fig. 11.3 shows the process of using the proposed RM technique. 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.

Scheduled Learning

In order to speed up the modeling convergence and recovery from stale models a schedule for learning the background model at each pixel based on the pixel's history is utilized. 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 $\alpha_t = \frac{1-\alpha_0}{h(t)} + \alpha_0$. In this context, α_t is the learning rate at time t and $\alpha_0 = 1/256 \times \sigma_\theta$ is a small target rate. σ_θ is the model variance. The function $h(t)$ is a monotonically increasing function $h(t) = t - t_0 + 1$. We denote the time at which a sudden global change is detected as t_0.

According to this schedule the learning occurs faster ($\alpha_t = 1$) shortly after a global illumination change is detected and decreases to converge to the target rate α_0. In section 11.6 we discuss the effect of this schedule on improving the convergence and recovery speed.

Algorithm 11.2 - The proposed Recursive Modeling (RM) algorithm

Initialization($\Delta, \alpha_0, \beta, \kappa$, th)
for each frame at time t **do**
  for each pixel:=uv **do**
    $x_r \leftarrow \text{frame}_t$[u,v]
    // **Training Stage:**
    $\alpha_t \leftarrow \frac{1-\alpha_0}{h_t} + \alpha_0$
    updateDelta(Δ)
    $\theta_t^B(x) \leftarrow (1-\beta_t)\theta_{t-1}^B(x) + \alpha_t \times H_\Delta(x - x_t)$
    if $(\theta_t^B(x) \leq th)$ **then**
      $\theta_t^F(x) \leftarrow (1-\beta_t)\theta_{t-1}^F(x) + \alpha_t \times H_\Delta(x - x_t)$
    end if
    // **Classification Stage:**
    if $(\ln(\frac{median(\theta_t^B(x)}{\theta_t^F(x)} \leq \kappa)$ **then**
      frame[u,v] $\leftarrow$ background
    else
      frame[u,v] $\leftarrow$ foreground
    end if
    // **Update Stage:**
    updateKappa(κ)
    updateTh(th)
  end for
end for

11.4 Support Vector Data Description Background Modeling

This section presents analytical background modeling techniques based on single class Support Vector classification techniques [26].

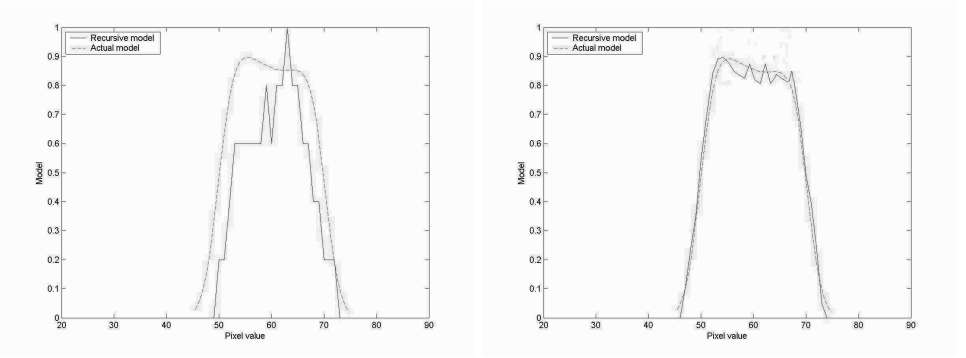
To overcome the main disadvantage of statistical learning tools in explicitly addressing the dependence of statistical models to probability estimation accuracy and the single-class classification problem a series of novel analytical background model learning tools are proposed in this section. In the following we present the SVDDM theory and the algorithm which detects foreground regions using this theory in detail.

11.4.1 The SVDD Theory

A normal data description is a description which gives a closed boundary around the data. A simple normal data description can be considered as a sphere with center $\mathbf{a}$ and radius $R > 0$, which encloses all of the training samples $\mathbf{x}_i$. The data description is achieved by minimizing the error function $F(R, \mathbf{a}) = R^2$ subject to $\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2$ for every smaple.

In order to allow for outliers in the training data set, the distance of each training sample $\mathbf{x}_i$ to the center of the sphere $\mathbf{a}$ should not be strictly smaller than R^2. However, large distances should be penalized. Therefore, after introducing slack variables $\epsilon_i \geq 0$ the minimization problem becomes:

$$F(R, \mathbf{a}) = R^2 + C \sum_i \epsilon_i \tag{11.3}$$



(a) (b)

FIGURE 11.3 Recursive modeling: Model after (a) 10 frames. (b) 467 frames (ⓒ(2009) Springer).

subject to the new constraints:

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \epsilon_i, \quad \forall i \tag{11.4}$$

where C controls the trade-off between the sphere volume and the description error.

The minimization problem in equation (11.3) can be solved by introducing the constraints of equation (11.4) to the error function using Lagrange multipliers:

$$L\left(R, \mathbf{a}, \alpha_i, \gamma_i, \epsilon_i\right) = R^2 + C\sum_i \epsilon_i - \sum_i \alpha_i \left[R^2 + \epsilon_i - \left(\|\mathbf{x}_i - \mathbf{a}\|^2\right)\right] - \sum_i \gamma_i \epsilon_i \tag{11.5}$$

where $\alpha_i \geq 0$ and $\gamma_i \geq 0$ are Lagrange multipliers. Optimization is achieved by minimizing for the value of L with respect to R the radius of the circle. According to our detailed explanation in [26], and after solving the minimization problems while replacing the results into equation (11.5) we have:

$$L = \sum_i \alpha_i(\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad \forall \alpha_i : 0 \leq \alpha_i \leq C \tag{11.6}$$

A set of α_i values can be achieved using equation (11.6). If a sample $\mathbf{x}_i$ satisfies the inequality in equation (11.4) its corresponding Lagrange multiplier will be zero ($\alpha_i = 0$). For all the training samples for which the equality in equation (11.4) is satisfied the Lagrange multipliers become greater than zero ($\alpha_i > 0$).

Note that the center of data descriptor from equation (11.3), $\mathbf{a}$, is a linear combination of the training samples. Only those training samples $\mathbf{x}_i$ which satisfy (11.4) by equality are needed to generate the description since their coefficients are not zero. These samples are called *Support Vectors* of the data descriptor.

The main assumption in the above theory states that the data description is normal –i.e. the data boundary is the smallest sphere surrounding the training samples. However, this simple, normal description is not enough for more complex data which does not fit into a sphere, i.e. the description needs more complex boundaries. To achieve a more flexible description, instead of a simple dot product of the training samples $(\mathbf{x}_i \cdot \mathbf{x}_j)$ in equation (11.6), we perform the dot product using a kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. This is done by using a mapping function Φ which maps the data into another (higher dimensional) space. By performing this mapping any complicated boundary (description of data) in low

dimension can be modeled by a hyper-sphere in a higher dimension. Several kernel functions have been proposed in the literature [31], among which the Gaussian kernel gives a closed data description:

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) \tag{11.7}$$

According to the above theory the proposed SVDDM method generates a Support Vector data description for each pixel in the scene using its history. These descriptions are then used to classify each pixel in new frames as a background or a novel/foreground pixel. In the following section the actual implementation of the system is presented.

11.4.2 The Algorithm

The methodology described above is used in our technique to build a descriptive boundary for each pixel in the background training frames in order to generate its model for the background. These boundaries are then used to classify their corresponding pixels in new frames as background or novel (foreground) pixels. There are several advantages in using the SVDD method in detecting foreground regions:

- It explicitly addresses the single-class classification problem. Existing statistical approaches try to estimate the probability of a pixel being background, and use (a set of) thresholds to classify pixels. It is impossible to have an estimate of the foreground probabilities, since there are no foreground samples in the training frames.

- The SVDD method has lower memory requirements compared to non-parametric density estimation techniques. This technique only requires a very small portion of the training samples, the *Support Vectors*, to classify new pixels.

Algorithm 11.3 shows the proposed algorithm in pseudo-code format. The only critical parameter is the number of training frames (`N`) that needs to be initialized. The Support Vector data description confidence parameter C is the target false reject rate of the system, which accounts for the system tolerance.

The background model in this technique is the description of the data samples (color and or intensity of pixels). The background training buffer is a First In First Out (FIFO) buffer with Round Robin replacement policy. The data description is generated in the training stage in which for each pixel Support Vectors and their Lagrange multipliers (α_i) are trained.

The Support Vectors and their corresponding Lagrange multipliers are stored as the classifier information for each pixel. This information is used for the classification step of the algorithm. The training stage can be performed off-line in cases where there are no global changes in the illumination or can be performed in parallel with the classification stage to achieve efficient foreground detection [26].

In the classification stage, for each frame its pixels are evaluated by their corresponding classifier to label them as background or foreground. To test each pixel $\mathbf{z}_t$ the distance to the center of the description hyper-sphere is calculated:

$$\|\mathbf{z}_t - \mathbf{a}\|^2 = (\mathbf{z}_t \cdot \mathbf{z}_t) - 2\sum_i \alpha_i(\mathbf{z}_t \cdot \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{11.8}$$

A pixel is classified as a background pixel if its distance to the center of the hyper-sphere is smaller or equal to the data descriptor's radius (i.e. $\|\mathbf{z}_t - \mathbf{a}\|^2 \leq R^2$).

Algorithm 11.3 - The SVDDM algorithm

```
// C: Confidence, N: No. of frames, σ: Bandwidth
Initialization(C, N, σ)
for each pixel:=uv do
    x_uv ← frames_u v[1 ··· N]
    // Training Stage:
    SVDD_uv ←trainSVDD(x_uv[1 ··· N])
end for
for each frame at time t do
    for each pixel:=uv do
        x_uv ← frames_u v[t]
        // Classification Stage:
        DV_uv ←classifySVDD(x_uv[t], SVDD_uv) // DV=Description Value
        if (DV_uv > 0) then
            pixel:=uv← foreground
        else
            pixel:uh← background
        end if
        // Update Stage:
        if (t%10) then
            SVDD_uv ←trainSVDD(x_uv[t − N ··· t])
        end if
    end for
end for
```

R^2 is the distance from the center of the hyper-sphere to its boundary which is also equivalent to the distance of each support vector to the center of the hyper-sphere:

$$R^2 = (\mathbf{x}_k \cdot \mathbf{x}_k) - 2\sum_i \alpha_i(\mathbf{x}_i \cdot \mathbf{x}_k) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{11.9}$$

11.5 Unified Framework for Target Detection and Tracking

Figure 11.4 illustrates the proposed framework for integrating target detection with tracking. This framework includes three main modules: (i) background modeling, (ii) target detection, and (iii) target tracking. The purpose of background modeling is to construct the intensity variation model of the pixels belonging to the background. Here, a SVR approach is exploited to fit the intensity distribution of background pixels using a Gaussian kernel. Target detection is performed by subtracting those pixels that fit the background model. Finally, the tracking module is used to establish a unique correspondence relationship among the detected targets over time.

In order to improve target-to-target correspondences over time, we calculate confidence coefficients based on shape, size, color and motion (i.e., velocity) information. Most importantly, the shape confidence coefficient computed in the tracking module is further exploited to iteratively update the threshold used in the detection stage to decide whether a pixel belongs to background or not. The detection threshold can be iteratively increased or reduced to improve detection results by considering the temporal correspondences of targets between adjacent frames. A voting-based strategy has been adopted to enhance matching results during target tracking under illumination changes and shape deformations due to

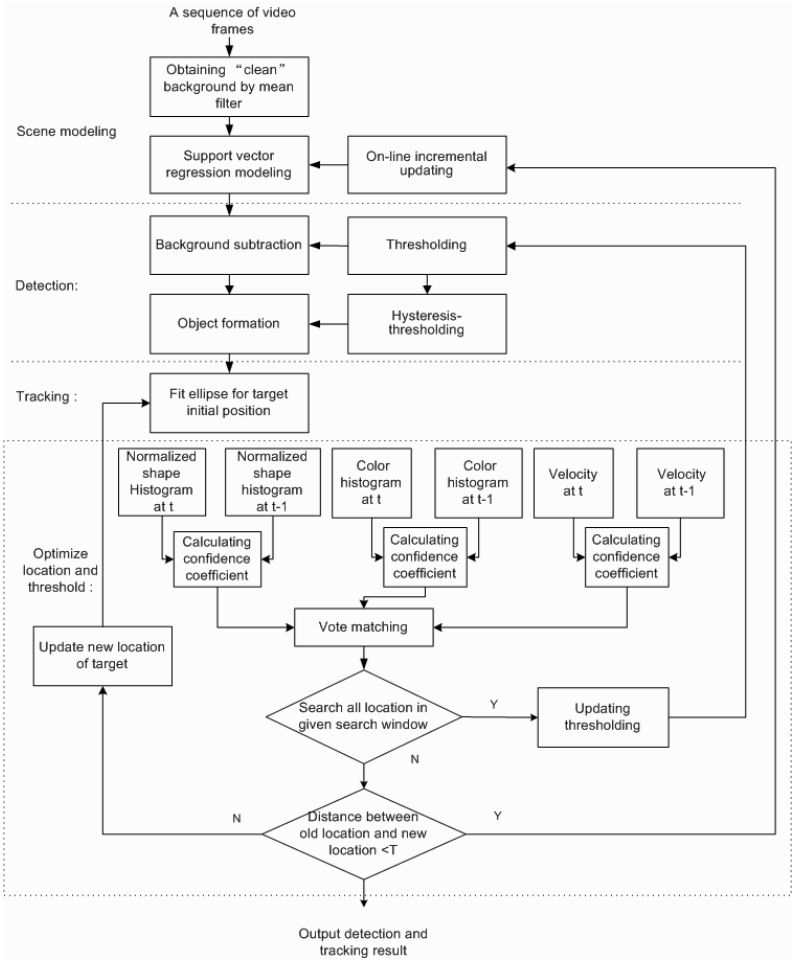


FIGURE 11.4 Framework for target detection and tracking (©(2008) Springer).

perspective projection. Specific details are provided in the following sections.

11.5.1 Background Modeling and Initialization of Target Location

In order to effectively detect the precise location of targets in a scene but also to avoid missing small targets, an accurate background model is required. Moreover, an effective way is required to incorporate background changes by updating the background model fast and effectively. The target detection state and background modeling may employ any of the statistical techniques presented in Section 11.3, the Support Vector Data Description in Section 11.4, or Support Vector Regression proposed in [18] and [13].

11.5.2 Integrating Target Detection with Tracking

When multiple targets are present, the proposed system maintains a list of targets which are actively tracked over time. The tracking is implemented through target feature match-

ing within continuous frames. This matching can build the correspondence relationships between the previously tracked targets and each potential targets at the current frame, detected by thresholding the outputs of background models. If the matching is successful and reliable, then the target is added to the list of targets for further tracking.

Specifically, the matching procedure searches iteratively for target candidates in the current frame that have similar shape and appearance with target models defined in the previous frame. First, we compute a similarity score based on weighted normalized shape projection histograms. Then, to discriminate between targets having similar shape, we compute additional information based on target's size, color and motion and apply a voting-based strategy. Targets that have been tracked consistently over a number of frames are added to the list of targets for tracking. This list is properly maintained to include new targets and remove targets that disappear from the scene. The same procedure is also used to handle undesired merging of targets. Potential targets in the list of detected objects are tracked using shape projection histograms only. The ratio between projection histograms of candidate and model targets, called confidence coefficient, is used to localize the targets accurately as well as to define the range of detection threshold.

In the following, we describe the framework for integrating target detection with tracking. First, we discuss our target representation scheme. Then, we describe the algorithm used to predict the location of targets in subsequent frames. Finally, we present the feedback mechanism for optimizing the detection threshold.

Target Representation

Our target representation scheme is based on shape, size, color and motion information. In order to make it robust to perspective projection, scale, and rotation transformations, we employ normalized shape projection histograms.

Normalized Shape Projection Histograms: The location of a target is denoted by (x_i, y_i) and it corresponds to the location of the best-fitting ellipse. To compute the projection histograms, we project the target horizontally and vertically by counting the number of pixels in each row and each column correspondingly. To make the projection histograms invariant to target orientation, first we transform the target to a default coordinate system. This is done in two steps: (i) we find the best-fitting ellipse of the target, and (ii) we align its major and minor axes with the x- and y-axis of the default coordinate system. The main assumption here is that the targets are approximately 2-D; this is a valid assumption in our application since the depth of the targets is much smaller compared to their distance from the camera.

Weighted Shape Projection Histograms: In order to reduce the effects of background noise and image outliers, we introduce weights to improve the robustness of matching. This is done by employing an isotropic kernel function $k(\cdot)$ in a similar way as in [3]. In particular, the role of the kernel function is to assign smaller weights to pixels farther away from the center bin of the projection histogram. Then, the weighted target model histograms, denoted as H_x^T and H_y^T are calculated according to [34].

To predict the location of targets in subsequent frames, we search a window of size $W \times H$. Candidate targets are identified in this window by thresholding the outputs of the background models [34].

Predicting Target Location

To find a target location in subsequent frames, we need to define a similarity measure between the target model, computed in previous frames, and the target candidate, detected in the current frame. A Manhattan distance based measure between the corresponding

weighted shape projection histograms of model and candidate targets is used.

To accurately localize a target in the search window, we minimize the objective function shown below in the case of horizontal shape projection histograms:

$$\begin{aligned}\Phi &= \min_k \sum_{m=1}^{M} [H_{x^k}^{C_k^S}(m) - H_x^T(m)] \\ &= \sum_k w_k \sum_{m=1}^{M} [H_{x^k}^{C_k^S}(m) - H_x^T(m)] \\ &= \sum_k w_k \sum_{x_i \in R} [H_{x^k}^{C_k^S}(x_i - x + M/2) \\ &\quad -H_x^T(x_i - x + M/2)] \\ &\quad \longrightarrow \min \text{over } S \text{ and } x^k \end{aligned} \tag{11.10}$$

where S is the threshold used to find the target candidates in the search window and w_k restricts the spatial position x^k of the target candidates around the geometric center x of the target model. $H_{x^k}^{C_k^S}(m)$ is the weighted shape projection histogram of the k-th target candidate detected using threshold S [34]. A similar calculation is applied on the vertical direction.

To perform the above minimization, an iterative scheme which gradually decreases the value of the threshold S used for target detection and changes the spatial center position of the search window is applied. The objective function is updated iteratively as follows:

$$\begin{aligned}\Phi(l) &= \sum_k w_k \sum_{m=1}^{M} [H_{x^k(l)}^{C_k^{S(l)}}(m) - H_x^T(m)] \\ &= \sum_k w_k \sum_{x_i \in R(l)} [H_{x^k(l)}^{C_k^{S(l)}}(x_i - x^k(l) + M/2) \\ &\quad -H_x^T(x_i - x + M/2)] \end{aligned} \tag{11.11}$$

where l corresponds to the iteration number.

Confidence Coefficient

A key issue in implementing the above idea is how to choose an appropriate function for decreasing S as well as to change the geometric center (x, y) of the candidate targets at each iteration l. For this, we use the ratio between the weighted shape projection histogram of the target model and the candidates. We refer to this ratio as the *confidence coefficient*. Its horizontal component is defined as follows:

$$\xi_x(l) = \sum_{x_i \in R(l)} \sqrt{\frac{H_{x^k(l)}^{S(l)}[x_i - x^k(l) + M/2]}{H_x^C[x - x^k(l) + M/2]}} \tag{11.12}$$

where x_i is the horizontal spatial location of pixels belonging to the candidate target $R(l)$. Similar calculation applies for the vertical location. The confidence coefficient becomes a weight factor in the iterative procedure used to update the spatial location of the targets as well as to select the threshold range for target detection (see next section). Specifically, using the confidence coefficient, the center of the search window is updated as follows:

TABLE 11.1 Per-pixel memory requirements for the AKDE, the RM and the SVDDM.

Memory Req.	Intensity	Chrominance	both	asymptotic
The AKDE [24]	$N+8$	$8N+20$	$9N+40$	$O(N)$
The RM [23]	1024	2048	3072	$O(1)$
The SVDDM	$[f(C,\sigma)\times 5]\geq 10$	$[f(C,\sigma)\times 8]\geq 24$	$f(C,\sigma)\geq 32$	$O(1)$

$$x^k(l) = x^k(l-1)\times\xi_x(l-1) \quad \text{and} \quad y^k(l) = y^k(l-1)\times\xi_y(l-1) \tag{11.13}$$

Adaptive Threshold Optimization

The confidence coefficient is also used to update the threshold S used in the target detection stage. Specifically, let us denote the threshold at the $l-1$ iteration as $S(l-1)$, then the threshold at the l iteration $S(l)$ is updated as follows:

$$S(l) = S(l-1) - \left[1 - \sqrt{\xi_x^2(l-1)+\xi_y^2(l-1)}\right] \tag{11.14}$$

This procedure decreases D_x and D_y while iteratively moving the spatial center of the search window closer to the geometric center of the target. The procedure terminates when the distance between the weighted shape projection histogram of target model and the target candidates is smaller than a threshold. However, when the confidence coefficient is too low, we increase the detection threshold to avoid under-segmentation which could cause differences in the shape of the targets in successive frames (see Fig. 11.10).

Tracking Multiple Targets

Using shape information alone to track multiple targets is not sufficient as it might lead to false matches. To eliminate such matches, we need to use additional information based on the target's size, color and motion. The key idea is using a voting strategy based on a majority rule – for more information please refer to [34].

It should be mentioned that the same equations used to compute the confidence coefficient in the case of shape projection histograms (i.e., Eq. (11.12)) can also be used to compute a confidence coefficient using size, color, and motion information.

11.6 Experimental Results and Comparison of Proposed Methods

In this section we compare the performance of proposed techniques using several real video sequences that pose significant challenges. In the real video experiments conducted below, $N = 300$ frames are used in the background training stage unless otherwise stated, corresponding to 10 seconds of the video's sequence. The background training buffer is a First In First Out (FIFO) buffer with Round Robin replacement policy.

11.6.1 Support Vector Modeling vs. Statistical Methods

From Table 11.1 we notice that the memory requirements of the AKDE technique presented in [24] and the SVDDM are lower than those of the RM method [23] if the number of training frames is small enough. That is, for situations when a small number of frames can cover most changes that occur in the background, by using a sliding window the AKDE method needs less memory than the RM technique. Note that SVDDM requires even less memory

(a) (b) (c) (d)

FIGURE 11.5 *Handshake* sequence (a). Detected foreground with AKDE (b), RM (c), SVDDM (d).

(a) (b) (c)

FIGURE 11.6 *Water* sequence (a), detected foreground region using AKDE (b) and SVDDM (c).

than the AKDE and like the RM its memory requirements are *independent* of the number of training samples.

Suitable Scenarios to Employ Statistical Background Modeling

For videos with rapidly changing backgrounds, the AKDE method has a better performance in terms of memory requirements and speed. Fig. 11.5 shows the detection results of the AKDE, RM and the SVDDM algorithms on the *Handshake* video sequence where the pixel values corresponding to monitors fluctuate rapidly. As it can be seen from this figure, capturing dependencies between chrominance features results in more accurate foreground regions (Fig. 11.5(b)), showing that AKDE performs better than both the RM and the SVDDM. Note that in this particular frame the color of foreground objects is very close to the background in some regions. The SVDDM technique results in very smooth and reliable foreground regions. Moreover, it uses the confidence factor C to guide the classification. This may lead to missing some parts of the foreground which are very close in color to the background.

Suitable Scenarios to Employ Support Vector Background Modeling

In videos with slowly changing or non-periodic backgrounds, the AKDE method needs more training frames to generate a good model for the background. This increases the memory requirements and drastically decreases its training and foreground detection speed. In these situations the SVDDM technique is a very good alternative, since its detection speed and memory requirements are independent of the number of training frames. Although the SVDDM like AKDE still require a large number of training frames, once the background model is trained the SVDDM only retains Support Vectors as pixel classifiers. As seen in Fig. 11.6 the SVDDM results are better than those of the AKDE.

(a) (b) (c)

FIGURE 11.7 *Mall* sequence (a), RM background model after 5 frames (b), and after 95 frames (c) (©(2009) Springer).

From this figure we can conclude that the SVDDM method has a better performance compared to the AKDE in situations where the background has a slow and irregular motion. Also in the AKDE there is a sliding window of limited size which may not cover all changes in the background resulting in an inaccurate probability density estimation but the model built in the SVDDM uses the decision boundaries of the single training class instead of bounding the training accuracy to the accuracy of the probability estimation.

Suitable Scenarios to Employ the RM Background Modeling

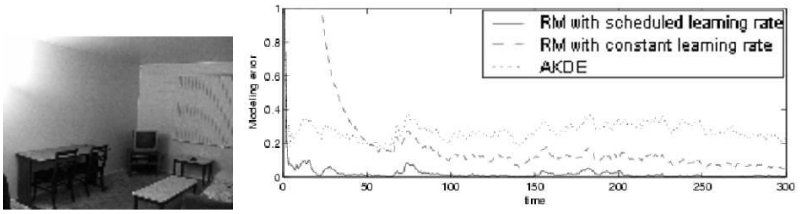
Fig. 11.7 shows the background model in the *Mall* video sequence in which the background is never empty. In this situation off-line methods fail unless a post-processing on the detected foreground regions is performed to generate models for uncovered parts of the background. In the RM method however, 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. Fig. 11.7(b) and (c) show the background model after 5 and 95 frames, respectively. 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.

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

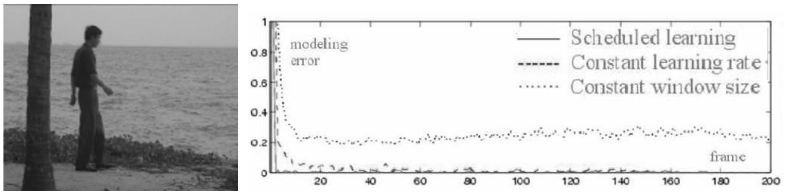
Fig. 11.8(a) shows an arbitrary frame of the *Room* video sequence. In this video a camera is held by hand and the scene only consists of the background. Fig. 11.8(b) compares the modeling error using different techniques. The modeling error using a constant window size in the AKDE (the dotted line) is between 20%-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.

Another important issue in the recursive learning is the convergence speed of the system (how fast the model converges to the actual background). Fig. 11.8(c)-(d) illustrates the convergence speed of the RM with scheduled learning, compared to constant learning and kernel density estimation with constant window size.

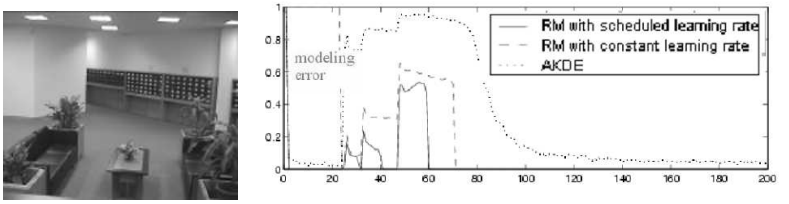
Fig. 11.8(e)-(f) shows the comparison of the recovery speed from an expired background



(a) (b)



(c) (d)



(e) (f)

FIGURE 11.8 *Room* sequence (a), modeling error (b). *Water* sequence (c), Convergence speed (d). *Lobby* sequence (e), Recovery speed (f).

model to the new one. In Fig. 11.8(e) lights go from on to off through three global but sudden changes occurring at frames 23, 31 and 47. 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.

Comparison Summary

Table 11.2 summarizes this study and provides a comparison between different traditional methods for background modeling and our proposed method. The comparison includes the classification type, memory requirements, computation cost and type of parameter selection.

11.6.2 Integration of Background Modeling and Tracking

The integration framework has been evaluated by detecting vehicles and pedestrians using visible and thermal video sequences. The visible video sequence was captured at a traffic

TABLE 11.2 Comparison between the proposed methods and traditional techniques.

	SVDDM	AKDE	RM	KDE	Sp-tmp [12]	MoG [20]	Wallflower [30]
Automated	Yes	Yes	Yes	No	No	No	No
Post proc.	No	No	No	No	Yes	No	No
Classifier	SVD	Bayes	MAP	Bayes	Bayes	Bayes	K-means
Memory req.*	$O(1)$	$O(N)$	$O(1)$	$O(N)$	$O(N)$	$O(1)$	$O(N)$
Comp. cost*	$O(N)$	$O(N)$	$O(1)$	$O(N)$	$O(N)$	$O(1)$	$O(N)$

$*$: Per-pixel
N: number of training frames

(a) (b) (c)

FIGURE 11.9 (See color insert.) Results using the proposed method (a) and frame-based (b). Final tracking results (c).

intersection and contains a total of two hours of video with a sampling rate 4 frames/second. The thermal video data was captured at a university campus walkway intersection over several days (morning and afternoon) using a Raytheon 300D thermal sensor core with 75mm lens mounted on an 8-story building [5].

In the following, the performance of the integration algorithm is demonstrated in terms of the following aspects: (1) detection alone, (2) integrating detection with tracking, (3) undesired merging of targets, and (4) comparisons with the state-of-the-art.

Fig. 11.9 presents comparison results between frame-based detection without feedback from tracking and the proposed method which integrates detection with tracking. Fig. 11.9 (a) and (c), show detection maps and tracking results using the proposed method. Fig. 11.9 (b) presents detection results using frame-difference and no threshold optimization. Among the results shown, it is interesting to note that the small target, labeled by a green rectangle in Fig. 11.9 (c), is very difficult to detect using frame-based detection and non-optimized thresholds as shown in Fig. 11.9 (b).

Table 11.3 shows quantitative comparisons in terms of true positives and false alarms for frame-based detection and the proposed approach. Obviously, the proposed approach has lower false alarm and higher true positive rates than frame-based detection.

TABLE 11.3 Quantitative comparisons in terms of True Positives (TP), False Alarms (FA), and Ground Truth (GT)

Data sets	Methods	Ground truth	True Positive	False Alarm
Visible video	Frame-based detection	346	296	30
	Integrating detection with tracking	346	340	5
Thermal video	Frame-based detection	371	371	35
	Integrating detection with tracking	371	371	0

Figs. 11.10 (a) and (b) show another quantitative comparison between frame-based detection and the proposed method by counting the number of pixels in two different seg-

mented regions moving away from the camera. The red curve indicates ground truth size. The green and blue curves show the performance of the proposed method and frame-based detection respectively. From the figure, the green curves are closer to the red curves, indicating that the proposed method higher accuracy compared to frame-based detection.

Fig. 11.10 (c) shows the adaptive threshold values over time for two targets with different motion characteristics (i.e., a car and a pedestrian). As it can be observed, the thresholds were iteratively decreased based on the confidence coefficient computed from the shape projection histogram matching process. To avoid under-segmentation, the threshold was reset to a higher value when the confidence coefficient fell below a certain value. Finally, Fig. 11.10 (d) demonstrates the average number of iterations for each frame.

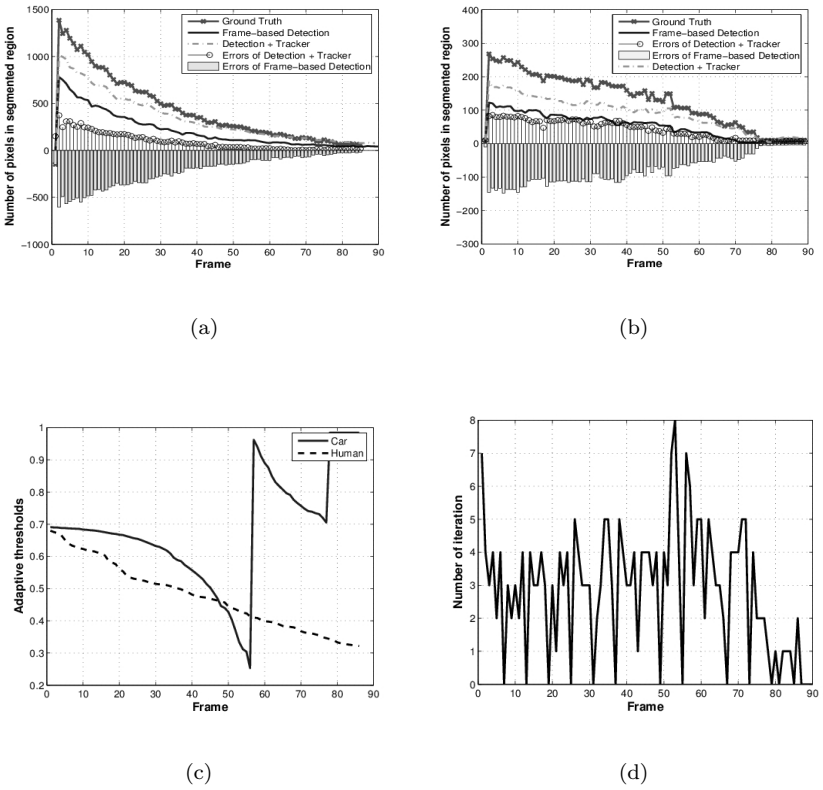


FIGURE 11.10 (See color insert.) (a),(b): Comparison results between the ground truth (red), frame-based detection (blue) and the proposed approach (green). (c): Adaptive threshold. (d): Average number of iterations (©(2008) Springer).

Tracking Merged Targets

The proposed detection and tracking approach can handle undesired target merging using the track-to-track stitching scheme reported in [1]. The voting-based matching scheme described is used to track accurately the targets when their shape is deformed due to perspective projection. Fig. 11.11 demonstrates how our proposed approach handles the undesired merging of targets. When two targets merge with each other, as shown in Fig. 11.11(b), a new track is assigned to these targets. After the undesired merge is resolved, the tracks are

recovered by stitching their new tracks with previous ones [1].

(a) (b) (c) (d)

FIGURE 11.11 An example of handling undesired merging of targets. Targets merge in (b) and the issue is resolved in (c).

11.7 Conclusion

In this chapter we have presented a number of novel techniques for learning background pixel models based on both statistical and analytical tools. As statistical modeling techniques, the non-parametric density estimation and recursive modeling approaches are discussed.

The advantage of the 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. The modeling method (RM) updates a statistical model for background pixels on-line. This method is superior and more robust than other techniques for situations in which background changes are slow and not periodic.

To overcome the issues inherent to statistical learning models, an alternative modeling tool is proposed to label pixels in video sequences into foreground and background classes using a Support Vector data description. The advantages of training Support Vectors for background modeling include:

- The model accuracy is not bounded to the accuracy of the estimated probability density functions.
- The memory requirements are lower than those of non-parametric techniques and are independent of the number of training samples.
- Support vector data description is more suitable for novelty detection since it explicitly models the decision boundary of the known class.

Furthermore, to enhance the quality of foreground detection, a framework for improving video-based surveillance by integrating target detection with tracking is presented. From Section 11.5, on-line SVR was used to model the background and to accurately detect the initial locations of the targets. To predict the location of targets in successive frames shape projection histograms were exploited. At the same time, a confidence coefficient based on shape matching was computed to suppress false alarms. Using weights derived from the confidence coefficient of shape matching, we were able to optimize the threshold used in the

target detection stage. Additional cues based on size, color, and motion were used to eliminate false positives when tracking multiple targets. Experiments show good performance, especially on small targets and targets undergoing perspective projection distortions.

References

1. A. Amer. Voting-based simultaneous tracking of multiple video objects. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(15):1448–1462, 2005.
2. Y. Bar-shalom and T. Fortmann. *Tracking and Data Association*. Academic Press, 1998.
3. D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *EEE Trans. on Pattern Analysis and Machine Intelligence*, 25(5):564–557, May 2005.
4. C. Criminisi, G. Gross, A. Blake, and V. Kolmogorov. Bilayer Segmentation of Live Video. *International Conference on Computer Vision and Pattern Recognition, CVPR 2006*, pages 17–22, June 2006.
5. J. W. Davis and M. A. Keck. A two-stage template approach to person detection in thermal imagery. *IEEE International Conference on Computer Vision and Pattern Recognition, CVPR 2005*, 2005.
6. R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, 2001.
7. A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *In proceedings of the IEEE*, 90:1151–1163., 2002.
8. N. Gordon, D. Salmond, and A. Smith. A novel approach to non-linear and non-Gaussian Bayesian state estimation. *Part-F: Radar and Signal Processing.*, 140:107–113, 1993.
9. K. Kim, D. Harwood, and L. S. Davis. Background updating for visual surveillance. *International Symposium on Visual Computing, ISVC 2005*, 1:337–346, December 2005.
10. G. Kitagawa. Non-Gaussian state-space modeling of nonstationary time-series. *Journal American Statistical Association*, 82:1032–1063, 1987.
11. D. Lee. Effective Gaussian mixture learning for video background subtraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):827–832, May 2005.
12. L. Li, W. Huang, I.Y. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, November 2004.
13. J. Ma and J. Theiler. Accurate on-line support vector regression. *Neural Computation*, 15:2683–2703, 2003.
14. N. Paragios and V. Ramesh. A MRF-based approach for real-time subway monitoring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:1030–1040, December 2001.
15. R. Pless, T. Brodsky, and Y. Aloimonos. Detecting independent motion: The statistics of temporal continuity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):68–73, 2000.
16. R. Pless, J. Larson, S. Siebers, and B. Westover. Evaluation of local models of dynamic backgrounds. *International Conference on Computer Vision and Pattern Recognition, CVPR 2003*, 2:73–78, June 2003.
17. Y. Sheikh and M. Shah. Bayesian object detection in dynamic scenes. *International Conference on Computer Vision and Pattern Recognition, CVPR 2005*, 1:74–79, June 2005.
18. A. Smola and B. Scholkopf. A tutorial on support vector regression. *NeuroCOLTS technical report Series NC2-TR-1998-030*, 1998.
19. C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. *International Conference on Computer Vision and Pattern Recognition, CVPR 1999*,

2:246–252, 1999.

20. C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.
21. Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection: A review. *IEEE Transactions on Pattern Anaysis and Machine Intelligence*, 28(5):694–711, 2006.
22. A. Tavakkoli, M. Nicolescu, and G. Bebis. Automatic robust background modeling using multivariate non-parametric kernel density estimation for visual surveillance. *International Symposium on Visual Computing, ISVC 2005*, LNSC 3804:363–370, December 2005.
23. A. Tavakkoli, M. Nicolescu, and G. Bebis. An adaptive recursive learning technique for robust foreground object detection. *International Workshop on Statistical Methods in Multi-Image and Video Processing in conjunction with ECCV 2006*, May 2006.
24. A. Tavakkoli, M. Nicolescu, and G. Bebis. Automatic statistical object detection for visual surveillance. *Southwest Symposium on Image Analysis and Interpretation, SSIAI 2006*, pages 144–148, March 2006.
25. A. Tavakkoli, M. Nicolescu, and G. Bebis. Robust recursive learning for foreground region detection in videos with quasi-stationary backgrounds. *International Conference on Pattern Recognition, ICPR 2006*, August 2006.
26. A. Tavakkoli, M. Nicolescu, and G. Bebis. A support vector data description approach for background modeling in videos with quasi-stationary backgrounds. *International Journal on Artificial Intelligence Tools*, 17(4):635658, August 2008.
27. A. Tavakkoli, M. Nicolescu, and G. Bebis. Non-parametric statistical background modeling for efficient foreground region detection. *International Journal of Machine Vision and Applications*, 20(6):395–409, October 2009.
28. D. Tax and R. Duin. Support vector domain description. *Pattern Recognition Letters*, 1(20):1191–1199, 1999.
29. D. Tax and R. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.
30. K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. *International Conference on Computer Vision, ICCV 1999*, 1:255–261, September 1999.
31. V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.
32. R. Verma, C. Schmid, and K. Mikolajczyk. Face detection and tracking in a video by propagating detection probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1215–1227, 2003.
33. J. Wang, H. Eng, A. Kam, and W. Yau. A framework for foreground detection in complex environments. *European Conference on Computer Vision, Workshop of Statistical Modeling for Video Processing*, pages 129–140, 2004.
34. X. Wang, G. Bebis, M. Nicolescu, M. Nicolescu, and R. Miller. Improving target detection by coupling it with tracking. *International Journal of Machine Vision and Applications*, 20(4):205–223, 2009.
35. L. Wixson. Detecting salient motion by accumulating directionary-consistent flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:774–780, August 2000.
36. C. Wren, A. Azarbayejani, T. Darrel, and A. Pentland. Pfinder: real-time tracking of human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.