Machine Vision and Applications (2009) 20:205–223
DOI 10.1007/s00138-007-0118-7

ORIGINAL PAPER

Improving target detection by coupling it with tracking

Junxian Wang · George Bebis · Mircea Nicolescu · Monica Nicolescu · Ronald Miller

Received: 27 December 2006 / Revised: 5 September 2007 / Accepted: 23 October 2007 / Published online: 6 February 2008

Abstract Target detection and tracking represent two fundamental steps in automatic video-based surveillance systems where the goal is to provide intelligent recognition capabilities by analyzing target behavior. This paper presents a framework for video-based surveillance where target detection is integrated with tracking to improve detection results. In contrast to methods that apply target detection and tracking sequentially and independently from each other, we feed the results of tracking back to the detection stage in order to adaptively optimize the detection threshold and improve system robustness. First, the initial target locations are extracted using background subtraction. To model the background, we employ Support Vector Regression (SVR) which is updated over time using an on-line learning scheme. Target detection is performed by thresholding the outputs of the SVR model. Tracking uses shape projection histograms to iteratively localize the targets and improve the confidence level of detection. For verification, additional information based on size, color and motion information is utilized. Feeding back the results of tracking to the detection stage restricts the range of detection threshold values, suppresses false alarms due to noise, and allows to continuously detect small targets as well as targets undergoing perspective projection distortions. We have validated the proposed framework in two different application scenarios, one detecting vehicles at a traffic intersection using visible video and the other detecting pedestrians at a university campus walkway using thermal video. Our experimental results and comparisons with frame-based detection and kernel-based tracking methods illustrate the robustness of our approach.

Keywords Visual surveillance · Background modeling · Support vector regression · Target detection · Target tracking · Integrate detection with tracking

J. Wang · G. Bebis (✉) · M. Nicolescu
Computer Vision Laboratory, University of Nevada, Reno, NV, USA
e-mail: bebis@cse.unr.edu

J. Wang
e-mail: junxian@cse.unr.edu

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

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

R. Miller
Vehicle Design R&A Department, Ford Motor Company, Dearborn, MI, USA
e-mail: rmille47@ford.com

1 Introduction

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 [1] (i.e., detect the target in the first frame and turn it over to the tracker in subsequent frames).

In detection alone schemes, various detection algorithms have been employed based on background subtraction [2–13], frame differencing [14,16], and optical flow [17]. Due to low computational complexity requirements, background subtraction is widely applied in real-time video-based surveillance systems when the cameras are fixed. In these systems, accurate and robust background modeling is a

Table 1 Overview of target detection approaches

Method	Model	Spectral	Spatial	Temporal	Decision	Updating scheme
Parametric-based approach						
W4 [2]	Minima and maxima of gray values	Pixel-based	–	Motion support map	Threshold	Re-estimate the model using new observations
Pfind [3]	Single Gaussian	Pixel-based	–	–	MAP	Adaptive filter
MoG [4]	Mixture of Gaussians	Pixel-based	–	–	Threshold	Adaptive filter
Matsuyama [5]	Gaussian of vector distances	Pixel-based	Neighboring block	Temporal co-occurrence	Threshold	Adaptive filter
DEWS [6]	Single Gaussian	Region-based	4-connected neighboring regions	–	Hysteresis threshold	Adaptive filter
Non-parametric-based approach						
Non-parametric [7]	Probability density of pixel density	Pixel-based	Neighboring pixels	–	Threshold	Re-estimate the model using new observations
Olvier [8]	Eigen background	Block-based	–	–	Threshold	Re-estimate the model using new observations
Monnet [9]	Principal components of block	Block-based	–	Auto-regressive model	Threshold	Incremental PCA
Wallflower [10]	Self-regression model	Block-based	4-connected blocks	–	Threshold	Updating prediction coefficients
Rittscher [11]	Discrete states	Pixel-based	–	–	Hidden Markov model	Dynamic state model
Seki [12]	Principal components of block	Block-based	Neighboring block	Color co-occurrence	Background likelihood	Adaptive filter
Liyuan [13]	Principal feature of pixels	Pixel-based	Gradient	Color co-occurrence	Bayesian decision	Adaptive filter

prerequisite step; however, due to significant intensity variations in images, statistical learning approaches have been exploited to provide more accurate models. In these approaches, the background is usually modeled using a parametric or non-parametric probability density estimation method. In the case of parametric approaches, it is assumed that the data follow a specific statistical distribution whose parameters are estimated from the training data. In the case of non-parametric approaches, the probability distribution of the data is assumed to have no specific form [7].

Table 1 presents an overview of parametric and non-parametric background subtraction methods for target detection. For each method, we report the model used, the level of information extracted (i.e., pixel-based, block-based, and region-based), the type of information extracted (e.g., spatial vs temporal), the decision rule, and the updating scheme (i.e., to deal with illumination changes). In the case of parametric background modeling, a single Gaussian [3] or Mixture of Gaussian (MoG) distributions [4] have been used for modeling the intensity distribution. A simple model was used in [2], based on the minimum and maximum gray-level background pixel values. Instead of using information at fixed spatial locations of the background scene, Dews [6] modeled homogeneous regions using a single Gaussian by considering color information. Non-parametric techniques estimate the density or color distribution using recent history values. Examples include kernel density estimation [7], Principal Component Analysis (PCA) [9], and one-step Wiener prediction filters [10].

In order to deal with illumination changes in outdoor environments, techniques based on background updating rely on slowly integrating background changes into the background model. One way that this can be implemented is by adding a portion of the difference between a new observation and the background model, where a learning factor is used to control the speed of insertion. Pixel statistics were used in [3] and MoG in [4] to estimate the learning factor and recursively update the parameters of the Gaussian distribution based on a simple adaptive filter. Alternatively, the background model can be updated by re-estimating the parameters of the model using new observations [7] or incremental PCA [9]. It should be noted that, many of the existing approaches utilize both spatial and temporal information to represent complex background scenes containing stationary and non-stationary information. As illustrated in Table 1, choosing an appropriate threshold value is critical for accurate and robust foreground detection. However, choosing a suitable threshold value in order to maximize true detections while minimizing false positives is a challenging issue. Hidden Markov models and probability decision models (i.e., Bayesian decision, maxima of a posteriori probabilities) are alternative approaches to using thresholding in foreground detection.

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 [19] and extended Kalman filters for linear predictions, as well as unscented Kalman filters [19] for non-linear predictions. The most general class of filters, however, includes particle filters [20], also called bootstrap filters [21], 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 [22].

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 [23]. 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 [24].

In this paper, we propose a framework for integrating target detection with tracking to improve detection results. Besides improving detection, integrating detection with tracking can help to initialize tracking automatically. In this framework, we employ SVR [26] to model the background along with an on-line learning scheme [27] to update it efficiently over time. The initial locations and representations of the targets are extracted by thresholding the outputs of the SVR model where the threshold is adaptively optimized using feedback from tracking. Tracking uses shape projection histograms to iteratively localize the targets and improve detection confidence between successive frames. Using feedback from tracking restricts the range of detection threshold values, suppresses false alarms due to noise, and allows for continuously detecting small targets, especially targets undergoing projection distortions. To reduce false positives, additional cues based on size, color and motion information are used when tracking multiple targets. We have validated the proposed framework in two different application scenarios, one detecting vehicles at a traffic intersection using visible video and the other detecting pedestrians in a university campus walkway using thermal video.

The rest of the paper is organized as follows. Section 2 gives a brief overview of the proposed target detection and tracking framework. Section 3 describes our background modeling approach and the process for initializing the location of targets. In Sect. 4, we discuss the process for integrating target detection and tracking. Our experimental results and comparisons are presented in Sect. 5. Finally, our conclusions and directions for future research are given in Sect. 6.

2 Framework for target detection and tracking

Figure 1 illustrates the proposed framework for integrating target detection with tracking. This framework includes three main modules: (a) background modeling, (b) target detection, and (c) 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

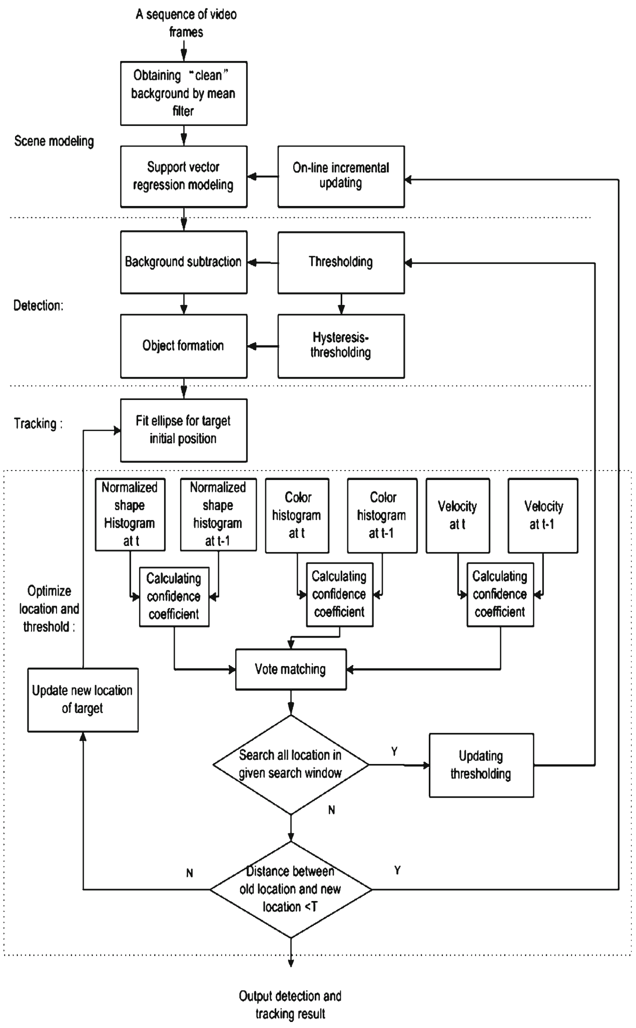

A sequence of video frames
Obtaining "clean" background by mean filter
Scene modeling
Support vector regression modeling
On-line incremental updating
Background subtraction
Thresholding
Detection:
Object formation
Hysteresis-thresholding
Tracking :
Fit ellipse for target initial position
Normalized shape Histogram at t
Normalized shape histogram at t-1
Color histogram at t
Color histogram at t-1
Velocity at t
Velocity at t-1
Optimize location and threshold :
Calculating confidence coefficient
Calculating confidence coefficient
Calculating confidence coefficient
Vote matching
Update new location of target
Search all location in given search window
Y
N
Updating thresholding
Distance between old location and new location <T
N
Y
Output detection and tracking result

Fig. 1 Framework for target detection and tracking

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

3 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. In this study, we propose using SVR for background modeling. SVR represents a statistical learning technique for estimating a function from a set of training data [26]. It estimates not only a function that does well on the training data, but also takes account of the possible deviation of the test data from the estimated function. Given a new input, SVR provides a tolerance which characterizes the regression estimate being away from the true estimate by a certain distance. To update the background model, we use an on-line SVR learning algorithm [27].

3.1 Background modeling using SVR

Given a set of training data, SVR fits a function by specifying an upper bound on a fraction of training data allowed to lie outside of a distance ε from the regression estimate. This type of SVR is usually referred to as ε-insensitive SVR [26]. For each pixel belonging to the background, a separate SVR is used to model it as a function of intensity. To classify a given pixel as background or not, we feed its intensity value to the SVR associated it and threshold the output of the SVR.

Let us assume a set of training data for some pixel p obtained from several frames $\{(x_1, y_1), \ldots, (x_l, y_l)\}$, where x_i corresponds to the intensity value of pixel p at frame i, and y_i corresponds to the confidence of pixel p being a background pixel. Once the SVR has been trained, the confidence of the pixel p in a new frame i, $f(x_i)$, is computed using the following linear regression function:

$$f(x_i) = \sum_{j=1}^{l} (a_i - a_j^*) k(x_i, x_j) + \zeta \tag{1}$$

where $k(x_i, x_j)$ is a kernel function.

The parameters a, a^* and ζ, called Lagrange multipliers, and are obtained by solving an optimization problem using the method of Lagrange multipliers [26]. Specifically, the solution of the ε-insensitive SVR corresponds to solving an optimization problem where the optimization criterion penalizes data points whose y_k-values differ from $f(x_k)$ by more than ε. Introducing Lagrange multipliers a, a^*, ζ, δ_i, μ_i and μ_i^*, the following quadratic objective function needs to be minimized:

$$\begin{aligned} W = {} & \frac{1}{2} \sum_{i}^{l} \sum_{j}^{l} (a_i - a_i^*) k(x_i, x_j)(a_j - a_j^*) \\ & - \sum_{i}^{l} y_i (a_i - a_i^*) + \varepsilon \sum_{i}^{l} (a_i + a_i^*) - \sum_{i}^{l} \delta_i (a_i + a_i^*) \\ & + \sum_{i}^{l} [\mu_i (a_i - C) + \mu_i^* (a_i^* - C)] + \zeta \sum_{i}^{l} (a_i - a_i^*) \end{aligned} \tag{2}$$

where $0 \le a_i, a_i^* \le C$ and $\sum_i (a_i - a_i^*) = 0$. By using different kernels, SVR implements a variety of estimation functions (e.g., a sigmoidal kernel corresponding to a two-layer sigmoidal neural network while a Gaussian kernel

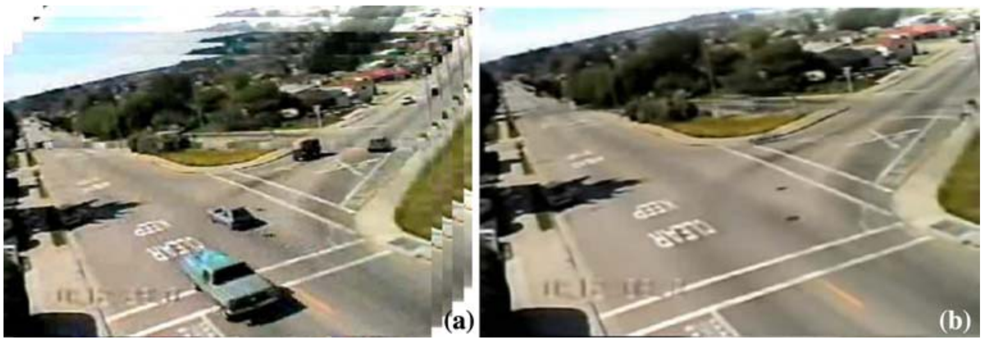


Fig. 2 Captured traffic scenes and the computed "clean" background scene using median filtering

corresponding to a radial basis function (RBF)). Here, we used a Gaussian kernel:

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{3}$$

By considering the dual optimization problem, Eq. (2) can be rewritten as follows:

$$\begin{aligned} L_W = {} & \frac{1}{2}\sum_{i=1}^{l}(a_i - a_i^*)(a_j - a_j^*) + \varepsilon\sum_{i=1}^{l}(a_i + a_i^*) \\ & -\sum_{i=1}^{l} y_i(a_i - a_i^*) + \sum_{i=1}^{l}[u_i(a_i - C) + u_i^*(a_i^* - C)] \\ & -\sum_{i=1}^{l}(\delta_i a_i + \delta_i^* a_i^*) + \xi\sum_{i=1}^{l}(a_i - a_i^*) \end{aligned} \tag{4}$$

where u_i,u_i^*, δ and ξ are also Lagrange multipliers. Optimizing this function leads to a set of conditions called the Karush-Kuhn-Tucker (KKT) conditions which state that at the point of the solution the product between dual variables and constraints has to vanish. In general, the KKT conditions are set of constraints that are necessary for a nonlinear programming solution to be optimal. In our problem, the KKT conditions can be expressed as follows:

$$\frac{\partial L_W}{\partial a_i} = \sum_{j=1}^{l} k(x_i, x_j)(a_j - a_j^*) + \varepsilon - y_i + \xi - \delta_i + u_i = 0 \tag{5}$$

$$\begin{aligned} \frac{\partial L_W}{\partial a_i^*} = {} & -\sum_{j=1}^{l} k(x_i, x_j)(a_j - a_j^*) \\ & + \varepsilon + y_i - \xi - \delta_i^* + u_i^* = 0 \end{aligned} \tag{6}$$

A detailed presentation of the SVR theory can be found in [26].

To illustrate the SVR-based background modeling approach, we use a video sequence captured at a traffic intersection assuming a fixed camera (see Fig. 2a). To train the SVR, we created B number of "clean" background images (i.e., without containing moving vehicles or pedestrians) by taking F number of successive frames and finding the median intensity value at each pixel location (see Fig. 2b). Here, we used a total of 90 frames to build B=30 clean background images using F=30 frames each time. It should be noted that, although all the images were captured using a fixed camera, there were still fluctuations in the intensity values in the "clean" background images due to light changes caused by outdoor environmental conditions. To train the SVR model assigned to a particular pixel location, we used the intensity values at this location from all clean images (i.e., x_i) and assigned a high confidence value to this pixel (i.e., y_i=1).

Figure 3 shows the results of background modeling at a fixed pixel location using Mixtures of Gaussians (MoG) and SVR. In each graph, the x-axis corresponds to the intensity of the pixel over the 30 "clean" background images (i.e., shown as red circles), while the y-axis corresponds to confidence of that pixel belonging to the background (i.e., set to 1). Figure 3a shows the SVR-based model, whereas Fig. 3b, c shows the MoG-based models using four and two Gaussians correspondingly. As it can be observed, SVR can find a better solution than MOG. Figure 3d shows an SVR-based solution corresponding to a different pixel location.

3.2 Extracting initial target locations

Given the SVR-based background model, the intensity of each pixel in a new frame forms the input to the SVR. The output of the SVR represents the confidence that a given pixel belongs to the background. Eventually, a pixel is labelled as background if its confidence is between a low threshold S_l and a high threshold S_h (i.e., hysteresis thresholding). Specifically, a binary foreground detection map $M_{x_i}^t$ is formed at frame t as follows:

$$M_{x_i}^t = \begin{cases} 0, & \text{background}, \quad S_l < f(x_i) < S_h \\ 1, & \text{foreground} \quad \text{otherwise} \end{cases} \tag{7}$$

where $f(x_i)$ is the SVR output and $S = \{S_l, S_h\}$ are the initial thresholds. Then, for each region in the binary map, we fit an

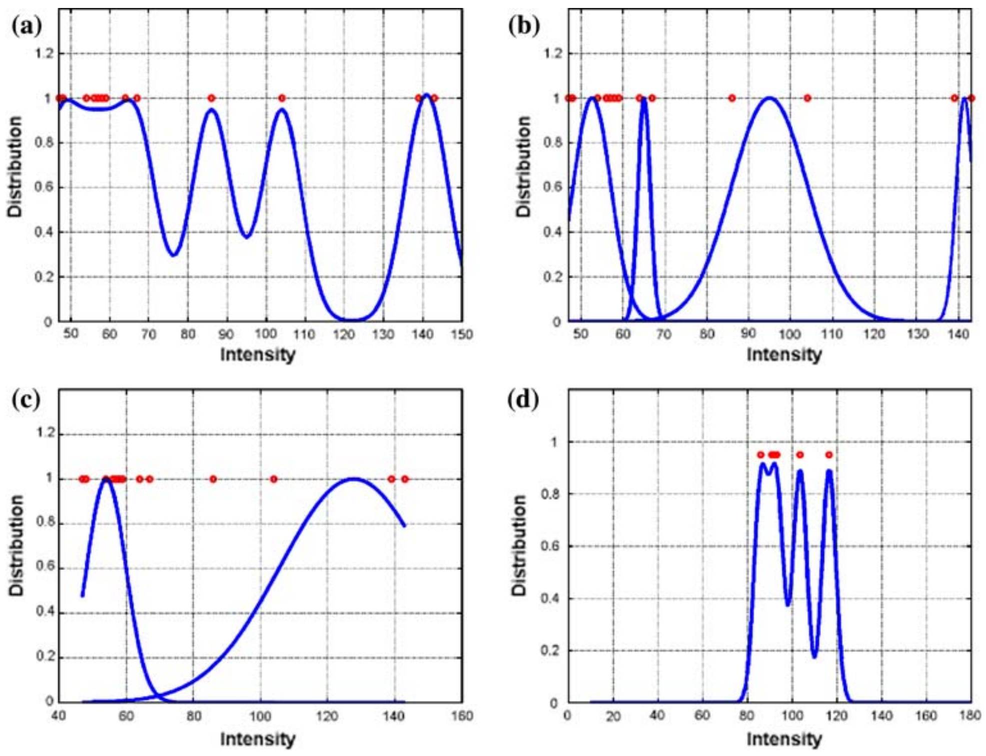


Fig. 3 Different solutions for background modeling: **a** SVR solution, **b** MoG solution using 4 Gaussians, **c** MoG solution using 2 Gaussians, **d** SVR solution at a different pixel location

ellipse. Figure 4 shows an incoming frame and corresponding detection results. Detection results using the low and high thresholds separately are shown in Fig. 4b, c. The results using hysteresis thresholding are shown in Fig. 4d. The initial location of each target was determined by best ellipse-fitting as shown in Fig. 4e.

3.3 On-line SVR learning

To update the background model over time, we need an efficient method that avoids re-training the SVR models which is very expensive. Here, we used an efficient on-line SVR learning algorithm which updates the SVR function whenever new training data becomes available [27].

The main idea behind on-line SVR learning is to gradually change the difference of Lagrange multipliers a and a^* corresponding to the new training data in a finite number of steps until the KKT conditions are satisfied [27]. Figure 5a, b illustrates the on-line SVR learning procedure where the training data is shown by red circles. After training, the estimated regression function in Fig. 5a contains a single peak (i.e., red dashed line). When new data comes along, shown by black circles, the regression function is updated using on-line learning. In this example, the regression function becomes bimodal (i.e., black line). Figure 5b shows another case where the regression function contains multiple peaks. In this case, the number of peaks does not change before (i.e., red dashed line) and after (i.e., black line) the addition of new examples, however, the peaks do shift to the right. As it can be observed, the SVR background model has the ability to adapt to new incoming data (i.e., the regression function shifts along the direction of new incoming data).

4 Integrating target detection with tracking

When multiple targets are present, out system maintains a list of targets which are actively tracked over time. The tracking is implemented through target feature matching within continuous frames. This matching can build the correspondence relationships between the previous tracked targets and each potential targets at the current frame, detected by thresholding the outputs of SVR-based background model as described in Sect. 3. If the matching is successful and reliable, then the target is added to the list of targets for further tracking.

Specifically, our 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

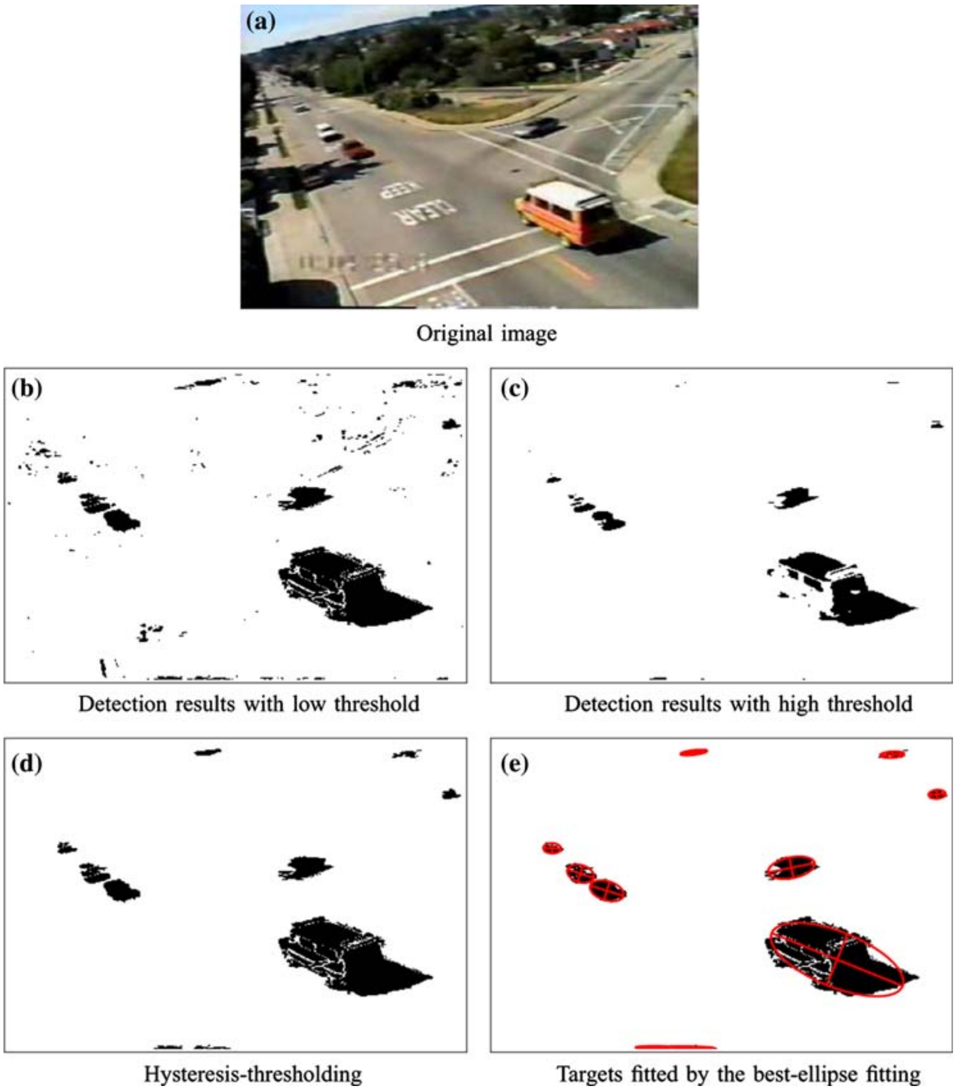

(a)
Original image
(b)
Detection results with low threshold
(c)
Detection results with high threshold
(d)
Hysteresis-thresholding
(e)
Targets fitted by the best-ellipse fitting

Fig. 4 Background subtraction with low and high thresholds and initial target locations represented by best-ellipse fitting

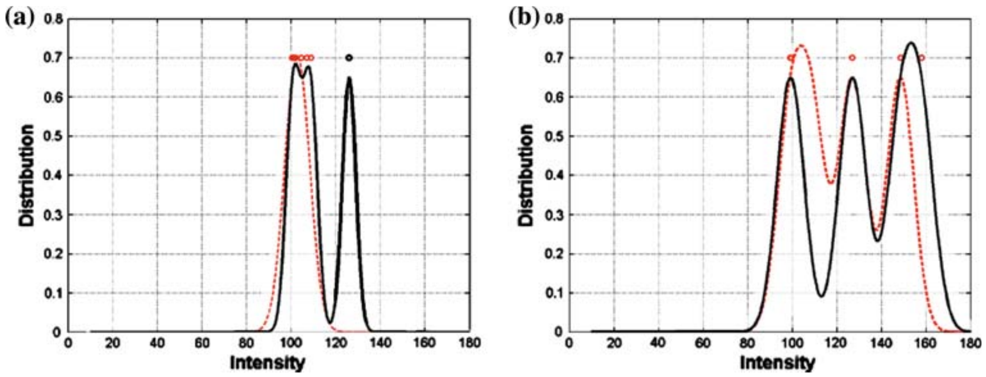


Fig. 5 Illustration of on-line SVR learning

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 occlusions. Targets in the list of targets 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.

4.1 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.

4.1.1 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: (a) we find the best-fitting ellipse of the target, and (b) we align its major and minor axes with the x- and y-axes 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.

Since projection histograms are sensitive to the location and size of the targets, we normalize them by shifting the middle bin of the histogram to the geometric center of the target and resizing the number of bins to a fixed size. Specifically, the normalized horizontal (i.e., $\bar{H}_x$) and vertical (i.e., $\bar{H}_y$) shape projection histograms are defined as follows:

$$\begin{aligned}&\bar{H}_x(m) = \{(x_i, y)|(x_i, y) \in R\}\\ &m = x_i - x + M/2\end{aligned} \tag{8}$$

$$\begin{aligned}&\bar{H}_y(n) = \{(x, y_j)|(x, y_j) \in R\}\\ &n = y_j - y + N/2\end{aligned} \tag{9}$$

where (x, y) is the geometric center of the target (i.e., the center of the best-fitting ellipse), m, n are indices, and M, N are the number of bins in the horizontal and vertical projection histograms.

4.1.2 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 [22]. In particular, the role of the kernel function is to assign smaller weights to pixels farther away from the center bin of the project histogram. Then, the weighted target model histograms, denoted as H_x^T and H_y^T, are defined as follows:

$$\begin{aligned}H_x^T(m) &= \frac{\bar{H}_x^T(m) + k(\cdot)}{\sum_{m=1}^{M} \bar{H}_x^T(m) + k(\cdot)}\\ H_y^T(n) &= \frac{\bar{H}_y^T(n) + k(\cdot)}{\sum_{n=1}^{N} \bar{H}_y^T(n) + k(\cdot)}\end{aligned} \tag{10}$$

where $k(x_i, y_j) = c - [(x_i - x)^2 + (y_j - y)^2]$, and $c = (w/2 + 1)^2 + (h/2 + 1)^2$ where the size of the target is $w \times h$.

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 SVR models. The weighted target candidate projection histograms, denoted as H_x^C and H_x^C, are defined as follows:

$$\begin{aligned}H_x^C(m) &= \frac{\bar{H}_x^C(m) + g(\cdot)}{\sum_{m=1}^{M} \bar{H}_x^C(m) + g(\cdot)}\\ H_y^C(n) &= \frac{\bar{H}_y^C(n) + g(\cdot)}{\sum_{n=1}^{N} \bar{H}_y^C(n) + g(\cdot)}\end{aligned} \tag{11}$$

where $g(x_i, y_j) = c - \left\{[(x_i - x)/h]^2 + [(y_j - y)/h]^2\right\}$, and c is calculated based on the size $h = W \times H$ of the search window. Figure 6 shows an example of shape projection histograms.

4.2 Predicting target location

To find the location of a target 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. Here, we use a similarity measure based on the Manhattan distance between the corresponding weighted shape projection histograms of model and candidate targets:

$$\begin{aligned}D_x &= \sum_{m=1}^{M}[H_x^C(m) - H_x^T(m)]\\ D_y &= \sum_{n=1}^{N}[H_y^C(n) - H_y^T(n)]\end{aligned} \tag{12}$$

To accurately localize a target in the search window, we minimize the objective function shown below in the case of horizontal shape projection histograms (similar derivations apply in the case of vertical shape projection histograms):

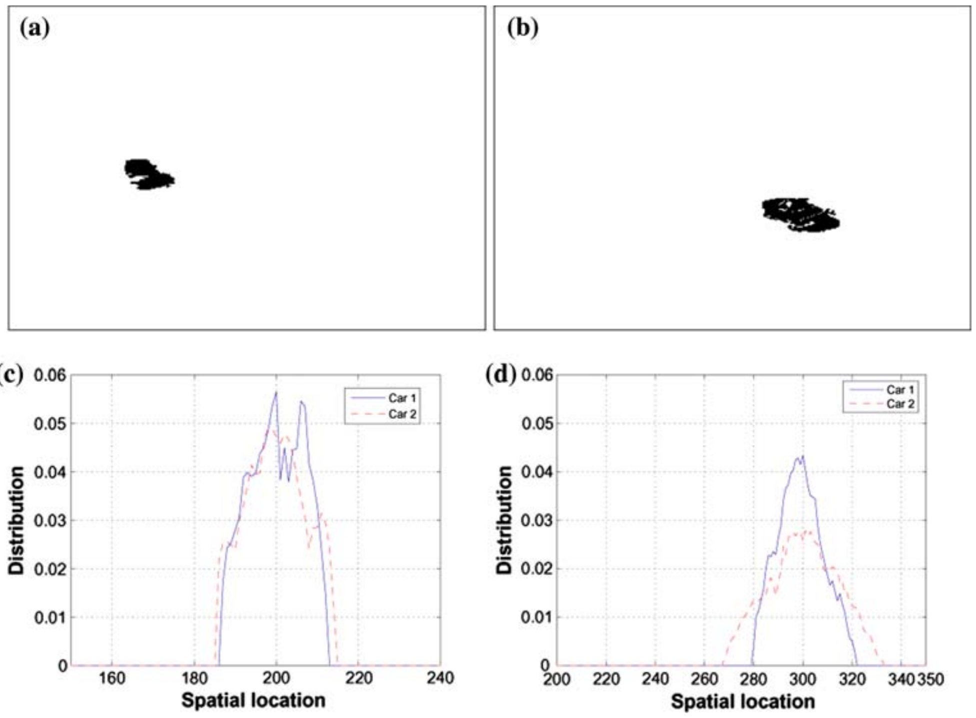


Fig. 6 **a**, **b**: Targets, **c** vertical shape projection histograms of the targets, **d** horizontal shape projection histograms of the targets

$$\begin{aligned}\Phi &= \min_k \sum_{m=1}^{M} \left[H_{x^k}^{C_k^S}(m) - H_x^T(m) \right] \\ &= \sum_k w_k \sum_{m=1}^{M} \left[H_{x^k}^{C_k^S}(m) - H_x^T(m) \right] \\ &= \sum_k w_k \sum_{x_i \in R} \left[H_{x^k}^{C_k^S}(x_i - x + M/2) \right. \\ &\quad \left. - H_x^T(x_i - x + M/2) \right] \qquad (13) \\ &\quad \longrightarrow \text{min over } S \text{ and } x^k\end{aligned}$$

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. In practice, we evaluate the weighted sum of $\sum_{m=1}^{M}[H_{x^k}^{C_k^S}(m) - H_i^T(m)]$, instead of evaluating $\min_k \sum_{m=1}^{M}[H_{x^k}^{C_k^S}(m) - H_x^T(m)]$ which is difficult to compute.

To perform the above minimization, we employ 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 as shown in the next two subsections. The objective function is updated iteratively as follows:

$$\begin{aligned}\Phi(l) &= \sum_k w_k \sum_{m=1}^{M} \left[H_{x^k(l)}^{C_k^{S(l)}}(m) - H_x^T(m) \right] \\ &= \sum_k w_k \sum_{x_i \in R(l)} \left[H_{x^k(l)}^{C_k^{S(l)}} \left(x_i - x^k(l) + M/2 \right) \right. \\ &\quad \left. - H_x^T(x_i - x + M/2) \right] \qquad (14)\end{aligned}$$

where l corresponds to the iteration number.

4.3 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* and it is defined as follows:

$$\xi_x(l) = \sum_{x_i \in R(l)} \sqrt{\frac{H_{x^k(l)}^{S(l)}[x_i - x^k(l) + M/2]}{H_x^C[x - x^k(l) + M/2]}} \qquad (15)$$

$$\xi_y(l) = \sum_{y_i \in R(l)} \sqrt{\frac{H_{y^k(l)}^{S(l)}[y_i - y^k(l) + N/2]}{H_y^C[y - y^k(l) + N/2]}} \qquad (16)$$

where x_i and y_i are the spatial location of pixels belonging to the candidate target $R(l)$.

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:

$$x^k(l) = x^k(l-1) \times \xi_x(l-1)$$
$$y^k(l) = y^k(l-1) \times \xi_y(l-1) \tag{17}$$

4.4 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:

$$S(l) = S(l-1) - \left[1 - \sqrt{\xi_x^2(l-1) + \xi_y^2(l-1)}\right] \tag{18}$$

The above iterative procedure decreases D_x and D_y while moving the spatial center of the search window iteratively closer to the geometric center of the target. The iterative procedure terminates when the distance between the weighted shape projection histogram of target model and the target candidates is smaller than a given value. 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. 12).

4.5 Tracking multiple targets

Using shape information alone (i.e., shape projection histograms) to track multiple tatgets 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.

Specifically, suppose that O_t^p and O_{t-1}^l correspond to the pth target in the current t frame and the lth target in the $t-1$ frame respectively. The correspondence between O_t^p and O_{t-1}^l is established if O_t^p is spatially close to O_{t-1}^l, and both targets have sufficiently close appearances in terms of shape, size, color, and motion information. The condition on the spatial locations of the targets is given as follows:

$$d(O_t^p, O_{t-1}^l) < R^l \tag{19}$$

where $d(O_t^p, O_{t-1}^l)$ is the Euclidean distance between the spatial locations of O_t^p and O_{t-1}^l, R^l is the radius of the search area centered at O_t^p. The condition on the appearance of the targets is given as follows:

$$\xi_f(O_t^p, O_{t-1}^l) > T_{\xi_f} \tag{20}$$

where $\xi_f(O_t^p, O_{t-1}^l)$ is the confidence coefficient between O_t^p and O_{t-1}^l in terms of a feature f (i.e., shape, size, color, and motion). T_{ξ_f} is the threshold used for that feature. The above two equations yield a total of five constraints (i.e., one constraint based on Eq. (19)) and four constraints based on Eq. (20). If more than half of the constraints are satisfied for a number of frames, then the correspondence between targets O_t^p and O_{t-1}^l is established and the target is added to the list of targets for tracking.

It should be mentioned that the same equations used to compute the confidence coefficient in the case of shape projection histograms (i.e., Eqs. (15) and (16)) can also be used to compute a confidence coefficient using size, color, and motion information.

5 Experimental results

The proposed framework has been evaluated by detecting vehicles and pedestrians using visible and thermal video sequences. The visible video sequence was captured at a traffic intersection and contains a total of two hours video with a sampling rate 4 frames/s. The thermal video data were captured at a university campus walkway intersection over several days (morning and afternoon) using a Raytheon 300D thermal sensor core with 75 mm lens mounted on an 8-story building [29]. Compared to visible image sensors, infrared image sensors exploit a combination of temperature difference, emissivity difference and "cold sky" reflection to generate images with high contrast between the target and the background. Infrared image sensor-based detection may enhance system performance for nighttime surveillance and has a relative higher resistance to poor weather (snow, rain and fog) [29]. In the following, we demonstrated the performance of the proposed algorithm in terms of the following aspects: (1) detection alone, (2) integrating detection with tracking, (3) new targets, (4) occlusion, and (5) comparisons using frame-based detection, kernel-based tracking, and our proposed method.

5.1 Results using detection alone

First, we demonstrate the performance of a system employing frame-based detection and SVR-based background modeling, without feedback from the tracking stage for threshold

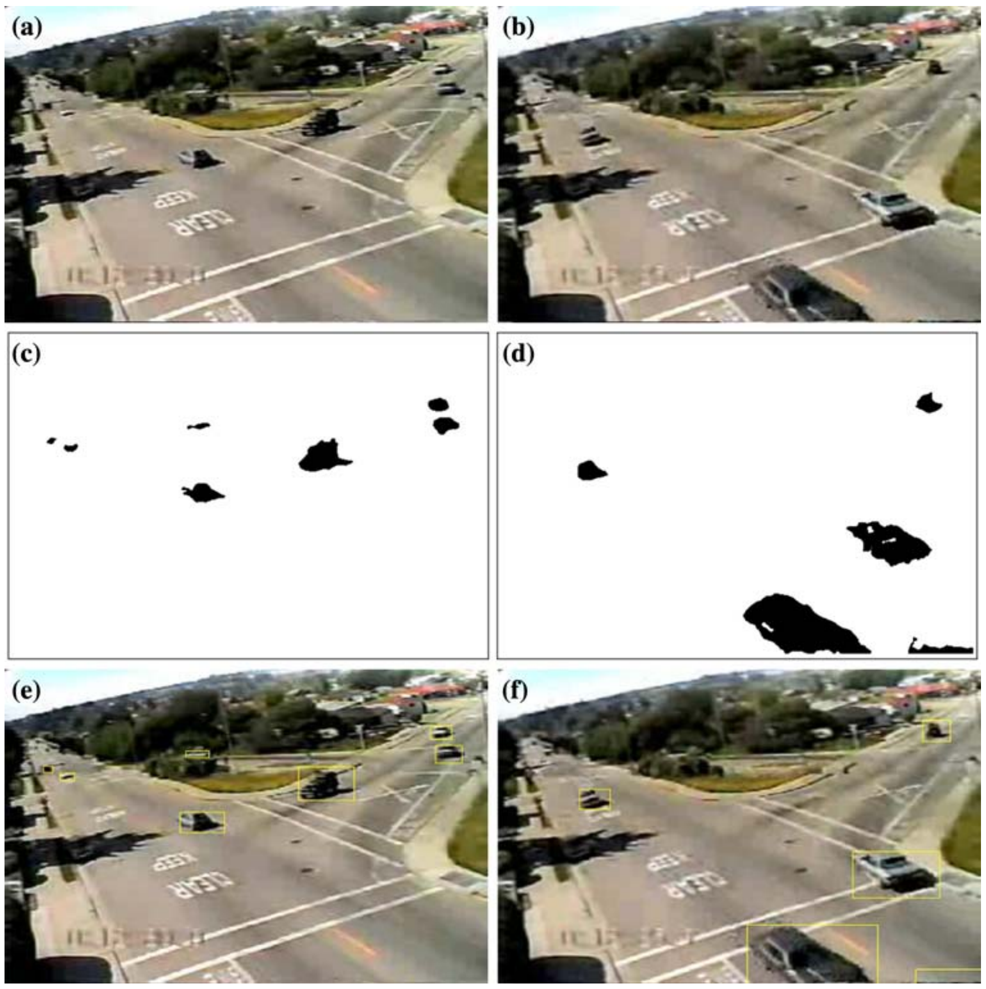


Fig. 7 Detection results using SVR-based background modeling, without feedback from tracking

optimization. Figures 7 and 8 show the locations of the targets found using this approach. Figure 9 shows comparison results between SVR-based background modeling and AdaBoost [29]. Our first observation is that SVR-based detection produces more accurate detections (i.e., the window enclosing the targets is much narrower). Moreover, it can observed in the left part of Fig. 9, a pedestrian who was detected as two separate entities by Adaboost (i.e., over-segmentation). On the right part of Fig. 9, however, the same pedestrian was detected as a single entity using SVR. Nevertheless, the performance of detection without employing some kind of feedback from tracking depends heavily on the choice of the threshold. If the threshold is not chosen properly, we might end up with many false alarms as shown in Fig. 4.

5.2 Results using integration of detection with tracking

Figures 10 and 11 present comparison results between frame-based detection without feedback from tracking and the proposed method which integrates detection with tracking. Each target is tracked and labeled with rectangles having different colors. The 1st and 2nd rows of Fig. 10, show tracking results and detection maps using the proposed method. The last row 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 the 1st row of Fig. 10, is very difficult to detect using frame-based detection and non-optimized thresholds as shown in the 3rd row of Fig. 10. On the other hand, the proposed method shows more accurate detection results by optimizing the threshold. In particular, the proposed method suppresses false alarms as shown in the 2nd row of Fig. 11. Table 2 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.

Figure 12a , b shows another quantitative comparison between frame-based detection and the proposed method by counting the number of pixels in two different segmented

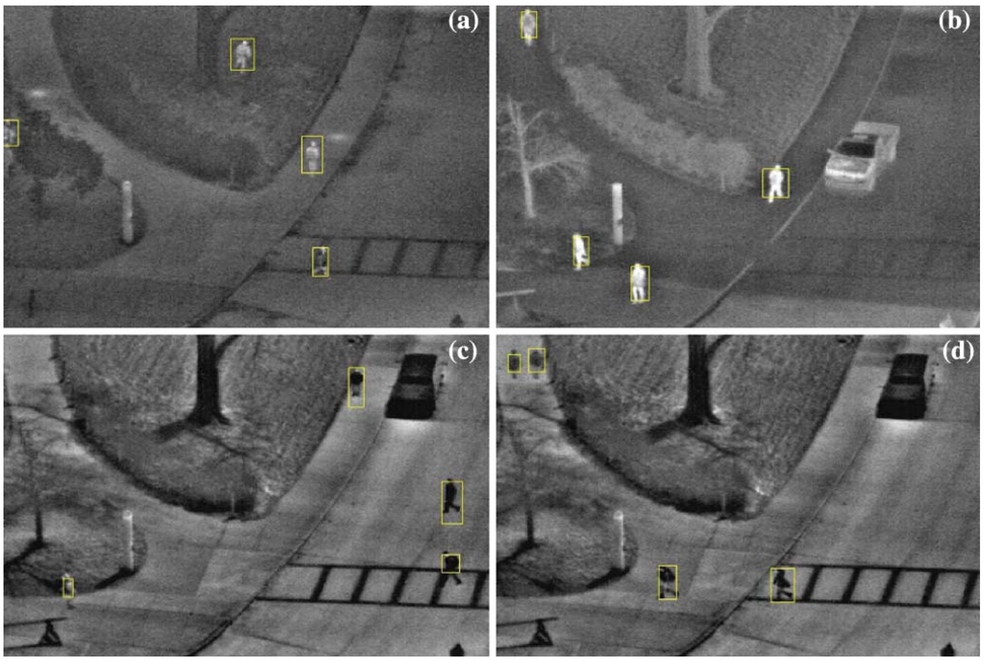


Fig. 8 Detection results using SVR-based background modeling, without feedback from tracking

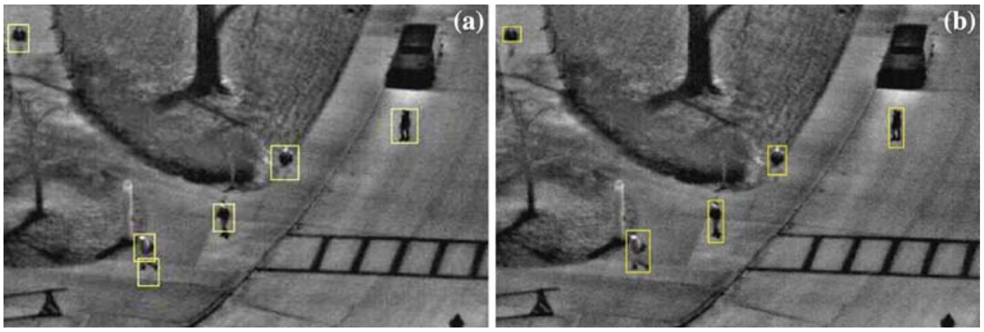


Fig. 9 Comparison results between AdaBoost [29] (*left*) and SVR (*right*)

regions. The red curve indicates ground truth information (i.e., the true number of pixels in the segmented region). The reason that the number of pixels decrease over time is because the target becomes smaller and smaller over time due to moving away from the camera. The green and blue curves show the performance of the proposed method and frame-based detection respectively. As it can be observed, the green curves are closer to the red curves, indicating that the proposed method makes less errors compared to frame-based detection.

Figure 12c 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 re-set to a higher value when the confidence coefficient fell below a certain value. Figure 12d demonstrates the average number of iterations for each frame. As it can be observed, the time complexity of this step is not high.

5.3 Tracking new targets

In our framework, target detection guides tracking by updating its predictions based on the latest observations. Therefore, tracking can quickly respond to the appearance of new targets. In addition, our tracking algorithm has two different modes, one that tracks targets actively and one that tracks targets temporarily in order to avoid propagating false alarms from the target detection stage to the tracking stage. The transition between temporary tracking to active tracking is based on detection continuity.

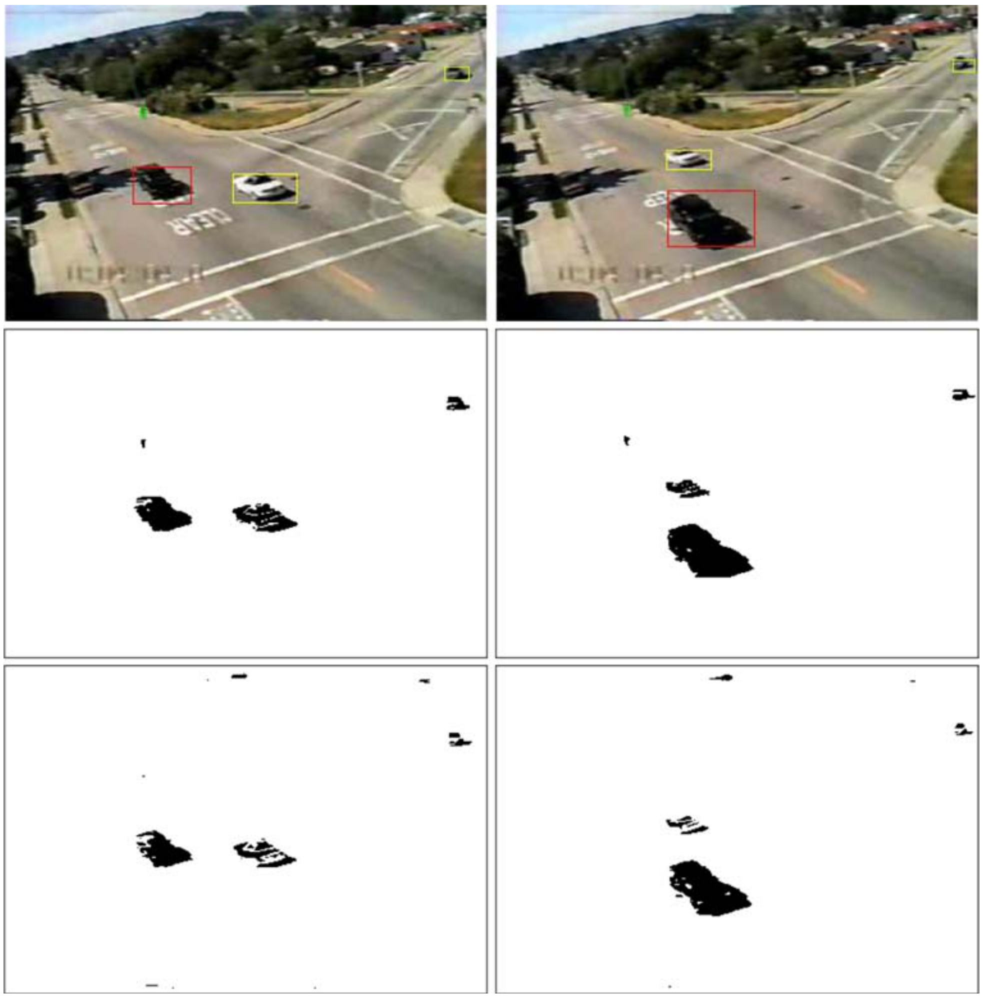


Fig. 10 Comparison results between the proposed method and frame-based detection using visible video: *1st* and *2nd rows* tracking results and corresponding detections using the proposed method; *3rd row* detection results using frame-based detection

Figure 13 shows target detection and tracking results when a new vehicle appears in the scene. It should be noted that the new target is not tracked immediately as shown in the 3rd frame of Fig. 13 although it has been detected. Initially, the target is tracked on a temporary basis. Only when the new target is detected continuously, then it is tracked actively based on the latest detection results as shown in the 4th frame of Fig. 13.

5.4 Tracking occluded targets

The proposed detection and tracking approach can handle target occlusion using the track-to-track stitching scheme reported in [30]. The voting-based matching scheme described in Sect. 4 is used to track accurately the targets when their shape is deformed due to perspective projection. Figure 14 demonstrates how our proposed approach handles target occlusion. When two targets occlude each other, as shown in Fig. 14b, a new track is assigned to the occluded targets. After occlusion, the tracks are recovered by stitching their new tracks with previous tracks as described in [30]. Figure 15 shows an example of tracking multiple targets and their trajectories. For each target, its trajectory is shown in the same color as its frame box.

5.5 Comparison with kernel-based tracking

In this section, we present comparison results between kernel-based tracking [22] and the proposed approach. Figure 16 shows tracking results for frames 4, 22, 26, 39 of a test sequence using the proposed method (1st row) and kernel-based tracking (2nd row). The 3rd row of Fig. 16 shows the detected targets using the proposed method. In order to make the comparison fair, kernel-based tracking was initialized using the initial target locations found by our approach,

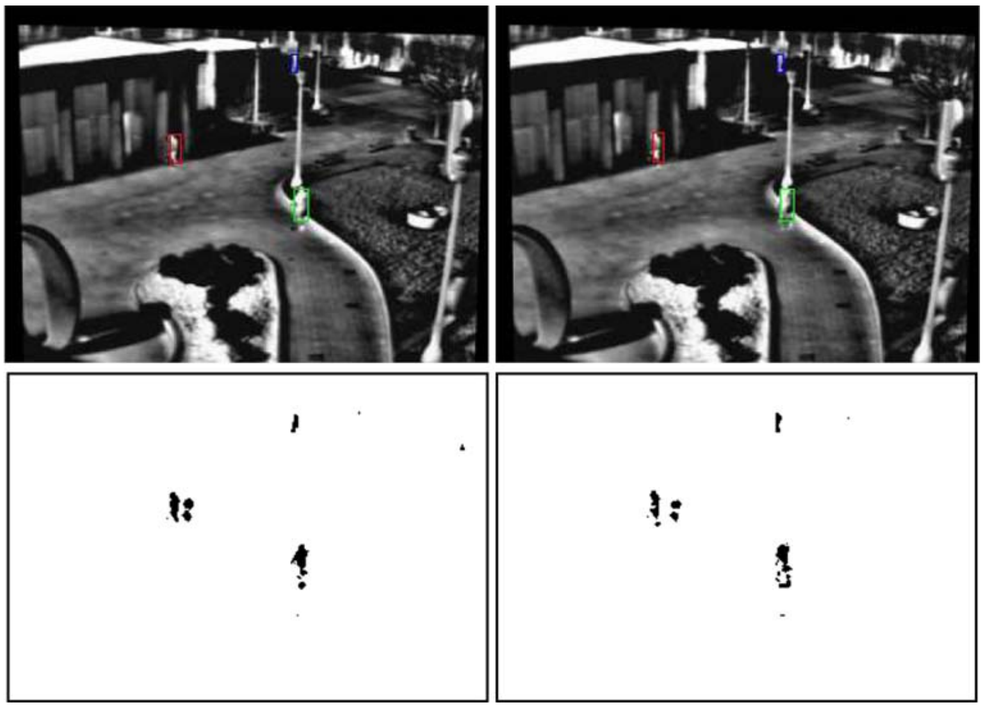


Fig. 11 Comparison results between the proposed method (*1st row*) and frame-based detection (*2nd row*) using thermal video: As it can be observed, frame-based detection yields false positives

Table 2 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

shown in the first column of Fig. 16. As it can be observed, kernel-based tracking has difficulties with tracking small targets (e.g., a small human walking along the road) and targets with perspective projection distortions. On the other hand, the proposed approach can handle these cases due to the adaptive thresholding scheme.

5.6 Running time requirements

Our algorithm was implemented in C++ using Microsoft foundation class (MFC) interface and runs on a 2.4 GHz standard desktop PC. The frame size of the test video sequences was 240 × 320. After the background has been learned, the object detection module takes 0.17 s per frame while the tracking module takes 0.015 s per frame. The time complexity of the proposed tracking module is comparable with that of kernel-based tracking, where the average processing time is 0.0169 s per frame. Overall, our algorithm can detect and track objects in real time with a sample rate of 4 frames/s.

6 Conclusions

We have proposed a framework for improving video-based surveillance by integrating target detection with tracking. The proposed framework was evaluated by detecting and tracking pedestrians and vehicles both in visible and thermal video sequences. On-line SVR was used to model the background and to accurately detect the initial locations of the targets. Moreover, shape projection histograms were exploited to predict the location of targets in successive frames. 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

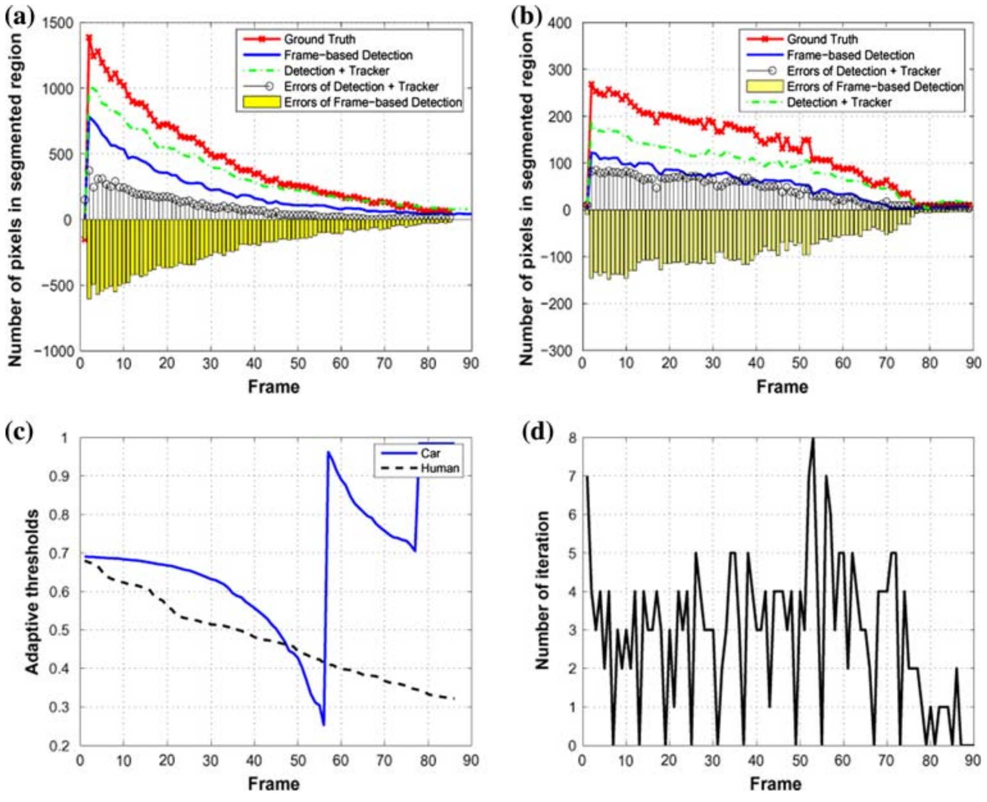


Fig. 12 **a**, **b** Comparison results between frame-based detection and the proposed approach by counting the number of pixels in two different segmented regions. The *red curve* indicates ground truth information while the *green and blue curves* indicate the performance of the proposed method and frame-based detection, respectively; **c** adaptive threshold values over time for two different targets; **d** average number of iterations

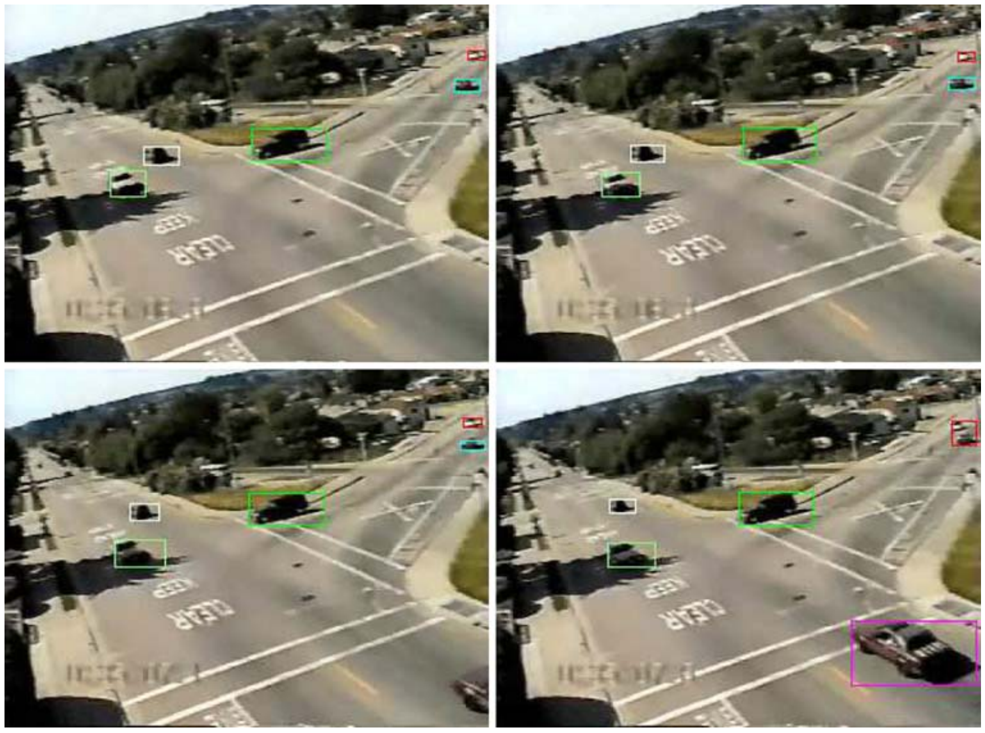


Fig. 13 Target detection and tracking results in the appearance of a new target

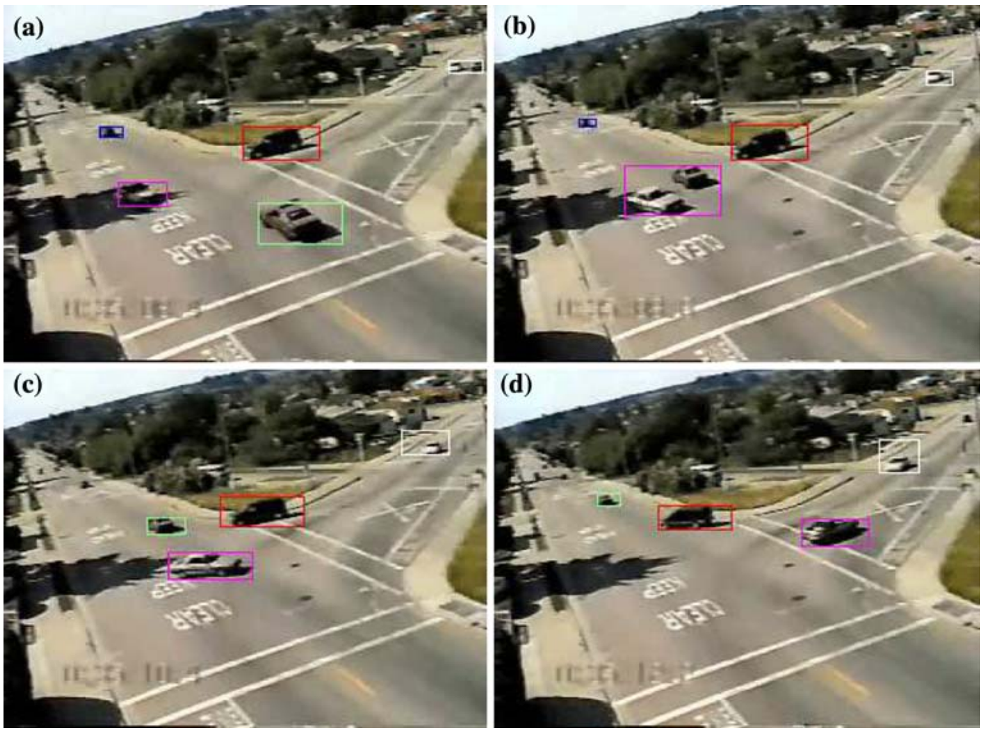


Fig. 14 An example of handling target occlusion. For each target, its trajectory and frame box are shown in the same color. When two targets occlude each other, occluded targets are labeled using the same color box. After occlusion, the trajectories of targets are successful recovered by stitching their tracks with previous tracks

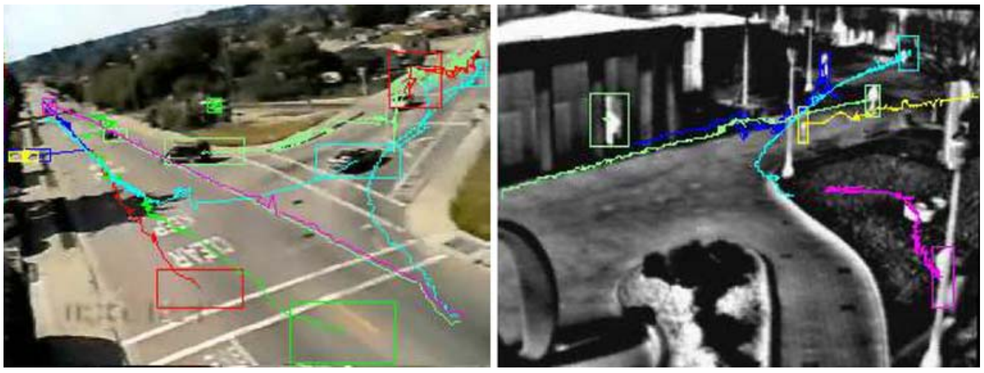


Fig. 15 Multiple target tracking and their trajectories. For each target, its trajectory is shown in the same color as its frame box

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. Our experimental results show good performance, especially when dealing with small targets and targets undergoing perspective projection distortions. Moreover, they show good suppression of false alarms due to noise.

For future work, we plan to improve the speed of background modeling in our method. Although we were able to achieve good speed in our experiments by sub-sampling the captured images, further improvements are necessary for true real-time performance. One way to improve the speed is by using region-based instead of pixel-based SVR models to represent the background scene.

Acknowledgments This research was supported in part by Ford Motor Company under grant No. 2001332R, and the University of Nevada, Reno (UNR) under an Applied Research Initiative (ARI) grant.

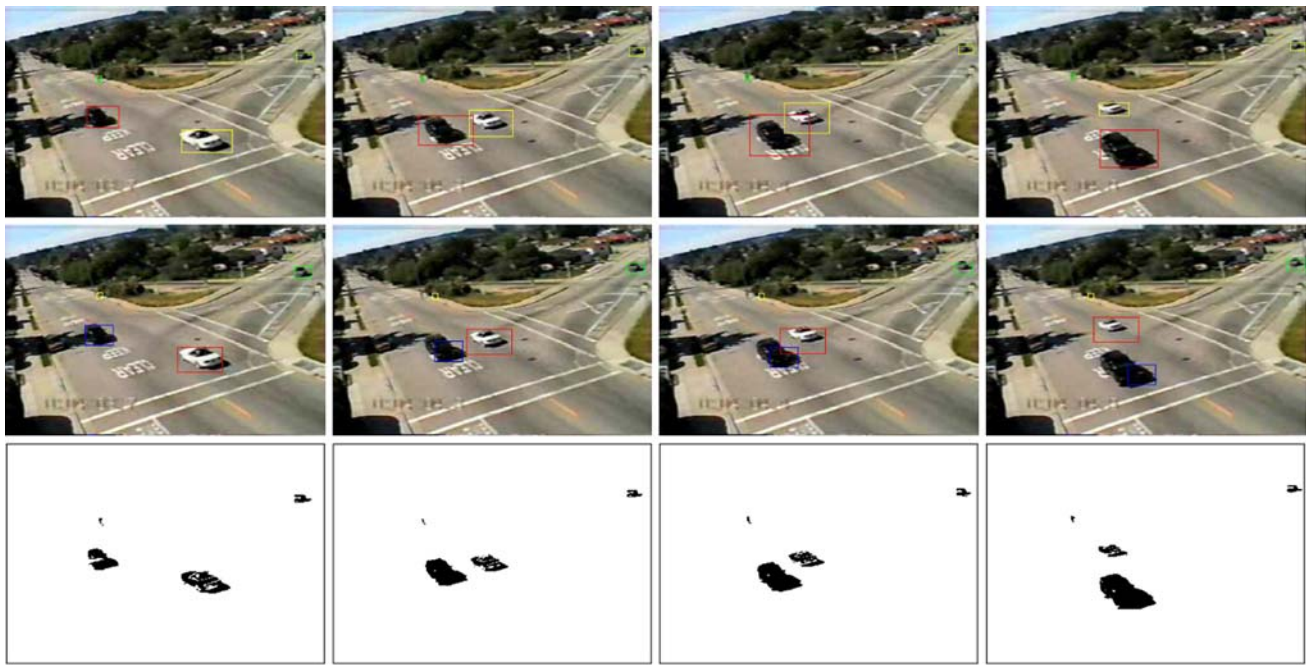


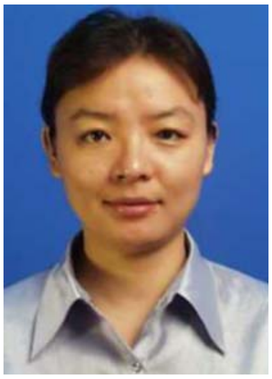
Fig. 16 Comparison results between kernel-based tracking and the proposed approach when tracking small targets and target with projection distortions. Tracking results are labelled using rectangles of different colors. Frames 4, 22, 26, 39 of the test sequence are shown. *1st row* tracking results using the proposed method. *2nd row* tracking results using kernel-based tracking where, the initial target locations were chosen to be the same to those found by our approach (i.e., shown in the first row). *3rd row* detection results using the proposed method

References

1. Sun, Z., Bebis, G., Miller, R.: On-road vehicle detection: a review. IEEE Trans. Pattern Anal. Mach. Intell. **28**(5), 694–711 (2006)
2. Haritaoglu, I., Harwood, D., Davis, L.: W^4: real-time surveillance of people and their activities. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 809–830 (2000)
3. Wren, W., Azarbaygaui, A., Darrell, T., Pentland, A.: Pfinder: real-time tracking of the human body. IEEE Trans. Pattern Anal. Mach. Intell. **19**(7), 780–785 (1997)
4. Stauffer, C., Grimson, W.: Learning patterns of activity using real-time tracking. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 747–757 (2000)
5. Matsuyama, T., Ohya, T., Habe, H.: Background subtraction for non-stationary scenes. In: Proceedings of the 4th Asian Conference on Computer Vision, pp. 662–667 (2000)
6. Eng, H., Wang, J., Kam, A.H., Yau, W.: Novel region-based modeling for human detection within highly dynamic aquatic environment. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. II-390–II-397 (2004)
7. Elgammal, A., Harwood, D., Davis, L.: Non-parametric model for background adaptation. In: European Conference on Computer Vision (2000)
8. Javed, O., Shafique, K., Shah, M.: A hierarchical apporach to robust background subtraction using color and gradient. IEEE Workshop on Motion and Video Computing, pp. 22–27 (2002)
9. Monnet, A., Mittal, A., Paragios, N., Ramesh, V.: Background modeling and subtraction of dynamic scenes. In: IEEE Internaltional Conference on Computer Vision, 2003
10. Toyama, K., Krumm, J., Brumitt, B., Meyers, B.: Wallflower: principles and practice of background maintenance. In: International Conference on Computer Vision, pp. 255–261 (1999)
11. Rittscher, J., Kato, J., Joga, S., Blake. A.: A probabilistic background model for tracking. European Conference on Computer Vision vol. 2, pp. 336–350, (2000)
12. Seki, M., Wada, T., Fujiwara, H., Sumi, K.: Background subtraction based on co-occurrence of image variations. In: International Conference on Computer Vision and Pattern Recognition vol. 2, 65 (2003)
13. Li, L., Leung, M.: Integrating intensity and texture differences for robust change detection. IEEE Trans. Image Process. **11**(2), 105–112 (2002)
14. Lipton, A.J., Fujiyoshi, H., Patil, R.S.: Moving target classification and tracking from real-time video on-road vehicle detection: a review. In: Proceedings of the IEEE workshop on applications of computer vision (2002)
15. Beren, J.R. et al.: A three frame algorithm for estimating two-component image motion. IEEE Trans. Pattern Anal. Mach. Intell, **14**(9), 886–896 (1992)
16. Beren, J.R. et al.: A three frame algorithm for estimating two-component image motion. IEEE Trans. Pattern Anal. Mach. Intell. **14**(9), 886–896 (1992)
17. Sharma, R., Aloimonos, Y.: Early detection of independent motion from active control of normal image flow patterns. IEEE Trans. Sys. Man Cybernet. Part B **26**(1), 42–52 (1996)
18. Avidan, S.: Support vector tracking. IEEE Trans. Pattern Anal. Mach. Intell. **26**(8), 1064–1072 (2004)
19. Bar-shalom, Y., Fortmann, T.: Tracking and Data Association. Academic Press, New York (1988)

20. Kitagawa, G.: Non-Gaussian state-space modeling of nonstationary time series. J. Am. Stat. Assoc. **82**, 1032–1063 (1987)
21. Gordon, N., Salmond, D., Smith, A.: A novel approach to nonlinear and non-Gaussian Bayesian state estimation. Proc. Part-F: Radar Signal Process. **140**, 107–113 (1993)
22. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Trans. Pattern Anal. Mach. Intell. **25**(5), (2005)
23. Wang, J., Eng, H., Kam, A., Yau, W.: A framework for foreground detection in complex environments. In: European Conference on Computer Vision, Workshop of Statistical Modeling for Video Processing, pp. 129–140 (2004)
24. Verma, R., Schmid, C., Mikolajczyk, K.: Face detection and trackign in a video by propagating detection probabilities. IEEE Trans. Pattern Anal. Mach. Intell. **25**(10), 1215–1227 (2003)
25. Hearst, M.: Trends and controversies—support vector machines. IEEE Intell. Syst. **13**(4), 18–28 (1998)
26. Smola, A., Scholkopf, B.: A tutorial on support vector regression. NeuroCOLTS Technical Report Series NC2-TR-1998-030, October 1998
27. Ma, J., Theiler, J.: Accurate on-line support vector regression. Neural Comput. **15**, 2683–2703 (2003)
28. Davis, J.W., Sharma, V.: Robust background-subtraction for person detection in thermal imagery. In: IEEE International Conference on Computer Vision and Pattern Recognition (2004)
29. Davis, J.W., Keck, M.A.: A two-stage template approach to person detection in thermal imagery. In: IEEE International Conference on Computer Vision and Pattern Recognition (2005)
30. Amer, A.: Voting-based simultaneous tracking of multiple video objects. IEEE Trans. Circuits Syst. Video Technol. **15**(11), 1448–1462 (2005)
31. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley, Chichester (2001)

Author biographies



Junxian Wang received the B.Eng. and M.E. degrees in computer science and engineering from Xidian University, Shaanxi, China, in 1994 and 1997, respectively, and the Ph.D. degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2003. From 2002 to 2004, she was an Associate Scientist with the Institute for Infocomm Research, Singapore. From 2004 to 2006, she was a Postdoctorate with the Computer Science and Engineering Department, University of Nevada, Reno. Currently, she is a research scientist at UtopiaCompression. Her research interests include image/video processing, object tracking, and machine learning.

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.

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.

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.

Ronald Miller received his BS in Physics in 1983 from the University of Massachusetts, and his PhD in Physics from the Massachusetts Institute of Technology in 1988. His research has ranged from computational modeling of plasma and ionospheric instabilities to automotive safety applications. Dr. Miller heads a research program at Ford Motor Company in intelligent vehicle technologies focusing on advanced RF communication, radar, and optical sensing systems for accident avoidance and telematics.