SIViP
DOI 10.1007/s11760-013-0576-5

ORIGINAL PAPER

Real-time background subtraction-based video surveillance of people by integrating local texture patterns

Hefeng Wu · Ning Liu · Xiaonan Luo · Jiawei Su · Liangshi Chen

Received: 2 March 2013 / Revised: 20 October 2013 / Accepted: 22 October 2013

Abstract This paper presents a real-time surveillance system for detecting and tracking people, which takes full advantage of local texture patterns, under a stationary monocular camera. A novel center-symmetric scale invariant local ternary pattern feature is put forward to combine with pattern kernel density estimation for building a pixel-level-based background model. The background model is then used to detect moving foreground objects on every newly captured frame. A variant of a fast human detector that utilizes local texture patterns is adopted to look for human objects from the foreground regions, and it is assisted by a head detector, which is proposed to find in advance the candidate locations of human, to reduce computational costs. Each human object is given a unique identity and is represented by a spatio-color-texture object model. The real-time performance of tracking is achieved by a fast mean-shift algorithm coupled with several efficient occlusion-handling techniques. Experiments on challenging video sequences show that the proposed surveillance system can run in real-time and is quite robust in segmenting and tracking people in complex environments that include appearance changes, abrupt motion, occlusions, illumination variations and clutter.

H. Wu · X. Luo
National Engineering Research Center of Digital Life, State-Province Joint Laboratory of Digital Home Interactive Applications, School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510006, China

H. Wu · X. Luo
Shenzhen Digital Home Key Technology Engineering Laboratory, Research Institute of Sun Yat-sen University in Shenzhen, Shenzhen 518057, China

N. Liu (✉) · J. Su · L. Chen
School of Software, Sun Yat-sen University, Guangzhou 510006 , China
e-mail: liuning2@mail.sysu.edu.cn

Keywords Local texture patterns · Background modeling · Human detection · People tracking · Spatio-color-texture representation

1 Introduction

Texture is a fundamental and powerful property of images and surfaces, and analysis of textures has applications ranging from image segmentation to object recognition and behavior understanding. So far, a variety of different texture descriptors have been presented in the literature, among which the local binary pattern (LBP) [17], maximum response 8 [20] and Gabor filter-based texture descriptors are some commonly studied texture analysis techniques. In this paper, some recent findings in texture analysis are explored and extended for detecting and tracking people in the surveillance applications under a stationary camera.

Background subtraction is often the first task in stationary video surveillance applications. It is quite challenging when real-world scenes contain complex dynamic backgrounds, e.g., moving vegetation, rippling water, etc. To address these situations, many background modeling methods [10,12,16] have been presented in the past few years. In this paper, we present a local texture descriptor termed center-symmetric scale invariant local ternary patterns (CS-SILTP) for background modeling. This improved local pattern feature encodes the local spatio-temporal texture information in a more compact way, and it is combined with pattern kernel density estimation techniques to build a pixel-wise background model.

After the scene has been modeled, the system watches for large deviations from this background model at every pixel location, and a binary foreground image is generated from every incoming frame of the scene. If a changed region of

Published online: 09 November 2013

the image is found, then the system proceeds to analyze the region in more detail to find objects of interest (i.e., human). A variant of a fast human detector C^4 [21] is presented for the task of finding human objects. The detector is based on the contour cues that are extracted by local texture descriptors. In addition, a simple head detection algorithm is carried out in advance to find possible locations of people, avoiding the time-consuming procedure of exhaustive detection.

When a new person is detected, the proposed surveillance system would assign it an unique identity and keep track of it. In order to achieve real-time performance, an efficient tracking algorithm is required. Although the research on object tracking has advanced a lot in recent years and many novel approaches [1,11,18] have been proposed for the tracking tasks in challenging scenes, most of them focus mainly on the accuracy and robustness of tracking objects and are commonly at the cost of high computational expense. In this paper, the proposed system uses a spatio-color-texture object representation for a detected person and adopts a fast mean-shift algorithm [4] for person tracking. In addition, a simple strategy is designed to make the tracking module more robust, introducing some efficient techniques for occlusion handling.

The rest of the paper is organized as follows. In Sect. 2, we describe the modeling of a scene and segmenting foreground regions, and Sect. 3 discusses the procedure of people detection and tracking. The experimental results are presented in Sect. 4. Finally, we conclude the paper in Sect. 5.

2 Background–foreground segmentation

2.1 CS-SILTP descriptor

Recently, a scale invariant local ternary pattern (SILTP) operator was proposed by Liao et al. [13] to describe local texture information. It is demonstrated to be effective for handling illumination variations and is used to model the pixel process for background subtraction. In this paper, we extend and improve the SILTP descriptor to introduce a novel center-symmetric scale invariant local ternary pattern (CS-SILTP) descriptor, by exploring spatial and temporal relationships of neighborhood.

SILTP is calculated by comparing the values of a center pixel and its neighboring pixels, which improves from LBP through adding a scale factor to tolerate noise and illumination variations. Given a pixel location (x_c, y_c), SILTP encodes it as:

$$\mathtt{SILTP}^{\tau}_{N,R}(x_c, y_c) = \bigoplus_{k=0}^{N-1} s_\tau(I_c, I_k), \tag{1}$$

where I_c is value of the center pixel (x_c, y_c), and $\{I_k\}_{k=0}^{N-1}$ is the values of its N neighboring pixels which are equally spaced on a circle of radius R, $\bigoplus$ denotes concatenation operator of binary strings, τ is scale factor indicating the comparing range, and s_τ is defined as

$$s_\tau(I_c, I_k) = \begin{cases} 01, & \text{if } I_k > (1+\tau)I_c, \\ 10, & \text{if } I_k < (1-\tau)I_c, \\ 00, & \text{otherwise.} \end{cases} \tag{2}$$

Motivated by Heikkilä et al. [7], we propose to encode the grayscale invariant local patterns in a more compact but not less effective way. We first give the definition of CS-SILTP in the 2D image plane and use CS-SILTP2 to denote it (a "2" is appended, standing for 2D). Then we extend the CS-SILTP2 descriptor into three dimensional context, including both spatial and temporal information, and attain the final CS-SILTP descriptor that we use for background modeling.

For any pixel location (x_c, y_c), the CS-SILTP2 operator encodes its local texture pattern as

$$\mathtt{CS\text{-}SILTP2}^{\tau}_{N,R}(x_c, y_c) = \bigoplus_{k=0}^{\left(\frac{N}{2}\right)-1} s_\tau\left(I_k, I_{k+\left(\frac{N}{2}\right)}\right). \tag{3}$$

The notations involved are defined the same as that of SILTP. The pixel locations in the neighborhood are numbered clockwise from 0 to $N-1$, so I_k and $I_{k+(N/2)}$ are values of center-symmetric pixel pairs that lie on a circle of radius R. Figure 1a illustrates the SILTP and CS-SILTP2 neighborhood when $R = 1$ and $N = 8$. We point out that one can strictly follow the circle layout and find the values of pixel locations that lie on the circle by interpolation as in [17], but we follow the discrete approximation of square layout, which is more

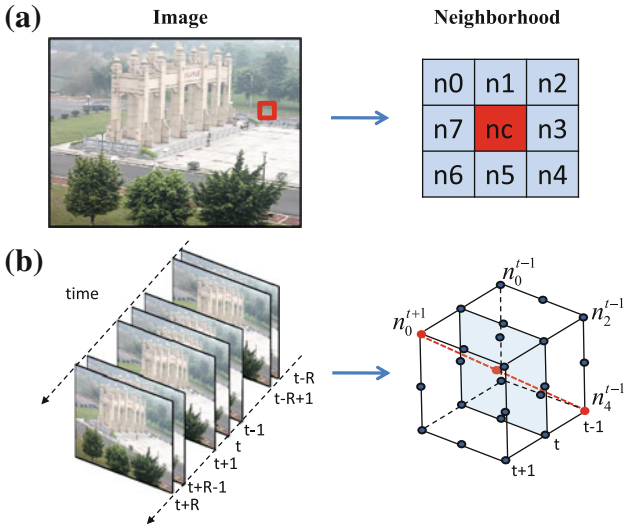

Fig. 1 Illustration of spatial and temporal neighborhoods when $R = 1$, $N = 8$. **a** The neighborhood of SILTP and CS-SILTP2 descriptors in a 2D image plane. **b** The neighborhood of the CS-SILTP descriptor, extended to the 3D spatio-temporal domain

commonly used and is easy to compute but not less effective. The SILTP descriptor has 16 bits when $N = 8$, while the CS-SILTP2 descriptor contains only 8 bits and thus is more compact in encoding the local texture.

When modeling the pixel process in a fixed scene, we can further take advantage of the temporal information. It can be imagined that the circle with radius R surrounding the center pixel in the 2D image plane is extended to a sphere with radius R in the 3D spatio-time domain. Here we again use the approximation of cubic layout for efficient computation. The CS-SILTP operator is formulated as below:

$$\texttt{CS-SILTP}^{\tau}_{N,R}(x_c, y_c) = \bigoplus_{r=-R}^{R} \bigoplus_{k=0}^{\frac{N}{2}-1} s_\tau(I_k^{t+r}, I_{k+\frac{N}{2}}^{t-r}), \quad (4)$$

where I^t denotes the scene image captured at the time instant t, I_k^{t+r} and $I_{k+(N/2)}^{t-r}$ are the center-symmetric pixel locations lying on the cubic surface. The number of pixel locations used in each image plane is the same (i.e., N) when using the cubic approximation. Figure 1b shows an example of the CS-SILTP neighborhood when $R = 1$ and $N = 8$. The red points (i.e., n_0^{t+1} and n_4^{t-1}) in the cubic surface denote a pair of center-symmetric pixels used in one comparison. As can be easily figured out, it is exactly the CS-SILTP2 operator when r just takes the value 0 in Eq. (4).

2.2 Background modeling

The pattern kernel density estimation technique [13] is utilized to build pixel-wise background model. Given a pixel location (x, y), if we maintain K most frequently occurred local patterns $\{p_i\}_{i=1}^K$, the pattern probability density function can be approximated smoothly by

$$\hat{f}(q) = \sum_{i=1}^{K} w_i f_{p_i}(q) = \sum_{i=1}^{K} w_i g(d(p_i, q)), \quad (5)$$

where g is a Gaussian-like weighting function, d is the Hamming distance between two binary patterns, w_i are weighting coefficients, and $\sum w_i = 1$. We sort the K patterns with the weights $\{w_i\}$ in descending order.

When a new pattern p_t is extracted from the current image frame I_t at the pixel location (x, y), we can update the estimated distribution $\hat{f}(q)$ accordingly. The pattern p_t is matched to the K sorted patterns $\{p_i\}$ in turn, and a match is found if $f_{p_i}(p_t) > T_m$, where T_m is a constant threshold controlling the matching. The weight of each pattern p_i is updated as

$$w_i = (1 - \alpha)w_i + \alpha\Gamma(p_i, p_t), \quad (6)$$

where α is a learning rate, and $\Gamma(p_i, p_t)$ is an indicator variable that takes 1 if p_i and p_t are matched and 0 otherwise. If none of the K patterns matches the current pattern p_t, the one with the lowest weight is replaced with p_t, and a low initial weight.

2.3 Foreground segmentation

Given a new image frame I_t, we can determine whether the pattern p_t at the specific location (x, y) is background or not. The first M patterns from the K maintained patterns are used to make the decision, and M is obtained by

$$M = \underset{k}{\operatorname{argmin}} \left(\sum_{i=1}^{k} w_i > T_b \right), \quad (7)$$

where $T_b \in [0, 1]$ is a threshold indicating how many data should be considered as background. Afterward, the probability of pattern p_t being background is estimated as

$$P_{\text{bg}}(p_t) = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} w_i f_{p_i}(p_t). \quad (8)$$

The corresponding pixel location (x, y) is marked as foreground if $P_{\text{bg}}(p_t)$ is less than T_{bg}, a predefined constant parameter.

We point out that background modeling can be carried out at different image scales. For the scale W, the original image I_t is downsampled by a factor W to be I_t^W, which can achieved by using mean values in non-overlapping $W \times W$ blocks as new pixel values. The higher is the scale, the more computational expense would be saved. However, it would also result in less precise accuracy, for the whole $W \times W$ block is considered as foreground if the corresponding pixel at the scale W is so.

When focusing heavily on accuracy performance, one can achieve it by fusing multiscale information. Background modeling is performed simultaneously at multiscales. For a pixel location at I_t, we can find its corresponding probabilities $\{P_{\text{bg}}^i\}_{i=1}^N$ at N image scales. Afterward, we can simply adopt the geometric average of these probabilities as the fusion probability:

$$F_{\text{bg}}(p_t) = \sqrt[N]{\prod_{i=1}^{N} P_{\text{bg}}^i(p_t)}. \quad (9)$$

The threshold T_{bg} can then be applied to make the foreground/background decision.

3 People detection and tracking

When the foreground image, in which foreground pixels are marked as white and background as black, is generated via background subtraction, we can then find and track the objects of interest in less effort.

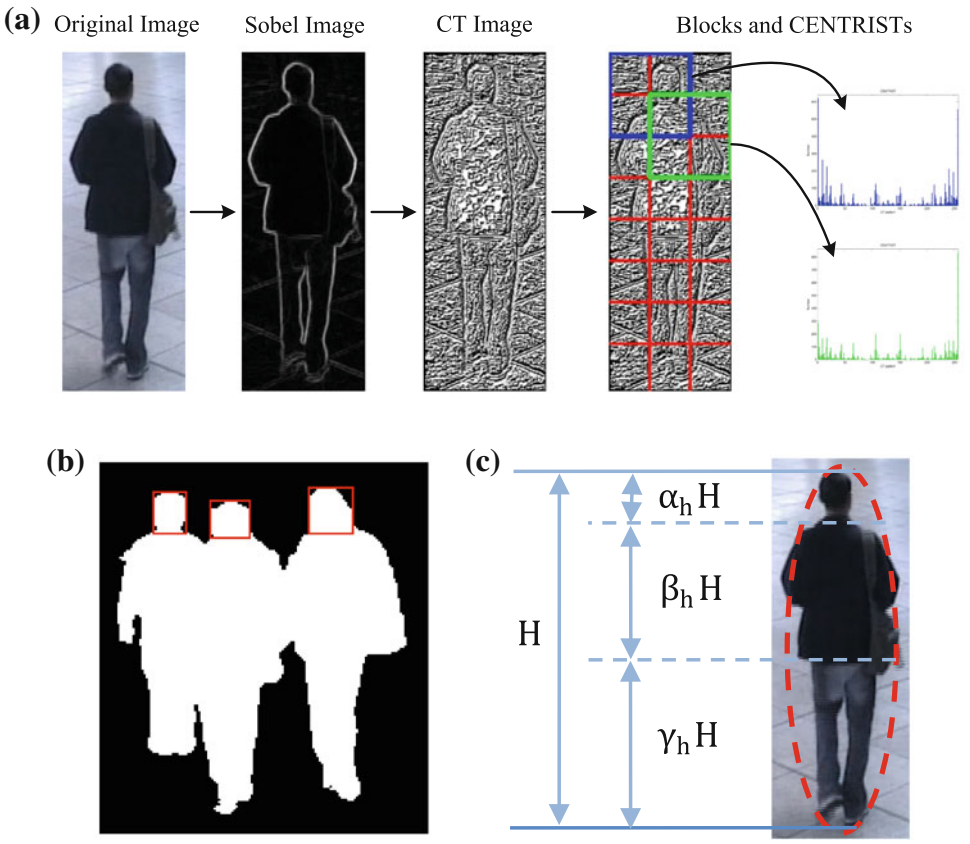

Fig. 2 Human detection and representation: **a** Feature extraction for human detection. **b** Head detection. **c** Spatial partition of human, where $\alpha_h + \beta_h + \gamma_h = 1$

3.1 People detection

To find the human objects from the detected foreground blobs, we present a variant of a fast human detector C^4 [21], which is recently developed using the contour-cues-based features CENTRIST. The C^4 human detector only requires O(1) steps to test an image patch when using a linear classifier and a fast scanning technique. Given an image patch with the feature vector $\mathbf{x}$, we train a linear SVM classifier $f(\mathbf{x}) = h^T\mathbf{x}$ to determine whether the patch contains a human or not.

$$\begin{array}{|c|c|c|} \hline 30 & 30 & 90 \\ \hline 50 & \mathbf{70} & 91 \\ \hline 20 & 20 & 20 \\ \hline \end{array} \Longrightarrow \begin{array}{ccc} 1 & 1 & 0 \\ 1 & & 0 \\ 1 & 1 & 1 \end{array}$$
$$\Longrightarrow \mathrm{CT} = (11010111)_2 = 215. \qquad (10)$$

The patch feature $\mathbf{x}$ is based on CENTRIST (CENsus TRansform hISTogram) [22]. The census transform (CT), proposed by Zabih and Woodfill [24], depends on intensity comparisons of a pixel with the set of pixels in its square neighborhood, as illustrated in Eq. (10). CT works similarly as a simple version of LBP that is systematically analyzed by Ojala et al. [17]. The CENTRIST descriptor of an image patch is a histogram of these CT values.

To be more noise-tolerant, we propose to define the comparison of a center pixel I_c and its neighbor I_k as

$$b_\tau(I_c, I_k) = \begin{cases} 1, & \text{if } I_k < (1-\sigma)I_c, \\ 0, & \text{otherwise}, \end{cases} \qquad (11)$$

by adding a predefined scale factor σ for resisting noise and illumination variations.

In order to capture the contour cues well, the CENTRIST descriptor is extracted from the Sobel version of an image rather than directly on the intensity version. Figure 2a illustrates the extraction process of CENTRIST descriptors. The third column of Fig. 2a visualize the CT image, every pixel of which is an improved CT value defined by Eqs. (10) and (11). The image patch is divided into $N_w \times N_h$ blocks, and each of adjacent 2×2 blocks is treated as a super-block. N_w and N_h are the numbers of blocks, respectively, in the horizontal and vertical directions. A CENTRIST descriptor is extracted from each super-block, as exhibited in Fig. 2a, and all the descriptors are then sequentially concatenated to form the feature vector $\mathbf{x}$.

The detector is trained in a bootstrap way. In the training phase, we collect a set of positive training image patches $\mathcal{P}$ and a set of person-free negative images $\mathcal{N}$. At first, a set of negative patches is randomly sampled from the images in $\mathcal{N}$ to form a negative training set $\mathcal{N}_1$. Using $\mathcal{P}$ and $\mathcal{N}_1$, we train

a linear SVM classifier H_1. Then, all the images in $\mathcal{N}$ are searched exhaustively for false positives ("hard examples") using H_1. An augmented negative set $\mathcal{N}_2$ is formed by $\mathcal{N}_1$ and the hard examples, and we train H_2 using $\mathcal{P}$ and $\mathcal{N}_2$. This bootstrap process is repeated several rounds to attain the final detector H_f.

Using the off-line trained human detector, we can find those human objects in the current frame. An exhaustive detection on every location of the foreground regions may find all the possible human objects, but at the price of high computational costs, and it does not satisfy the real-time requirements. Another way is to detect once at each foreground blob; however, it is only suitable for those foreground blobs that contain a single person, and problems arise when an foreground blob contains multiple people, as shown in Fig. 2b. Inspired by [6,19], we adopt a trade-off method between accuracy and real-time constraints, and a fast head detection algorithm is presented to check whether multiple people exist within a foreground blob and to find the possible locations for later examination by the human detector.

Given a foreground blob, the fast head detection algorithm is carried out in the following steps:

1. Find the minimum bounding box B_r that contains the foreground blob.
2. Create a vector H^v of heights within the bounding box B_r:
   (a) For each column i in B_r, examine the pixel values from the top until reaching a non-isolated pixel which is classified as foreground.
   (b) The distance of this pixel to the bottom of B_r is stored in $H^v(i)$ as the height at column i.
3. Scan H^v for intervals $H_j^v \subset H^v$ with the following properties (assume $h_j^* = \max H_j^v$):
   (a) The values within H_j^v do not differ significantly (i.e., $\forall h \in H_j^v,\ h_j^* - h \leq T_h$).
   (b) The values $\{\underline{h}\}$ near to the interval H_j^v at both sides are significantly lower than the values inside H_j^v (i.e., $h_j^* - \underline{h} > T_h$).
4. For each H_j^v found in the previous step, define the point $(x_j, H^v(x_j))$ as a head location in B_r, where x_j is the center coordinate of H_j^v, and $H^v(x_j)$ is its corresponding height.

For those foreground blobs where no head is detected, the human detector is also applied to check sparsely whether a person exists. Specifically, the image is divided into blocks of size $W_s \times W_s$, and we check at the center location of every block.

3.2 Tracking using spatio-color-texture representation

For each detected human individual, we maintain its corresponding state, e.g., an identity (ID), locations, sizes, etc. In order to achieve real-time performance, the efficient mean-shift method is exploited to track human individuals. We present a novel spatio-color-texture representation for modeling the specific appearance of a human object, improved from the color model used by Comaniciu et al. [4].

As shown in Fig. 2c, we first partition the appearance of an human into three parts: head, torso and lower limbs, for these three parts generally exhibit different characteristics due to clothing. The ratios are, respectively, α_h, β_h and γ_h, where $\alpha_h + \beta_h + \gamma_h = 1$. Then, besides color features, texture features are also taken into consideration. Specifically, the compact CS-SILTP2 feature, introduced in Sect. 2.1, is utilized. Assume that the color histogram of a given object is $\{\hat{q}_u\}_{u=1\ldots m}$, where m stands for the number of the color bins. When further adding the texture bins and the spatial bins (i.e., the three partitioned parts of human appearance), the color histogram would become a spatio-color-texture histogram $\{\hat{q}_{u,v,w}\}_{u=1\ldots m, v=1\ldots n, w=1\ldots o}$, where $n = 3$ and o are, respectively, the numbers of spatial bins and texture bins.

After applying a spatially weighting kernel, we calculate each component $\hat{q}_{u,v,w}$ of the normalized spatio-color-texture histogram as:

$$\hat{q}_{u,v,w} = C_q \sum_{i=1}^{n_q} k\left(\|\mathbf{x}_i\|^2\right) \delta\left[b_u\left(\mathbf{x}_i\right) - u\right] \delta\left[b_v\left(\mathbf{x}_i\right) - v\right] \delta\left[b_w\left(\mathbf{x}_i\right) - w\right], \tag{12}$$

where $\mathbf{x}_i$ is a pixel location, n_q is the number of pixels belonging to the target, $b_u(\mathbf{x})$, $b_v(\mathbf{x})$ and $b_w(\mathbf{x})$ are a function that maps the given pixel, respectively, to the color, spatial and texture bins, $k(x)$ is the weighting kernel, and $\delta[x]$ is the Kronecker Delta Function, which outputs 1 if $x = 0$ and outputs 0 otherwise. The normalization constant C_q makes $\{\hat{q}_{u,v,w}\}$ sum up to 1. We adopt the Epanechnikov function for $k(x)$ as suggested in [4].

The model $\hat{p}(\mathbf{y})$ of a target candidate located at $\mathbf{y}$ can be calculated similarly. We use the Bhattacharyya coefficient as the similarity measure between two models $\hat{q}$ and $\hat{p}(\mathbf{y})$, which is formulated as

$$\rho\left[\hat{p}(\mathbf{y}), \hat{q}\right] = \sum_{u=1}^{m}\sum_{v=1}^{n}\sum_{w=1}^{o} \sqrt{\hat{p}_{u,v,w}(\mathbf{y})\hat{q}_{u,v,w}}. \tag{13}$$

With the spatio-color-texture histogram representation of human objects and the similarity metric, the fast kernel-based mean-shift method from [4] can easily be applied to find where the target is most likely to be, by maximizing $\rho\left[\hat{p}(\mathbf{y}), \hat{q}\right]$ in an iterative scheme.

3.3 Strategy of tracking multiple people

First of all, we maintain two lists of tracks. One list is termed TrackList, including the individuals that are successfully being tracked, and the other list (LoseList) maintains those individuals that are temporarily lost. For each track in the TrackList, the fast mean-shift tracking method is applied to find the location of the individual. We can also adjust the individual's size according to the overlapping foreground blob. If an individual in the TrackList has not been tracked successfully (the similarity between the candidate and the model is below a constant threshold), it will be moved to the LoseList. For each individual, a Kalman filter is also initialized to predict its locations. When an individual is unable to be located for the first time, we assign it with a label. If it is occluded by a foreground blob (i.e., the predicted location falls in the foreground blob), the label is the corresponding ID of that blob, and is -1 otherwise (occluded by background).

For each track in the LoseList, the Kalman predictor is used to look for the corresponding individual, however, it will not be updated until the track is re-gained. For an individual that is occluded by a foreground blob, we further try to find it from that foreground blob: The head detection algorithm is carried out to look for an extra human, and a few locations in the periphery of the foreground blob are sampled to match the corresponding object model. For an individual occluded by the background, newly detected humans are used to look for a match. If a track in the LoseList is re-gained, it will be moved back to the TrackList. However, it will be removed forever when it can not be recovered for a preset number of frames and be considered permanently lost. This simple strategy can improve the robustness of tracking multiple people under the real-time constraints.

4 Experimental results

The proposed surveillance system has been implemented in C++, and extensive experiments are carried out to verify the effectiveness of the proposed background subtraction method and analyze the performance of the proposed surveillance system under different scenes.

4.1 Evaluation of background subtraction

To evaluate the proposed background model, we ran exhaustive experiments on fifteen publicly available video sequences, which contain challenging backgrounds including moving cast shadows, waving water, sudden changes of lighting, etc. Nine of the test sequences are from the I2R dataset [13], which are captured using a static camera in various environments. Most of these sequences contain several thousand video frames, with 20 frames manually labeled for each sequence as the groundtruth data. The other six test sequences are gray image sequences taken from the UCSD dataset [15], with the groundtruth data provided as well. The UCSD dataset contains various sequences captured by both static and moving cameras, and we select among them six sequences captured by static cameras.

Our method is compared against six existing background subtraction methods, including the simple frame differencing (FrmDiff) method [14], mixture of Gaussians (MoG) proposed by KaewTraKulPong and Bowden [8], the complex background model with Bayesian decisions (ACMMM03) from [12], codebook [10], SILTP [13], and ViBe [3]. For FrmDiff, we adjust the threshold between 20 and 40 to obtain its best performance. For MoG and ACMMM03, we use the OpenCV library, and the library provided by the authors is used for ViBe. We use the $\text{SILTP}^{0.05}_{8,1}$ and $\text{CS-SILTP}^{0.05}_{8,1}$ operators, with consistent parameters of density estimation for both SILTP and CS-SILTP: $K = 5$, $T_b = 0.7$, $T_m = 0.01$, $T_{\text{bg}} = 0.008$, and $\alpha = 0.005$. In addition, the background model for SILTP and CS-SILTP is built by combining two scale layers, and three frames are used for the CS-SILTP operator in Eq. (4).

We show the qualitative comparison results of six test sequences in Fig. 3. The foreground segmentation results of the compared algorithms are demonstrated with one frame for each sequence, of which the classified backgrounds are masked with green color.

A quantitative evaluation is also done on these test sequences. The results are showed in Table 1, using the F score metric from [13]. The F score measures the segmentation accuracy by considering both the recall and the precision. As shown in Table 1, we mark the highest F score of each test sequence in bold, and the second highest with italics. From the table you can see that the proposed method outperforms all the compared state-of-the-art methods on ten out of the fifteen test sequences, and ranks second on another three sequences. The recently proposed ViBe method also gives impressive performance on some of the test sequences, however, its performance is relatively unstable, e.g., performing badly under sudden change of light (Lobby). It should also be noted that, the proposed method based on the CS-SILTP operator generally has an improved performance from the SILTP-based method. It is worth pointing out that, we can incorporate more sophisticated techniques like complex feature representation and adaptive kernel variances, which are reported in [16] to yield improved performance.

4.2 Quantitative analysis of the proposed system

To give a detailed analysis of the proposed surveillance system, we test it on the CAVIAR dataset [5], which is annotated with the positions and sizes of every person of inter-

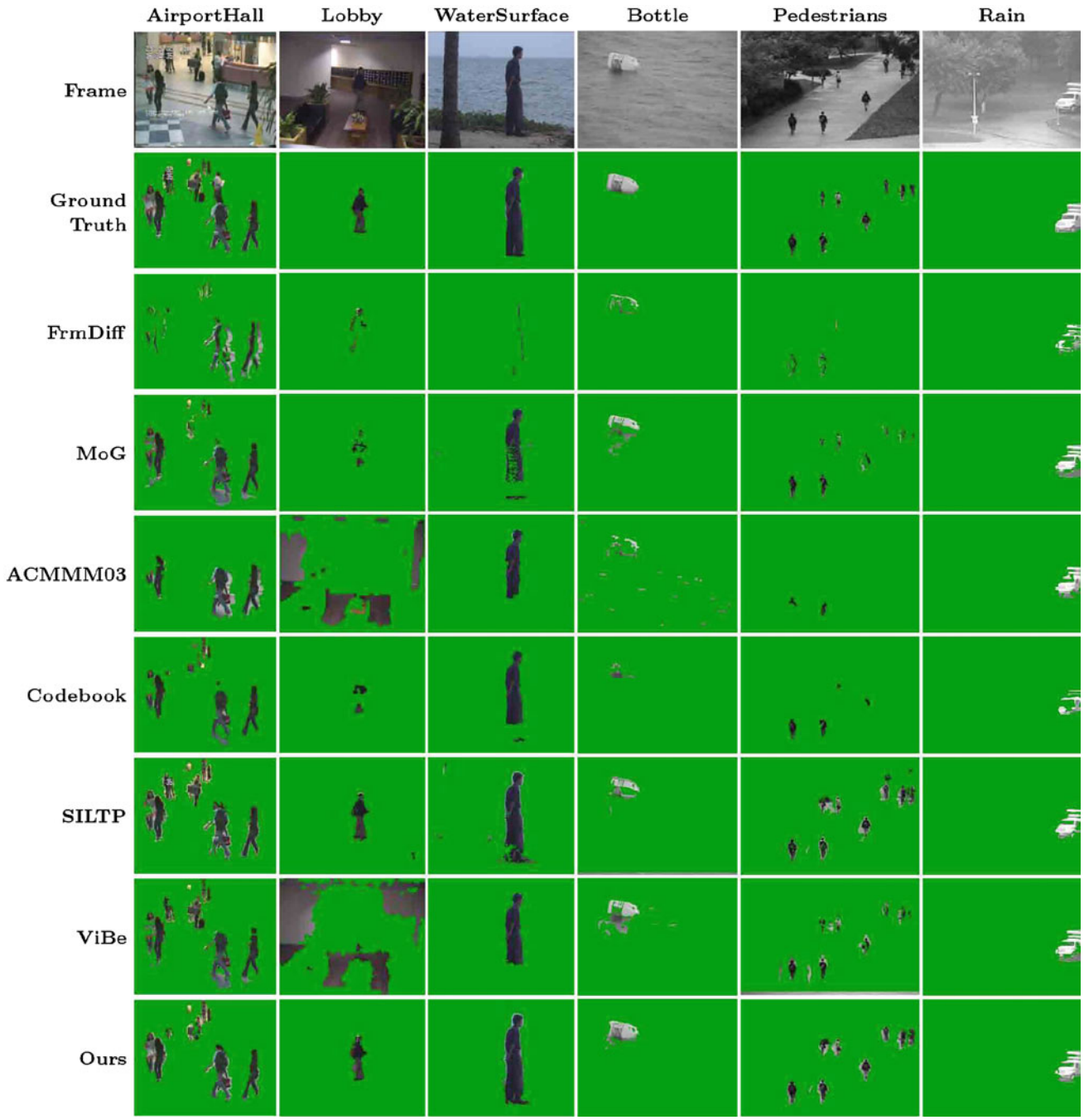

Fig. 3 Foreground segmentation results of different background subtraction methods

est. We train the linear SVM human detector on improved CENTRIST in the way introduced in Sect. 3.1, with a set of 1000 positive image patches and 500 person-free negative images. Consistent parameters are used for feature vector extraction: $\sigma = 0.08$, $N_w = 3$ and $N_h = 6$. In addition, we set $T_h = 6$, $W_s = 5$ for head detection, and $\alpha_h{:}\beta_h{:}\gamma_h = 1{:}3{:}4$, $m = 64$, $n = 3$, $o = 4$ for histogram representation in tracking. If a person is lost for 30 frames, it is considered to be permanently lost.

For quantitative evaluation of human detection, we select 500 images from the tested CAVIAR videos (with no overlap with the training data), along with the binary images resulted from background subtraction. There are a total of 1,837 human objects. We run the detector on foreground regions in an exhaustive way as [21]. The detector with the original CENTRIST descriptor, which does not have the noise-tolerant factor σ in Eq. (11), achieves 89.6 % precision (the ratio of correctly detected humans to all detected humans)

Table 1 Performance of F score (%) on the challenging video sequences

Sequences	FrmDiff	MoG	ACMMM03	Codebook	SILTP	ViBe	Ours
I2R dataset							
AirportHall	23.35	58.68	50.21	51.67	68.48	*71.15*	**73.54**
Bootstrap	35.36	56.29	60.43	63.66	73.32	**78.26**	*76.35*
Curtain	14.97	67.42	57.08	55.81	*91.05*	86.22	**93.87**
Escalator	21.72	41.17	32.60	49.82	*65.88*	64.72	**70.72**
Fountain	25.43	76.91	56.51	61.36	*86.23*	60.96	**87.46**
Lobby	16.33	47.92	30.31	25.51	*78.57*	26.55	**80.23**
Shoppingmall	28.03	69.33	67.82	59.87	**82.19**	76.88	*81.24*
Trees	41.96	*72.49*	**75.43**	54.23	65.83	68.18	69.11
WaterSurface	24.26	79.70	63.66	73.09	84.36	*86.82*	**87.58**
UCSD dataset							
Birds	19.63	28.35	28.82	21.21	*30.11*	**31.92**	29.76
Bottle	19.31	51.12	23.44	23.09	55.23	*57.50*	**63.41**
Freeway	26.81	53.06	31.88	38.48	51.94	*54.62*	**55.54**
Ocean	10.07	28.95	19.03	22.00	*57.87*	26.33	**60.63**
Pedestrians	24.28	79.63	28.89	64.30	*81.09*	80.10	**85.70**
Rain	41.88	74.81	81.28	43.28	85.24	**93.70**	*89.74*

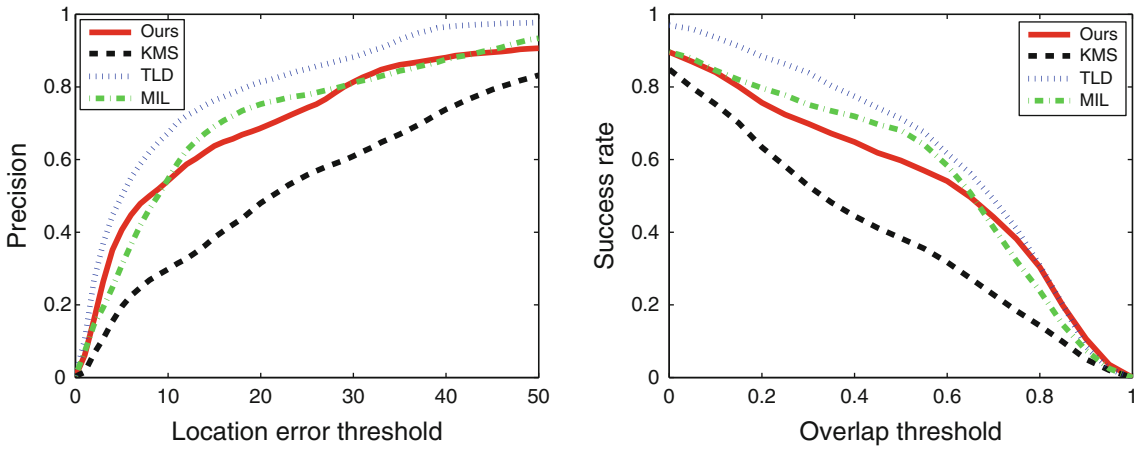

Fig. 4 Evaluation plots of tracking methods. *Left* Precision plot; *Right* Success rate plot

with 82.8 % recall (the ratio of correctly detected humans to all ground truth humans). With the same training process and setting, the detector with the improved CENTRIST descriptor achieves 93.6 % precision with 83.2 % recall. When carrying out the head detection algorithm in advance, the detector achieves 94.7 % precision with 79.3 % recall. Though there is a slight decrease on recall, the speed is almost 13 times faster.

To evaluate the tracking performance of our surveillance system, we collect tracking statistics of 67 human objects from sixteen CAVIAR sequences. Our tracking module is compared with three state-of-the-art trackers KMS [4], MIL [2], TLD [9], whose source codes are publicly available. The tested trackers track each of the 67 human objects with the same initial bounding box found by our human detector.

Two evaluation metrics from [23], i.e., precision and success rate, are used for quantitative analysis. The precision shows the percentage of frames whose center location error (CLE) is within the given threshold, and CLE is defined as the Euclidean distance between the center locations of the tracked object and the ground truth. The success rate measures the percentage of frames whose overlap score is larger than the given threshold. The overlap score is defined as the area ratios between the intersection and union of the tracked bounding box and the ground truth. Figure 4 shows the evaluation plots of the tested trackers' overall performance.

As revealed in Fig. 4, though not as well as TLD, our tracker performs comparably with MIL. With the spatio-color-texture representation and some simple strategies, our tracker has an obvious improvement from the KMS tracker.

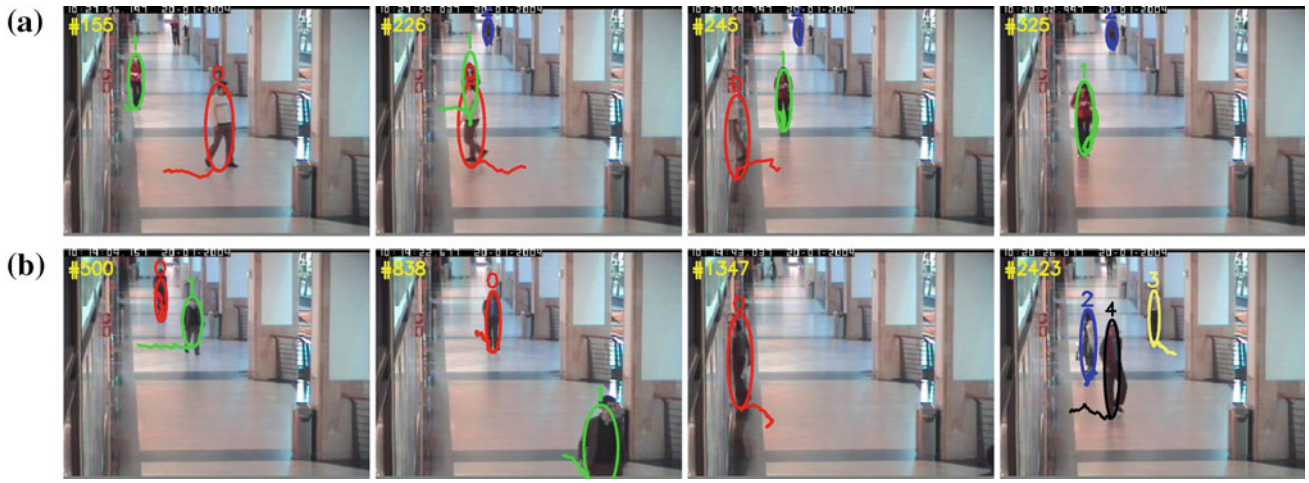

Fig. 5 Example tracking results. **a** The OneLeaveShopRe-enter1cor sequence, **b** The OneStopEnter2cor sequence

When tracking a single human object, our tracker averagely runs at 48 Hz (frames per second), several times faster than MIL (11 Hz) and TLD (9 Hz), slightly slower than KMS (53 Hz). In addition, we have found that using the SILTP texture descriptor in the object representation performs roughly the same as using the CS-SILTP2 descriptor.

Example results of two representative sequences (i.e., OneLeaveShopReenter1cor and OneStopEnter2cor) are shown in Fig. 5. These two sequences are, respectively, 388 and 2,723 frames long, capturing the corridor scenes of passers-by using fixed cameras. Each tracked human object is marked with an ellipse, and is given a unique identity as well. A short trajectory indicating the recent movement of the tracked person is also drawn.

The OneLeaveShopReenter1cor sequence is relatively simple. As shown in Fig. 5a, a man (ID0) leaves the shop and then reenters, briefly occluding a woman in red (ID1) while going halfway back. The tracking module handles the appearance changes of ID0 and the partial occlusion well. We also note that the person with ID2 almost stays still and is not annotated in the ground truth, and he could not be detected until some of his motions are big enough to be marked as foreground.

Figure 5b exhibits example tracking results of the 2,723-frame-long OneStopEnter2cor sequence. The man with ID0 walks along the corridor, and stops halfway to look into the store. After a while, he turns around to enter the store. It should be noted that the tracking module can handle the apparent scale changes of the man when he walks along the corridor, which is fulfilled by appropriate adjustment according to the size of the corresponding foreground blob. However, the man is given a new identity (ID4) when he walks out from the store and reappears again several hundred frames later. This is because the information of ID0 is discarded when it is considered lost for a predefined threshold and this new detected object could not match against the model of ID0. Another three persons also appear in this long sequence, as displayed in Fig. 5b, which adds to the sequence more challenges of appearance changes, occlusions and clutter.

Table 2 Average execution times in ms for foreground detection and handling different numbers of people in videos

Stage	Time (ms)
Foreground detection	27.793
People detection and tracking	
No person	3.801
One person	15.630
Two persons	28.532
Three persons	45.364

The proposed surveillance system has been implemented in C++ and runs on a dual-core 2.27 GHz notebook with the Window 7 operating system. The average execution times for foreground detection and handling different numbers of people are given in Table 2, on the statistical results of processing the sixteen videos. You can notice that the time for detecting and tracking a single person has an evident increase. The reason is that detecting humans from the foreground and other related processing also heavily add to computational load.

In order to achieve real-time performance, the foreground detection module processes the incoming frames at scale 2 (i.e., downsampling the original image by a factor of 2). As revealed in Table 2, the foreground detection module on average takes 28 ms for an image with resolution of 384 × 288, and can run at about 36 frames per second.

Figure 6 shows the execution times needed for processing the OneLeaveShopReenter1cor and OneStopEnter2cor sequences. We plot the execution times used in every frame for two major parts: foreground detection, people detection and tracking. As shown in Fig. 6, the times for foreground

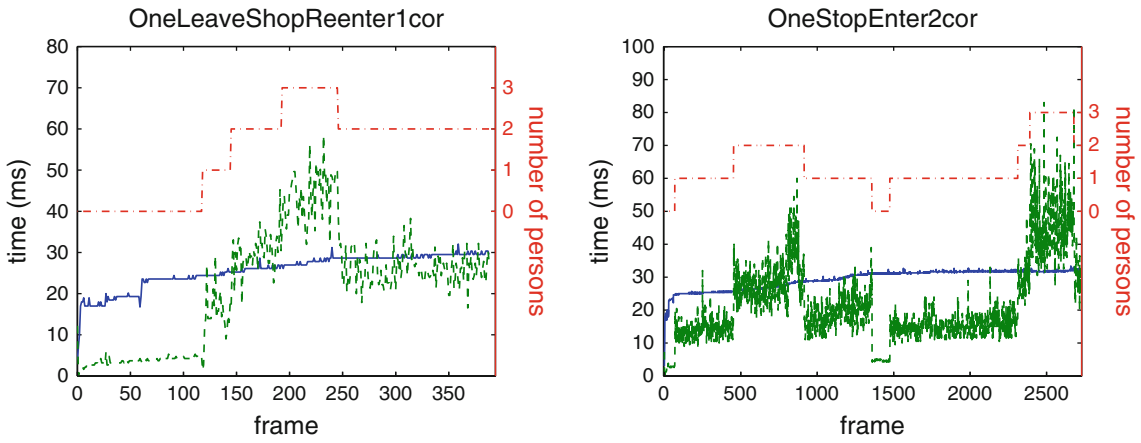

Fig. 6 Execution times in each frame. The *solid blue line* shows the execution time used for foreground detection, while the time used for people detection and tracking is plotted in *dashed green line*. The *dash-dotted red line* gives the number of persons being tracked in each frame, which is plotted with respect to the right *y* axis (color figure online)

detection are plotted in solid blue line. You can see that times for the first few dozen of frames are relatively less, since it is in the phase of initializing the background model. After the background model is built, the foreground detection time for each frame would also be affected by the updating of the background model. The postprocessing stage of the foreground image would also influence the execution time, and may cause some small jumps in the solid blue lines. You can notice that the times for foreground detection increase slowly and gradually become stable. This process differs a little with respective to different video sequences.

The execution times for people detection and tracking exhibit a relatively large and frequent fluctuation (as demonstrated by the dashed green lines in Fig. 6). To make it easier to clarify, we also plot the number of persons being tracked in each frame (the dash-dotted red lines). Generally speaking, the execution time increases as the number of tracked persons does. However, we could not fit an appropriate relational model to the variations of the two, though they may manifest an approximately linear relationship in some simple scenes. Challenging situations (e.g., occlusions, dynamic background clutter) would cause great fluctuations to the execution times. Some big jumps in the plotted curves can also be partly attributed to the effort of trying to detect and find new human objects from the newly detected foreground blobs.

From the average execution times shown in Table 2, you can easily infer that the proposed system can run at around 23 Hz when handling a scene containing a single person, but would run at around 14 Hz if simultaneously handling three persons.

4.3 Surveillance results under different scenes

To further evaluate the performance of the proposed surveillance system, we test it extensively on more video clips of different scenes.

In the AVSSEasy sequence shown in Fig. 7a, the proposed system tracks a man in the subway station. As can been seen, when the man is fully occluded by a pillar at around Frame 1,210 and reappears again, the system can recover his identity and continue the tracking. The second full occlusion of the man is also well handled by the system, thanks to the maintenance of the LoseList and the Kalman predictor.

The Parking sequence (Fig. 7b) captures the scene of a parking lot. This scene contains dynamic waving trees, and also have some cars moving in and out. Our system is able to separate the pedestrian correctly and track it through the scene.

In Fig. 7c, we show the performance of our proposed system on the crowded scenes containing groups of people. The TwoEnterShop2cor sequence illustrated in Fig. 7c mainly captures two people (ID5 and ID6) walk together along the corridor and then enter the store halfway. However, another six people are also involved in this process and make the scene quite challenging. Though the processing speed is a bit slower, our surveillance system gives an impressive performance on this scene, as verified by the tracking results.

5 Conclusion

In this paper, by integrating local texture patterns, a novel real-time surveillance system for detecting and tracking people is presented. A center-symmetric scale invariant local ternary pattern descriptor is introduced to model the scene of interest and separate the foreground regions. With the help of a head detector, individuals from a group of people can be segmented fast. We represent the appearance of each detected person with a spatio-color-texture histogram and track them separately. The situations of occlusions are also properly handled by efficient strategies. Experiments show that the

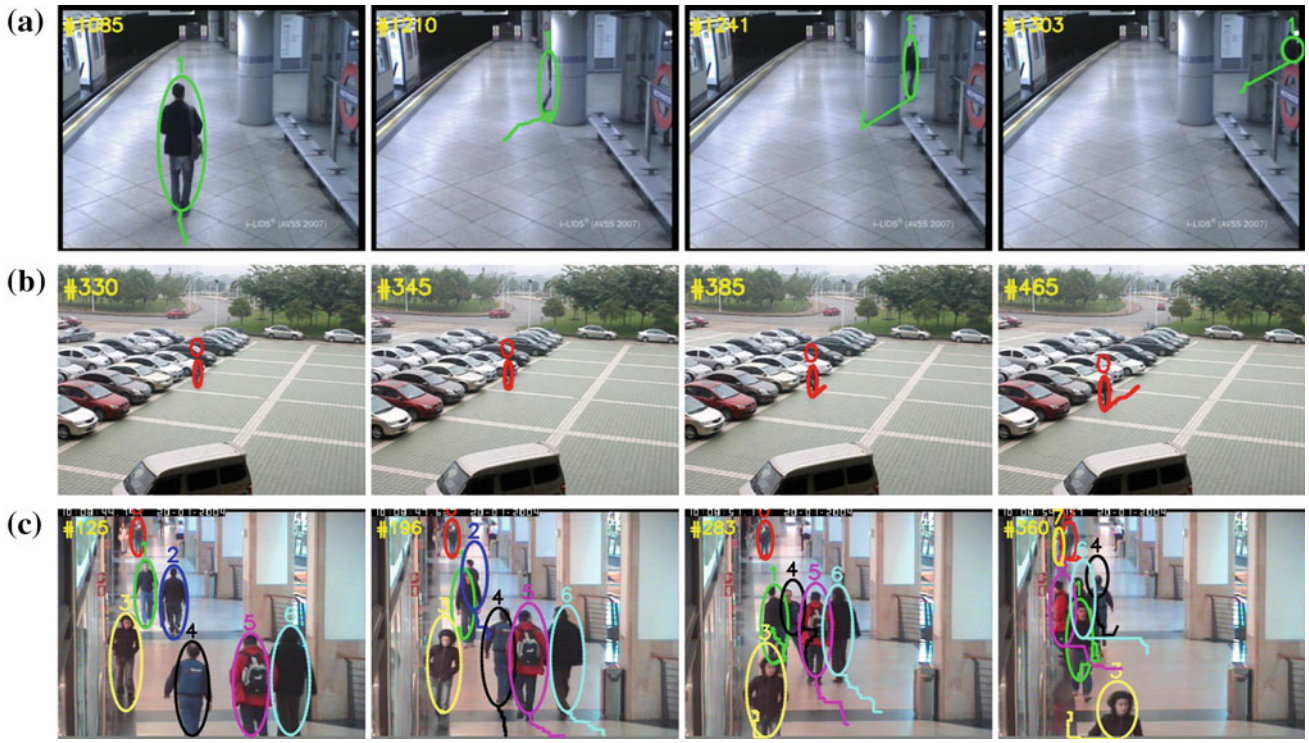

Fig. 7 Surveillance results of people in different scenes: **a** The AVSSEasy sequence, **b** The parking sequence, **c** The TwoEnterShop2cor sequence

proposed method can handle challenging scenes and exhibit impressive performance.

Acknowledgments This research is supported by NSFC-Guangdong Joint Fund (U1135003), the Natural Science Foundation of China (61370186, 6110008), the Industry-academy-research Project of Guangdong (2012B091000104, 2012B091100410), the Special Foundation of Industry Development for Biology, Internet, New Energy and New Material of Shenzhen (JC201104220324A), and the Fundamental Research Funds for the Central Universities (2010620003161035).

References

1. Allili, M., Ziou, D.: Active contours for video object tracking using region, boundary and shape information. Signal Image Video Process. **1**(2), 101–117 (2007)
2. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. IEEE Trans. Pattern Anal. Mach. Intell. **33**(8), 1619–1632 (2011)
3. Barnich, O., Van Droogenbroeck, M.: ViBe: a universal background subtraction algorithm for video sequences. IEEE Trans. Image Process. **20**(6), 1709–1724 (2011)
4. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Trans. Pattern Anal. Mach. Intell. **25**(5), 564–575 (2003)
5. Fisher, R.B.: The PETS04 surveillance ground-truth data sets. In: Proceedings of IEEE Workshop on Performance Evaluation of Tracking and Surveillance, pp. 1–5 (2004)
6. Haritaoglu, I., Harwood, D., Davis, L.S.: W^4: real-time surveillance of people and their activities. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 809–830 (2000)
7. Heikkilä, M., Pietikäinen, M., Schmid, C.: Description of interest regions with local binary patterns. Pattern Recognit. **42**(3), 425–436 (2009)
8. KaewTraKulPong, P., Bowden, R.: An improved adaptive background mixture model for real-time tracking with shadow detection. In: Proceedings of 2nd European Workshop on Advanced Video-Based Surveillance Systems, pp. 1–5 (2001)
9. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking–learning-detection. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1409–1422 (2012)
10. Kim, K., Chalidabhongse, T., Harwood, D., Davis, L.: Real-time foreground–background segmentation using codebook model. Real-Time Imaging **11**(3), 172–185 (2005)
11. Kwon, J., Lee, K.M.: Tracking by sampling trackers. In: Proceedings of the IEEE International Conference Computer Vision, pp. 1195–1202 (2011)
12. Li, L., Huang, W., Gu, I., Tian, Q.: Foreground object detection from videos containing complex background. In: Proceedings of the ACM Conference Multimedia, pp. 2–10 (2003)
13. Liao, S., Zhao, G., Kellokumpu, V., Pietikainen, M., Li, S.Z.: Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In: Proceedings of IEEE Conference Computer Vision and Pattern Recognition, pp. 1301–1306 (2010)
14. Lipton, A.J., Fujiyoshi, H., Patil, R.S.: Moving target classification and tracking from real-time video. In: Proceedings of IEEE Workshop on Applications of Computer Vision, pp. 8–14 (1998)
15. Mahadevan, V., Vasconcelos, N.: Spatiotemporal saliency in dynamic scenes. IEEE Trans. Pattern Anal. Mach. Intell. **32**(1), 171–177 (2010)
16. Narayana, M., Hanson, A., Learned-Miller, E.: Background modeling using adaptive pixelwise kernel variances in a hybrid feature space. In: Proceedings of IEEE Conference Computer Vision and Pattern Recognition, pp. 2104–2111 (2012)

17. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multi-resolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 971–987 (2002)
18. Shan, D., Zhang, C.: Visual tracking using IPCA and sparse representation. Signal Image Video Process (2013). doi:10.1007/s11760-013-0525-3. http://link.springer.com/article/10.1007/s11760-013-0525-3
19. Siebel, N.T., Maybank, S.J.: Fusion of multiple tracking algorithms for robust people tracking. In: Proceedings of the European Conference Computer Vision, pp. 373–387 (2002)
20. Varma, M., Zisserman, A.: A statistical approach to texture classification from single images. Int. J. Comput. Vis. **62**(1), 61–81 (2005)
21. Wu, J., Geyer, C., Rehg, J.M.: Real-time human detection using contour cues. In: Proceedings of IEEE International Conference Robotics and Automation, pp. 860–867 (2011)
22. Wu, J., Rehg, J.M.: CENTRIST: a visual descriptor for scene categorization. IEEE Trans. Pattern Anal. Mach. Intell. **33**(8), 1489–1501 (2011)
23. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: a benchmark. In: Proceedings of IEEE Conference Computer Vision and Pattern Recognition (2013)
24. Zabih, R., Woodfill, J.: Non-parametric local transforms for computing visual correspondence. In: Proceedings of the European Conference Computer Vision, pp. 151–158 (1994)