Optical Engineering 49(11), 117203 (November 2010)

Keyframe labeling technique for surveillance event classification

Ediz Şaykol
Muhammet Başstan
Uğur Güdükbay
Özgür Ulusoy
Bilkent University
Department of Computer Engineering
06800 Bilkent, Ankara, Turkey
E-mail: gudukbay@cs.bilkent.edu.tr

Abstract. The huge amount of video data generated by surveillance systems necessitates the use of automatic tools for their efficient analysis, indexing, and retrieval. Automated access to the semantic content of surveillance videos to detect anomalous events is among the basic tasks; however, due to the high variability of the audio-visual features and large size of the video input, it still remains a challenging task, though a considerable amount of research dealing with automated access to video surveillance has appeared in the literature. We propose a keyframe labeling technique, especially for indoor environments, which assigns labels to keyframes extracted by a keyframe detection algorithm, and hence transforms the input video to an event-sequence representation. This representation is used to detect unusual behaviors, such as crossover, deposit, and pickup, with the help of three separate mechanisms based on finite state automata. The keyframes are detected based on a grid-based motion representation of the moving regions, called the *motion appearance mask*. It has been shown through performance experiments that the keyframe labeling algorithm significantly reduces the storage requirements and yields reasonable event detection and classification performance. [DOI: 10.1117/1.3509270]

Subject terms: video surveillance; scenario-based querying and retrieval; anomaly detection; keyframe detection; after-the-fact analysis; finite state automata.

Paper 100021RR received Jan. 8, 2010; revised manuscript received Sep. 6, 2010; accepted for publication Sep. 23, 2010; published online Nov. 22, 2010.

1 Introduction

Video surveillance has become an interesting and challenging application domain in video processing. Automated access to the semantic content of surveillance videos of interest, basically to detect anomalous situations in the scene. An automated video surveillance system should support both real-time alarm generation and offline inspection components to satisfy the requirements of the operators.[1] On either side, the input video stream should be processed adequately so that the actions are correctly analyzed. The primary challenges are the large input size and the high variability of the audio-visual features[2]; hence it remains a challenging issue to access the semantic content of the videos automatically.

Automated video surveillance processing generally starts with the detection of moving regions/objects as the first step in most of the existing surveillance systems (e.g., Refs. 3 and 4). Background foreground segmentation is widely studied, and techniques based on the running average with learning constant,[3] the running Gaussian average,[5] the mixture of Gaussians,[6] the average median of a set of previous frames,[7] the kernel density estimators,[8] and the codebook model[9] exist in the literature. Temporal template-based methods are also used to detect moving objects.[10,11] This first step is followed by tracking, classification,[12] and modeling activities to detect unusual object behaviors,[13] especially human activities.

One of the basic aims in understanding the behaviors of the objects is detecting the anomalies in the activities of the objects, having observed their patterns.[14,15] *Anomaly detection* refers to the problem of finding patterns in data that do not conform to expected behavior.[16] Abnormal situations and anomalies are reported to the operator and/or stored in a database for later inspection,[17] which requires efficient processing, indexing, and retrieval.

We propose a keyframe labeling technique for event classification in indoor surveillance with a fixed camera, based on a simple yet effective keyframe detection scheme. The underlying data model is constructed with respect to the moving regions in each frame, which are represented by a grid-based foreground mask, called the *motion appearance mask* (MAM). A keyframe is detected if a change occurs in the MAM of a frame compared to the previous frame. The keyframes are categorized into four simple types, namely JOIN, SPLIT, MOVE, and STOP, based on the appearance of the identified moving regions. The input stream is represented as a temporally ordered sequence of keyframe labels, and the event classification is carried out on this compact representation. Since the input size is significantly reduced with this representation, the detection and after-the-fact analysis tasks are facilitated.

We also provide mechanisms to detect a set of events, including *crossover*, *deposit*, and *pickup*, which may be considered peculiar for an indoor surveillance system. To this end, we devise three separate finite state automata (FSAs) to recognize the sequences corresponding to these behaviors. The inputs of these FSAs are the keyframe labels that we assign to the extracted keyframes. The basic aim in devising these FSA-based mechanisms is to validate the use of our keyframe labeling technique in surveillance event classification. It has been shown through the experimental results that the use of our keyframe labeling technique with the FSA-based mechanisms yields a reasonable event detection and classification performance.

0091-3286/2010/$25.00

The rest of the paper is organized as follows: Section 2 discusses related studies. The keyframe labeling technique that we propose is discussed in Sec. 3. The algorithms based on finite state automata to detect a set of basic events, such as crossover, deposit, and pickup, are provided in Sec. 4. The results of the performance experiments are presented in Sec. 5, and Sec. 6 concludes the paper.

2 Related Work

There exist quite a number of systems in the literature dealing with monitoring and anomalous behavior detection (e.g., Refs. 3, 4, and 18). Many techniques are proposed for dynamic scene segmentation as the preprocessing phase of anomaly detection. One of the widely used approaches for scene segmentation is background foreground segmentation. Depending on the complexity of the background, various techniques based on the running average with learning constant,[3] the running Gaussian average,[5] the mixture of Gaussians,[6] the average of median of a set of previous frames,[7] the kernel density estimators,[8] and the codebook model[9] are employed. The background model generally requires an initialization step, which can be applied as a part of the model or as a separate scheme (e.g., Ref. 19). Temporal template-based methods[10,11] are also used for dynamic scene segmentation to detect moving objects.

One of the challenging tasks in monitoring is detecting the abnormal actions caused by moving objects in the scene. The video surveillance data have both spatial and temporal characteristics, and the anomalies are caused by motion or insertion of foreign object(s) into the scene.[15] Each data point has a few continuous attributes, such as color, lightness, and texture, and the anomalies to be detected are either anomalous points or regions in the scene.[16] The activities of the moving objects has to be modeled to detect unusual object behaviors in terms of the observed continuous attributes (e.g., Refs. 10, 14, and 20). Anomalous events are also detected by analyzing motion trajectories of objects by employing an unsupervised clustering approach.[21] One of the key challenges in this domain is the large input size. Online anomaly detection techniques are required as well as offline processing support[1] for a complete video surveillance system. Below, some of the existing surveillance systems are discussed.

The techniques for anomaly detection are generally employed within video surveillance systems designed for continuously monitoring the environments. The video surveillance and monitoring (VSAM) system proposed by Collins et al. is one of the complete prototypes for object detection, tracking, and classification.[3] The hybrid algorithm developed in that work is based on adaptive background subtraction by three-frame differencing. The background maintenance scheme is based on a classification of pixels (either moving or nonmoving) performed by a simple threshold test. A model is provided on temporal layers for pixels and pixel regions in order to better detect stop-and-go movements.

IBM's Smart Surveillance System (S3)[22] is an advanced surveillance system that provides the capability to automatically monitor a scene, manage the surveillance data, perform event-based retrieval, receive real-time event alerts through standard web infrastructure, and extract long-term statistical patterns of activity. It also provides middleware for surveillance, namely MILS (Middleware for Large Scale Surveillance),[23] which provides a complete solution for video surveillance, including data-management services that can be used for building large-scale systems. MILS also provides query services for surveillance data, including object-specific attributes. The system employs relevance feedback and data-mining facilities to increase its effectiveness.

Knight[24] is an automatic multicamera surveillance system deployed in a variety of sites ranging from railway security to law enforcement. It detects, categorizes, and tracks moving objects in the scene, flags significant events, and presents a summary in terms of keyframes and a textual description of observed activities to a human operator for final analysis and decision.

Haritaoğlu et al.[10] propose a model for real-time analysis of people's activities. Their model uses a stationary camera and background subtraction to detect the regions corresponding to a specific person(s). Their system, called W^4, uses shape information to locate people and their body parts (head, hands, feet, and torso). The system operates on monocular grayscale video data, and no color cues are used. Creating models of people's appearances helps track interactions (e.g., occlusions) and simultaneous activities. The system uses a statistical background model holding a bimodal distribution of intensity at each pixel to locate people. The system is capable of detecting a single person, multiple persons, and multiple-person groups, in various postures.

Lyons et al.[25] present a system called Video Content Analyzer (VCA), the main components of which are background subtraction, object tracking, event reasoning, graphical user interface, indexing, and retrieving. VCA differentiates between people and objects, and the main events it recognizes are *entering scene*, *leaving scene*, *splitting*, *merging*, and *depositing/picking-up*. Brodsky et al.[26] describe a system for indoor visual surveillance, specifically for use in retail stores and homes. They assume a stationary camera and use background subtraction. A list of events that the object participates in is stored for each object, simply, *entering, leaving, merging*, and *splitting*. Both of these techniques operate at the pixel level and the region level, whereas we provide techniques to transform the input stream into an event sequence representation, which is easier to process and has lower storage costs.

Kim and Hwang present an object-based video abstraction model, where a moving-edge detection scheme is used for video frames.[27,28] A semantic shot-detection scheme is employed to select object-based keyframes. When a change occurs in the number of moving regions, the current frame is declared as a keyframe, indicating that an important event has occurred. This scheme also facilitates the detection of important events. If the number of moving objects remains the same in the next frame, a shape-based change detector is applied to the remaining frames. The use of keyframes in this approach is very similar to our keyframe detection scheme; however, we utilize a keyframe detection scheme with inverted tracking data model and extend it further by assigning descriptive labels to the keyframes.

In Ref. 29, a view-based multiple-object tracking system is proposed, including a human action recognition scheme. The basic aim in that work is to recognize human actions in an interactive virtual environment even when the actions are not abnormal. The blob-tracking phase that they have developed assigns labels to each blob based on its previous motion and current motion. The labels used are *continue*, *merge*, *split*, *appear*, and *disappear*, and an inference graph is maintained to track multiple objects simultaneously. The labeling mechanisms of this scheme and ours are similar;

```
Inputs:  V, the input video stream;
         t_s, the size threshold;
         t_o, the temporal object appearance threshold;
         t_d, the temporal detection threshold;
Outputs: K, the list of keyframe labels;

1.  for each frame f in V
        /* moving region extraction */
2.      extract the set of moving regions R_f
3.      for each element r in R_f
4.          if size(r) < t_s
5.              remove r from R_f
6.          else
7.              increment temporal-appearance(r)
                /* inverted tracking */
8.              if temporal-appearance(r) > t_o
9.                  set the cell(r) to 1 in MAM_f
        /* keyframe detection */
10.     if isKeyframe(MAM_f)
11.         compute keyframe label l_f
12.         increment temporal-count(l_f)
13.         if temporal-count(l_f) > t_d
14.             push l_f onto K
15.             MAM_{f-1} ← M_f
16.             clear M_f
17. endfor
```

Fig. 1 The pseudocode of the keyframe labeling algorithm.

however, we assign labels to a frame as a global representation of the events that occurred at the frame.Moreover, we apply a keyframe-based technique to narrow the storage and processing requirements.

3 Keyframe Labeling

In video processing, storage requirement is a very crucial issue due to huge size of a video data set. Keyframe-based video-processing techniques are popular because they reduce the storage requirements significantly by storing only the data at the keyframes. A keyframe is generally identified when there is a change in the spatiotemporal relations among the salient objects in the scene. In video surveillance, there are abnormal behaviors to be detected, and hence, there could be other conditions, based on the change in the global motion of the scene, to detect keyframes. Our keyframe detection algorithm categorizes the keyframes into four primitive types, namely JOIN, SPLIT, MOVE, and STOP, based on the appearances of the extracted moving regions. These four labels are among the primitive event types, and it is observed that they can be used to detect typical abnormal behaviors such as crossover, deposit, and pickup. As a result of this step, a label is assigned for each keyframe, and the input video stream is represented as a sequence of events.

The keyframe labeling technique relies on moving-region extraction and tracking steps, where the extracted moving regions are indexed with respect to a grid-based representation. The appearances of the identified moving regions are stored in the *motion appearance mask* (MAM) *for each frame* f; a 1 in this mask represents the presence of a motion in that cell. The keyframe labeling computations for f are performed based on MAM_f and MAM_{f-1}. Hence, the keyframe labeling technique produces a temporal ordering of the keyframe labels as an event sequence that can be used to classify a set of basic potentially abnormal events, such as crossover, deposit, and pickup.

The pseudo code of our keyframe-labeling algorithm is given in Fig. 1. At the first step, the moving regions are extracted for each frame. This step includes a foreground extraction scheme where morphological operations are applied *a priori*. The morphological operations are used to group the moving pixels into moving regions by the help of size filters, in terms of minimum and maximum object size. The salient regions are extracted through these grouping and filtering operations at the end of the first step. At the next step, the motion appearance mask of the frame is computed and compared with that of the previous frame in order to detect whether the current frame is a keyframe. At the last step, the identified keyframe is labeled. In both the moving-region detection and the keyframe detection step, temporal filtering is applied to minimize the effect of temporal noise. Temporal filtering is applied in both the moving-region detection and the keyframe detection steps. For the former, the number of frames that the extracted region has appeared is used in a thresholding scheme. For the latter, the number of frames that a keyframe label is identified consecutively is thresholded.

(a) (b) (c)

Fig. 2 The noise reduction filters applied on a sample frame at a moving-region extraction step: (a) original frame from PETS 2006 S1-T1-C data set[30] at t_1, (b) processed frame at t_1, and (c) processed frame at $t_2 = t_1 + 0.25$ s.

3.1 *Moving-Region Extraction*

We employ an adaptive background maintenance scheme to extract the moving regions, similar to the one proposed in Ref. 3. We combine the scheme with three-frame differencing to detect the moving pixels. Then, we apply region grouping methods and morphological operations to these pixels to identify the moving regions.

This technique can be described as follows: Let $I_f(x, y)$ denote the intensity value of a pixel at (x, y) in video frame f. Hence, $M_f(x, y) = 1$ if (x, y) is moving in frame f, where $M_f(x, y)$ is a vector holding moving pixels. A threshold vector $T_f(x, y)$ for a frame f is needed for detecting pixel motions. The basic test condition to detect moving pixels with respect to $T_f(x, y)$ can be formulated as

$$M_f(x, y) = \begin{cases} 1 & \text{if } (|I_f(x, y) - I_{f-1}(x, y)| > T_f(x, y)) \text{ and} \\ & \quad (|I_f(x, y) - I_{f-2}(x, y)| > T_f(x, y)), \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The (moving) pixel intensities larger than the background intensities $[B_f(x, y)]$ are used to fill in the region of a moving object. This step requires a background maintenance task based on the previous intensity values of the pixels. Similarly, the threshold is updated based on the observed moving-pixel information in the current frame. A statistical background and threshold maintenance scheme is employed, as follows:

$$B_0(x, y) = 0, \tag{2}$$

$$B_f(x, y) = \begin{cases} \alpha B_{f-1}(x, y) + (1 - \alpha) I_{f-1}(x, y), & M_f(x, y) = 0, \\ B_{f-1}(x, y), & M_f(x, y) = 1, \end{cases} \tag{3}$$

$$T_0(x, y) = 1, \tag{4}$$

$$T_f(x, y) = \begin{cases} \alpha T_{f-1}(x, y) + (1 - \alpha)[k \\ \quad \times |I_{f-1}(x, y) - B_{f-1}(x, y)|], & M_f(x, y) = 0, \\ T_{f-1}(x, y), & M_f(x, y) = 1, \end{cases} \tag{5}$$

where α is the learning constant and the constant k is set to 5 in Eq. (5).[3]

We employ a view-based motion-tracking approach similar to the motion history image (MHI) technique proposed in Ref. 11. The MHI technique detects and tracks the parameters (i.e., structure and orientation) of the moving regions. In an MHI, the pixel intensity is encoded as a function of the temporal history of the motion at that pixel, where the pixels that moved more recently are brighter. $\mathrm{MHI}_f(x, y)$ of f is constructed by the update rule

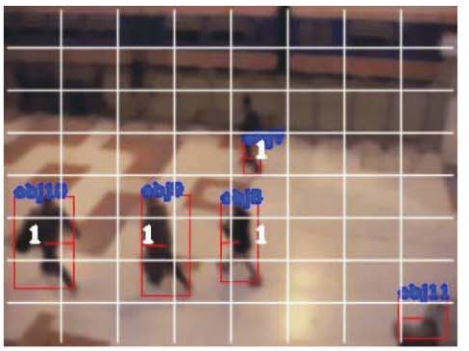
$$\mathrm{MHI}_f(x, y) = \begin{cases} \tau, & M_f(x, y) = 1, \\ \max(0, \mathrm{MHI}_{f-1}(x, y) - 1), & M_f(x, y) = 0, \end{cases} \tag{6}$$

where τ denotes the temporal extent of a motion.

A set of filters is applied to reduce the effect of noise in moving-region detection. First of all, a distance filter is applied to the extracted moving regions, such that the closer regions are joined. The distance threshold is adjusted with respect to the perimeter of the smaller region to be joined. As the next step, a size filtering is applied to the moving regions, which filters out the ones below the size threshold t_s in terms of both area and perimeter. The distance and size filtering thresholds are computed as functions of the width and height of the frame to preserve the scale invariance.

The last filtering scheme that we employ is temporal filtering. The temporal appearances of a moving region are counted, and the region is filtered out if it fails to be present in a predefined number of frames. The temporal threshold t_0 is computed according to the frame rate of the input stream. For example, for a temporal threshold duration of 0.25 s, t_0 will be 6 if the input video stream is 24 frames/s. To elaborate further, the noise reduction filters that we applied are shown in Fig. 2 on a sample frame. In Fig. 2(a) and 2(b), the original frame and its corresponding processed version at time t_1 are shown, respectively. The object in the bottom right corner in Fig. 2(b) has failed to pass the temporal filter, since it has not appeared in a sufficient number of frames. However, as shown in Fig. 2(c), the same object is extracted, since it passed the temporal filter t_0. On the other hand, the smallest rectangle without an object label in the mid-left part of the scene in Fig. 2(c) is failed to pass the size threshold.

(a)



(b)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	**1**	0	0	0
0	0	0	0	0	0	0	0
1	0	**1**	0	**1**	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(c)

Fig. 3 Motion appearance mask (MAM) computation on a sample frame: (a) original frame from PETS 2006 S1-T1-C data set,[30] (b) processed frame for 8×8 grid, and (c) MAM of the frame.

3.2 *Computation of the Motion Appearance Mask*

The actual content of a frame consists of the extracted moving regions, and in our approach a grid-based mapping is held along with the actual content. In this scheme, instead of processing moving regions, the corresponding grid-based representation, which we call the *motion appearance mask* (MAM), is processed for semantic analysis of the input stream. The video frame $I(x, y)$ is divided into a predefined number of cells corresponding to the subdivisions in the x and y directions. The appearances of the identified moving regions are stored in the MAM of a frame f; a 1 in this mask represents the presence of a motion in the corresponding cell. The moving-object appearances in a cell are computed with respect to the center of mass (c_m) of a region corresponding to the moving object. Figure 3 illustrates the computation of the MAM that we employ on a sample video frame by using an 8×8 grid.

This grid-based representation of the extracted moving regions at each frame significantly reduces the storage space and the processing cost during the keyframe detection process. The reason is that the amount of information to be processed is significantly lower while using MAMs for keyframe detection instead of the pixel-level information of the extracted moving regions. The effectiveness of this grid-based representation depends on the effectiveness of the moving-region extraction step, which is an expected prerequisite for most of the event classification techniques. In our scheme, the effect of false detection in moving-region extraction is minimized with the use of distance, size, and temporal filters. Another well-known difficulty is the occlusion problem during moving-region extraction. Since a MAM holds the appearance of the motion at a specific cell without knowing the actual moving region, the occlusion of moving regions is not a problem in our grid-based representation.

The most challenging task in this scheme is the selection of the grid size, which is crucial for the structure of the motion appearance mask. Even though the camera is generally fixed in surveillance videos, an appropriate grid size has to be specified to represent the motion effectively, since the field of view of the cameras or depth of the scenes may change in different applications. Object-based heuristics could be applied to select the grid size as a function of the smallest or the largest object/region size. These heuristics might work, but in our opinion, the grid size has to be selected in a way that is independent of the pixel level parameters (e.g., size, perimeter). The main reason is that the grid-based index representation is a view-based data model on top of the pixel level, and hence, the view-based scheme has more adaptive processing capabilities on various data sets when loosely coupled with the pixel-level representation. Another reason is that using a variable grid size based on the pixel-level parameters of the objects makes on-the-fly processing complex. Hence, we used fixed grid sizes in our experiments.

3.3 *Keyframe Detection*

The keyframe detection scheme uses only the motion appearance masks of the current frame and the previous frame, which makes the processing easier. The pseudocode of the keyframe detection algorithm is shown in Fig. 4. This algorithm corresponds to steps 10 and 11 of the keyframe labeling algorithm shown in Fig. 1, where a temporal filtering is applied after the keyframe detection to reduce the misdetection rate based on sudden changes in MAM_f. If the keyframe fails to be the same for a duration of t_d frames, it is considered as a sudden change in the scene. A keyframe is formally defined as follows: Frame f is a *keyframe* if $(\mathrm{MAM}_f \neq \mathrm{MAM}_{f-1}) \vee (\mathrm{MAM}_f = \mathrm{MAM}_{f-1} = \cdots = \mathrm{MAM}_{f-k}$ if $k \geq t_{\mathrm{stop}})$, where t_{stop} denotes the maximum allowed number of frames without motion.

We employed another filtering mechanism to reduce the number of keyframes. The current frame is not detected as a keyframe if the current and previous frames are labeled as MOVE by the keyframe labeling algorithm. The idea behind this filtering scheme is that consecutive MOVE labels have no use in the mechanisms to classify surveillance events. Figure 5 shows the effect of this keyframe filtering step on two sample videos in the PETS 2006[30] data set. The subjective ground truth for the S1-T1-C data set [see Fig. 5(a)] is 148 keyframes, and for S2-T3-C data set [see Fig. 5(b)] is 142 keyframes. The extracted keyframe count is significantly reduced after the filtering step, and the final keyframe counts are 154 and 138 on the average for these two sample videos, respectively. The average is computed using different values of the grid size parameter, as shown in Fig. 5(a) and 5(b).

Besides reducing the number of extracted keyframes, another gain is reducing the criticality of selecting appropriate values for the grid size parameter.As shown in Fig. 5(a) and 5(b), the number of extracted frames without filtering increases with the grid size, since the number of frames with

Inputs: f, the frame of the input video stream;
MAM_f, the motion appearance mask of f;
MAM_{f-1}, the motion appearance mask of $f-1$;
t_{stop}, the temporal threshold for detecting stop;
$|MAM_f|$ denotes the total number of 1s in MAM_f;
Outputs: l_f, the label of the keyframe;

```
1.    if |MAM_f| > |MAM_{f-1}|
2.       l_f = SPLIT
3.    else if |MAM_f| < |MAM_{f-1}|
4.       l_f = JOIN
5.    else if MAM_f ∧ MAM_{f-1} ≠ 0
6.       l_f = MOVE
7.    else /* MAM_f = MAM_{f-1} */
8.       stop-count ← stop-count +1
9.       if stop-count > t_stop
10.         l_f = STOP
```

Fig. 4 The pseudocode of the keyframe detection algorithm.

the MOVE label increases with smaller grid size values. However, the final keyframe count is almost constant when this keyframe filtering scheme is applied.

Since the labels assigned to the extracted keyframes do not change with this filter, the surveillance event classification performance is not affected. Hence, the input size is reduced by decreasing the number of extracted keyframes, and the effect of the grid size parameter is minimized without degrading the classification performance.

4 Surveillance Event Classification

Providing a general solution to anomalous event detection in video surveillance is still an open research area. Reasonable accuracies can be achieved for specific video surveillance applications by restricting the variation of the video data. Detecting anomalous events requires tracking the actions of moving regions. A typical video stream has too many frames, and hence too many moving regions to deal with. Keyframe-based techniques reduce the number of regions to be processed, but effective data models are required for surveillance event detection and classification. Pixel-level or region-level detection techniques may have high processing costs or performance limitations for on-the-fly detection, due to their large input size.

The keyframe labeling algorithm (see Fig. 1) that we employ transforms the input video stream to a textual representation of sequence of events in the video. The steps dealing with moving regions are performed once, and a textual representation of the video with a relatively small size is achieved. The event classification, then, becomes detecting a sequence of event labels in this input sequence. We devise three finite state automata for crossover, deposit, and pickup. Formally,

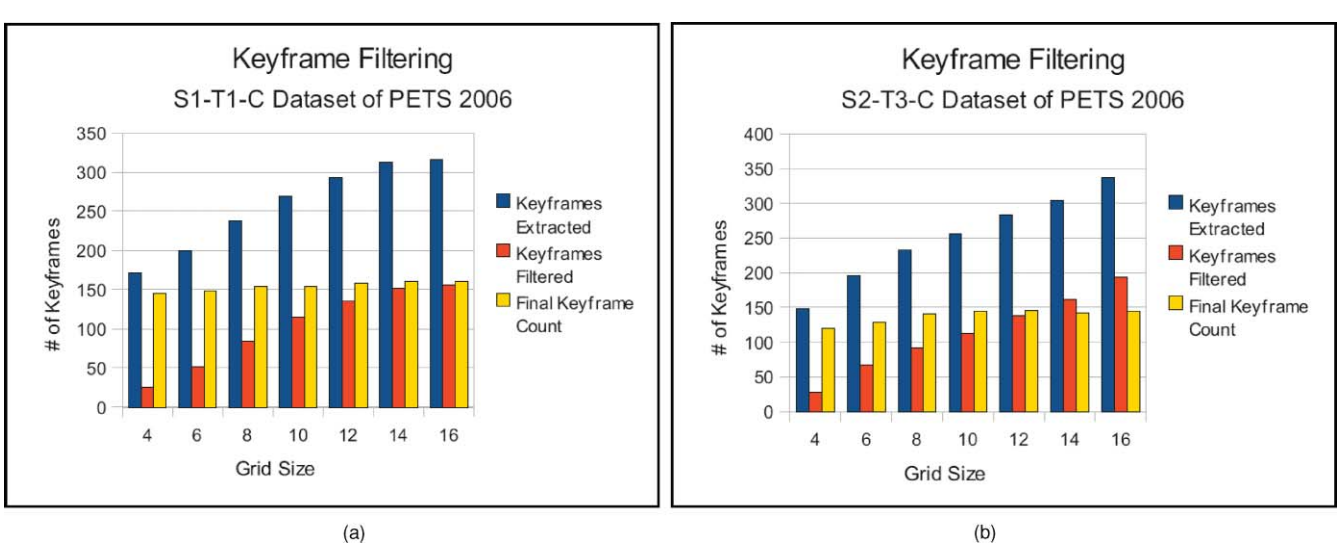
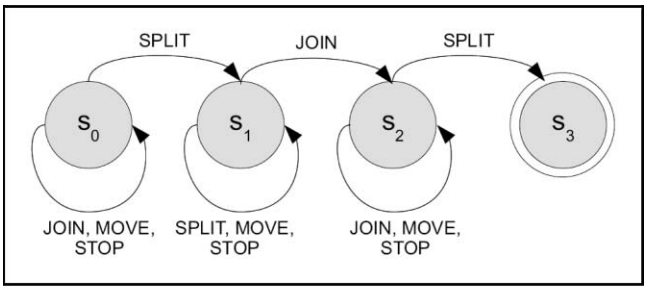


Fig. 5 Keyframe filtering applied to PETS 2006 data sets. (a) S1-T1-C; ground truth is 148 keyframes; 154 keyframes are extracted on the average. (b) S2-T3-C; ground truth is 142 keyframes; 138 keyframes are extracted on the average.



(a)

	SPLIT	JOIN	MOVE	STOP
S_0	S_1	S_0	S_0	S_0
S_1	S_1	S_2	S_1	S_1
S_2	S_3	S_2	S_2	S_2
S_3	S_0	S_0	S_0	S_0

(b)

Fig. 6 (a) The FSA for recognizing an event sequence for crossover. (b) The state transition function δ_C for the automaton detecting crossover. Here S_0 is the initial state, and S_3 is the final state accepting an event sequence for crossover.

a deterministic FSA is denoted as a quintuple $(\Sigma, S, S_0, \delta, F)$, where Σ is the input alphabet (a finite, nonempty set of symbols); S is a finite, nonempty set of states; S_0 is an initial state where $S_0 \in S$; δ is the state transition function such that $\delta : S \times \Sigma \rightarrow S$; and F is the set of final states, where $F \subset S$. The FSAs that we devised for crossover, deposit, and pickup are discussed with on this notation. The input to these FSAs is the sequence of keyframe labels representing the input video stream. Reasonable detection accuracies are achieved in our experiments.

4.1 *Crossover*

A *crossover* situation occurs when at least two moving objects have passed through each other in the video scene. For the two-object case, this event may occur in two different forms:

1. if the objects move in the same direction and the faster object passes the slower object, and
2. if the moving objects move in opposite directions and cross each other.

These situations can be extended for more than two objects in a similar manner. In both cases, tracking the moving objects according to their locations to detect a crossover situation imposes a high processing cost. The FSA-based approach proposed for crossover detection operates effectively because the input size is reduced.

Let $\text{FSA}_C = (\Sigma, S_C, S_0, \delta_C, F_C)$ represent the FSA detecting crossover event occurrences $S_C = \{S_0, S_1, S_2, S_3\}$, $\Sigma = \{\text{JOIN, SPLIT, MOVE, STOP}\}$, and $F_C = \{S_3\}$. Figure 6 presents the automation FSA_C in (a), and the state transition function δ_C in (b).

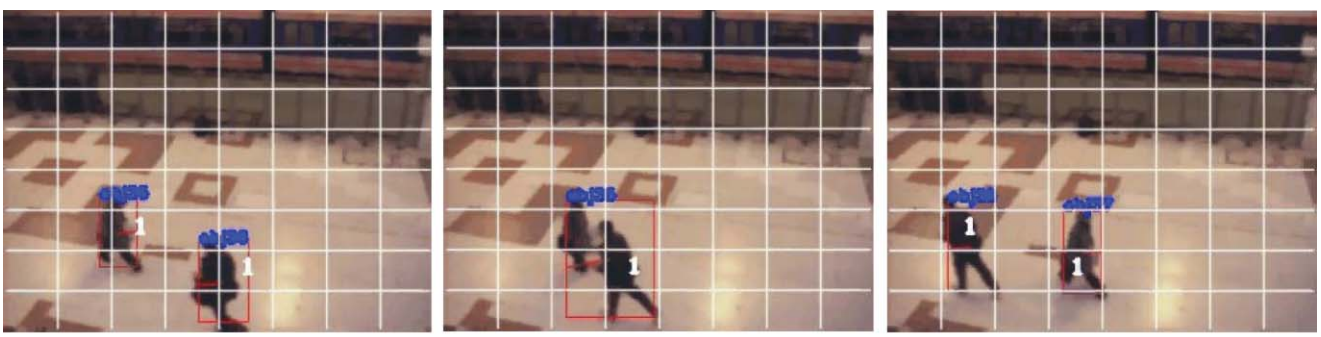
A sample crossover detection by FSA_C on a sample video of the PETS 2006 data set[30] is shown in Fig. 7. At the beginning, the active state s_c of FSA_C is at S_0. When the objects shown in Fig. 7(a) enter the scene, s_c becomes S_1, and eventually $|\text{MAM}_f| = 2$, where $|\text{MAM}_f|$ denotes the total number of 1's in MAM_f. When the execution reaches Fig. 7(b), $|\text{MAM}_f| = 1$ and $|\text{MAM}_{f-1}| = 2$, which signals JOIN; hence s_c reaches S_2. Finally, in Fig. 7(c), $|\text{MAM}_f| = 2$ again and $|\text{MAM}_{f-1}| = 1$, which signals SPLIT and makes s_c reach the final state S_3.

4.2 *Deposit*

A *deposit* situation occurs when a moving object leaves a smaller object (e.g., suitcase, bag) in the video scene. Effective motion models are required for detecting the moving object's action.

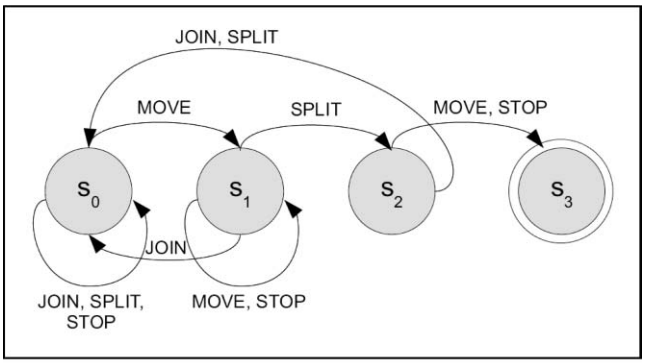
Let $\text{FSA}_D(\Sigma, S_D, S_0, \delta_D, F_D)$ represent the finite state automaton detecting deposit event occurrences $S_D = \{S_0, S_1, S_2, S_3\}$, $\Sigma = \{\text{JOIN, SPLIT, MOVE, STOP}\}$, and $F_D = \{S_3\}$. Figure 8 presents the automaton FSA_D in (a), and the state transition function δ_D in (b).

A sample deposit detection by FSA_D on a sample video of the PETS 2004 data set[31] is shown in Fig. 9. At the beginning, the active state s_d of FSA_D is at S_0. When the object shown in Fig. 9(a) enters the scene, s_d becomes S_1, and eventually $|\text{MAM}_f| = 1$. When the execution reaches Fig. 9(b), $|\text{MAM}_f| = 2$ and $|\text{MAM}_{f-1}| = 1$, which signals SPLIT; hence s_d reaches S_2. Finally, in Fig. 9(c), MOVE is detected when $|\text{MAM}_f| = 2$ and $|\text{MAM}_{f-1}| = 2$, which makes s_d reach the final state S_3.



(a) (b) (c)

Fig. 7 FSA state transitions for crossover detection for PETS 2006 S1-T1-C data set[30]: (a) transition from S_0 to S_1, (b) transition from S_1 to S_2, (c) transition from S_2 to S_3 and crossover detection.



(a)

	SPLIT	JOIN	MOVE	STOP
S_0	S_0	S_0	S_1	S_0
S_1	S_2	S_0	S_1	S_1
S_2	S_0	S_0	S_3	S_3
S_3	S_0	S_0	S_0	S_0

(b)

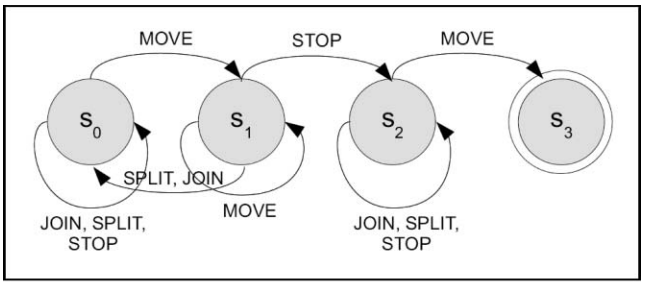
Fig. 8 (a) The FSA for recognizing an event sequence for deposit. (b) The state transition function δ_D for the automaton detecting deposit. Here S_0 is the initial state, and S_3 is the final state accepting the event sequence for deposit.

4.3 *Pickup*

Pickup can be considered as the dual of deposit; thus a pickup situation occurs when a moving object picks up a smaller object (e.g., suitcase, bag) in the video scene. Similarly, effective motion models are required for detecting the moving object's action.

Let $\text{FSA}_P = (\Sigma, S_P, S_0, \delta_P, F_P)$ represent the FSA detecting pickup event occurrences $S_P = \{S_0, S_1, S_2, S_3\}$, $\Sigma = \{\text{JOIN, SPLIT, MOVE, STOP}\}$, and $F_P = \{S_3\}$. Figure 10 presents the automation FSA_P in (a), and the state transition function δ_P in (b).

A sample pickup detection by FSA_P on a sample video of the PETS 2004 data set[31] is shown in Fig. 11. At the beginning, the active state s_p of FSA_P is at S_0. When the object shown in Fig. 11(a) enters the scene, s_d becomes S_1, and eventually $|\text{MAM}_f| = 1$. When the execution reaches Fig. 11(b), $|\text{MAM}_f| = 1$ but sufficiently many frames have passed for t_{stop}, which signals STOP; hence s_d reaches S_2. Finally, in Fig. 11(c), MOVE is detected when $|\text{MAM}_f| = 1$ and $|\text{MAM}_{f-1}| = 1$, which makes s_d reach the final state S_3.



(a)

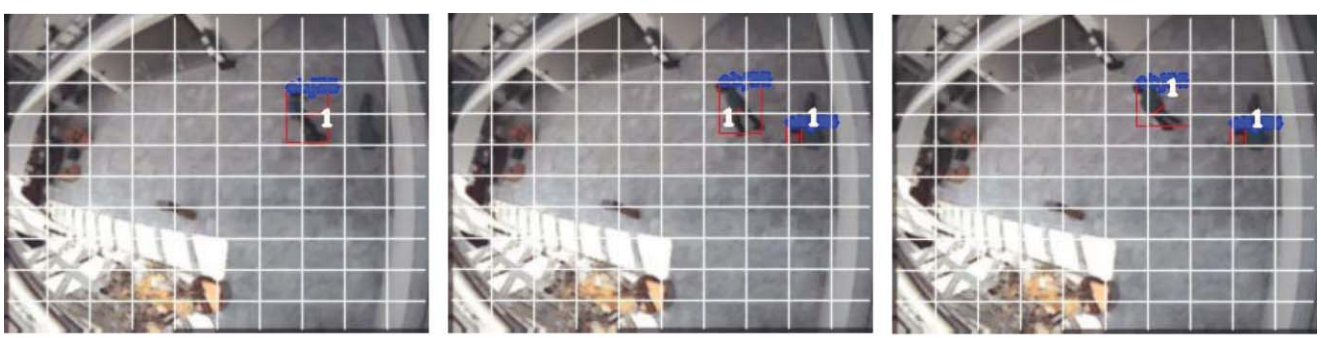
	SPLIT	JOIN	MOVE	STOP
S_0	S_0	S_0	S_1	S_0
S_1	S_0	S_0	S_1	S_2
S_2	S_2	S_2	S_3	S_2
S_3	S_0	S_0	S_0	S_0

(b)

Fig. 10 (a) The FSA for recognizing an event sequence for pickup. (b) The state transition function δ_P for the automaton detecting pickup. Here S_0 is the initial state, and S_3 is the final state accepting the event sequence for pickup.

5 Performance Experiments

To evaluate the performance of our event detection technique, we manually annotated two sample videos from PETS 2004,[31] namely leftbag and meetsplit, having 426 and 543 frames, respectively, and two sample videos from PETS 2006,[30] namely S1-T1-C3 and S2-T3-C3, having 2526 and 2763 frames, respectively. We employed a fivefold cross-validation method for the following experimental evaluation. Although there exist performance evaluation techniques based on object tracking[32] and event detection[33] in PETS 2006, a direct comparison of the performance of our keyframe labeling technique with that of the ones in PETS 2006 (e.g., Ref. 34) is not quite possible, since the latter use metrics based on the object position for performance evaluations. Hence, we utilized receiver operating characteristic (ROC) analysis based on the parameters of our technique



(a) (b) (c)

Fig. 9 FSA state transitions for deposit detection for PETS 2004 leftbag data set[31]: (a) transition from S_0 to S_1, (b) transition from S_1 to S_2, (c) transition from S_2 to S_3 and deposit detection.

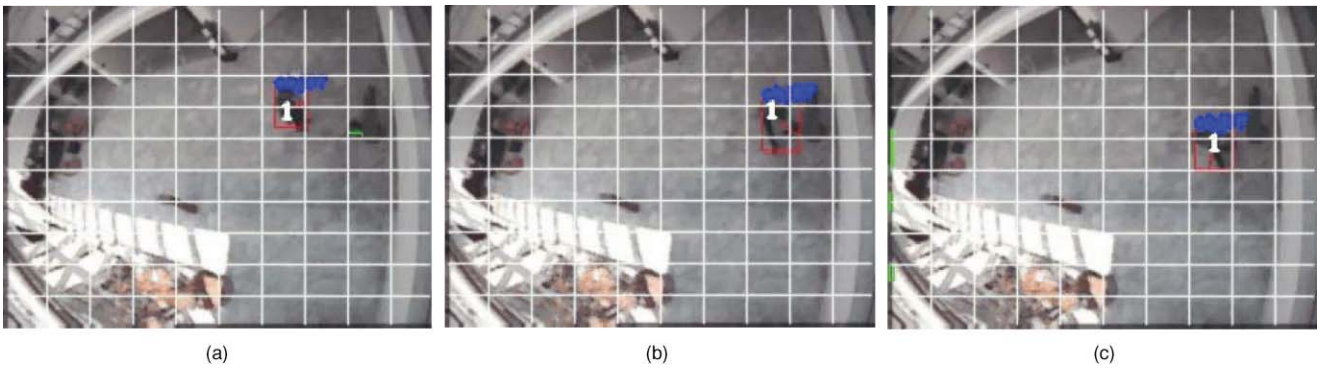


Fig. 11 FSA state transitions for pickup detection for the PETS 2004 leftbag data set[31]: (a) transition from S_0 to S_1, (b) transition from S_1 to S_2, (c) transition from S_2 to S_3 and pickup detection.

to validate its applicability for surveillance event classification using our ground truth data. The benchmark data sets provided by PETS 2004 and PETS 2006 are used in these analysis. The use of these widely accepted data sets enables us to evaluate the effectiveness of our technique.

ROC analysis is used to inspect the effect of a single parameter on the classifier by plotting the true-positive rate (TPR) and false-positive rate (FPR) values that are calculated while keeping all the other parameters fixed. It indicates the effectiveness of the classifier by altering values of a single parameter. Since our keyframe labeling algorithm yields exact keyframe labels instead of label percentages for keyframes, and our FSA-based schemes give binary output for surveillance event classification, a set of points is plotted on ROC curves. The points above the $x = y$ line are considered as good classification results, whereas the ones below are bad.

Figure 12 shows the ROC analysis results for inspecting the effect of the grid size parameter in surveillance event classification. TPR and FPR values were computed using the outputs of the FSA-based detection algorithms. The positive output frames for each of the classes were annotated manually, and a similar set of negative output frames was annotated for the analysis. For this experimental setup, the temporal detection threshold t_d is set to 3 frames for the PETS 2006 and 2 frames for the PETS 2004 data set. In Fig. 12(a), the only bad classification occurs when the grid size is 6, and the detection algorithm gives best results for the grid size 8. In Fig. 12(b), the classification for the grid sizes 10 and 12 are among the good ones, and 10 gave better results. The main reason behind this difference among data sets is the variation in the pixel-level properties, such as object sizes, object average velocities, etc. Obtaining a formula to express the appropriate grid size in terms of the pixel-level parameters is very hard. Hence, ROC analysis can be performed, as just discussed, to determine effective grid size values for the data sets. However, since the camera is generally fixed for surveillance data sets, this step can be taken once in preprocessing for each different camera setting, to minimize the overall processing cost.

Figure 13 shows the ROC analysis results for inspecting the effect of the temporal detection threshold parameter t_d in surveillance event classification. The TPR and FPR values are based on the outputs of the FSA-based detection algorithms. The positive output frames for each class are annotated manually, as well as a similar set of negative output frames for the analysis. For this experimental setup, the grid size is set

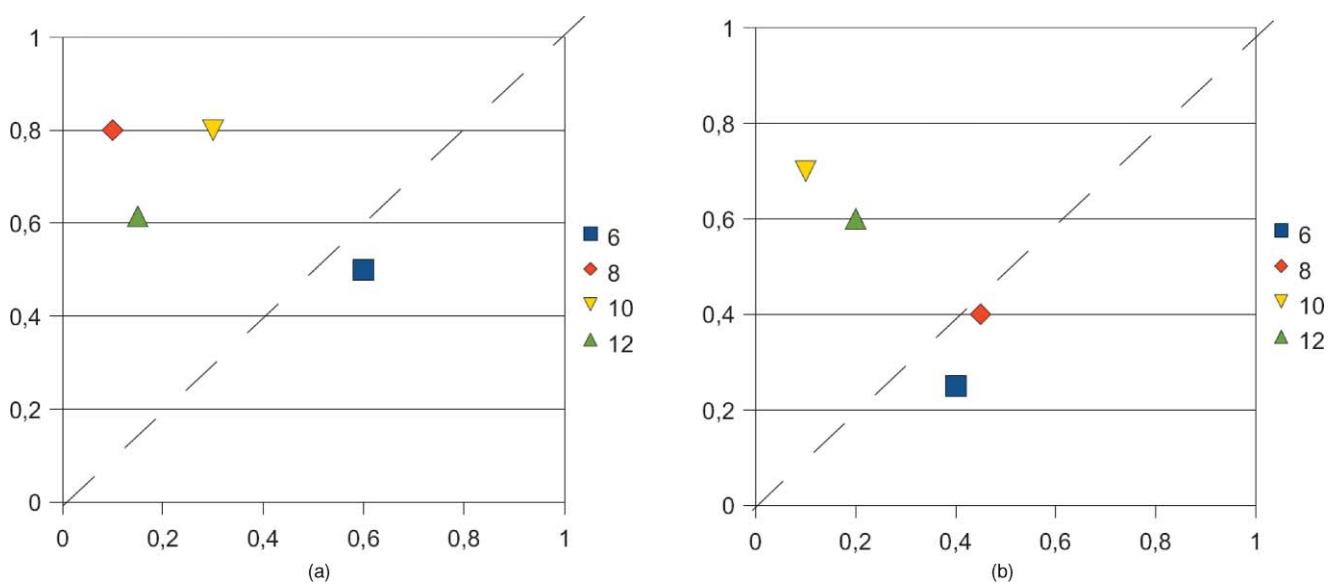


Fig. 12 ROC curve analysis for the grid size parameter: (a) PETS 2006 data set, (b) PETS 2004 data set.

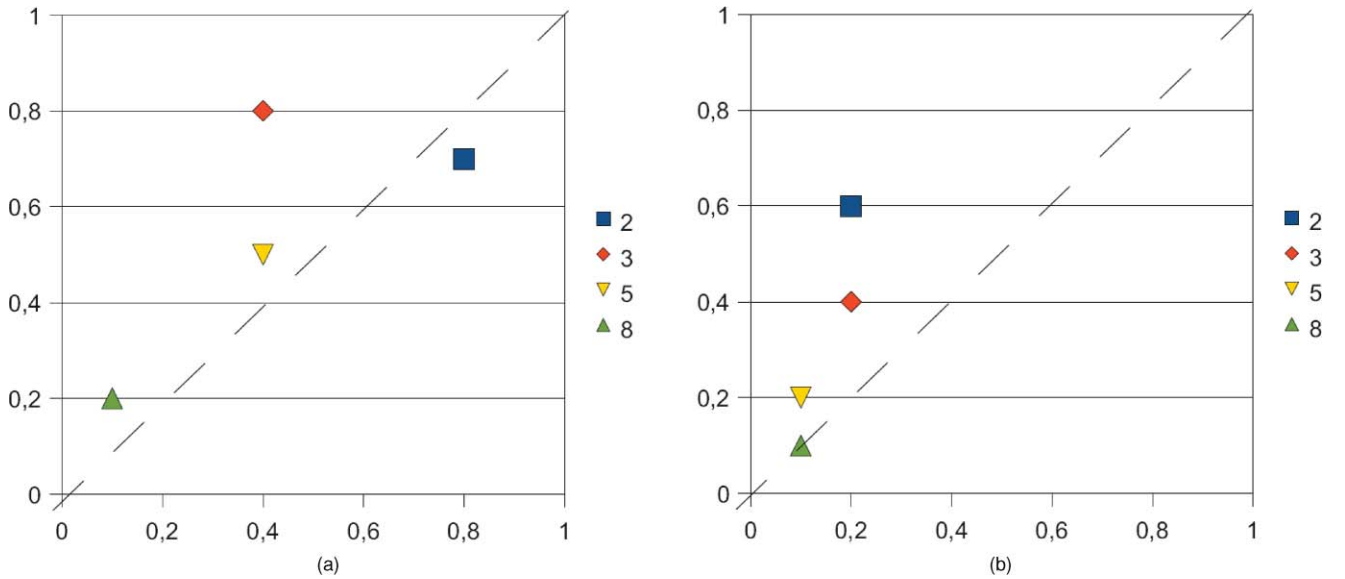


Fig. 13 ROC curve analysis for the temporal detection threshold parameter t_d: (a) PETS 2006 data set, (b) PETS 2004 data set.

to 8 for the PETS 2006 and 10 for the PETS 2004 data set. In Fig. 13(a), $t_d = 3$ gives the best results, whereas in Fig. 13(b), $t_d = 2$ detects the anomalies better than the other values. As expected, increasing the temporal detection threshold frame count lowers the detection accuracy significantly.

In our keyframe labeling technique, the outcome of the moving-region extraction scheme is crucial for the rest of the steps. By the help of the filtering steps that we employ to reduce the noise and improve the detection performance, the keyframe detection algorithm yields reasonable performance for event classification. To elaborate on this, we provide the results of a set of experiments in Table 1. The experiments were carried out on the PETS 2006 data set; the grid size was set to 6, and the temporal detection threshold was set to 3. The false-detection rate of the moving-region extraction step is significantly lowered by means of the filters. Since the keyframe detection scheme depends on the MAM of the frame, and since the grid-based foreground mask depends on the existence of the motion at a specific grid, occlusion would not be a problem for the keyframe labeling algorithm.

One of the primary advantages of our keyframe labeling scheme is the gain in storage, which is obtained simply by reducing the input size. In general, the storage gain of keyframe-based techniques in video processing can be expressed by the ratio of the number of keyframes extracted to the total number of frames. In order to compare the storage gain, a fair approach is to compare the input sizes of different techniques. In our keyframe labeling scheme, the processing for event detection and classification can be handled with the help of the extracted event label sequence, which yields a significant storage gain over object-based approaches. The input sizes of the object-level approaches are estimated by the total number of extracted objects, since the extracted objects have to be processed for event detection, whereas only the extracted keyframe labels are to be processed in our approach. To be fair, the number of objects is computed after completing all of the filtering steps. Figure 14 presents the results of

Table 1 The effect of filtering on moving-region extraction. Experiments on S1-T1-C3 and S2-T3-C3 data sets were performed with grid size set to 6 and temporal detection threshold set to 3.

	Count	
	S1-T1-C3	S2-T3-C3
Prior to filtering	5986	6027
Distance filtering	4212	4365
Size filtering	3743	4056
Temporal filtering	2419	2620
Ground truth	2480	2745
Frame count	2526	2763

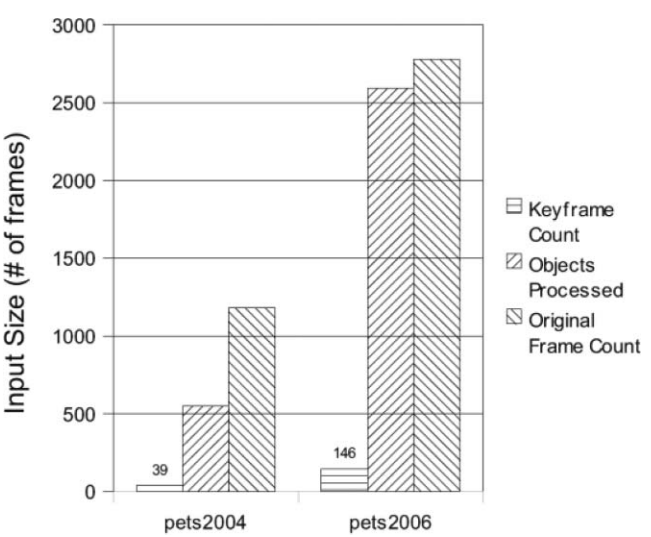


Fig. 14 Storage gain and reduction of the input size for detection for PETS 2004 and PETS 2006 data sets. The keyframe count that is extracted by our technique significantly reduces not only the input size for event detection but also the storage space for after-the-fact analysis.

this analysis. As expected, the keyframe-based approach has a significantly lower storage cost and input size.

The major drawback of our keyframe labeling algorithm is that it may not be suitable for crowded scenes, such as video streams of the PETS 2007[35] data set. The ROC analysis gives poor results for this data set, and the detection accuracy is low. The main reasonis that it is very hard to identify the keyframe with a single label in a crowded scene. Too many SPLIT, JOIN, and MOVE events occur simultaneously. Another drawback is the algorithm's behevior when the object size is very large (e.g., an object occupies nearly one-fourth of the video frame). In such cases, forming a grid for representing the moving regions in the MAM does not bring significant improvement over ordinary detection techniques.

6 Conclusion

We propose a keyframe labeling technique, which simply assigns labels to the keyframes extracted by a keyframe detection algorithm. Our keyframe detection technique relies on a grid-based index representation, which is used to compute the motion appearance mask (MAM) of the frame. A keyframe is detected if a change occurs in the MAM of the frame with respect to that of the previous frame. The keyframes are categorized into four simple types based on the appearance of the identified moving regions. As a result of the keyframe labeling process, the input stream is represented as a temporally ordered sequence of keyframes. The surveillance event classification task is carried out on this sequence; hence, the complexity of the detection is reduced. The keyframe labeling technique reduces the large input size for on-the-fly processing, and thus reduces the storage requirements for after-the-fact analysis.

We also provide FSA-based mechanisms to detect a typical set of anomalous behaviors. We devise three separate FSAs to recognize sequences corresponding to a typical set of events, the inputs of which are the sequence of keyframe labels that we assign to the extracted keyframes. The performance experiments based on the benchmark data sets PETS 2004 and PETS 2006 show that the FSA-based approach, together with the keyframe labeling technique, provides effective on-the-fly anomaly detection, in that reasonable detection performance is achieved.

Acknowledgments

This work was supported in part by the European Commission's 6th Framework Program's MUSCLE Network of Excellence Project, with grant No. FP6-507752, and by the Scientific and Technical Research Council of Turkey (TÜBİTAK), with grant No. EEEAG-105E065.

References

1. C. S. Regazzoni, V. Ramesh, and G. L. Foresti, "Scanning the issue/technology: special issue on video communications, processing, and understanding third generation surveillance systems," *Proc. IEEE* **89**(10), 1355–1367 (2001).
2. N. Haering, P. L. Venetianer, and A. Lipton, "The evolution of video surveillance: an overview," *Mach. Vis. Appl.* **19**(5–6), 279–290 (2008).
3. R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, and L. Wixson, "A system for video surveillance and monitoring," Technical Report CMU-RI-TR-00-12, Carnegie Mellon Univ., Robotics Insti. (2000).
4. D. Duque, H. Santos, and P. Cortez, "The OBSERVER: an intelligent and automated video surveillance system," in *Proc. Int. Conf. on Image Analysis and Recognition (ICIAR)*, A. Campilho and M. Kamel, Eds., pp. 898–909, Springer-Verlag, Berlin (2006).
5. C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Trans. Pattern Anal. Mach. Intell.* **19**(7), 780–785 (1997).
6. C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition (CVPR'99)*, pp. 246–252 (1999).
7. R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts and shadows in video streams," *IEEE Trans. Pattern Anal. Mach Intell.* **25**(10), 1337–1342 (2003).
8. A. Elgammal, R. Duraiswami, D. Harwood, and L.S. Davis, "Background and foreground modeling using non-parametric kernel density estimation for visual surveillance," *Proc. IEEE*, **90**, 1151–1163 (2002).
9. K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground/background segmentation using codebook model," *Real-time Imaging* **11**(3), 172–185 (2005).
10. İ. Haritaoğlu, D. Harwood, and L. Davis, "W4: real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 809–830 (2000).
11. A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *IEEE Trans. Pattern Anal. Mach. Intell.* **23**(3), 257–267 (2001).
12. E. Rivlin, M. Rudzsky, R. Goldenberg, U. Bogomolov, and S. Lepchev, "A real-time system for classification of moving objects," in *Proc. Int. Conf. on Pattern Recognition*, Vol. 3, pp. 688–691 (2002).
13. T. Xiang and S. Gong, "Incremental and adaptive abnormal behaviour detection," *Comput. Vis. Image Understanding* **111**, 59–73 (2008).
14. C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 747–757 (2000).
15. W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Trans. Systems Man Cybernet. C* **34**(3), 334–352 (2004).
16. V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," *ACM Comput. Surveys* **41**(3), Art. 15 (2009).
17. N. Durak, A. Yazıcı, and R. George, "Online surveillance video archive system," in *Proc. 13th Int. Multimedia Modeling Conf. Part I*, T.-J. Cham , J. Cai, C. Dorai, D. Rajan, T.-S. Chua, and L.-T. Chia, Eds., pp. 376–385, Springer-Verlag, Berlin (2007).
18. G. Paschos and F. P. Valavanis, "A color texture based visual monitoring system for automated surveillance," *IEEE Trans. Systems Man Cybernet. C* **29**(2), 298–307 (1999).
19. D. Gutchess, M. Trajkovic, E. Cohen-Solal, D. Lyons, and A.K. Jain, "A background model initialization algorithm for video surveillance," in *Proc. Int. Conf. on Computer Vision*, pp. 733–740 (2001).
20. R. Hamid, A. Johnson, S. Batta, A. Bobick, C. Isbell, and G. Coleman, "Detection and explanation of anomalous activities: representing activities as bags of event *n*-grams," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1031–1038 (2005).
21. Y. Zhou, S. Yan, and T. S. Huang, "Detecting anomaly in videos from trajectory similarity analysis," in *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME)*, pp. 1087–1090 (2007).
22. Y. Tian, L. Brown, A. Hampapur, M. Lu, A. Senior, and C. Shu, "IBM Smart Surveillance System (S3): event based video surveillance system with an open and extensible framework," *Mach. Vis. Appl.* **19**(5–6), 315–327 (2008).
23. A. Hampapur, L. M. Brown, J. Connell, M. Lu, H. Merkl, S. Pankanti, A. W. Senior, C.-F. Shu, and Y.-L. Tian, "Multi-scale tracking for smart video surveillance," *IEEE Trans. Signal Process.* **22**(2), 38–51 (2005).
24. M. Shah, O. Javed, and K. Shafique, "Automated visual surveillance in realistic scenarios," *IEEE Multimedia* **14**(1), 30–39 (2007).
25. D. M. Lyons, T. Brodsky, E. Cohen-Solal, and A. Elgammal, "Video content analysis for surveillance applications," in *Proc. Philips Digital Video Technologies Workshop* (2000).
26. T. Brodsky, R. Cohen, E. Cohen-Solal, S. Gutta, D. Lyons, V. Philomin, and M. Trajkovic, "Visual surveillance in retail stores and in the home," in *Advanced Video-Based Surveillance Systems*, Kluwer Academic Publishers, pp. 50–61 (2001).
27. C. Kim and J. N. Hwang, "Fast and automatic video object segmentation and tracking for content-based applications," *IEEE Trans. Circuits Systems Video Technol.* **12**(2), 122–129 (2002).
28. C. Kim and J. N. Hwang, "Object-based video abstraction for video surveillance systems," *IEEE Trans. Circuits Systems Video Technol.* **12**(12), 1128–1138 (2002).
29. J. Choi, Y. Cho, K. Cho, S. Bae, and H. S. Yang, "A view-based multiple objects tracking and human action recognition for interactive virtual environments," *Int. J. Virtual Reality* **7**(3), 71–76 (2008).
30. "PETS 2006," in *Ninth IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance Benchmark Data*, http://www.cvg.rdg.ac.uk/PETS2006/data.html (2006).
31. "PETS 2004," in *Seventh IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance Benchmark Data*, http://www-prima.inrialpes.fr/PETS04/caviar_data.html (2004).

32. F. Bashir and F. Porikli, "Performance evaluation of object detection and tracking systems," in *Proc. Ninth IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2006)*, pp. 7–14 (2006).
33. X. Desurmont, R. Sebbe, F. Martin, C. Machy, and J.-F. Delaigle, "Performance evaluation of frequent events detection systems," in *Proc. Ninth IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2006)*, pp. 15–22 (2006).
34. K. Smith, P. Quelhas, and D. Gatica-Perez, "Detecting abandoned luggage items in a public space," in *Proc. Ninth IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2006)*, pp. 75–82 (2006).
35. "PETS 2007," in *Tenth IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance*, http://www.pets2007.net (2007).

Ediz Şaykol received the BSc degree from the Computer Engineering and Information Science Department, Bilkent University, Ankara, Turkey, in 1999. He received his MSc and PhD degrees from the Computer Engineering Department, Bilkent University, in 2001 and 2009, respectively. His current research interests include multimedia database and video surveillance systems, and semantic and low-level feature extraction in video.

Muhammet Baştan is a PhD candidate in the Department of Computer Engineering at Bilkent University, Ankara, Turkey. He has an MS in electronics engineering and computer science from Sabanci University, Istanbul, Turkey. His research interests include computer vision, pattern recognition, multimedia retrieval, MPEG-7, image and video processing, saliency, segmentation, and annotation.

Uğur Güdükbay received a BSc degree in computer engineering from Middle East Technical University, Ankara, Turkey, in 1987. He received his MSc and PhD degrees, both in computer engineering and information science, from Bilkent University, Ankara, Turkey, in 1989 and 1994, respectively. Then, he conducted research as a postdoctoral fellow at the University of Pennsylvania, Human Modeling and Simulation Laboratory. Currently, he is an associate professor at Bilkent University, Department of Computer Engineering. His research interests include multimedia database and video surveillance systems, and various aspects of computer graphics, including physically based modeling, human modeling and animation, multiresolution modeling and rendering, and visualization. He is a senior member of the IEEE and ACM.

Özgür Ulusoy received the PhD degree in computer science from the University of Illinois at Urbana-Champaign. He is currently a professor in the Computer Engineering Department, Bilkent University, Ankara, Turkey. His research interests include multimedia database and video surveillance systems, wireless data access, data management for mobile systems, web query languages and data models, and real-time and active database systems. He coedited a special issue on real-time databases in *Information Systems Journal* and a special issue on current trends in database technology in the *Journal of Database Management*. He also coedited a book on current trends in data management technology. He has published more then 50 articles in archived journals and conference proceedings. He is a member of the IEEE Computer Society, the ACM, and the ACM SIGMOD. He was the program cochair of the International Workshop on Issues and Applications of Database Technology, held in Berlin, Germany, in July 1998.