EURASIP Journal on Applied Signal Processing 2005:14, 2330–2340

Robust Background Subtraction with Foreground Validation for Urban Traffic Video

Sen-Ching S. Cheung

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94550, USA

Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY 40506-0503, USA
Email: cheung@engr.uky.edu

Chandrika Kamath

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94550, USA
Email: kamath2@llnl.gov

Received 15 January 2004; Revised 29 December 2004

Identifying moving objects in a video sequence is a fundamental and critical task in many computer-vision applications. Background subtraction techniques are commonly used to separate foreground moving objects from the background. Most background subtraction techniques assume a single rate of adaptation, which is inadequate for complex scenes such as a traffic intersection where objects are moving at different and varying speeds. In this paper, we propose a foreground validation algorithm that first builds a foreground mask using a slow-adapting Kalman filter, and then validates individual foreground pixels by a simple moving object model built using both the foreground and background statistics as well as the frame difference. Ground-truth experiments with urban traffic sequences show that our proposed algorithm significantly improves upon results using only Kalman filter or frame-differencing, and outperforms other techniques based on mixture of Gaussians, median filter, and approximated median filter.

Keywords and phrases: background subtraction, foreground validation, urban traffic video.

1. INTRODUCTION

Identifying moving objects in a video sequence is a fundamental and critical task in video surveillance, traffic monitoring and analysis, human detection and tracking, and gesture recognition in human-machine interface. A common approach to identifying the moving objects is background subtraction, where each video frame is compared against a reference or background model. Pixels in the current frame that deviate significantly from the background are considered to be moving objects. These "foreground" pixels are further processed for object localization and tracking. Since background subtraction is often the first step in many computer vision applications, it is important that the extracted foreground pixels accurately correspond to the moving objects of interest. Requirements of a good background subtraction algorithm include fast adaptation to changes in environment, robustness in detecting objects moving at different speeds, and low implementation complexity.

At the heart of any background subtraction algorithm is the construction of a statistical model that describes the background state of each pixel. Different algorithms build the background model differently, ranging from a single-state estimate based on median filter [1, 2, 3, 4, 5], Weiner filter [6], or Kalman filter [7, 8, 9, 10, 11, 12], to a full density estimation based on Gaussian mixture models [13, 14, 15, 16, 17, 18] or histograms [19]. All of these algorithms have a fixed design parameter, typically the size of a frame buffer or a recursive update parameter, that determines how adaptive the model is to the change in pixel value. We argue that a fixed parameter is inadequate for scenes such as a traffic intersection where objects move at a variety of speeds. We illustrate this with an example.

The top plot in Figure 1a shows an example of how a pixel changes its value throughout a period of time. It begins at level 50, then moves to and stays at 150 for a long period of time. A fast-adapting background subtraction algorithm may output its background/foreground decision similar to the middle plot in Figure 1a: it declares the pixel to be foreground for a short period of time before absorbing the new value as part of the background state. On the other hand, the output decision from a slow-adapting algorithm, shown in the bottom plot in Figure 1a, stays in foreground for a much longer period.

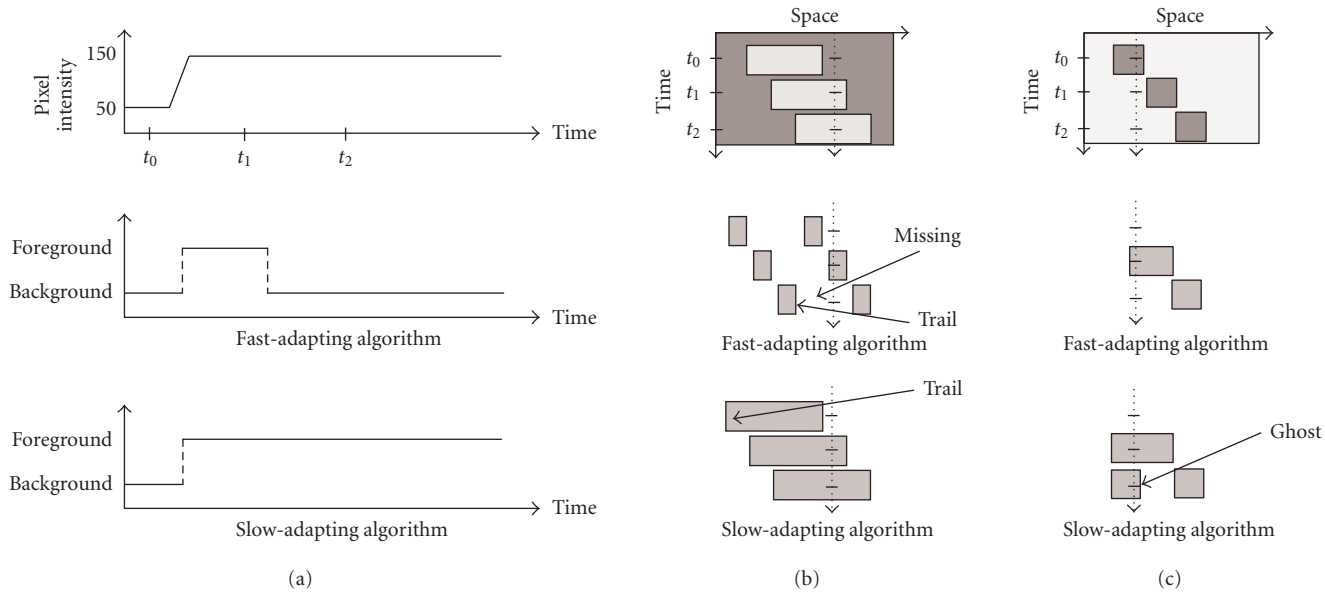


Figure 1: The top plot in (a) shows the changes of a pixel over time. The middle and bottom plots in (a) show the foreground/background decisions of a fast-adapting and a slow-adapting background subtraction algorithm, respectively. (b) and (c) show two possible scenarios that may correspond to the plots in (a) at time-step t_0, t_1, and t_2. The top illustration in (b) shows an object moving to the right. The top illustration in (c) shows another object that is stationary at t_0 but starts to move to the right afterwards, revealing the background behind. The middle illustrations show the foreground masks produced by a fast-adapting algorithm for the two scenarios at the same time-steps, and the bottom illustrations show those of a slow-adapting algorithm. The dotted arrows indicate where the pixel values and foreground/background decisions in (a) are sampled at the three time-steps.

The issue here is that depending on the particular situation, either of these algorithms can be wrong. One scenario is shown in the top illustration in Figure 1b, where an object of gray value 150 is moving slowly to the right over a background of gray value 50. The resulting foreground masks of a fast-adapting and a slow-adapting algorithms are shown in the middle and bottom illustrations. The slow-adapting algorithm clearly produces better results as the fast-adapting algorithm misses most parts of the moving object. The problem of a fast-adapting algorithm failing to detect the motion of a homogeneous object is known as the aperture problem in computer vision [20]. Notice that both algorithms leave a trail of erroneous foreground pixels behind the object as their background models are temporarily corrupted by it. Another possible scenario is shown in Figure 1c, where an object of gray value 50, stationary at first, starts moving and reveals a background of value 150. The fast-adapting algorithm quickly absorbs the newly revealed background value and correctly identifies the moving object. The slow-adapting algorithm, however, leaves a ghost impression of the object long after it is gone. These two scenarios demonstrate that a single fixed rate of adaptation is not sufficient for objects moving at different and varying speeds.

The above example also illustrates that it is impossible to determine the correct rate of adaptation by just considering the causal history of a single pixel. In this paper, we propose a novel algorithm that first builds a foreground mask based on a slow-adapting algorithm, and then validates individual foreground pixels by a simple moving object model built using both the foreground and background statistics as well as a fast-adapting algorithm. Our primary focus is on detecting moving vehicles and pedestrians in urban traffic video taken during day time. This would help us to track the objects in the video, enabling us to build models of normal activity at a scene, and detect anomalous events. These models could include the number of vehicles, the paths followed by the vehicles, their speed, and so forth. This paper is organized as follows: we describe the proposed algorithm in Section 2 and contrast it with related work in Section 3. We apply our proposed algorithm and other background subtraction techniques on a set of urban traffic video sequences. Results of subjective evaluations and objective performance measurements with respect to a ground-truth are presented in Section 4. In Section 5, we conclude the paper by discussing limitations of our algorithm and possible future work.

2. PROPOSED ALGORITHM

This section describes our proposed algorithm for validating a foreground mask computed by a slow-adapting background subtraction algorithm. Figure 2 shows the schematic diagram of our algorithm. The output is a binary foreground mask F_t at time t with $F_t(\mathbf{p}) = 1$ indicating a foreground pixel detected at location $\mathbf{p}$. There are three inputs to the algorithm: (1) I_t is the video frame at time t; (2) P_t is the binary foreground mask from a slow-adapting background subtraction algorithm; (3) D_t denotes the foreground mask obtained

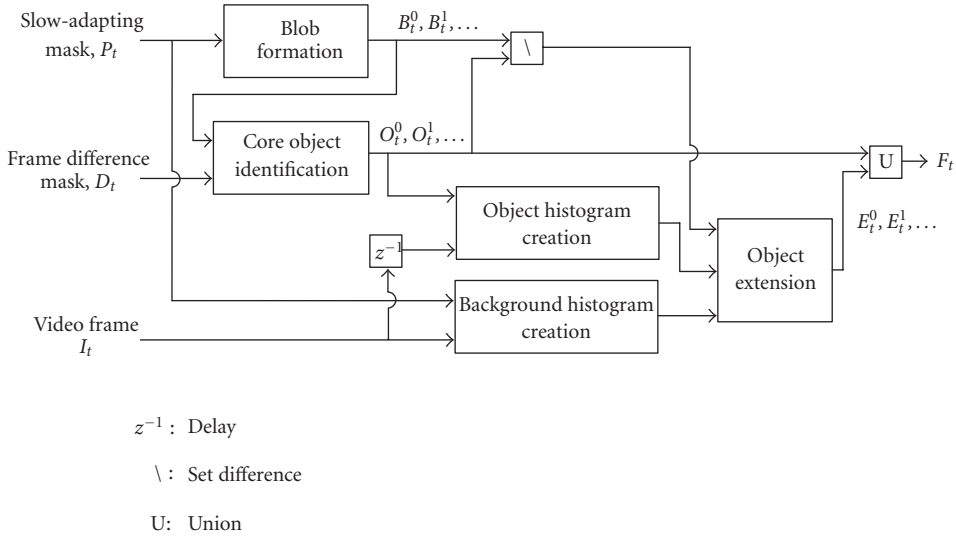


Figure 2: Structure of the data validation module.

by thresholding on the normal statistics of the difference between I_t and I_{t-1}, that is, $D_t(\mathbf{p}) = 1$ if

$$\frac{\left| I_t(\mathbf{p}) - I_{t-1}(\mathbf{p}) - \mu_d \right|}{\sigma_d} > T_d, \tag{1}$$

and zero otherwise. μ_d and σ_d are the mean and the standard deviation of $I_t(\mathbf{q}) - I_{t-1}(\mathbf{q})$ for all spatial locations $\mathbf{q}$. Frame-differencing is the ultimate fast-adapting background subtraction algorithm. Even though it suffers from severe aperture problem, we choose frame-differencing because it leaves only a short foreground trail behind a moving object as its memory does not extend beyond the previous frame. There are five key components in our algorithm: blob formation, core object identification, background histogram creation, object histogram creation, and object extension. Their functions are explained in the following sections.

2.1. Blob formation

In blob formation, all the foreground pixels in P_t are grouped into disconnected blobs $B_t^0, B_t^1, \ldots, B_t^N$ based on the assumption that each foreground pixel is connected to all of its eight adjacent foreground pixels [20]. A blob may contain (1) no object, (2) part of a moving object, (3) a single moving object with possible foreground trail, and (4) multiple moving objects. The first case corresponds to the foreground ghost as explained in Section 1. The second case is likely the result of the aperture problem. Since P_t is computed by a slow-adapting algorithm, the aperture problem occurs only when an object is starting to move. Most blobs fall into the third case of a single object. The last case of multiple objects occur when multiple vehicles start moving after a traffic light has turned green. We ignore the last case as the large blob is likely to break down into multiple single-object blobs once the traffic disperses. The main goals of our algorithm are (1) to eliminate all the ghost blobs, (2) to maintain the partial-object blobs so that they can grow to contain the full objects, and (3) to produce better localization for single-object blobs by removing any foreground trail. We accomplish these goals by validating each blob with the frame-difference mask D_t in the core object identification module.

2.2. Core object identification

The core object identification module first eliminates all the blobs that do not contain any foreground pixels from D_t. This step removes all the ghost blobs which produce no significant frame differences as there are no moving objects in them. The module then computes a core object O_t^i for each of the remaining blobs B_t^i. O_t^i is defined as follows:

$$O_t^i = \text{bounding ellipse}\{\mathbf{p} : \mathbf{p} \in B_t^i, D_t(\mathbf{p}) = 1\} \cap B_t^i. \tag{2}$$

We illustrate our definition of O_t^i using a single moving object as shown in Figure 3a. The blob contains both the object and its foreground trail. The frame-difference mask D_t captures the front part of the object and the small area trailing the object, but completely ignores the rest of the foreground trail of the blob. Taking advantage of the shape of a typical vehicle, we assume that the object is contained within the bounding ellipse of all the foreground pixels from D_t inside the blob. The key idea is that we can use the bounding ellipse to exclude most of the foreground trail from the blob. The bounding ellipse is computed by first calculating its two foci and orientation based on the first- and second-order moments of the foreground pixels in D_t [21], and then increasing the length of its major axis until it contains all the foreground pixels. Finally, we output the intersection between the bounding ellipse and the blob shown in Figure 3b as the core object O_t^i.

2.3. Background histogram creation

Our experience with urban traffic sequences indicates that most moving objects can be adequately represented by their

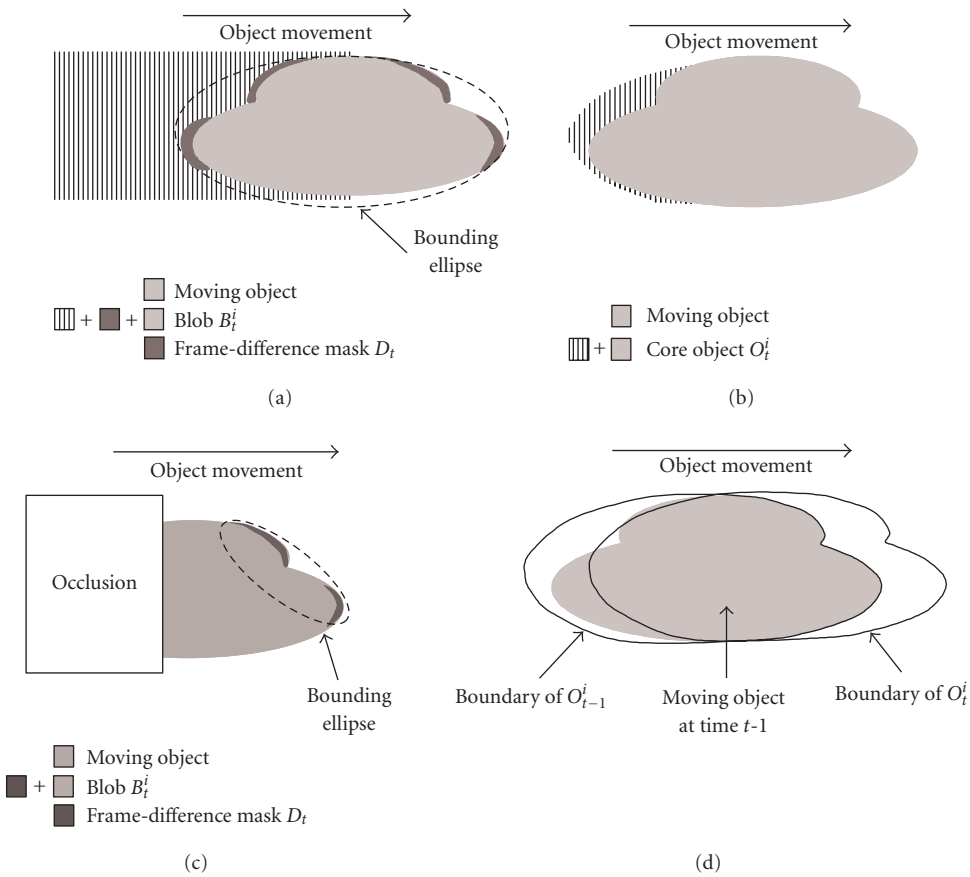


Figure 3: (a) The bounding ellipse is defined by the frame-difference foreground pixels within the blob; (b) the intersection of the bounding ellipse and the blob identifies the core object O_t^i; (c) the bounding ellipse may fail to include the full object emerging from occlusion; (d) the intersection of the core objects at time $t-1$ and t is used for estimating the object histogram.

corresponding core objects. Nevertheless, there are situations where the core object captures only a small portion of the entire moving object. Consider the example of a vehicle emerging from an occlusion as shown in Figure 3c. Even though the blob may contain the entire moving object, the foreground pixels from D_t are present only in the front, resulting in a small core object O_t^i covering that part of the object. The object extension module in Figure 2 is responsible for thrusting back some of the blob pixels outside O_t^i back into foreground. For every pixel $\mathbf{p}$ in B_t^i outside O_t^i, the object extension module declares $\mathbf{p}$ to be foreground if $\mathbf{p}$ is more likely to be part of the core object O_t^i than part of the background. This process is explained in more detail in Section 2.4. The module thus needs to estimate the probability density functions (PDF) of both the background and the object.

To estimate the background PDF, we first partition the current video frame into M rectangular regions $R^1, R^2, \ldots, R^M$. Then, for each region R^j, we compute a background histogram h_t^j for all the pixels in R^j that are not part of P_t, that is,

$$h_t^j(s) = \frac{\left|\left\{\mathbf{p} : \mathbf{p} \in R^j,\ P_t(\mathbf{p}) = 0,\ I_t(\mathbf{p}) = s\right\}\right|}{\left|\left\{\mathbf{p} : \mathbf{p} \in R^j,\ P_t(\mathbf{p}) = 0\right\}\right|} \quad (3)$$

for $j = 1, 2, \ldots, M$. In our implementation, we partition I_t into $M = 64$ identical rectangular regions.

2.4. Object histogram creation and object extension

To build the object histogram, we notice from Figure 3b that the core object O_t^i, as defined in (2), may contain pixels that are not part of the object. It is shown in [6] that the only pixels guaranteed to be part of the object are pixels from I_{t-1} that are foreground in both D_t and D_{t-1}. Based on our experience, this approach does not always produce sufficient number of pixels to reliably estimate the object histogram. Instead, for each core object O_t^i, we first identify the corresponding core object at time $t-1$, which we denote as O_{t-1}^i. We accomplish this by finding the core object at time $t-1$ that has the biggest overlap with O_t^i. Then, we compute the intersection between O_t^i and O_{t-1}^i and build the histogram of the pixels from I_{t-1} under this intersection. This procedure is illustrated in Figure 3d. The object histogram g_t^i for core object O_t^i can now be defined as follows:

$$g_t^i(s) = \frac{\left|\left\{\mathbf{p} : \mathbf{p} \in O_t^i \cap O_{t-1}^i,\ I_{t-1}(\mathbf{p}) = s\right\}\right|}{\left|\left\{\mathbf{p} : \mathbf{p} \in O_t^i \cap O_{t-1}^i\right\}\right|}. \quad (4)$$

Combining the histograms from (3) and (4), the object extension module defines the object extension E_t^i as those pixels within B_t^i but outside O_t^i that are more likely to be part of the core object than of the background:

$$E_t^i = \{\mathbf{p} : \mathbf{p} \in B_t^i \setminus O_t^i,\ g_t^j(I_t(\mathbf{p})) \geq h_t^j(I_t(\mathbf{p})) \text{ with } \mathbf{p} \in R^j\}. \tag{5}$$

The final output mask F_t is simply the union of all the core objects O_t^i and the object extensions E_t^i.

3. RELATED WORK

In this section, we review related work in background modeling and foreground validation. We first summarize some of the representative schemes for background modeling. A more detailed exposition can be found in [22]. We classify background modeling algorithms into nonrecursive and recursive techniques. A nonrecursive technique maintains a buffer of video frames and uses a sliding-window approach for background estimation. In order to keep a long history with low storage requirement, video frames can be stored into the buffer at a frame rate r lower than the input rate. The most commonly used nonrecursive technique is median filtering [1, 2, 3, 4, 5]. The background estimate is defined to be the median at each pixel location of all the frames in the buffer. Wiener filter is used in [6], where the filter coefficients are estimated at each frame time based on the sample covariances. Unlike median filter or Wiener filter which produce a single background estimate, Elgammal et al. [19] build a background PDF using a Gaussian kernel estimator. The advantage of using a full density function over a single estimate is the ability to handle multi modal background distribution. Examples of multi modal background include pixels from moving leaves of a tree or pixels near high-contrast edges which flicker under small camera movement.

Recursive techniques recursively update a single background model and do not store a buffer of video frames. The two simplest recursive techniques are approximated median filter [23, 24] and Kalman filter [7, 8, 9, 10, 11, 12]. Approximated median filter increments a running estimate of the median by one if the input pixel is larger than the estimate, and decrements by one if the opposite is true. Kalman filter is a widely used recursive technique for tracking linear dynamical systems under Gaussian noise. Many different versions have been proposed for background modeling, differing mainly in the state spaces they use for tracking. We provide a brief description of the popular scheme used in [7]: the internal state at pixel location $\mathbf{p}$ of the Kalman filter is described by the background intensity $S_t(\mathbf{p})$ and its temporal derivative $S'_t(\mathbf{p})$, which are recursively updated as follows:

$$\begin{bmatrix} S_t(\mathbf{p}) \\ S'_t(\mathbf{p}) \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} S_{t-1}(\mathbf{p}) \\ S'_{t-1}(\mathbf{p}) \end{bmatrix} + \mathbf{K} \cdot \left(I_t(\mathbf{p}) - \mathbf{H} \cdot \mathbf{A} \cdot \begin{bmatrix} S_{t-1}(\mathbf{p}) \\ S'_{t-1}(\mathbf{p}) \end{bmatrix} \right). \tag{6}$$

Matrix $\mathbf{A}$ describes the background dynamics and $\mathbf{H}$ is the measurement matrix. The Kalman gain matrix $\mathbf{K}$ switches between a slow adaptation rate α_1 and a fast adaptation rate $\alpha_2 > \alpha_1$ based on the feedback of the foreground mask F_{t-1}:

$$\mathbf{K} = \begin{bmatrix} \alpha_1 \\ \alpha_1 \end{bmatrix} \quad \text{if } F_{t-1}(\mathbf{p}) = 1, \qquad \begin{bmatrix} \alpha_2 \\ \alpha_2 \end{bmatrix} \quad \text{otherwise.} \tag{7}$$

Another popular recursive technique is the mixture of Gaussian (MoG), which tracks the background distribution as a linear sum of K Gaussian component densities [13, 14, 15, 16, 17, 18]. If the new input pixel $I_t(\mathbf{p})$ is close to one of the Gaussian components, the mean and the standard deviation of that component is updated. Otherwise, the least probable one is deleted and a new component is added, centered at $I_t(\mathbf{p})$ [14]. The weights of all the components are decayed at a rate of $(1-\alpha)$, except for the weight of the updated component which is incremented by α. To determine which components correspond to the background, all components are first ranked by the ratios between their weights and standard deviations. Then, the first N components that satisfy the following criterion are declared to be the background components:

$$\sum_{k=i_1}^{i_N} \omega_{k,t} \geq \Gamma, \tag{8}$$

where $\omega_{k,i_1}, \ldots, \omega_{k,i_M}$ are the weights of the components after ranking, and $\Gamma > 0$ is the weight threshold. I_t is declared as background if it is within D times the standard deviation from the mean of any one of the background components.

It should be noted that any one of the above-mentioned background modeling algorithms can be used to generate the slow-adapting mask in our proposed algorithm. As described in Section 4, we use Kalman filter in our system primarily because of its simplicity.

Many foreground validation techniques have also been proposed in the literature. Some of them incorporate knowledge from the high-level applications such as tracking [9, 25], or use extra information such as depth [25, 26] to improve the background model. Optical flow is also commonly used to detect ghost blobs [5, 27]. In contrast to these approaches, our algorithm does not rely on any external information and uses the much simpler frame-differencing for ghost-blob removal. Algorithms have also been proposed to combine multiple background models running at different adaptation rates [12, 19]. However, the combination is done by a simple conjunction at the pixel level, rather than at the blob level as in our algorithm. Pixel-level combination might lead to the aperture problem as the fast-adapting algorithm can fail to detect slow-moving objects. Blob-level processing has been proposed in [6] for foreground validation. Similar to our algorithm, [6] builds core objects based on the intersection of a slow-adapting mask and the frame-difference mask, and grows the core objects using object histograms and connected component grouping. Nevertheless, lackluster results are reported in [6] because, by growing the core objects from a waving tree, the blob-level processing turns part of the sky into foreground. There are three key differences between our proposed algorithm and that proposed in

Table 1: Background modeling schemes and their parameters.

Schemes	Fixed parameters	Test parameter
Frame-differencing (FD)	None	Foreground threshold T_d
Approximated median filter (AMF)	None	Foreground threshold T
Kalman filter (KF)	$\alpha_1 = 0.001$, $\alpha_2 = 0.05$	Foreground threshold T_k
Median filter (MF)	Buffer size $L = 9$ Buffer sampling rate $r = 10$ frame/s	Foreground threshold T
Mixture of Gaussian (MoG)	Number of components $K = 3$ Adaptation rate $\alpha = 0.05$ Weight threshold $\Gamma = 0.25$ Initial variance $\sigma_o^2 = 36$ Initial weight $\omega_o = 0.1$	Deviation threshold D

[6]. First, in our proposed algorithm, the object growing is confined within the slow-adapting mask, thus limiting the amount of false-positives the algorithm can introduce. Second, we use both the object histogram and the background histogram to achieve reliable object growing. Finally, using a bounding ellipse as a first-order approximation to the object leads to a significant speedup as it already accounts for most of the foreground pixels.

To summarize, the main advantages of our validation algorithm over others in the literature are

(1) our algorithm has low complexity as it does not rely on information from other sensory sources or from sophisticated computer-vision algorithms. It also minimizes the complex blob processing steps by making assumptions about the general shape of the foreground objects,

(2) our algorithm achieves good object localization by utilizing both foreground and background statistics in combining the slow-adapting blobs with the frame-difference foreground masks.

4. EXPERIMENTAL RESULTS

In this section, we compare the performance of our proposed algorithm with other algorithms in the literature. We apply the background subtraction algorithms to luminance sequences only. For preprocessing, we first apply a three-frame temporal erosion to the test sequence, that is, we replace I_t with the minimum of I_{t-1}, I_t, and I_{t+1}. This step can reduce temporal camera noise and mitigate the effect of snowfall present in one of our test sequences. Such an erosion step is also effective in removing rainfall because raindrops tend to be much brighter than the surrounding background [28]. On the other hand, this step does not affect the performance of moving object extraction under normal weather condition because typical object movements are much slower and last much longer than the short transient distortion introduced by the erosion. Afterwards, a 3×3 spatial Gaussian filter is used to reduce spatial camera noise.

Even though our proposed algorithm can work with any slow-adapting background subtraction algorithm, we have chosen Kalman filter as described in Section 3 for its simplicity in implementation. To propagate the validation results to future frames, we use the output mask F_t from the foreground validation for feedback in the Kalman filter. The parameters of the Kalman filter are set as follows [7]:

$$\mathbf{A} = \begin{bmatrix} 1 & 0.7 \\ 0 & 0.7 \end{bmatrix}, \qquad \mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad \alpha_1 = 0.001, \qquad \alpha_2 = 0.05. \tag{9}$$

Similar to the frame-difference thresholding in (1), $P_t(\mathbf{p}) = 1$ if

$$\frac{\left| I_t(\mathbf{p}) - S_t(\mathbf{p}) - \mu_k \right|}{\sigma_k} > T_k, \tag{10}$$

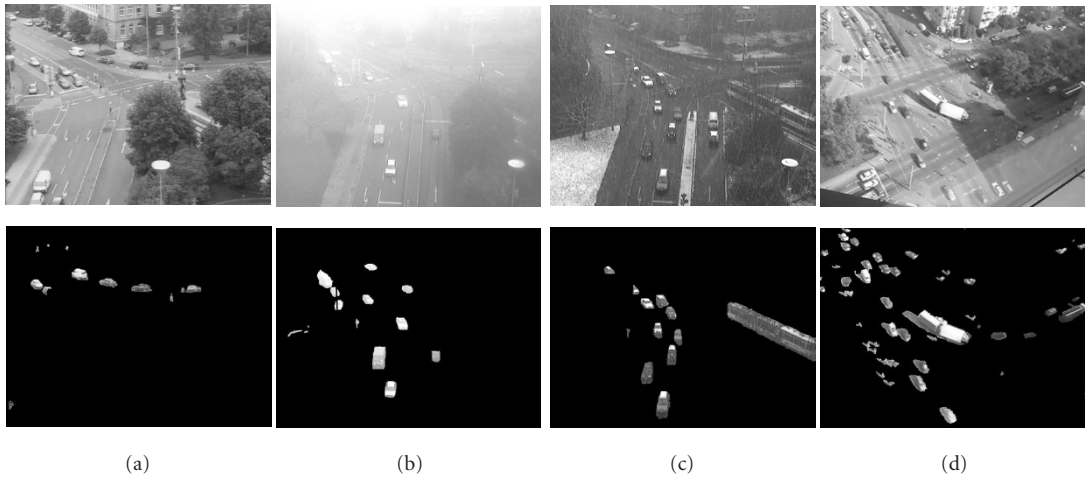
where $S_t(\mathbf{p})$ is the internal state of the Kalman filter, and μ_k and σ_k are the mean and the standard deviation of $I_t(\mathbf{q}) - S_t(\mathbf{q})$ for all spatial locations $\mathbf{q}$ in the frame. In our experiments, we set the frame-difference foreground threshold T_d to be 2, and vary the Kalman filter foreground threshold T_k to show the trade-off between false-positives and false-negatives in foreground detection.

The set of algorithms used for comparison includes frame-differencing (FD), approximated median filter (AMF), Kalman filter without validation (KF), median filter (MF), and mixture of Gaussian (MoG). Based on our earlier work in [22], we have selected particular values for the parameters in these algorithms that perform well in our test sequences. These fixed parameters are listed in Table 1. The most sensitive parameter in each algorithm is used as the test parameter to show the trade-off between false-positives and false-negatives.

4.1. Test sequences

We have selected four publicly available urban traffic video sequences from the website maintained by KOGS/IAKS, Universitaet Karlsruhe.[1] A sample frame from each sequence is shown in the first row of Figure 4. The first sequence is called

[1]The URL is http://i21www.ira.uka.de/image_sequences. All sequences are copyrighted by H.-H. Nagel of KOGS/IAKS Universitaet Karlsruhe.



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

Figure 4: Sample frames and the corresponding ground-truth frames from the four test sequences: (a) Bright, (b) Fog, (c) Snow, and (d) Busy.

"Bright," which is 1500 frames long showing a traffic intersection in bright daylight. The second sequence is called "Fog," which is 300 frames long showing the same traffic intersection in heavy fog. The third sequence "Snow" is also 300 frames long and shows the intersection while snowing. Fog and Snow were originally in color; we have first converted them into luminance and discarded the chroma channels. The first three sequences all have low to moderate traffic, and contain "stop-and-go" traffic—vehicles come to a stop in front of a red light and start moving once the light turns green. The last sequence "Busy" is 300 frames long. It shows a busy intersection with the majority of the vehicle traffic flowing from the top left corner to the right side.

4.2. Evaluation

In order to have a quantitative evaluation of the performance, we have selected ten frames at regular intervals from each test sequence, and manually highlighted all the moving objects in them. These "ground-truth" frames are selected from the latter part of each of the test sequences[2] to minimize the effect of the initial adaptation of the algorithms. This sampling rate allows the vehicles to move a reasonable distance, making each ground-truth frame sufficiently different from others. In the manual annotation, we highlight only the pixels belonging to vehicles and pedestrians that are actually moving at that frame. Since we do not use any shadow suppression scheme in our comparison, we also include the shadow pixels cast by moving objects. The ground-truth frames showing only the moving objects are shown in the second row of Figure 4.

We use two information retrieval measurements, recall and precision, to quantify how well each algorithm matches the ground-truth [29]. They are defined in our context as follows:

$$\begin{aligned} \text{Recall} &= \frac{\text{Number of foreground pixels correctly identified by the algorithm}}{\text{Number of foreground pixels in ground-truth}}, \\ \text{Precision} &= \frac{\text{Number of foreground pixels correctly identified by the algorithm}}{\text{Number of foreground pixels detected by the algorithm}}. \end{aligned} \tag{11}$$

Recall and precision values are both within the range of 0 and 1. When applied to the entire sequence, the recall and precision reported are averages over all the measured frames. Typically, there is a trade-off between recall and precision—recall usually increases with the number of foreground pixels detected, which in turn may lead to a decrease in precision. A good background algorithm should attain as high a recall value as possible without sacrificing precision.

By varying the testing parameter of each algorithm, we obtain the precision-recall (PR) curves for the test sequences as shown in Figures 5a, 5b, 5c, and 5d. Notice that the PR curves of all the nonrecursive techniques (FD, AMF, MF)

[2] The ground-truth frames are selected from the last 1000 frames in the Bright sequence, and the last 200 frames in the remaining three sequences.

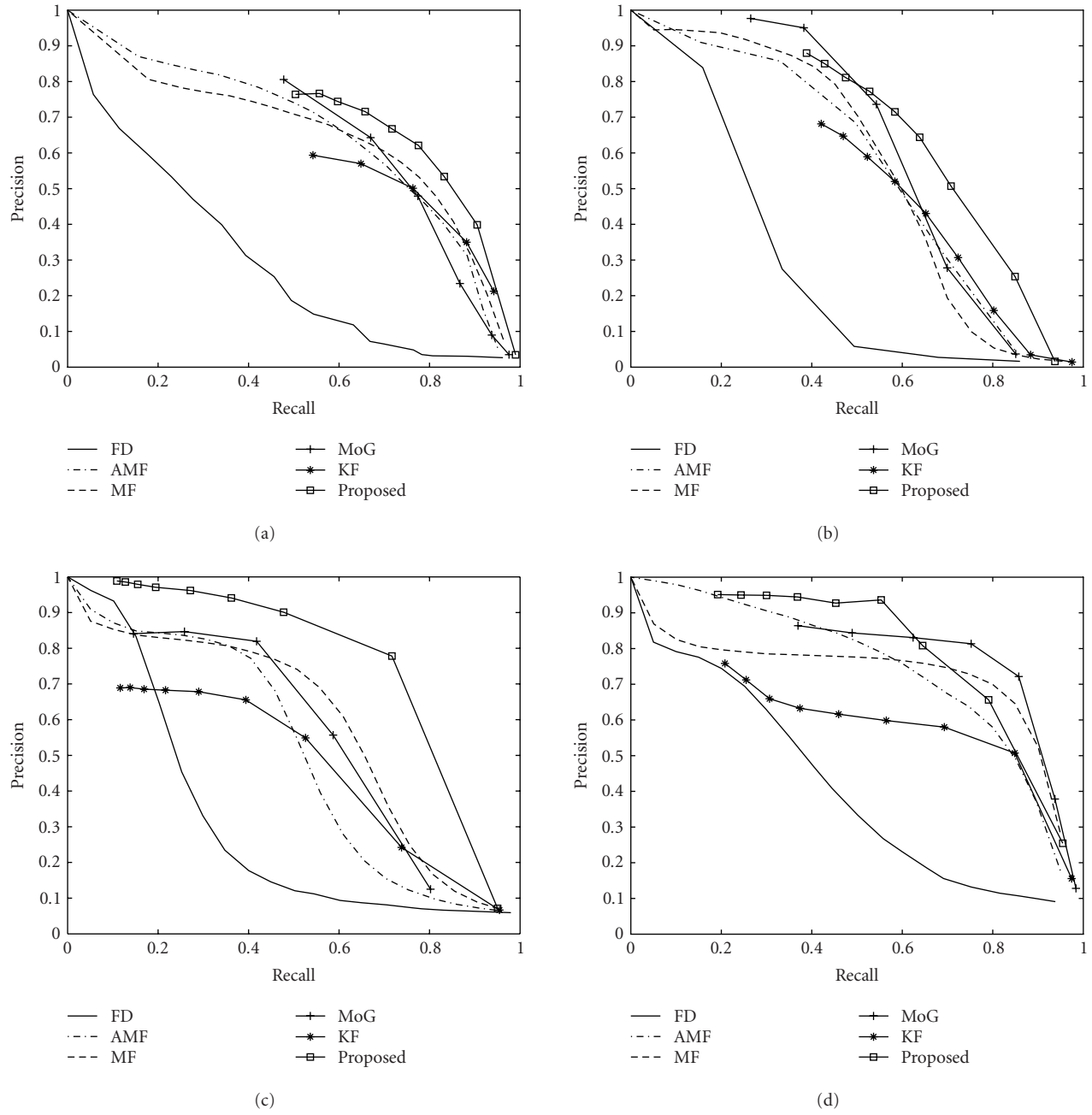
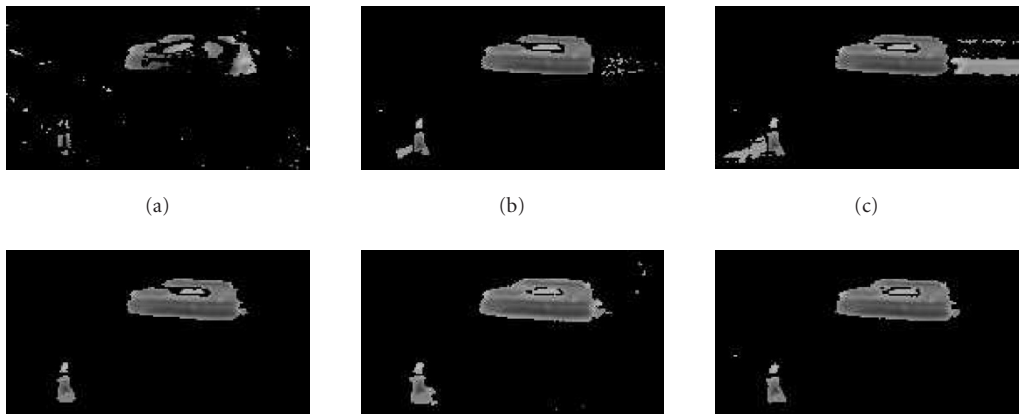


Figure 5: Precision-recall plots for (a) Bright, (b) Fog, (c) Snow, and (d) Busy.

are almost continuous over the entire range of recall values, while those from the recursive ones (MoG, KF, proposed) are discrete, occupying shorter ranges of recall. The reason is that nonrecursive techniques do not have a feedback loop so that it is possible to run the simulation once and compute the precision and the recall for any test parameter value. Recursive techniques require a separate simulation for each different test parameter value, and thus we only obtain results at a few operating points.

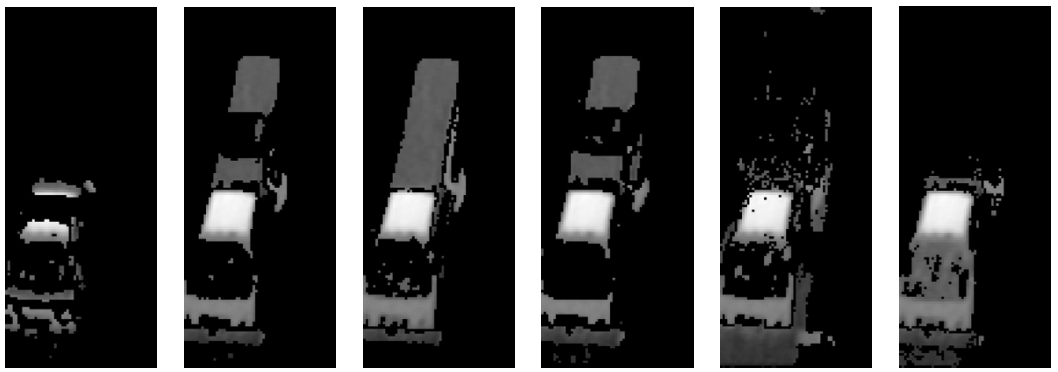
Figure 5a shows the results of the sequence Bright. The worst performer is FD and the best performer, at least for recall above 60%, is our proposed algorithm. Our proposed algorithm shares a similar shape with KF but has a much better precision. MoG outperforms the proposed algorithm at low recall primarily because MoG can adapt its threshold for each pixel individually, while our proposed algorithm relies on the global standard deviation. As some of the ground-truth frames have only a few moving objects, the global standard deviation becomes quite small. This turns some of the background pixels into foreground erroneously and thus lowers the precision values. The PR curves of Fog in Figure 5b follow a similar trend as those in Figure 5a.



(a) (b) (c)

(d) (e) (f)

Figure 6: Foreground images identified by various algorithms showing a moving car and a pedestrian in sequence Bright. (a) FD, (b) AMF, (c) KF, (d) MF, (e)MoG, and (f) proposed.



(a) (b) (c) (d) (e) (f)

Figure 7: Foreground images identified by various algorithms showing a car that starts to move after being stationary for awhile in sequence Snow. (a) FD, (b) AMF, (c) KF, (d) MF, (e)MoG, and (f) proposed.

Figure 5c shows that our proposed algorithm significantly outperforms all the other schemes in the sequence Snow. This can be explained by the two thresholds T_d and T_k used in our proposed algorithm. The Snow sequence is very noisy and all other algorithms need high foreground thresholds to prevent excessive false foreground, thus resulting in low recall values. On the other hand, our proposed algorithm can use a small threshold T_k in the Kalman filter to get good coverage of the moving objects, but uses a large frame-difference threshold T_d for building the correct shapes of the objects and removing the ghost foreground blobs.

In the final sequence Busy, our proposed algorithm performs worse than MoG and MF for recall values above 60%. This reversal of performance is due to the large number of moving objects clustered together in Busy. At high recall values, the foreground threshold of the Kalman filter is small, creating a large blob that contains many moving objects. The resulting bounding ellipse is not able to eliminate any of the false foreground between vehicles, leading to a low precision value.

We further illustrate the differences between the algorithms using two sets of extracted foreground images in Figures 6 and 7. The corresponding original images are shown in Figure 8. Figure 6 is extracted from Bright showing a car moving to the left and a pedestrian walking to the right. FD is noisy and captures only fragments of the car due to the aperture problem. Both AMF and KF leave a foreground trail behind each moving object as their background states are corrupted. MF, MoG, and the proposed algorithm produce similar results. Figure 7 is extracted from Snow showing a car starting to move to the bottom after being stationary for a long time. AMF, KF, and MF leave a ghost foreground behind the moving car. MoG is less problematic but still has much noise. FD and the proposed algorithm have no ghost at all. Only the proposed algorithm produces an almost perfect localization of the car. Notice that even the low-contrast windshield is captured due to the two thresholds used in the proposed algorithm as explained earlier.

(a) (b)

Figure 8: (a) and (b) show the original images corresponding to the foreground masks in Figures 6 and 7.

5. CONCLUSIONS

In this paper, we have introduced a new algorithm to validate foreground regions or blobs captured by a slow-adapting background subtraction algorithm. By comparing the blobs with bounding ellipses formed by frame-difference foreground pixels, the algorithm can eliminate false foreground trails and ghost blobs that do not contain any moving object. Better object localization under occlusion is accomplished by extending the ellipses using the object and background pixel distributions. Ground-truth experiments with urban traffic sequences have shown that our proposed algorithm produces performances that are comparable or better than other background subtraction techniques.

Our proposed algorithm, however, has a number of limitations. First, the use of bounding ellipses may not be appropriate for complex-shaped objects such as human beings. Second, we use frame-differencing as it produces minimal false foreground trails behind objects. This assumption may not hold if the input video frame rate is very low. Third, our algorithm relies on the object and background pixel distributions to determine the shape of the foreground objects. If the distributions are not estimated correctly due to lack of samples, or the objects and the foreground have similar pixel distributions, the results will be adversely affected. This problem may be mitigated by extending the time window to incorporate more data as well as using features other than pure pixel intensities. Fourth, in Section 4, we have tested a large range of threshold values and identified the appropriate operating points for our algorithm. In a real system, the thresholds must be automatically adjusted based on the environment. We are currently investigating techniques to adjust thresholds automatically by validating the output with an a priori statistical model of foreground objects. Finally, as described in Section 4.2, the proposed algorithm does not perform well when there are multiple moving objects close to each other. We are currently improving our algorithm by building multiple ellipses to identify moving objects and using the background distribution to identify the background area among them. We will include additional sequences taken under different conditions, as well as additional ground-truth frames, to test how well these enhancements perform in real traffic sequences.

ACKNOWLEDGMENTS

We would like to thank the two anonymous reviewers for their valuable comments which helped us tremendously in improving the manuscript. UCRL-JRNL-201916: this work was performed under the auspices of the U S Department of Energy by the University of California Lawrence Livermore National Laboratory under Contract no. W-7405-Eng-48.

REFERENCES

[1] B. Gloyer, H. Aghajan, K.-Y. Siu, and T. Kailath, "Video-based freeway monitoring system using recursive vehicle tracking," in *Image and Video Processing III*, vol. 2421 of *Proceedings of SPIE*, pp. 173–180, San Jose, Calif, USA, February 1995.

[2] R. Cutler and L. Davis, "View-based detection and analysis of periodic motion," in *Proc. 14th International Conference on Pattern Recognition (ICPR '98)*, vol. 1, pp. 495–500, Brisbane, Australia, August 1998.

[3] B. Lo and S. Velastin, "Automatic congestion detection system for underground platforms," in *Proc. International Symposium on Intelligent Multimedia, Video, and Speech Processing*, pp. 158–161, Kowloon Shangri-La, Hong Kong, May 2001.

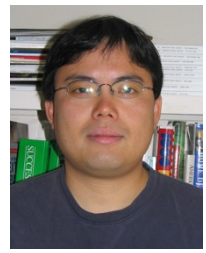
[4] Q. Zhou and J. Aggarwal, "Tracking and classifying moving objects from videos," in *Proc. 2nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance (PETS '01)*, Kauai, Hawaii, USA, December 2001.

[5] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 10, pp. 1337–1342, 2003.

[6] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance," in *Proc. 7th IEEE International Conference on Computer Vision (ICCV '99)*, vol. 1, pp. 255–261, Kerkyra, Greece, September 1999.

[7] K.-P. Karmann and A. Brandt, "Moving object recognition using and adaptive background memory," in *Time-Varying Image Processing and Moving Object Recognition*, V. Cappellini, Ed., vol. 2, pp. 289–307, Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1990.

[8] D. Koller, J. Weber, and J. Malik, "Robust multiple car tracking with occlusion reasoning," Tech. Rep. UCB/CSD-93-780, EECS Department, University of California, Berkeley, Calif, USA, October 1993.
[9] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 780–785, 1997.
[10] J. Heikkila and O. Silven, "A real-time system for monitoring of cyclists and pedestrians," in *Proc. 2nd IEEE Workshop on Visual Surveillance (VS '99)*, pp. 74–81, Fort Collins, Colo, USA, June 1999.
[11] G. Halevy and D. Weinshall, "Motion of disturbances: detection and tracking of multi-body non-rigid motion," *Machine Vision and Applications*, vol. 11, no. 3, pp. 122–137, 1999.
[12] T. Boult, R. Micheals, X. Gao, et al., "Frame-rate omnidirectional surveillance and tracking of camuflaged and occluded targets," in *Proc. 2nd IEEE Workshop on Visual Surveillance (VS '99)*, pp. 48–55, Fort Collins, Colo, USA, June 1999.
[13] N. Friedman and S. Russell, "Image segmentation in video sequences: a probabilistic approach," in *Proc. 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI '97)*, pp. 175–181, Morgan Kaufmann Publishers, San Francisco, Calif, USA, August 1997.
[14] C. Stauffer and W. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 747–757, 2000.
[15] X. Gao, T. Boult, F. Coetzee, and V. Ramesh, "Error analysis of background adaption," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, vol. 1, pp. 503–510, Hilton Head Island, SC, USA, June 2000.
[16] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Proc. 2nd European Workshop on Advanced Video-Based Surveillance Systems (AVBS '01)*, Kingston, UK, September 2001.
[17] P. W. Power and J. A. Schoonees, "Understanding background mixture models for foreground segmentation," in *Proc. Image and Vision Computing New Zealand*, pp. 267–271, Auckland, New Zealand, November 2002.
[18] D.-S. Lee, J. Hull, and B. Erol, "A Bayesian framework for Gaussian mixture background modeling," in *Proc. IEEE International Confererence on Image Processing (ICIP '03)*, vol. 3, pp. 973–976, Barcelona, Spain, September 2003.
[19] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Proc. IEEE ICCV '99 Frame-rate workshop*, Corfu, Greece, September 1999.
[20] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1982.
[21] R. Mukundan and K. R. Ramakrishnam, *Moment Functions in Image Analysis*, World Scientific, Singapore, Singapore, 1998.
[22] S.-C. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Proc. Video Communications and Image Processing, SPIE Electronic Imaging*, San Jose, Calif, USA, January 2004.
[23] N. McFarlane and C. Schofield, "Segmentation and tracking of piglets in images," *Machine Vision and Applications*, vol. 8, no. 3, pp. 187–193, 1995.
[24] P. Remagnino, A. Baumberg, T. Grove, et al., "An integrated traffic and pedestrian model-based vision system," in *Proc. 8th British Machine Vision Conference*, pp. 380–389, Essex, UK, September 1997.
[25] M. Harville, "A framework for high-level feedback to adaptive, per-pixel, mixture-of-Gaussian background models," in *Proc. 7th European Conference on Computer Vision (ECCV '02), Part III*, pp. 543–560, Copenhagen, Denmark, May 2002.
[26] Y. Ivanov, A. Bobick, and J. Liu, "Fast lighting independent background," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 199–207, 2000.
[27] D. Gutchess, M. Trajkovic, E. Cohen-Solal, D. Lyons, and A. K. Jain, "A background model initialization algorithm for video surveillance," in *Proc. 8th IEEE International Conference on Computer Vision (ICCV '01)*, vol. 1, pp. 733–740, Vancouver, British Columbia, Canada, July 2001.
[28] K. Garg and S. Nayar, "Detection and removal of rain from video," in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR '04)*, vol. 1, pp. 528–535, Washington, DC, USA, June 2004.
[29] C. J. van Rijsbergen, *Information Retrieval*, Butterworth, London, UK, 2nd edition, 1979.



Sen-Ching S. Cheung is an Assistant Professor at the Department of Electrical and Computer Engineering, University of Kentucky. He also holds a joint appointment at the Center of Visualization and Virtual Environments, where he and his research group are working on various signal processing, machine learning, and data mining problems in large-scale multimedia systems. Before joining the University of Kentucky, he spent almost two years at Lawrence Livermore National Laboratory working with other data miners on finding abnormal patterns in simulation and surveillance data. He received his Ph.D. degree 2002 from the University of California, Berkeley, based on his work on video summarization, indexing, and clustering. Between 1995 and 1998, he was a Researcher at Compression Labs Inc., San Jose, California, where he developed algorithms for teleconferencing and satellite broadcast systems. During the same period, he was also an active participant in the ITU-T H.263 (version 2) and MPEG-4 standardization activities.

Chandrika Kamath is a Computer Scientist at the Center for Applied Scientific Computing, Lawrence Livermore National Laboratory (LLNL), where she leads the Sapphire project in scientific data mining. Her research interests include image processing, pattern recognition, and practical applications of data mining. She received her Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign in 1986. Prior to joining LLNL, she worked at Digital Equipment Corporation, developing mathematical software for high-performance computers. She is a Member of ACM, IEEE, SIAM, and SPIE.