A Background Model Initialization Algorithm for Video Surveillance

D. Gutchess†, M. Trajković‡, E. Cohen-Solal‡, D. Lyons‡, A. K. Jain†

†Computer Science and Engineering
Michigan State University
East Lansing, MI 48843

‡Adaptive Systems
Philips Research
Briarcliff Manor, NY 10510

Abstract

Many motion detection and tracking algorithms rely on the process of background subtraction, a technique which detects changes from a model of the background scene. We present a new algorithm for the purpose of background model initialization. The algorithm takes as input a video sequence in which moving objects are present, and outputs a statistical background model describing the static parts of the scene. Multiple hypotheses of the background value at each pixel are generated by locating periods of stable intensity in the sequence. The likelihood of each hypothesis is then evaluated using optical flow information from the neighborhood around the pixel, and the most likely hypothesis is chosen to represent the background. Our results are compared with those of several standard background modeling techniques using surveillance video of humans in indoor environments.

1. Introduction

Automated video surveillance has emerged as an important research topic in the computer vision community. The growth in this area is being driven by the increased availability of inexpensive computing power and image sensors, as well as the inefficiency of manual surveillance and monitoring systems. Applications such as event detection, human action recognition, and semantic indexing of video are being developed in order to partially or fully automate the task of surveillance. To be successful, such applications require real-time motion detection and tracking algorithms, which provide low-level functionality upon which higher level recognition capabilities can be built.

The detection of motion in many current tracking systems relies on the technique of background subtraction. By comparing incoming image frames to a reference image, regions of the image which have changed are efficiently located. The use of background subtraction is popular in real-time tracking applications such as surveillance [1, 2], and also in the video coding community [3, 4], which focuses on problems such as video compression and content-based functionality for multimedia.

Recently, there has been a large amount of work addressing the issues of background model representation and adaptation [5, 6, 7, 8, 9, 10, 11, 12]. However, a third problem which has received little attention is model initialization (also called bootstrapping in [11]). Often the assumption is made that an initial model can be obtained by using a short training sequence in which no foreground objects are present. However, in some situations, e.g., public areas, it is difficult or impossible to control the area being monitored. In such cases it may be necessary to train the model using a sequence which contains foreground objects. This paper presents a new algorithm which is able to learn a model of the background when moving (foreground) objects are present in the scene. The algorithm is applied to the problem of tracking humans in an indoor environment, and its performance is compared with several other background modeling techniques.

In the following subsection, we will define the problem of background initialization, and discuss the assumptions made in our solution. Section 2 provides a summary of previous work relevant to background modeling. The next two sections describe our optical flow-based algorithm, and the results we obtained from experiments. Section 5 concludes by describing some of the important characteristics of the algorithm, and directions for future research.

1.1. Problem Definition

The background initialization problem is defined as follows. The input is a short video sequence in which any number of moving objects may be present. The goal is to output a single background model describing the scene (see Figure 1). Several assumptions are necessary to make the task feasible:

1. Each pixel in the image will reveal the background for

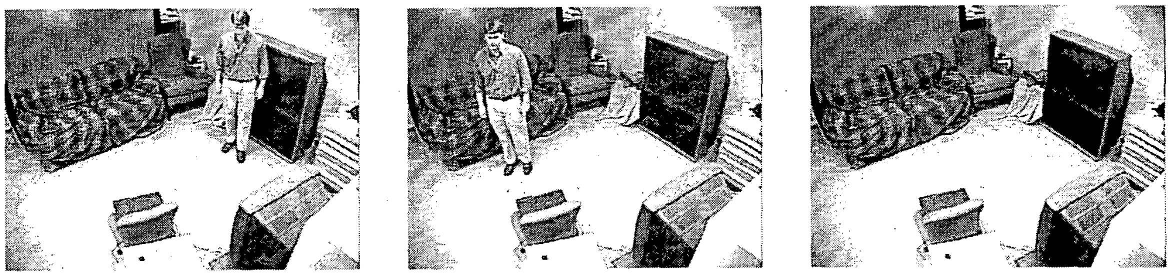

Figure 1. The background initialization problem uses a sequence of frames, (a) and (b), containing foreground objects as input, and outputs a background model (c) representing the stationary parts of the scene.

at least a short interval of the sequence.

2. The background is approximately stationary; only small background motion may occur.

3. A short processing delay is allowed subsequent to acquiring the training sequence.

The first assumption is necessary to avoid randomly choosing background appearance. If an object occludes a certain area for the entire training period, it is impossible to accurately estimate the background pixel values in that area. Allowing parts of the background to be moving would require discrimination between different types of motion. Motion of the foreground objects would have to be distinguishable from that of the background, and task-specific assumptions would be necessary to distinguish them. The recognition of background motion could be added to this algorithm, which is a topic of future research. The final assumption allows batch processing of the sequence, which gives the algorithm certain advantages over sequential processing. Batch algorithms make use of information from the entire sequence and output a single decision, while background updating schemes must perform sequential decision-making using information from past frames only. For most applications, the penalty of a small processing delay at system startup caused by an initialization procedure is outweighed by the resulting performance improvement.

2. Previous Work

Prior work on background modeling has mainly focused on model representation and techniques for adaptation. Table 1 provides a comparison of some popular methods. The ability to represent multiple modes for the background values allows some techniques to model motion which is part of the background [5, 10, 11]. Unimodal representations, which store a single mean per pixel, are more prone to creating false detections in the case of a moving background. The methods may also be grouped into those which estimate background values using temporal smoothing [5, 8, 10, 11, 12], and those which choose a single value from the set of past observations [1, 9, 13].

A popular choice of representation includes a measure of central tendency and a measure of dispersion for each pixel. Alternative background model representations have also been proposed. The W^4 system stored minimum and maximum intensity values, and maximum temporal derivative for each pixel [6]. Elgammal et al. [5] used a histogram to model each pixel and determined when multiple modes in the distribution were present. Using mixture models to represent moving backgrounds was investigated by Grimson et al. [10]. Yang and Levine proposed an edge-based background representation called the background primal sketch [14].

Temporal smoothing is often performed to recursively update the model. For example, Karmann and von Brandt [8] used a Kalman filter, and the Wallflower algorithm [11] used a Wiener filter. Others have used weighted moving averages, such as exponential smoothing [12]. All of these adaptation techniques may exhibit the unwanted side effect of blending pixel values in areas of the image which have recently changed. This blending problem can result in both false positives and false negatives in the foreground detection stage. Gao et al. [15] discuss how the proper blending rate can be computed for given false alarm and miss detection probabilities.

Background initialization using the median intensity value for each pixel was used in a traffic monitoring system [13]. Use of the median relies on an assumption that the background at every pixel will be visible more than fifty

Method	Advantages	Disadvantages
DBS [1]	efficiency, avoids blending	unimodal, slow moving foreground objects become part of background
Kalman filter [8]	adapts quickly	unimodal, blending
Exp. smoothing [12]	efficiency, variable adaptation rate parameter	unimodal, blending
Median [13]	avoids blending, output corresponds to some input value	assumption that background is visible more than 50 percent of the time, unimodal
Wallflower [11]	operates at multiple spatial scales	blending, region filling causes some errors
Gaussian mixture [10]	ability to represent a moving background	blending
Nonparametric model [5]	ability to represent a moving background	memory requirements
Adaptive smoothness [9] (AS)	avoids blending	unimodal, assumption: intensity of background is stable for long period

Table 1. Comparison of several background modeling techniques.

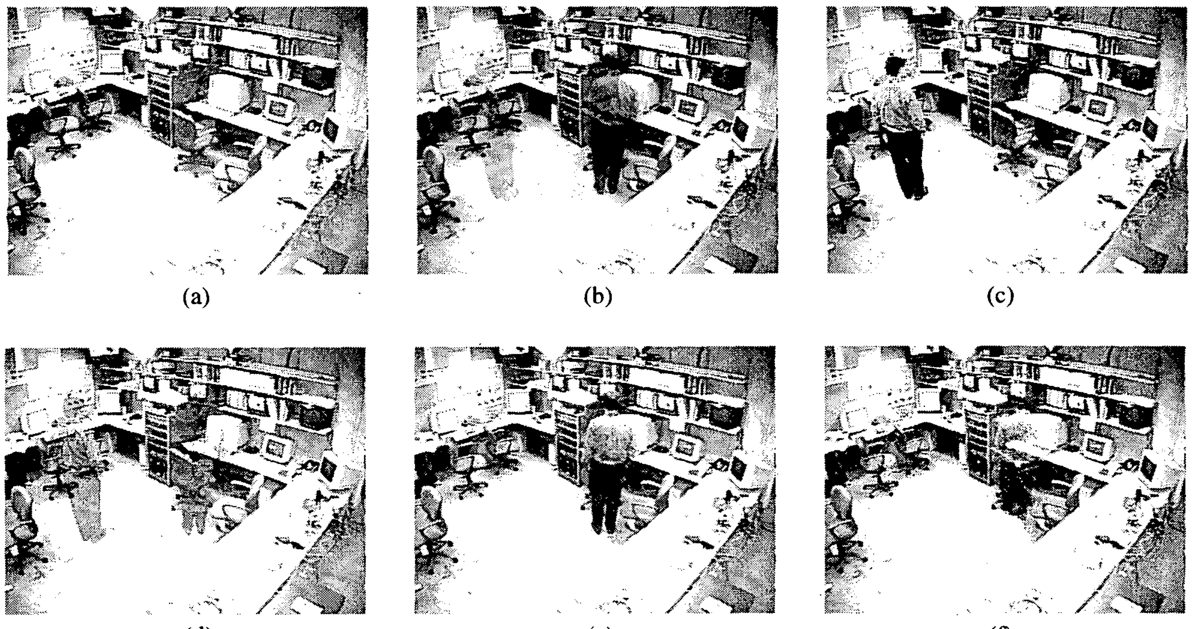

Figure 2. Example output from five background modeling techniques for the scene shown in (a): (b) mean value, (c) DBS, (d) Kalman filter, (e) median value, and (f) adaptive smoothness detector. In this example, a subject stands on the right side of the image frame for approximately 70% of the sequence, then walks to the left side of the image frame, and stands there for the final 20% of the sequence.

percent of the time during the training sequence. The output of the median filter typically will contain large errors when this assumption is false, as in Figure 2(e). The advantage of using the median rather than the mean is that it avoids blending pixel values. The mean of a pixel's intensity over time may not correspond to any of the pixel's actual values during that time, in which case it is likely to be in error. Dawson-Howe proposed a similar technique called dynamic background subtraction (DBS) [1] using a fixed window of three frames to recursively update the background, avoiding the rather expensive cost of computing the median.

An algorithm proposed by Long and Yang [9] also avoids the problem of blending. Their algorithm, called the adaptive smoothness method, finds intervals of stable intensity, and uses a heuristic which chooses the longest, most stable interval as the one most likely to represent the background. Our algorithm, called the Local Image Flow algorithm, is similar to adaptive smoothness in that we generate hypotheses by locating intervals of relatively constant intensity.

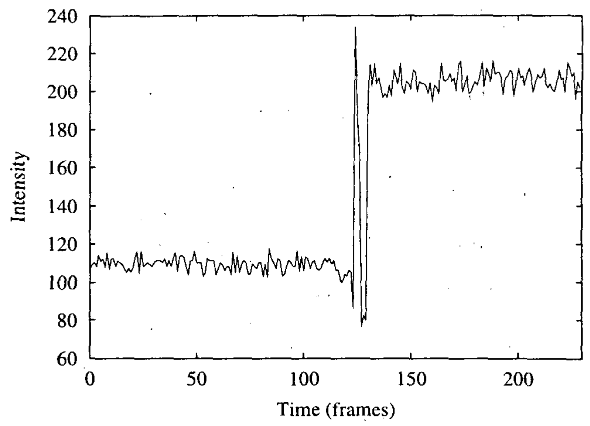

Figure 3. Example intensity history plot of a pixel. The transition at $t = 130$ is caused by an object occluding the background.

3. Local Image Flow Algorithm

3.1. Motivation

Most background updating techniques operate at the pixel-level, making decisions independently for each pixel. Figure 3 shows a typical plot of intensity over time for one pixel in the image. An accurate decision about the true value of the background based on this information alone is impossible. At time $t = 130$, a transition occurs, but it cannot be determined whether the transition was from foreground to background, or background to foreground. A third possibility is a transition between two different foreground objects.

The pixel-level data is, however, useful in narrowing the number of possible values of the background. In Figure 3, it can be reasonably assumed that the approximate intensity value of the background is either 110 or 210. Now the decision is simplified from a choice between 255 values to a choice between only two values.

Our approach is motivated by Long and Yang's adaptive smoothness detector [9]. Their method performs well when all foreground objects are in motion throughout the sequence. However, for sequences in which foreground objects were stationary for a long period of time, many pixels are incorrectly classified. Like most other algorithms, adaptive smoothness makes the decision for each pixel independently, and does not utilize other information from the sequence. To overcome this, our method uses information about motion in the vicinity of a pixel as an additional heuristic. We consider the optical flow in the neighborhood surrounding each pixel. If the direction of optical flow in the neighborhood is toward the pixel, then there is likely to be a moving object approaching the pixel. However, if the majority of optical flow is directed away from the pixel, it is likely that the moving object is leaving the area. Figure 4 illustrates these two situations.

One simple way of using image flow would be to compute the probability of each transition being a "cover" or "uncover" event. For example, these probabilities could be computed by counting the number of flow vectors entering and exiting the neighborhood of a pixel during the event. However, it is very difficult to precisely define a transition event using the intensity history alone. While most transitions appear as large changes in intensity over a short time interval, as in Figure 3, more complex sequences sometimes contain extended intervals of instability. Our algorithm instead locates intervals of stable intensity, which are easily detected using a sliding window. These stable intervals form a set of reasonable hypotheses of the actual intervals of background visibility.

3.2. Algorithm

The first step of the algorithm is to locate, for each pixel, time intervals over which the intensity is stable. That is, we find all sub-intervals in the training sequence of length at least w frames, in which intensity changes at most δ_{max}. Let $\langle a_0, \ldots, a_n \rangle$ represent the intensity of a pixel over the entire sequence. Locate all non-overlapping subsequences $\langle a_i, \ldots, a_j \rangle$ such that the following two conditions hold:

$$\begin{gathered} w \leq j - i, \\ \forall(s,t)\ |a_s - a_t| \leq \delta_{max}. \end{gathered} \tag{1}$$

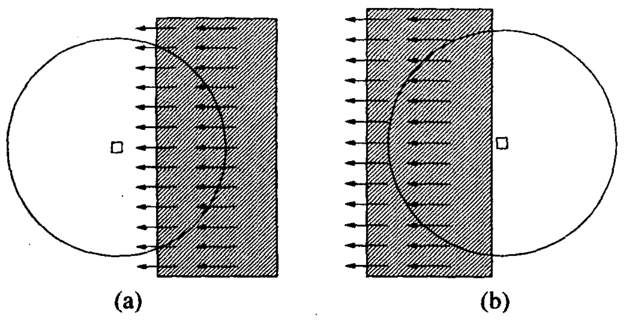
Figure 4. Flow fields in the neighborhood of a pixel (□) may be used to detect (a) covering events, and (b) uncovering events. The shaded regions represent part of a foreground object.

In our implementation, values of $\delta_{max} = 10$, and $w = 6$ frames were used. These intervals represent a set of candidate hypotheses, of which one will be chosen to model the background.

The second stage of the algorithm estimates the likelihood of background visibility for each pixel, for every frame of the training sequence. This is done by first computing optical flow for each consecutive pair of images. In our implementation optical flow was computed using 3x3 block matching on sub-sampled images of size 80x60. This produces m flow vectors whose heads have positions $(x_{i,1}, y_{i,1})$, and whose tails have positions $(x_{i,2}, y_{i,2})$. We then estimate the magnitude of the flow moving toward each pixel (input flow), and the magnitude of the flow moving away from each pixel (output flow). Denote the input flow between the images at time $t-1$ and t as F_t^+, and the output flow as F_t^-. (Lower-case letters denote single pixel values, and capital letters denote entire image frames.) We defined the *input flow* of the pixel located at (x, y) as the sum of Gaussian-weighted distances to each vector head in the local neighborhood $\mathcal{N}$:

$$f^+(x,y) = \sum_{i \in \mathcal{N}} \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}[(x - x_{i,1}/\sigma)^2 + (y - y_{i,1}/\sigma)^2]}, \quad (2)$$

and the *output flow* as the sum of Gaussian-weighted distances to each vector tail:

$$f^-(x,y) = \sum_{i \in \mathcal{N}} \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}[(x - x_{i,2}/\sigma)^2 + (y - y_{i,2}/\sigma)^2]}. \quad (3)$$

The input flow F_t^+ and output flow F_t^- are then combined to form the "net flow" N_t by subtracting the output flow from the input flow:

$$N_t = F_t^+ - F_t^-. \quad (4)$$

Therefore N_t will have large positive values in regions of the image where moving objects are approaching, large negative values in regions where moving objects are leaving, and values near zero in regions where no motion is occurring. Figure 5(c) graphically illustrates the net flow between two frames.

The accumulated net flow S_t gives an estimate of the total amount of input and output flow up to time t:

$$S_t = \sum_{i=0}^{t} N_i. \quad (5)$$

This measure is recursively computed for each frame of the sequence. We then use $s_t(x, y)$ to estimate the likelihood of a foreground object occluding the background at position (x, y) at time t. The corresponding likelihood of background visibility at time t is inversely related to the likelihood of foreground visibility, so the background likelihood is defined as $L_t = -S_t$.

In the final step, the candidate hypotheses at each pixel are evaluated by taking the average likelihood $\bar{l}_{[t_1,t_2]}$ over each stable interval $[t_1, t_2]$, calculated as:

$$\bar{l}_{[t_1,t_2]} = \frac{\sum_{i=t_1}^{t_2} l_i}{t_2 - t_1}. \quad (6)$$

The interval $[\alpha, \beta]$ with the greatest average likelihood is chosen to represent the background.

$$[\alpha, \beta] = \arg\max_{[t_1,t_2]} \bar{l}_{[t_1,t_2]} \quad (7)$$

We then compute the mean and variance of the pixel's intensity over this interval and use them as parameters of the background model. An outline of this algorithm is given below.

Algorithm Summary

1. For each pixel, find all intervals of stable intensity.
2. Initialize accumulated net flow and likelihood to zero: $S_0 = 0$, $L_0 = 0$.
3. Compute optical flow for each pair of consecutive image frames I_t and I_{t+1}.
4. Calculate the net flow N_{t+1} between the two frames using Eqs. 2-4.
5. Compute the accumulated net flow for time $t+1$: $S_{t+1} = \sum_{i=1}^{t+1} N_i$.
6. The likelihood of the pixel being background at time $t+1$ is $L_{t+1} = -N_{t+1}$.

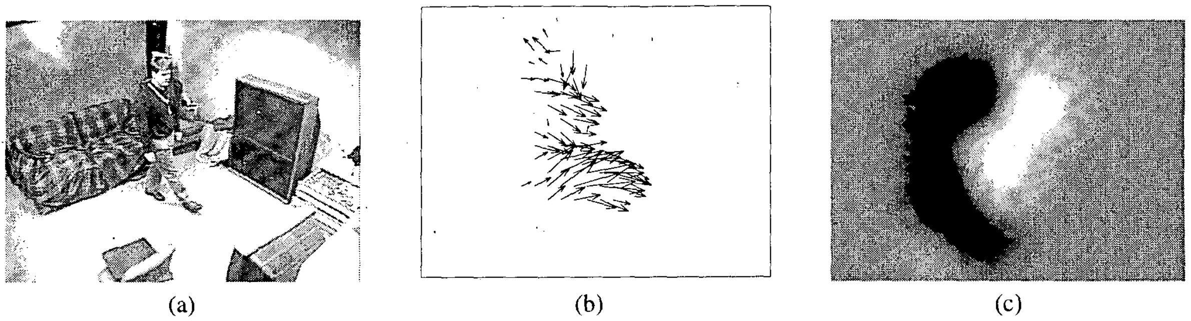

Figure 5. Intermediate stages of computation: (a) image frame I_{t+1}, (b) optical flow between I_t and I_{t+1}, and (c) net flow N_{t+1}. In (c), positive values (light) indicate that the moving object is approaching, and negative values (dark) indicate that the object is leaving. Since the subject is walking from left to right, peaks appear on the right side of the net flow image, and valleys appear on the left side.

7. For each pixel, calculate the average likelihood over each stable interval, and choose the interval with maximum likelihood.

8. The mean and variance of intensity over the chosen interval are used as parameters of the background model.

4. Experimental Results

The video sequences used for testing were taken using ceiling-level dome security cameras. The video was sampled at a resolution of 320x240 and a rate of 10 frames per second. Once the frames were loaded into memory, our algorithm averaged 17 frames per second on a 600 MHz Pentium III. Although we used grayscale video as input, and output a grayscale background model, it is straightforward to extend the algorithm to use color images. Many systems represent the background using color spaces in which illumination has little effect, such as the U-V [12] and chromatic color spaces [5].

We compared our algorithm's output with that of several existing techniques used for background modeling. Tests were performed on several sequences representative of situations which might be commonly encountered in surveillance video. The average performance statistics of several algorithms over a variety of sequences are compared in Table 2. The output of our algorithm for several examples is shown in Figure 6.

To measure the accuracy of the algorithms, we considered three performance criteria: total gray-level error, total number of error pixels, and number of "clustered" error pixels. Error pixels are defined as pixels whose mean value differs from the ideal value by more than some threshold τ. (We used a threshold of $\tau = 20$). A clustered error pixel is defined as any error pixel whose 4-connected neighbors are also error pixels. Of these measures, the number of clustered error pixels may be the most relevant when the background model is to be used for moving object segmentation. A dense region of error in the background model will cause highly inaccurate segmentation output, while sparse errors are easily removed from the segmentation results using standard noise removal techniques such as size thresholding.

The first example in Figure 6 was difficult because a foreground object is present in every frame of the sequence. In this example the results of our algorithm are far better than the others (see Figure 2). The performance of most algorithms was the best for the second sequence, in which the subject continuously moves. Since the foreground occludes the background in any one location for only a small percentage of the total time, smoothing techniques result in a relatively small error. In the third example, two people walk toward one another, pause briefly to shake hands, and then walk away in opposite directions. This sequence was intended to create a difficult scenario for our algorithm, since large occlusion caused by the meeting of two people resulted in an error in the optical flow field. Although the average background estimation error for this sequence was the highest among these three examples, our algorithm significantly out-performed all others. Other tests under much more crowded conditions show that our algorithm is robust to errors in the flow field caused by occlusions.

5. Conclusions and Future Work

In this paper, a new algorithm for background model estimation was presented, which is useful in environments

Method	Avg. Gray-level Error	No. error pixels	No. clustered pixels
Mean	14.29	15029	13003
Exp. Smooth	11.21	9936	6711
Kalman Filter	6.76	4628	2571
DBS	5.64	2807	1493
Median	4.22	2297	1296
Adaptive Smoothness	5.39	3641	766
Local Image Flow (proposed)	2.43	268	22

Table 2. Average results of seven different algorithms run on eight different video sequences.

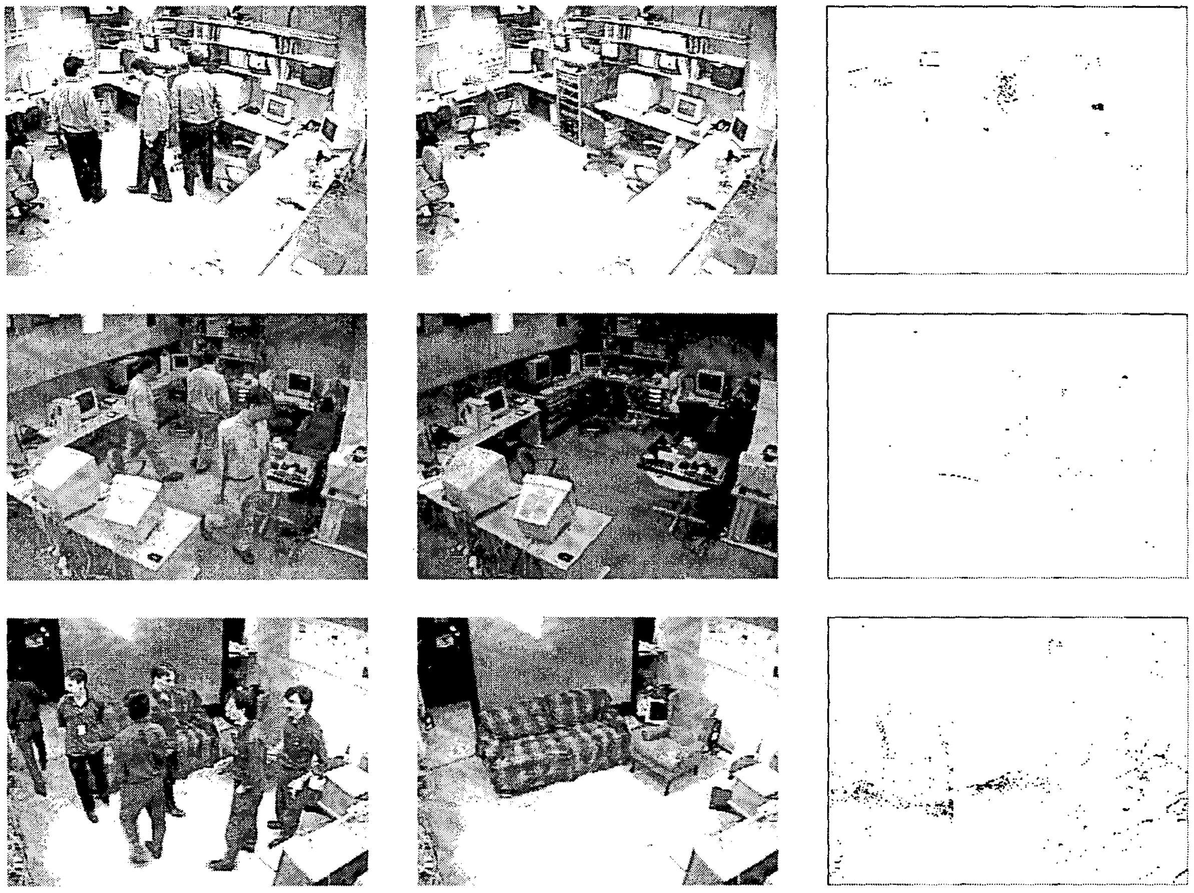

Figure 6. Results of running our algorithm on three different sequences: (a) a snapshot of the training sequence, (b) the computed background model, (c) map of error pixels (black). Compare the results of the first row with those of alternative methods in Figure 2.

where an unobstructed view of the background is not always available. The approach may be applied to many types of input video, since it uses only low-level motion information to construct the background. Like the median filter and adaptive smoothness methods, it avoids the problem of blending pixel values present in many current methods. However, the heuristics used in our technique, based on local image flow, are stronger than those used by the median filter and adaptive smoothness.

The main strength of the algorithm is that while the decision at each pixel is independent of its neighbors, it is not only based on past values observed at that pixel, but also local motion information. Additionally, by estimating the background independently at the pixel-level rather than at the region-level, our algorithm reduces the amount of spatial clustering in the error, and is therefore well suited for segmentation of moving objects. Sparse error is simple to remove from the background-subtracted image using median and size filters.

We are currently exploring possible modifications to the algorithm for handling background adaptation. While the current approach uses batch processing, it is possible to detect stable intervals in a sequential manner by examining a fixed-length window of past frames. Because adaptation must operate over extended periods of time, a method will be required to prune the continually-growing number of hypotheses.

References

[1] K. Dawson-Howe, "Active surveillance using dynamic background subtraction," Tech. Rep. TCD-CS-96-06, Trinity College, 1996.

[2] W. Grimson, C. Stauffer, R. Romano, and L. Lee, "Using adaptive tracking to classify and monitor activities in a site," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, (Santa Barbara, CA), pp. 22–29, 1998.

[3] A. Neri, S. Colonnese, G. Russo, and P. Talone, "Automatic moving object and background separation," *Signal Processing*, vol. 66, pp. 219–232, 1998.

[4] J. Vass, K. Palaniappan, and X. Zhuang, "Automatic spatio-temporal video sequence segmentation," in *Proc. IEEE International Conference on Image Processing*, (Chicago, IL), 1998.

[5] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Proc. 6th European Conference on Computer Vision*, (Dublin, Ireland), 2000.

[6] I. Haritaoglu, D. Harwood, and L. Davis, "W^4: Who? When? Where? What? A real time system for detecting and tracking people," in *Proc. 3rd International Conference on Face and Gesture Recognition*, (Nara, Japan), 1998.

[7] T. Horprasert, D. Harwood, and L. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," in *ICCV Frame-rate Workshop*, 1999.

[8] K. Karmann and A. von Brandt, "Moving object recognition using an adaptive background memory," in *Time-Varying Image Processing and Moving Object Recognition II* (V. Cappellini, ed.), pp. 289–296, Elsevier Science, 1990.

[9] W. Long and Y. H. Yang, "Stationary background generation: An alternative to the difference of two images," *Pattern Recognition*, vol. 23, no. 12, pp. 1351–1359, 1990.

[10] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, (Fort Collins, CO), pp. 246–252, 1999.

[11] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *International Conf. on Computer Vision*, (Kerkyra, Greece), pp. 255–261, 1999.

[12] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[13] B. Gloyer, H. K. Aghajan, K. Y. Siu, and T. Kailath, "Video-based freeway monitoring system using recursive vehicle tracking," in *Proc. of IS&T-SPIE Symposium on Electronic Imaging: Image and Video Processing*, 1995.

[14] Y. H. Yang and M. D. Levine, "The background primal sketch: An approach for tracking moving objects," *Machine Vision and Applications*, vol. 5, pp. 17–34, 1992.

[15] X. Gao, T. Boult, F. Coetzee, and V. Ramesh, "Error analysis of background adaption," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, (Hilton Head, SC), pp. 503–510, 2000.