ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Real-Time Imaging ▮ (▮▮▮▮) ▮▮▮–▮▮▮

www.elsevier.com/locate/rti

Real-time foreground–background segmentation using codebook model

Kyungnam Kim[a,*], Thanarat H. Chalidabhongse[b], David Harwood[a], Larry Davis[a]

[a]Computer Vision Lab, Department of Computer Science, University of Maryland, College Park, MD 20742, USA
[b]Faculty of Information Technology, King Mongkut's Institute of Technology, Ladkrabang, Bangkok 10520, Thailand

Abstract

We present a real-time algorithm for foreground–background segmentation. Sample background values at each pixel are quantized into codebooks which represent a compressed form of background model for a long image sequence. This allows us to capture structural background variation due to periodic-like motion over a long period of time under limited memory. The codebook representation is efficient in memory and speed compared with other background modeling techniques. Our method can handle scenes containing moving backgrounds or illumination variations, and it achieves robust detection for different types of videos. We compared our method with other multimode modeling techniques.

In addition to the basic algorithm, two features improving the algorithm are presented—layered modeling/detection and adaptive codebook updating.

For performance evaluation, we have applied perturbation detection rate analysis to four background subtraction algorithms and two videos of different types of scenes.

1. Introduction

The capability of extracting moving objects from a video sequence captured using a static camera is a typical first step in visual surveillance. A common approach for discriminating moving objects from the background is detection by background subtraction. The idea of background subtraction is to subtract or difference the current image from a reference background model. The subtraction identifies non-stationary or new objects.

1.1. Related work

The simplest background model assumes that the intensity values of a pixel can be modeled by a single unimodal distribution. This basic model is used in [1,2]. However, a single-mode model cannot handle multiple backgrounds, like waving trees. The generalized mixture of Gaussians (MOG) in [3] has been used to model complex, non-static backgrounds. Methods employing MOG have been widely incorporated into algorithms that utilize Bayesian frameworks [4], dense depth data [5], color and gradient information [6], mean-shift analysis [7], and region-based information [8].

MOG does have some disadvantages. Backgrounds having fast variations are not easily modeled with just a few Gaussians accurately, and it may fail to provide sensitive detection (which is mentioned in [9]). In addition, depending on the learning rate to adapt to background changes, MOG faces trade-off problems. For a low learning rate, it produces a wide model that has difficulty in detecting a sudden change to the background. If the model adapts too quickly, slowly moving foreground pixels will be absorbed into the background model, resulting in a high false negative rate. This is the foreground aperture problem described in [10].

*Corresponding author. Tel.: +1 301 405 8368; fax: +1 301 314 9658.

E-mail addresses: knkim@umiacs.umd.edu (K. Kim), thanarat@it.kmitl.ac.th (T.H. Chalidabhongse), harwood@umiacs.umd.edu (D. Harwood), lsd@umiacs.umd.edu (L. Davis).

URL: http://www.cs.umd.edu/~knkim.

1077-2014/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rti.2004.12.004

To overcome these problems, a non-parametric technique estimating the probability density function at each pixel from many samples using kernel density estimation technique was developed in [9]. It is able to adapt very quickly to changes in the background process and to detect targets with high sensitivity. A more advanced approach using adaptive kernel density estimation was recently proposed in [11].

However, the non-parametric technique in [9] cannot be used when long-time periods are needed to sufficiently sample the background—for example when there is significant wind load on vegetation—due mostly to memory constraints. Our algorithm constructs a highly compressed background model that addresses that problem.

Pixel-based techniques assume that the time series of observations is independent at each pixel. In contrast, some researchers [5,8,10] employ a region- or frame-based approach by segmenting an image into regions or by refining low-level classification obtained at the pixel level. Markov random field techniques employed in [12,13] can also model both temporal and spatial context. Algorithms in [14,15] aim to segment the foreground objects in dynamic textured backgrounds (e.g., water, escalators, waving trees, etc.). Furthermore, Amer et al. [16] describes interactions between low-level object segments and high-level information such as tracking or event description.

1.2. Proposed algorithm

Our codebook (CB) background subtraction algorithm was intended to sample values over long times, without making parametric assumptions. Mixed backgrounds can be modeled by multiple codewords. The key features of the algorithm are

- an adaptive and compact background model that can capture structural background motion over a long period of time under limited memory. This allows us to encode moving backgrounds or multiple changing backgrounds;
- the capability of coping with local and global illumination changes;
- unconstrained training that allows moving foreground objects in the scene during the initial training period;
- layered modeling and detection allowing us to have multiple layers of background representing different background layers.

In Section 2, we describe the codebook construction algorithm and the color and brightness metric, used for detection. We show, in Section 3, that the method is suitable for both stationary and moving backgrounds in different types of scenes, and applicable to compressed videos such as MPEG. Important improvements to the above algorithm are presented in Section 4—layered modeling/detection and adaptive codebook updating. In Section 5, a performance evaluation technique—perturbation detection rate analysis—is used to evaluate four pixel-based algorithms. Finally, conclusion and discussion are presented in last Section 6.

2. Background modeling and detection

The CB algorithm adopts a quantization/clustering technique, inspired by Kohonen [18,19], to construct a background model from long observation sequences. For each pixel, it builds a codebook consisting of one or more codewords. Samples at each pixel are clustered into the set of codewords based on a color distortion metric together with brightness bounds. Not all pixels have the same number of codewords. The clusters represented by codewords do not necessarily correspond to single Gaussian or other parametric distributions. Even if the distribution at a pixel were a single normal, there could be several codewords for that pixel. The background is encoded on a pixel-by-pixel basis.

Detection involves testing the difference of the current image from the background model with respect to color and brightness differences. If an incoming pixel meets two conditions, it is classified as background—(1) the color distortion to some codeword is less than the detection threshold, and (2) its brightness lies within the brightness range of that codeword. Otherwise, it is classified as foreground.

2.1. Construction of the initial codebook

The algorithm is described for color imagery, but it can also be used for gray-scale imagery with minor modifications. Let $\mathscr{X}$ be a training sequence for a single pixel consisting of N RGB-vectors: $\mathscr{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. Let $\mathscr{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_L\}$ represent the codebook for the pixel consisting of L codewords. Each pixel has a different codebook size based on its sample variation.

Each codeword $\mathbf{c}_i$, $i = 1 \ldots L$, consists of an RGB vector $\mathbf{v}_i = (\bar{R}_i, \bar{G}_i, \bar{B}_i)$ and a 6-tuple $\mathbf{aux}_i = \langle \check{I}_i, \hat{I}_i, f_i, \lambda_i, p_i, q_i \rangle$. The tuple $\mathbf{aux}_i$ contains intensity (brightness) values and temporal variables described below:

$\check{I}, \hat{I}$	the *min* and *max* brightness, respectively, of all pixels assigned to this codeword
f	the *frequency* with which the codeword has occurred
λ	the *maximum negative run-length* (MNRL) defined as the longest interval during the training period that the codeword has NOT recurred
p, q	the *first* and *last* access times, respectively, that the codeword has occurred

In the training period, each value, $\mathbf{x}_t$, sampled at time t is compared to the current codebook to determine which codeword $\mathbf{c}_m$ (if any) it matches (m is the matching codeword's index). We use the matched codeword as the sample's encoding approximation. To determine which codeword will be the best match, we employ a color distortion measure and brightness bounds. The detailed algorithm is given below.

Algorithm for Codebook construction

```
I. $L \leftarrow 0$[1], $\mathscr{C} \leftarrow \emptyset$ (empty set)
II. **for** $t = 1$ to $N$ **do**
    (i) $\mathbf{x}_t = (R, G, B)$, $I \leftarrow \sqrt{R^2 + G^2 + B^2}$
    (ii) Find the codeword $\mathbf{c}_m$ in $\mathscr{C} = \{\mathbf{c}_i | 1 \leqslant i \leqslant L\}$ matching to $\mathbf{x}_t$ based on two conditions (a) and (b).
        (a) $\mathit{colordist}(\mathbf{x}_t, \mathbf{v}_m) \leqslant \varepsilon_1$
        (b) $\mathit{brightness}(I, \langle \check{I}_m, \hat{I}_m \rangle) = \mathbf{true}$
    (iii) If $\mathscr{C} = \emptyset$ or there is no match, then $L \leftarrow L + 1$. Create a new codeword $\mathbf{c}_L$ by setting
        • $\mathbf{v}_L \leftarrow (R, G, B)$
        • $\mathbf{aux}_L \leftarrow \langle I, I, 1, t-1, t, t \rangle$.
    (iv) Otherwise, update the matched codeword $\mathbf{c}_m$, consisting of
        $\mathbf{v}_m = (\bar{R}_m, \bar{G}_m, \bar{B}_m)$ and $\mathbf{aux}_m = \langle \check{I}_m, \hat{I}_m, f_m, \lambda_m, p_m, q_m \rangle$, by setting
        • $\mathbf{v}_m \leftarrow \left( \frac{f_m \bar{R}_m + R}{f_m + 1}, \frac{f_m \bar{G}_m + G}{f_m + 1}, \frac{f_m \bar{B}_m + B}{f_m + 1} \right)$
        • $\mathbf{aux}_m \leftarrow \langle \min\{I, \check{I}_m\}, \max\{I, \hat{I}_m\}, f_m + 1, \max\{\lambda_m, t - q_m\}, p_m, t \rangle$.
    **end for**
III. For each codeword $\mathbf{c}_i$, $i = 1, \ldots, L$, wrap around $\lambda_i$ by setting $\lambda_i \leftarrow \max\{\lambda_i, (N - q_i + p_i - 1)\}$.
```

The two conditions (a) and (b) in the Step II(ii), detailed in Eqs. (2, 3) later, are satisfied when the pure colors of $\mathbf{x}_t$ and $\mathbf{c}_m$ are close enough and the brightness of $\mathbf{x}_t$ lies between the acceptable brightness bounds of $\mathbf{c}_m$. Instead of finding the nearest neighbor, we just find the first codeword to satisfy these two conditions. ε_1 is the sampling threshold (bandwidth). One way to improve the speed of the algorithm is to relocate the most recently updated codeword to the front of the codebook list. Most of the time, the matched codeword was the first codeword thus relocated, making the matching step efficient.

Note that reordering the training set almost always results in codebooks with the same detection capacity. Reordering the training set would require maintaining all or a large part of it in memory. Experiments show that one-pass training is sufficient. Retraining or other simple "batch" processing methods do not affect detection significantly.

2.2. Maximum negative run-length

We refer to the codebook obtained from the previous step as the fat codebook. It contains all the codewords that represent the training image sequence, and may include some moving foreground objects and noise.

In the temporal filtering step, we refine the fat codebook by separating the codewords that might contain moving foreground objects from the true background codewords, thus allowing moving foreground objects during the initial training period. The true background, which includes both static pixels and moving background pixels, usually is quasi-periodic (values recur in a bounded period). This motivates the temporal criterion of MNRL (λ), which is defined as the maximum interval of time that the codeword has not recurred during the training period. For example, as shown in Fig. 1, a pixel on the tip of the tree was sampled to plot its intensity variation over time. The codeword of sky-color has a very small λ, around 15, and that of tree-color has 100. However, the codeword of the person's body has a very large λ, 280.

Let $\mathscr{M}$ and $T_{\mathscr{M}}$ denote the background model (which is a refined codebook after temporal filtering) and the threshold value, respectively. Usually, $T_{\mathscr{M}}$ is set equal to half the number of training frames, $N/2$,

$$\mathscr{M} = \{\mathbf{c}_m \,|\, \mathbf{c}_m \in \mathscr{C} \wedge \lambda_m \leqslant T_{\mathscr{M}}\}. \tag{1}$$

Codewords having a large λ will be eliminated from the codebook by Eq. (1). Even though one has a large frequency 'f', its large λ means that it is mostly a foreground event which was stationary only for that period f. On the other hand, one having a small f and a small λ could be a rare background event occurring quasi-periodically. We can use λ as a feature to

[1] ← means assignment.

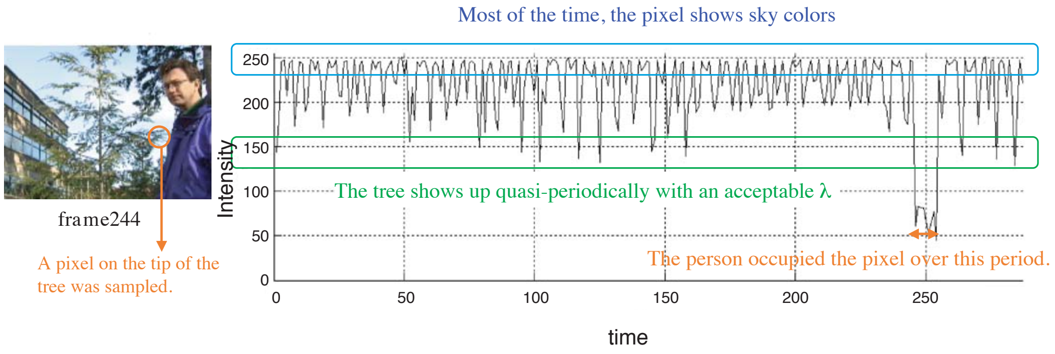


Fig. 1. Example showing how MNRL is used.

discriminate the actual background codewords from the moving foreground codewords. If $T_{\mathscr{M}} = N/2$, all the codewords should recur at least every $N/2$ frames. We note that we also experimented with the combination of the frequency f and λ, but that λ alone performs almost the same as that combination.

Experiments on many videos reveal that only 6.5 codewords per pixel (on average) are required for the background acquisition in order to model 5 min of outdoor video captured at 30 frames/s. By contrast, indoor videos are simpler, having one or two background values nearly everywhere. This reasonable number of codewords means that our method achieves a high compression of the background model. This allows us to capture variable moving backgrounds over a very long period of training time with limited memory.

2.3. Color and brightness

To deal with global and local illumination changes such as shadows and highlights, algorithms generally employ normalized colors (color ratios). These techniques typically work poorly in dark areas of the image. The dark pixels have higher uncertainty[2] than the bright pixels, since the color ratio uncertainty is related to brightness. Brightness should be used as a factor in comparing color ratios. This uncertainty makes the detection in dark regions unstable. The false detections tend to be clustered around the dark regions. This problem is discussed in [17].

Hence, we observed how pixel values change over time under lighting variation. Fig. 2(b) shows the pixel value distributions in the RGB space where 4 representative pixels are sampled from the image sequence of the color-chart in Fig. 2(a). In the sequence captured in a lab environment, the illumination changes over time by decreasing or increasing the light strength to make the pixel values darker or brighter. The pixel values are mostly distributed in elongated shape along the axis going toward the origin point $(0,0,0)$.

Based on this observation, we developed a color model depicted in Fig. 3 to perform a separate evaluation of color distortion and brightness distortion. The motivation of this model is that background pixel values lie along the principal axis of the codeword along with the low and high bound of brightness, since the variation is mainly due to brightness. When we have an input pixel $\mathbf{x}_t = (R, G, B)$ and a codeword $\mathbf{c}_i$ where $\mathbf{v}_i = (\bar{R}_i, \bar{G}_i, \bar{B}_i)$,

$$\|\mathbf{x}_t\|^2 = R^2 + G^2 + B^2,$$
$$\|\mathbf{v}_i\|^2 = \bar{R}_i^2 + \bar{G}_i^2 + \bar{B}_i^2,$$
$$\langle \mathbf{x}_t, \mathbf{v}_i \rangle^2 = (\bar{R}_i R + \bar{G}_i G + \bar{B}_i B)^2.$$

The color distortion δ can be calculated by

$$p^2 = \|\mathbf{x}_t\|^2 \cos^2\theta = \frac{\langle \mathbf{x}_t, \mathbf{v}_i \rangle^2}{\|\mathbf{v}_i\|^2},$$
$$colordist(\mathbf{x}_t, \mathbf{v}_i) = \delta = \sqrt{\|\mathbf{x}_t\|^2 - p^2}. \tag{2}$$

Our color distortion measure can be interpreted as a brightness-weighted version in the normalized color space. This is equivalent to geometrically rescaling (normalizing) a codeword vector to the brightness of an input pixel. This way, the brightness is taken into consideration for measuring the color distortion, and we avoid the instability of normalized colors.

To allow for brightness changes in detection, we store $\check{I}$ and $\hat{I}$ statistics, which are the min and max brightness of all pixels assigned to a codeword, in the 6-tuple defined in Section 2.1. We allow the brightness change to vary in a certain range that limits the shadow level and highlight level. The range is $[I_{low}, I_{hi}]$, for each codeword, defined as

$$I_{low} = \alpha\hat{I}, \quad I_{hi} = \min\left\{\beta\hat{I}, \frac{\check{I}}{\alpha}\right\},$$

[2] Consider two pairs of two color values at the same Euclidean distance in RGB space—$\langle 10,10,10\rangle$ and $\langle 9,10,11\rangle$ for dark pixels, $\langle 200,200,200\rangle$ and $\langle 199,200,201\rangle$ for bright pixels. Their distortions in normalized colors are $\frac{2}{30} = \frac{|10-9|+|10-10|+|10-11|}{30}$ and $\frac{2}{200} = \frac{|200-199|+|200-200|+|200-201|}{200}$, respectively.

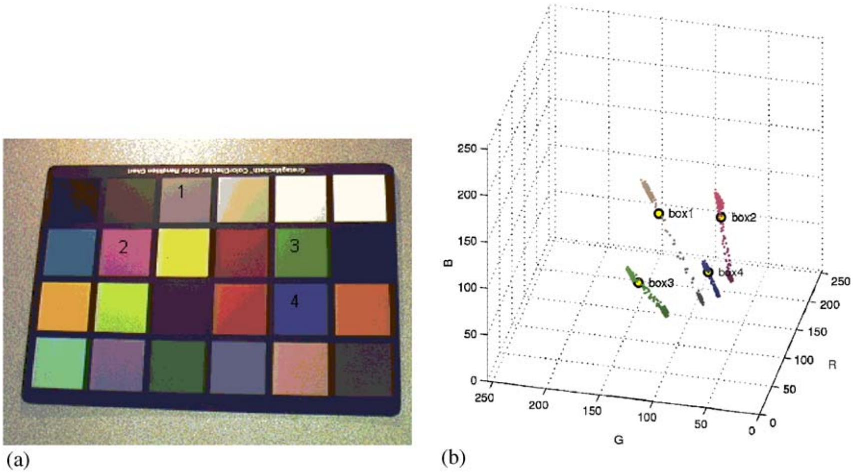


Fig. 2. The distributions of 4 pixel values of the color-chart image sequence having illumination changes over time: (a) original color-chart image, (b) 3D plot of pixel distributions.

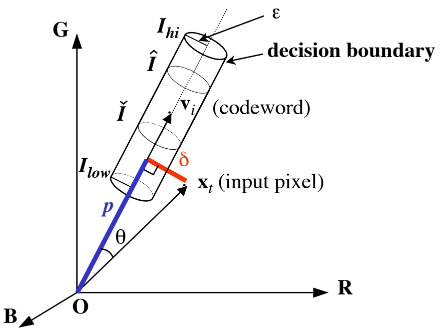


Fig. 3. The proposed color model—a separate evaluation of color distortion and brightness distortion.

where $\alpha < 1$ and $\beta > 1$. Typically, α is between 0.4 and 0.7,[3] and β is between 1.1 and 1.5.[4] This range $[I_{low}, I_{hi}]$ becomes a stable range during codebook updating. The logical brightness function in Section 2.1 is defined as

$$brightness(I, \langle \check{I}, \hat{I} \rangle) = \begin{cases} \textbf{true} & \text{if } I_{low} \leqslant \|\mathbf{x}_t\| \leqslant I_{hi}, \\ \textbf{false} & \text{otherwise.} \end{cases} \tag{3}$$

2.4. Foreground detection

Subtracting the current image from the background model is straightforward. Unlike MOG or [9] which compute probabilities using costly floating point operations, our method does not involve probability calculation. Indeed, the probability estimate in [9] is dominated by the nearby training samples. We simply compute the distance of the sample from the nearest cluster mean. This is very fast and shows little difference in detection compared with the probability estimate. The subtraction operation $BGS(\mathbf{x})$ for an incoming pixel value $\mathbf{x}$ in the test set is defined as:

Algorithm for Background subtraction

```
I.   $\mathbf{x} = (R, G, B),\ I \leftarrow \sqrt{R^2 + G^2 + B^2}$
II.  For all codewords in $\mathcal{M}$ in Eq. (1), find the
     codeword $\mathbf{c}_m$ matching to $\mathbf{x}$ based on two
     conditions:
       • $colordist(\mathbf{x}, \mathbf{c}_m) \leqslant \varepsilon_2$
       • $brightness(I, \langle \check{I}_m, \hat{I}_m \rangle) = \textbf{true}$
     Update the matched codeword as in Step II (iv)
     in the algorithm of codebook construction.
III. $BGS(\mathbf{x}) = \begin{cases} \textbf{foreground} & \text{if there is no match} \\ \textbf{background} & \text{otherwise.} \end{cases}$
```

ε_2 is the detection threshold. The pixel is detected as foreground if no acceptable matching codeword exists. Otherwise it is classified as background.

2.5. Review of multimode modeling techniques

Here, we compare our method with other multimode background modeling techniques—MOG [3] and Kernel [9]. The characteristics of each algorithm are listed in Table 1.

- Unlike MOG, we do not assume that backgrounds are multimode Gaussians. If this assumption, by chance, were correct, then MOG would get accurate parameters, and would be very accurate. But this is not always true. The background distribution could be very different from normal, as we see in compressed videos such as MPEG.

[3] These typical values are obtained from experiments. 0.4 allows large brightness bounds, but 0.7 gives tight bounds.

[4] β is additionally used for limiting I_{hi} since shadows (rather than highlights) are observed in most cases.

Table 1
Characteristics of background modeling algorithms

	MOG [3]	Kernel [9]	CB (proposed)
Model representation	Mixture of Gaussians	Kernel density	Codebook
Model evaluation	Probability density estimation	Probability density estimation	Distance
Parametric modeling	Yes	No	No
Color metric	RGB only	Normalized color r, g and s (brightness)	Rescaled RGB and brightness
Background memorization capacity	As much as *K* Gaussians hold	Short-term (*N* samples) Long-term (*N* samples)	Almost practically infinite memory
Memory usage	Small	Large	Compact
Processing speed	Slow	Slow	Fast
Model maintenance	Online updating with *K* Gaussians	Short- and long-term models	Layered modeling and detection using *cache*

- Also, in contrast to Kernel, we do not store raw samples to maintain the background model. These samples are huge, but do not cover a long period of time. The codebook models are so compact that we can maintain them with very limited memory.
- Ours handles *multi-backgrounds* well. There is no restriction of the number of backgrounds. It can model trees which move longer than the raw sample size of Kernel. Even the rare background events, which meet the quasi-periodicity condition, survive as backgrounds.
- *Unconstrained* training using MNRL filtering allows moving foreground objects in the training sequence.
- Our codebook method does not evaluate probabilities, which is very computationally expensive. We just calculate the distance from the cluster means. That makes the operations fast.
- MOG uses the original RGB variables and does not separately model brightness and color. MOG currently does not model covariances, which are often large and caused by variation in brightness. It is probably best to explicitly model brightness. Kernel uses normalized colors and brightness; the normalized color has uncertainty related to brightness. To cope with the problem of illumination changes such as shading and highlights, we calculates a brightness difference as well as a color difference of rescaled RGB values.

3. Detection results and comparison

Most existing background subtraction algorithms fail to work with low-bandwidth compressed videos mainly due to spatial block compression that causes block artifacts, and temporal block compression that causes abnormal distribution of encoding (random spikes). Fig. 4(a) is an image extracted from an MPEG video encoded at 70 kbits/s. Fig. 4(b) depicts 20-times scaled image of the standard deviations of blue(B)-channel values in the training set. It is easy to see that the distribution of pixel values has been affected by the blocking effects of MPEG. The unimodal model in Fig. 4(c) suffers from these effects. For the compressed video, CB eliminates most compression artifacts—see Figs. 4(c)–(f).

In a compressed video, pixel intensities are usually quantized into a few discontinuous values based on an encoding scheme. Their histograms show several spike distributions in contrast to continuous bell-shaped distributions for an uncompressed video. MOG has low sensitivity around its Gaussian tails and less frequent events produce low probability with high variance. Kernel's background model, which contains a recent N-frame history of pixel values, may not cover some background events which were quantized before the N frames. If Gaussian kernels are used, the same problems occur as in the MOG case. CB is based on a vector quantization technique. It can handle these discrete quantized samples, once they survive temporal filtering (λ-filtering).

Fig. 5 illustrates the ability of the codebooks to model multiple moving backgrounds—The trees behind the person moving significantly in the video. For the test sequence[5] used in Fig. 5(a), further comparison of our method was done with 10 different algorithms, and the results are described in [10].

In areas such as building gates, highways, or pathways where people walk, it is difficult to obtain good background models without filtering out the effects of foreground objects. We applied the algorithms to a test video in which people are always moving in and out a building (see Fig. 6). By λ-filtering, our method was able to obtain the most complete background model.

[5] We would like to thank K. Toyama and J. Krumm at Microsoft Research, for providing us with this image sequence.

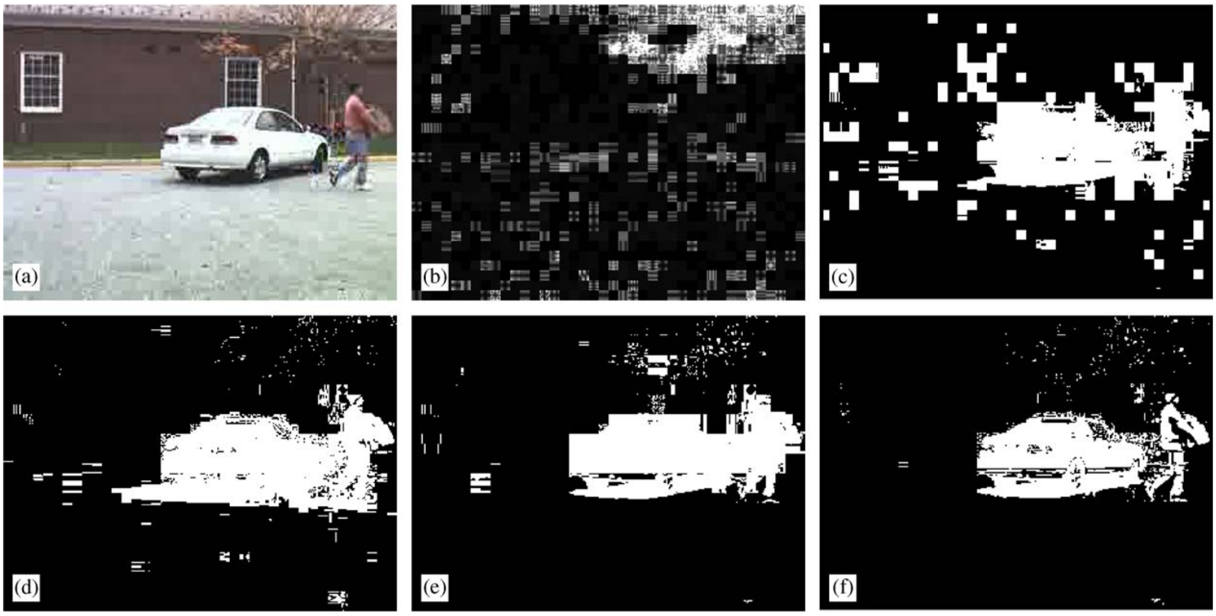


Fig. 4. Detection results on a compressed video: (a) original image, (b) standard deviations, (c) unimodal model in [2], (d) MOG, (e) Kernel, (f) CB (proposed).

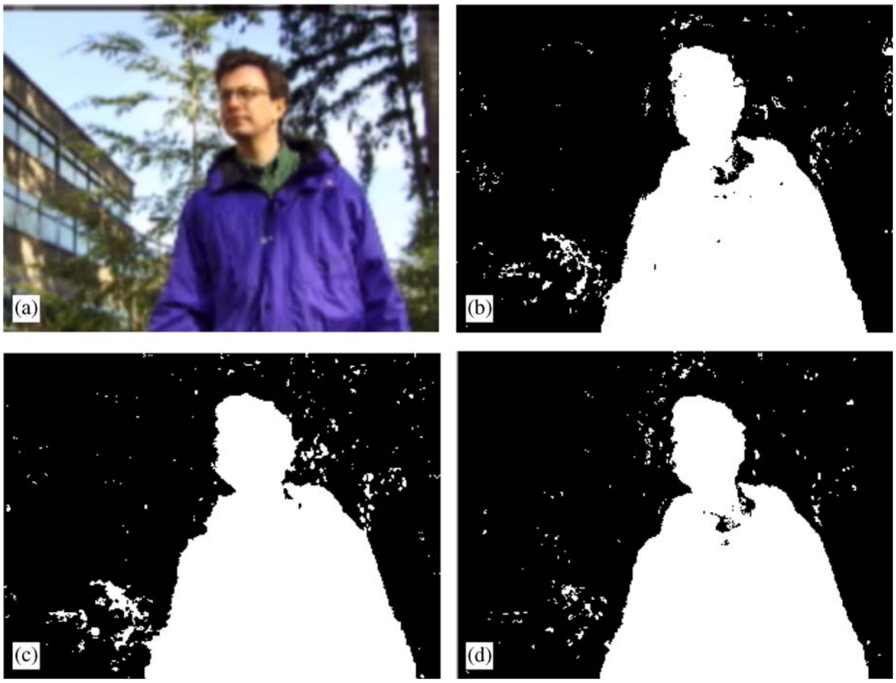


Fig. 5. Detection results on multiple moving backgrounds: (a) original image, (b) MOG, (c) Kernel, (d) CB (proposed).

Multiple backgrounds moving over a long period of time cannot be well trained with techniques having limited memory constraints. A sequence of 1000 frames recorded at 30 frames/s (fps) was trained. It contains trees moving irregularly over that period. The number of Gaussians allowed for MOG was 10. A sample of size 300 was used to represent the background. Fig. 7 shows that CB captures most multiple background events; here we show typical false alarms for a frame containing no foreground objects. This is due to a compact background model represented by quantized codewords.

The implementation of the approach is quite straightforward and is faster than MOG and Kernel. Table 2 shows the speeds to process the results in Figs. 7(b)–(d) on a 2 GHz Dual Pentium system. Note that the training time of Kernel is mostly used for reading and storing samples.

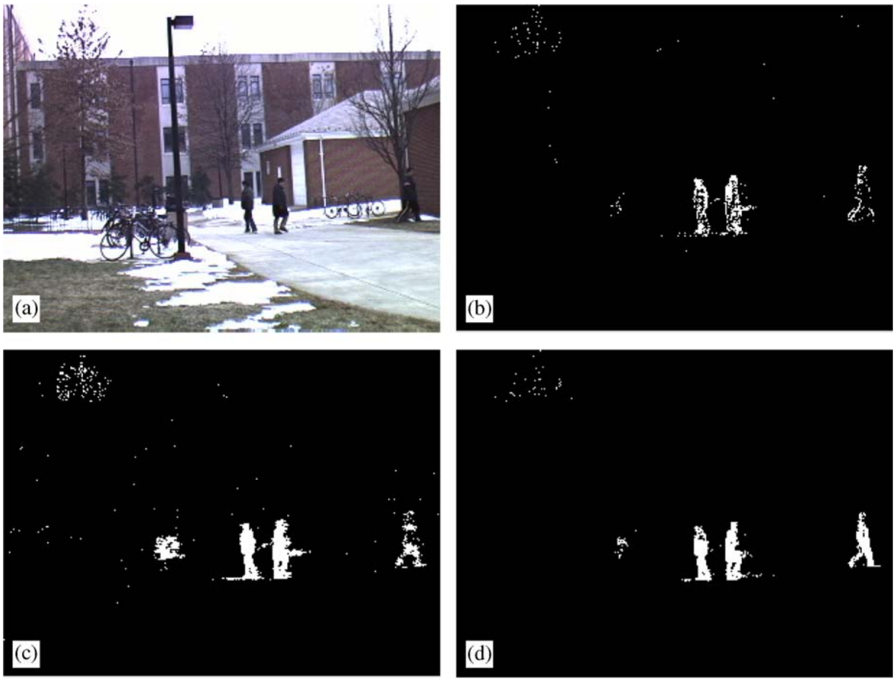


Fig. 6. Detections results on training of non-clean backgrounds: (a) original image, (b) MOG, (c) Kernel, (d) CB (proposed).

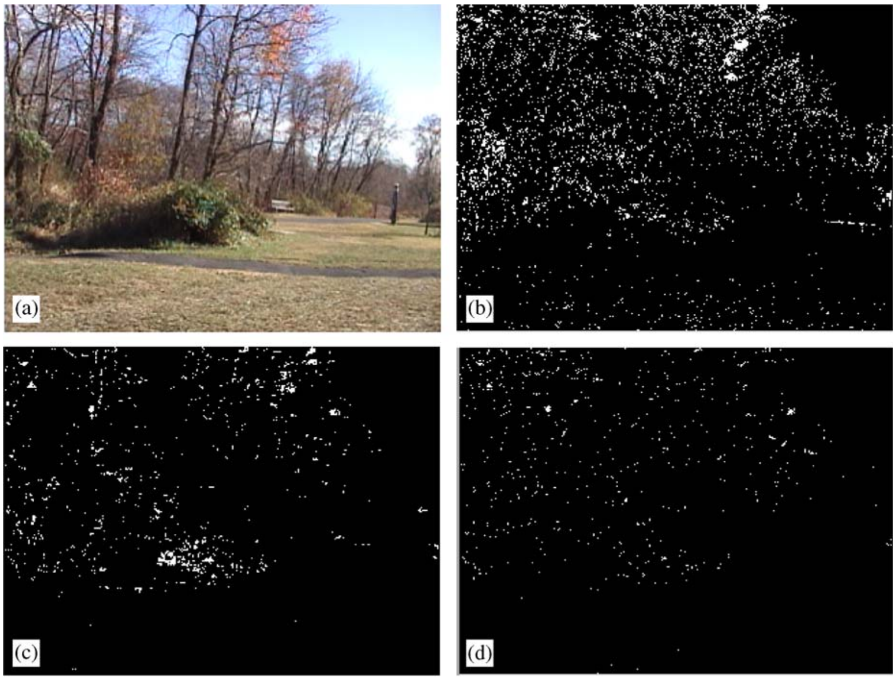


Fig. 7. Detections results on very long-time backgrounds: (a) original image, (b) MOG, (c) Kernel, (d) CB (proposed).

Regarding memory usage for the results in Figs. 7(b)–(d), MOG requires 5 floating point numbers[6] RGB means, a variance, a weight for each distribution—10 Gaussians correspond to *200* bytes. Kernel needs 3 bytes for each sample—300 samples amount to *900* bytes. In CB, we have 5 floating point numbers ($\bar{R}, \bar{G}, \bar{B}, \check{I}, \hat{I}$) and 4 integers ($f, \lambda, p, q$)—the average[7] number of codewords in each pixel, 4 codewords, can be stored in *112* bytes.

[6] Floating point: 4 bytes, integer: 2 bytes.

[7] The number of codewords depends on the variation of pixel values.

4. Improvements

In order to make our technique more practically useful in a visual surveillance system, we improved the basic algorithm by layered modeling/detection and adaptive codebook updating.

4.1. Layered modeling and detection—model maintenance

The motivation of layered modeling and detection is to still be able to detect foreground objects against new backgrounds which were obtained during the detection phase. If we do not have those background layers, interesting foreground objects (e.g., people) will be detected mixed with other stationary objects (e.g., car).

The scene can change after initial training, for example, by parked cars, displaced books, etc. These changes should be used to update the background model. We do this by defining an additional model $\mathscr{H}$ called a *cache* and three parameters—$T_{\mathscr{H}}$, T_{add}, and T_{delete}. The periodicity of an incoming pixel value is filtered by $T_{\mathscr{H}}$, as we did in the background modeling. The values re-appearing for a certain amount of time (T_{add}) are added to the background model as non-permanent, short-term background. We assume that the background obtained during the initial background modeling is permanent. Background values not accessed for a long time (T_{delete}) are deleted from the background model. Thus, a pixel can be classified into four subclasses—(1) background found in the permanent background model, (2) background found in the non-permanent background model, (3) foreground found in the cache, and (4) foreground not found in any of them. This adaptive modeling capability also allows us to capture changes to the background scene (see Fig. 8). Only two layers of background are described here, but this can be extended to multiple layers. The detailed procedure is given below:

Table 2
Processing speed in frames/s

	MOG	Kernel	CB
Background training	8.3	40.8	39.2
Background subtraction	12.1	11.1	30.7

I. After training, the background model $\mathscr{M}$ is obtained. Create a new model $\mathscr{H}$ as a cache.
II. For an incoming pixel $\mathbf{x}$, find a matching codeword in $\mathscr{M}$. If found, update the codeword.
III. Otherwise, try to find a matching codeword in $\mathscr{H}$ and update it. For no match, create a new codeword $\mathbf{h}$ and add it to $\mathscr{H}$.
IV. Filter out the cache codewords based on $T_{\mathscr{H}}$.

$$\mathscr{H} \leftarrow \mathscr{H} - \{\mathbf{h}_i | \mathbf{h}_i \in \mathscr{H}, \lambda \text{ of } \mathbf{h}_i \text{ is longer than } T_{\mathscr{H}}\}$$

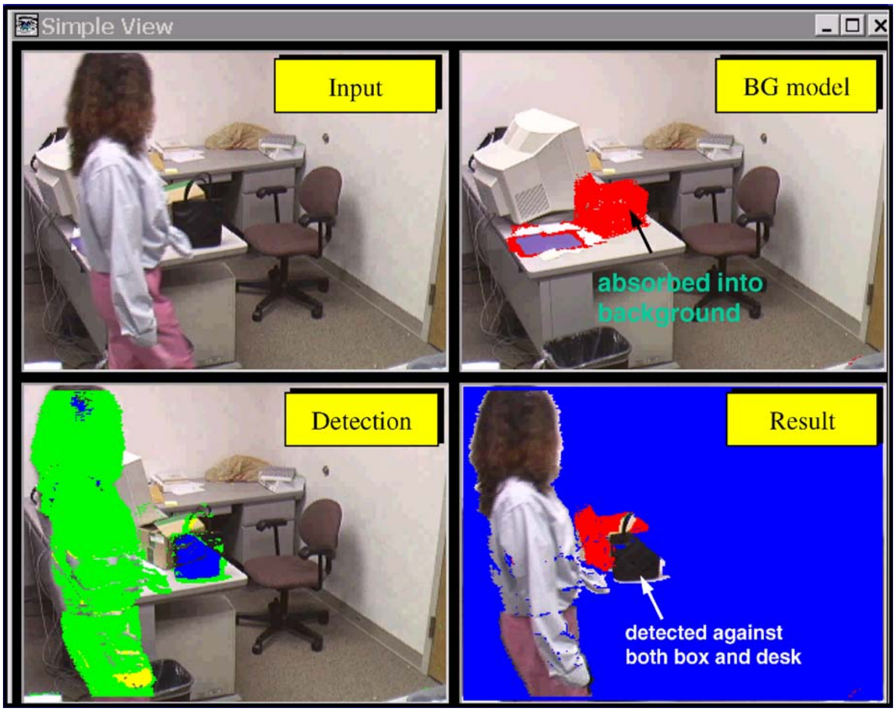


Fig. 8. Layered modeling and detection—a woman placed a box on the desk and then the box has been absorbed into the background model as non-permanent. Then a purse is put in front of the box. The purse is detected against both the box and the desk.

```
V. Move the cache codewords staying for enough
   time, to $\mathcal{M}$.
   $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathbf{h}_i | \mathbf{h}_i \in \mathcal{H},\ \mathbf{h}_i$ stays longer than $T_{add}\}$
VI. Delete the codewords not accessed for a long time
   from $\mathcal{M}$.
   $\mathcal{M} \leftarrow \mathcal{M} - \{\mathbf{c}_i | \mathbf{c}_i \in \mathcal{M}, \mathbf{c}_i$ not accessed for $T_{delete}\}$
VII. Repeat the process from the Step II.
```

Layered modeling and detection can also be used for the further analysis of scene change detection. As shown in Fig. 9, a man unloads two boxes after parking the car. The car and the two boxes are labeled with different coloring based on their 'first-access-times' as non-permanent backgrounds while the man is still detected as foreground.

4.2. Adaptive codebook updating—detection under global illumination changes

Global illumination changes (for example, due to moving clouds) make it difficult to conduct background subtraction in outdoor scenes. They cause over-detection, false alarms, or low sensitivity to true targets. Good detection requires equivalent false alarm rates over time and space. We discovered from experiments that variations of pixel values are different (1) at different surfaces (shiny or muddy), and (2) under different levels of illumination (dark or bright). Codewords should be adaptively updated during illumination changes. Exponential smoothing of codeword vector and variance with suitable learning rates is efficient in dealing with illumination changes. It can be done by replacing the updating formula of $\mathbf{v}_m$ with

$$\mathbf{v}_m \leftarrow \gamma \mathbf{x}_t + (1-\gamma)\mathbf{v}_m$$

and appending

$$\sigma_m^2 \leftarrow \rho\delta^2 + (1-\rho)\sigma_m^2$$

to Step II (iv) of the algorithm for codebook construction. γ and ρ are learning rates. Here, σ_m^2 is the overall variance of color distortion in our color model, not the variance of RGB. σ_m is initialized when the algorithm starts. Finally the function *colordist*() in Eq. (2) is modified to

$$colordist\,(\mathbf{x}_t, \mathbf{v}_i) = \frac{\delta}{\sigma_i}.$$

We tested a PETS'2001[8] sequence which is challenging in terms of multiple targets and significant lighting variation. Fig. 10(a) shows two sample points (labeled 1 and 2) which are significantly affected by illumination changes and Fig. 10(b) shows the brightness changes of those two points. As shown in Fig. 10(d), adaptive codebook updating eliminates the false detection which occurs on the roof and road in Fig. 10(c).

5. Performance evaluation using PDR analysis

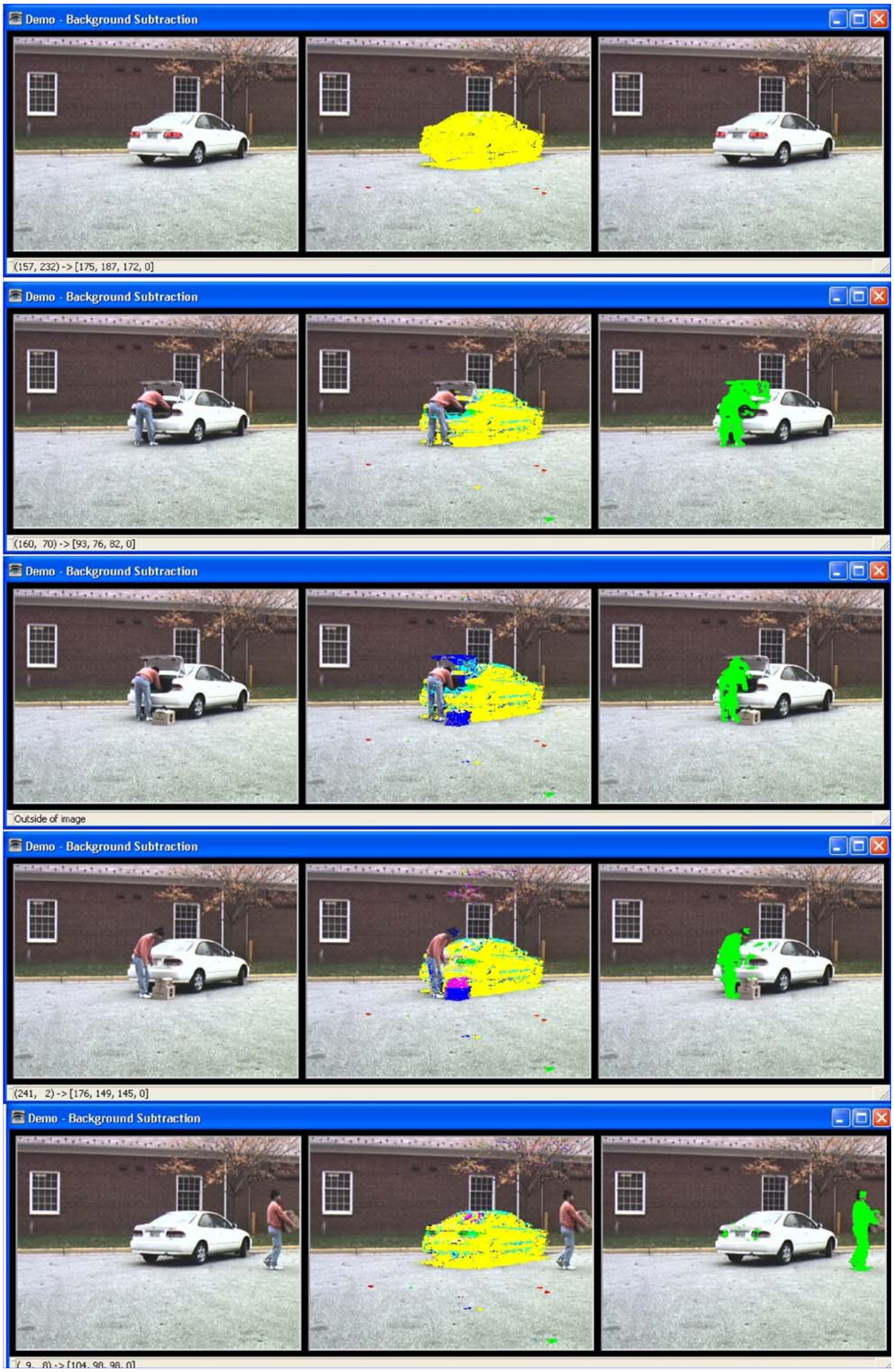
In this section we evaluate the performance of several background subtraction algorithms using perturbation detection rate (PDR) analysis. PDR measures, given a false alarm rate (FA-rate), the sensitivity of a background subtraction algorithm in detecting low contrast targets against a background as a function of contrast (Δ), also depending on how well the model captures mixed (moving) background events. As an alternative to the common method of ROC analysis, it does not require foreground targets or knowledge of foreground distributions. PDR graphs show how sensitively an algorithm detects foreground targets at a certain contrast (Δ) to the background as the contrast increases. A detailed discussion of PDR analysis is reported in [21].

We evaluate four algorithms—CB (proposed), MOG [3], KER [9], and UNI [2]. UNI was added to evaluate single-mode technique in contrast to multi-mode ones. Since the algorithm in [9] can work with either normalized colors (KER) or RGB colors (KER.RGB), it has two separate graphs. Fig. 11 shows the representative empty frames from two test videos.

Fig. 12 depicts an example of foreground detection, showing differences in detection sensitivity for two algorithms due to differences in the color metrics. These differences reflect the performance shown in the PDR graph in Fig. 13. The video image in Fig. 12(a) shows someone with a red sweater standing in front of the brick wall of somewhat different reddish color shown in Fig. 11(a). There are detection holes through the sweater (and face) and more shadows behind the person in the MOG result (Fig. 12(b)). The holes are mainly due to difference in color balance and not overall brightness. The CB result in Fig. 12(c) is much better for this small contrast. After inspection of the image, the magnitude of contrast Δ was determined to be about 16 in missing spots. Fig. 13 shows a large difference in detection for this contrast, as indicated by the vertical line.

Fig. 14 shows how sensitively the algorithms detect foregrounds against a scene containing moving backgrounds (trees). In order to sample enough moving background events, 300 frames are allowed for training. A windows is placed to represent 'moving backgrounds' as shown in Fig. 11(b). PDR analysis is performed on the window with the FA-rate obtained only within the

[8] IEEE International Workshop on Performance Evaluation of Tracking and Surveillance 2001 at http://www.visualsurveillance.org/PETS2001.


Demo - Background Subtraction
Demo - Background Subtraction
Demo - Background Subtraction
Demo - Background Subtraction
Demo - Background Subtraction

(157, 232) -> [175, 187, 172, 0]

(160, 70) -> [93, 76, 82, 0]

Outside of image

(241, 2) -> [176, 149, 145, 0]

(9, 8) -> [104, 98, 98, 0]

Fig. 9. The leftmost column: original images, the middle column: color-labeled non-permanent backgrounds, the rightmost column: detected foreground. The video shows that a man parks his car on the lot and takes out two boxes. He walks away to deliver them.

window—a ‘window’ false alarm rate (instead of ‘frame’ false alarm rate).

The PDR graph (Fig. 14) for the moving background window is generally shifted right, indicating reduced sensitivity of all algorithms for moving backgrounds. Also, it shows differences in performance among algorithms, with CB and KER performing best. CB and KER, both of which model mixed backgrounds and separate color/brightness, are most sensitive, while, as expected, UNI does not perform well as in the previous case because it was designed for single-mode backgrounds. KER.RGB and MOG are also less sensitive outdoors.

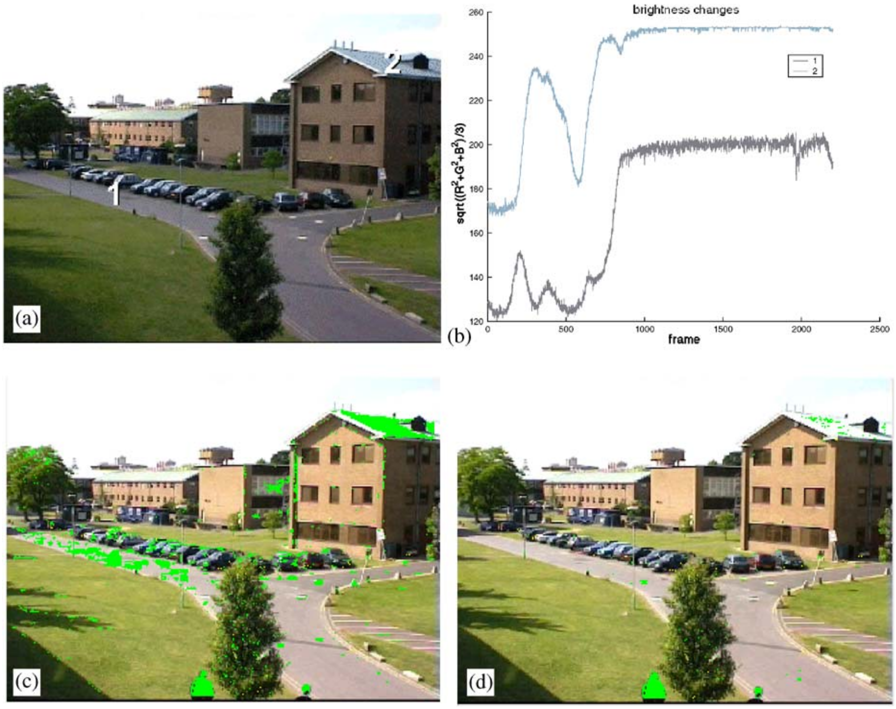


Fig. 10. Results of adaptive codebook updating for detection under global illumination changes. Detected foregrounds on the frame 1105 are labeled with green color: (a) original image—frame 1, (b) brightness changes, (c) before adaptive updating, (d) after adaptive updating.

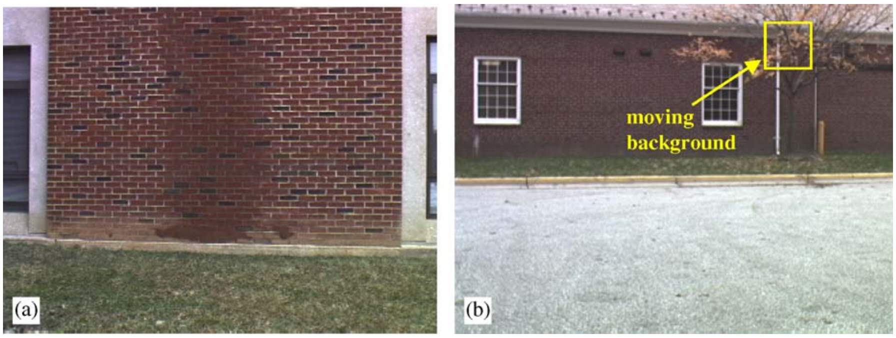


Fig. 11. The sample empty-frames of the two videos used in the experiments: (a) red-brick wall, (b) parking lot.

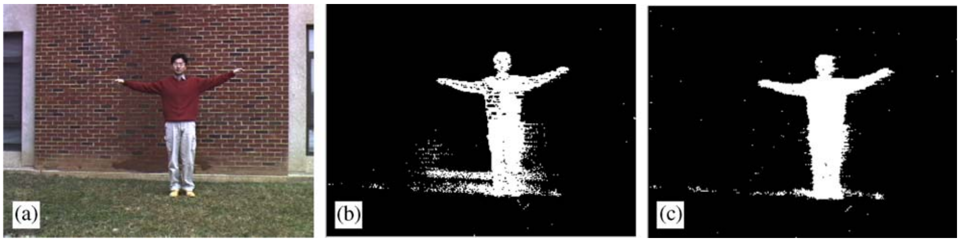


Fig. 12. Sensitive detection at small contrast showing the differences in color metrics of the algorithms: (a) a 'red-brick wall' frame including a person in a red sweater, (b) MOG, (c) CB (proposed).

6. Conclusion and discussion

Our new adaptive background subtraction algorithm, which is able to model a background from a long training sequence with limited memory, works well on moving backgrounds, illumination changes (using our color distortion measures), and compressed videos having irregular intensity distributions. It has other desirable features—unconstrained training and layered modeling/detection. Comparison with other multimode

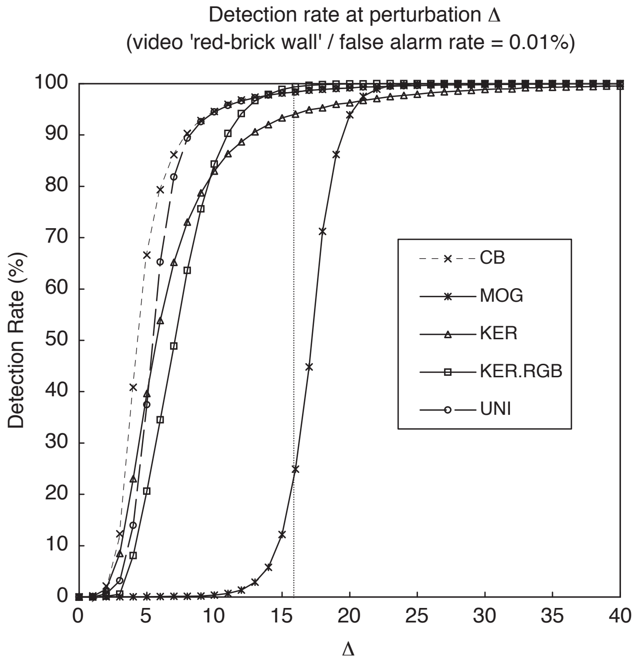


Fig. 13. PDR for ‘red-brick wall’ video in Fig. 11(a).

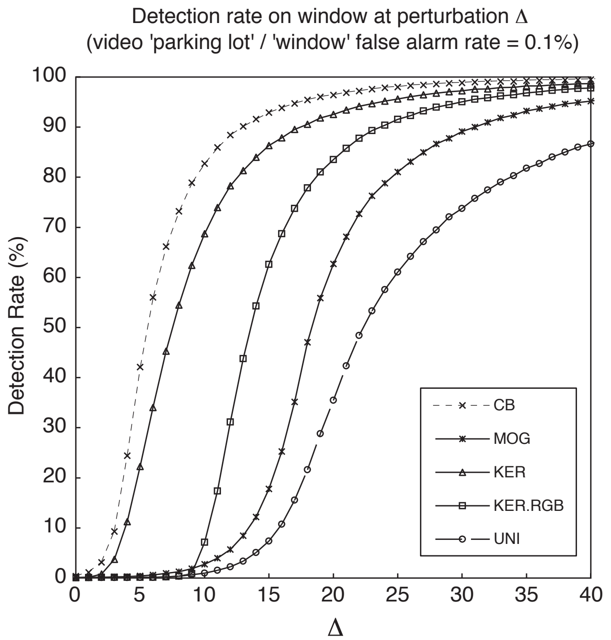


Fig. 14. PDR for window on moving background (Fig. 11(b)).

modeling algorithms shows that the codebook algorithm has good properties on several background modeling problems.

In summary, our major contributions are as follows:

(1) We propose a background modeling technique efficient in both memory and speed. Experiments show that nearest neighbor ‘classification’, which is computationally very efficient, is as effective as probabilistic classification (both kernel and MOG) for our application. Practically, even when computing probabilities of pixel measurements coming from the background, these probabilities are dominated by the nearest component of the background mixture.
(2) The most important lesson, based on our experience, for analyzing color videos is that using an appropriate color model is critical for obtaining accurate detection, especially in low light conditions such as in shadows. Using RGB directly lowers detection sensitivity because most of the variance at a pixel is due to brightness, and absorbing that variability into the individual RGB components results in a lower true detection rate for any desired false alarm rate. In other words, an algorithm would have to allow greater color variability than the data actually requires in order to accommodate the intrinsic variability in brightness. Using normalized colors, on the other hand, is undesirable because of their high variance at low brightness levels; in order to maintain sufficiently low detection error rates at low brightness, one necessarily sacrifices sensitivity at high brightness. This is due to using an angular measure between normalized color coordinates for detection. The color model proposed in this paper, on the other hand, maintains a constant false alarm rate across, essentially, the entire range of brightness levels. One would expect that modifying other background subtraction algorithms, such as the MOG algorithm, to use this more appropriate color model would bring their performance much closer to that of the codebook algorithm.

We have applied the PDR analysis to four background subtraction algorithms and two videos of different types of scenes. The results reflect obvious differences among the algorithms as applied to the particular type of background scenes. We also provided a real video example of differences among the algorithms with respect to sensitive foreground detection which is consistent with the PDR simulation.

Automatic parameter selection is an important goal for visual surveillance systems as addressed in [20]. Two of our parameters, ε_1 in Section 2.1 and ε_2 in Section 2.4, can be automatically determined. Their values depend on variation within a single background distribution, and are closely related to false alarm rates. Preliminary experiments on many videos show that automatically chosen threshold parameters ε_1 and ε_2 are sufficient. However, they are not always acceptable, especially for highly compressed videos where we cannot always measure the robust parameter accurately. In this regards, further investigation could be done to obtain robust parameters.

References

[1] Wren CR, Azarbayejani A, Darrell T, Pentland A. Pfinder: real-time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence 1997;19(7):780–5.

[2] Horprasert T, Harwood D, Davis LS. A statistical approach for real-time robust background subtraction and shadow detection. IEEE Frame-Rate Applications Workshop, Kerkyra, Greece; 1999.

[3] Stauffer C, Grimson WEL. Adaptive background mixture models for real-time tracking. IEEE International Conference on Computer Vision and Pattern Recognition 1999;2:246–52.

[4] Lee DS, Hull JJ, Erol B. A Bayesian framework for Gaussian mixture background modeling. IEEE International Conference on Image Processing 2003.

[5] Harville M. A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models. European Conference on Computer Vision 2002;3:543–60.

[6] Javed O, Shafique K, Shah M. A hierarchical approach to robust background subtraction using color and gradient information. IEEE Workshop on Motion and Video Computing (MOTION'02); 2002.

[7] Porikli F, Tuzel O. Human body tracking by adaptive background models and mean-shift analysis. IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-ICVS); 2003.

[8] Cristani M, Bicego M, Murino V. Integrated region- and pixel-based approach to background modelling. Proceedings of IEEE Workshop on Motion and Video Computing; 2002.

[9] Elgammal A, Harwood D, Davis LS. Non-parametric model for background subtraction. European Conference on Computer Vision 2000;2:751–67.

[10] Toyama K, Krumm J, Brumitt B, Meyers B. Wallflower: principles and practice of background maintenance. International Conference on Computer Vision 1999; 255–61.

[11] Mittal A, Paragios N. Motion-based background subtraction using adaptive kernel density estimation. IEEE Conference in Computer Vision and Pattern Recognition 2004.

[12] Paragios N, Ramesh V. A MRF-based real-time approach for subway monitoring. IEEE Conference in Computer Vision and Pattern Recognition 2001.

[13] Wang D, Feng T, Shum H, Ma S. A novel probability model for background maintenance and subtraction. The 15th International Conference on Vision Interface; 2002.

[14] Zhong J, Sclaroff S. Segmenting foreground objects from a dynamic textured background via a robust Kalman filter. IEEE International Conference on Computer Vision 2003.

[15] Monnet A, Mittal A, Paragios N, Ramesh V. Background modeling and subtraction of dynamic scenes. IEEE International Conference on Computer Vision 2003.

[16] Amer A, Dubois E, Mitiche A. Real-time system for high-level video representation: application to video surveillance. Proceedings of SPIE International Symposium on Electronic Imaging, Conference on Visual Communication and Image Processing (VCIP); 2003.

[17] Greiffenhagen M, Ramesh V, Comaniciu D, Niemann H. Statistical modeling and performance characterization of a real-time dual camera surveillance system. Proceedings of International Conference on Computer Vision and Pattern Recognition 2000;2:335–42.

[18] Kohonen T. Learning vector quantization. Neural Networks 1988;1:3–16.

[19] Ripley BD. Pattern recognition and neural networks. Cambridge: Cambridge University Press; 1996.

[20] Scotti G, Marcenaro L, Regazzoni C. A S.O.M. based algorithm for video surveillance system parameter optimal selection. IEEE Conference on Advanced Video and Signal Based Surveillance 2003.

[21] Chalidabhongse TH, Kim K, Harwood D, Davis L. A perturbation method for evaluating background subtraction algorithms. Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS); 2003.