BACKGROUND MODELING AND SUBTRACTION BY CODEBOOK CONSTRUCTION

Kyungnam Kim[1], *Thanarat H. Chalidabhongse*[2], *David Harwood*[1], *Larry Davis*[1]

[1]Computer Vision Lab, University of Maryland, College Park, MD 20742, USA
[2]Faculty of Information Technology, King Mongkut's Institute of Technology, Thailand
[1]{knkim|harwood|lsd@umiacs.umd.edu}, [2]{thanarat@it.kmitl.ac.th}

ABSTRACT

We present a new fast algorithm for background modeling and subtraction. 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. Our method can handle scenes containing moving backgrounds or illumination variations (shadows and highlights), and it achieves robust detection for compressed videos. We compared our method with other multimode modeling techniques.

1. INTRODUCTION

In visual surveillance, a common approach for discriminating moving objects from the background is detection by background subtraction. Some background models assume that the series of 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, such as waving trees. The generalized mixture of Gaussians (MOG) has been used to model complex, non-static backgrounds [3, 4]. The MOG has some disadvantages. Background having fast variations cannot be accurately modeled with just a few Gaussians, causing problems for sensitive detection. To overcome these problems, a non-parametric technique [5] was developed for estimating background probabilities at each pixel from many recent samples over time using Kernel density estimation. These pixel-based techniques assume that the time series of observation is independent at each pixel. In contrast, some researchers employ a region- or frame-based approach by segmenting an image into regions or by refining low-level classification obtained at the pixel level [4, 6].

Our codebook (CB) background subtraction algorithm was intended to sample values over long times, without making parametric assumptions. The key features of our algorithm are in the followings: (1) **resistance to artifacts** of acquisition, digitization and compression, (2) capability of coping with **illumination changes**, (3) **adaptive and compressed** background models that can capture structural background motion over a long period of time under limited memory, (4) **unconstrained training** that allows moving foreground objects in the scene during the initial training period.

In Sec.2, we describe the background modeling method along with the color metric and foreground detection. We show, in Sec.3, that the method is suitable for both stationary and moving backgrounds and is robust with respect to image quality. Conclusions and future work are presented in Sec.4.

2. BACKGROUND MODELING

The CB algorithm adopts a quantization/clustering technique [7], to construct a background model. Samples at each pixel are clustered into the set of codewords. The background is encoded on a pixel by pixel basis.

2.1. Construction of the codebook

Let $\mathcal{X}$ be a training sequence for a single pixel consisting of N RGB-vectors: $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$. Let $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_L\}$ represent the codebook for the pixel consisting of L codewords. Each pixel has a different codebook size based on its sample variation. Each codeword $\mathbf{c}_i$, $i = 1 \dots L$, 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, that the codeword accepted;
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.  $L \leftarrow 0$ ($\leftarrow$ means assignment),   $\mathcal{C} \leftarrow \emptyset$ (empty set)
II. for $t$=1 to $N$ do
    i.   $\mathbf{x}_t = (R, G, B)$,   $I \leftarrow R + G + B$
    ii.  Find the codeword $\mathbf{c}_m$ in $\mathcal{C} = \{\mathbf{c}_i | 1 \leq i \leq L\}$ matching to
         $\mathbf{x}_t$ based on two conditions (a) and (b).
           (a) $colordist(\mathbf{x}_t, \mathbf{v}_m) \leq \epsilon_1$
           (b) $brightness(I, \langle \check{I}_m, \hat{I}_m \rangle) = \mathbf{true}$
    iii. If $\mathcal{C} = \emptyset$ 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), detailed in Eq.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).

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 (i.e., iteration of the codebook construction algorithm) does not affect detection significantly.

2.2. Maximum Negative Run-Length

We refer to the codebook obtained from the previous step as the fat codebook. 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.

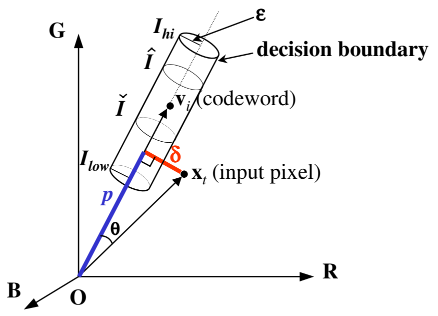


Fig. 1. The proposed color model - separate evaluation of color distortion and brightness distortion.

Let $\mathcal{M}$ denote the background model (a new codebook after temporal filtering):

$$\mathcal{M} = \{\mathbf{c}_m | \mathbf{c}_m \in \mathcal{C} \;\; \wedge \;\; \lambda_m \leq T_{\mathcal{M}}\}. \qquad (1)$$

Usually, a threshold $T_{\mathcal{M}}$ is set equal to half the number of training frames, $\frac{N}{2}$.

A codeword 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.

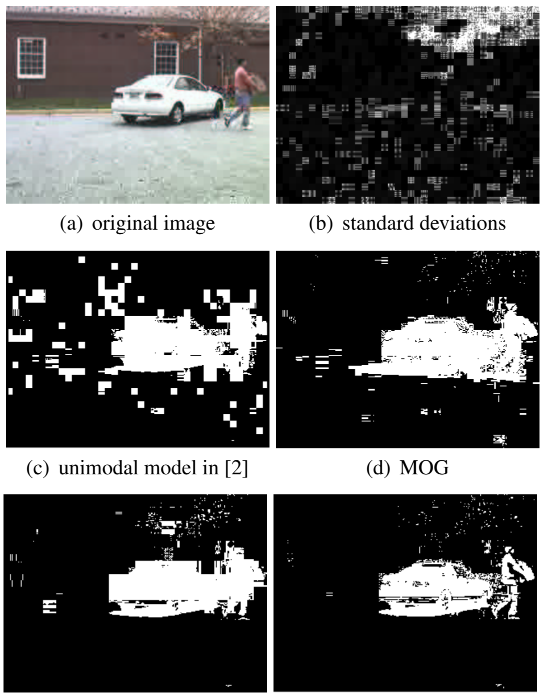
Experiments show that only 6.5 codewords per pixel (on average) are required for the background acquisition in order to model 5 minutes of 30 fps outdoor video. This reasonable number of codewords means that our method achieves a high compression of the background model. This allows us to capture variable moving backgrounds over a very long period of time with limited memory.

2.3. Color and Brightness

To cope with the problem of illumination changes such as shading and highlights, we utilize a color model separating the color and brightness components.

We observe that the pixel values sampled over time are mostly distributed along the axis going toward the origin point $(0, 0, 0)$. We developed a new color model depicted in Fig.1 to perform a separate evaluation of color distortion and brightness distortion. The motivation of this model is that the background pixel values lie along the principal axis of the codeword with the low and high bound of brightness since the variation is mainly due to the brightness. When we have an input pixel $\mathbf{x}_t = (R, G, B)$ and a codeword $\mathbf{c}_i$ where $\mathbf{v}_i = (\bar{R}_i, \bar{G}_i, \bar{B}_i)$,

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



(a) original image (b) standard deviations

(c) unimodal model in [2] (d) MOG

(e) Kernel (f) CB

Fig. 2. Detection results on a compressed video

The color distortion δ can be calculated by

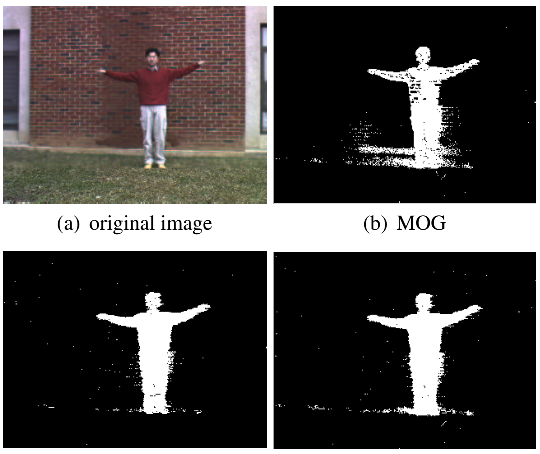
$$\begin{aligned} p^2 &= \|\mathbf{x}_t\|^2 \cos^2\theta = \tfrac{\langle \mathbf{x}_t, \mathbf{v}_i\rangle^2}{\|\mathbf{v}_i\|^2} \\ colordist(\mathbf{x}_t, \mathbf{v}_i) &= \delta = \sqrt{\|\mathbf{x}_t\|^2 - p^2}. \end{aligned} \tag{2}$$

To allow for brightness changes in detection, we consider $\check{I}$ and $\hat{I}$ statistics in the 6-tuple defined in Sec.2.1. In detection, 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 $I_{low} = \alpha\hat{I}, I_{hi} = min\{\beta\hat{I}, \frac{\check{I}}{\alpha}\}$ where $\alpha < 1$ and $\beta > 1$. Typically, α is between 0.4 – 0.7, and β is between 1.1 – 1.5. This range $[I_{low}, I_{hi}]$ becomes a stable range during codebook updating. The logical brightness function in Sec.2.1 is defined as

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

2.4. Foreground Detection

Subtracting the current image from the background model is straightforward. Unlike MOG or [5] which compute probabilities using costly floating point operations, our method does not involve probability calculation. Indeed, the probability estimate in [5] is dominated by the nearby training samples. We simply compute the distance of the sample



(a) original image (b) MOG

(c) Kernel (d) CB

Fig. 3. Differences in the color metrics of the algorithms

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.   $\mathbf{x} = (R, G, B), \quad I \leftarrow R + G + B$
II.  For all codewords in $\mathcal{M}$ in Eq.1, find the codeword $\mathbf{c}_m$ matching
     to $\mathbf{x}$ based on two conditions:
       • $colordist(\mathbf{x}, \mathbf{v}_m) \le \epsilon_2$
       • $brightness(I, \langle\check{I}_m, \hat{I}_m\rangle) = \textbf{true}$
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.

3. DETECTION RESULTS AND COMPARISON

This section demonstrates the performance of the proposed algorithm compared with MOG [3] and Kernel [5].

Fig.2(a) is an image extracted from the MPEG video encoded at 70 kbits/sec. Fig.2(b) depicts a 20-times scaled image of the standard deviations of green(G)-channel values in the training set. The distribution of pixel values has been affected by the blocking effects of MPEG. The unimodal model in Fig.2(c) suffers from these effects. For compressed videos having very abnormal distributions, CB eliminates most compression artifacts - see Fig.2(c)-2(f).

Fig.3(a)-3(d) depict an example of foreground detection, showing differences in the color metrics of the algorithms. The video image in Fig.3(a) shows someone with a red sweater standing in front of a brick wall of somewhat different reddish color. In the MOG result, there are detection holes through the sweater (and the face) and more

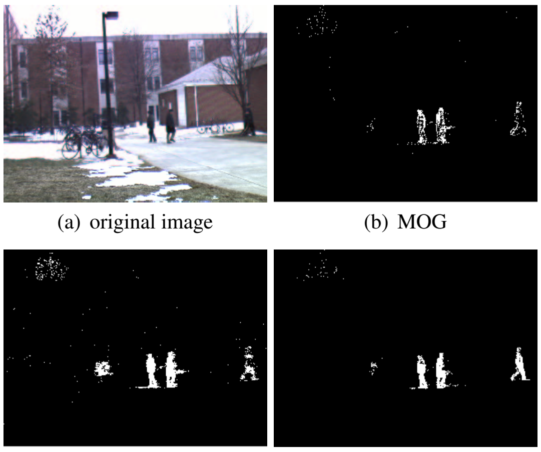

(a) original image (b) MOG (c) Kernel (d) CB

Fig. 4. Detection results on training of non-clean backgrounds

shadows behind the person (Fig.3(b)). The holes are mainly due to difference in color balance and not overall brightness. Kernel shows some miss-detection on the face.

To test unconstrained training, we applied the algorithms to a video in which people are almost always moving in and out a building (see Fig.4(a)-4(d)). By λ-filtering, CB was able to obtain the most complete background model.

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 per second (fps) was trained. It contains trees moving irregularly over that period. The number of Gaussians allowed for MOG was 20. A sample of size 300 was used to represent the background for Kernel. Fig.5(a)-5(d) shows that CB captures most multiple background events. This is due to a compact background model represented by quantized codewords. The processing speeds for this sequence are reported in Table1. The implementation of our approach is straightforward and it is faster than MOG and Kernel.

Further performance evaluation of the algorithms was reported in [8] where a Perturbation Detection Rate (PDR) analysis measures the sensitivity of a background subtraction algorithm in detecting low contrast targets against background as a function of contrast. The results in [8] reflect obvious differences among the algorithms as applied to the particular type of background scenes.

4. CONCLUSIONS AND FUTURE WORK

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. Comparison with other multimode modeling algorithms shows that the proposed one has good properties on several background modeling problems.

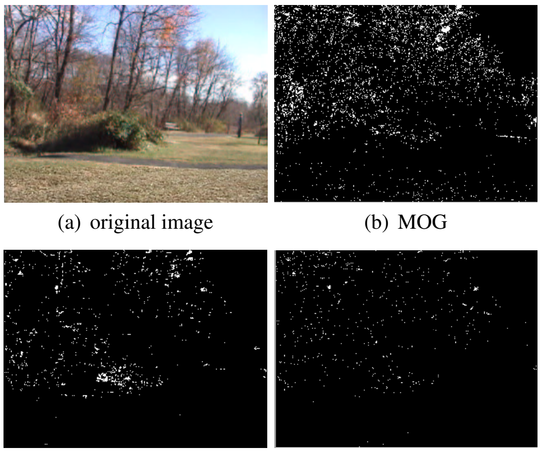

(a) original image (b) MOG (c) Kernel (d) CB

Fig. 5. Detection results on very long-time backgrounds

	MOG	**Kernel**	**CB**
background training	8.3	40.8	39.2
background subtraction	12.1	11.1	30.7

Table 1. Processing speed (fps) on 2GHz Pentium system

The future work will be focused on improving the CB algorithm in terms of automatic parameter estimation and integration of detection and tracking.

5. REFERENCES

[1] C.R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on PAMI*, Vol. 19, no. 7, pp. 780-785, 1997.

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

[3] C. Stauffer and W.E.L. Grimson, "Adaptive background mixture models for real-time tracking," *Int. Conf. Computer Vision and Pattern Recognition*, Vol. 2, pp. 246-252, 1999.

[4] M. Harville, "A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models," *European Conf. Computer Vision*, Vol. 3, pp. 543-560, 2002.

[5] A. Elgammal, D. Harwood, and L.S. Davis, "Non-parametric model for background subtraction," *European Conf. Computer Vision*, Vol. 2, pp. 751-767, 2000.

[6] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," *Int. Conf. Computer Vision*, pp. 255-261, 1999.

[7] T. Kohonen, "Learning vector quantization," *Neural Networks*, Vol. 1, pp. 3-16. 1988.

[8] T. H. Chalidabhongse, K. Kim, D. Harwood and L. Davis, "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.