A BLOCK-BASED BACKGROUND MODEL FOR VIDEO SURVEILLANCE

Xiaoyu Deng, Jiajun Bu, Zhi Yang, Chun Chen, Yi Liu

College of Computer Science, Zhejiang University
Hangzhou, China
Email: {dengxy, bjj, yangzh, chenc, yiliu}@zju.edu.cn

ABSTRACT

Background modeling is an important component of many computer vision systems. The numerous approaches to this problem differ in the statistical models used to describe the temporal behavior of single pixels. Without proper use of spatial coherence between pixel values, these models suffer greatly from memory consumption. In order to reduce spatial redundancy in the data, we propose a novel block-based background model which clusters pixel values within each small block of frames, and build weighted indexes for each pixel to track color values temporally. Compared with traditional models, the proposed model greatly reduces average number of bytes needed to model a pixel, and can be used in real-time video surveillance systems.

Index Terms— Video signal processing, Surveillance, Object detection

1. INTRODUCTION

The extraction of moving objects from video is the first step in many computer vision systems. A common method used to achieve automatic extraction is background subtraction, which detects motion pixels by comparing each new frame with a model of the background, then classifies pixels into different types, i.e., foreground, background.

The simplest background model is to treat the intensity values at each pixel as a single Gaussian distribution [1]. However, such single-mode models cannot handle dynamic backgrounds, such as waving trees, lighting changes and shadow casts. Stauffer and Grimson use a mixture of Gaussians model to represent dynamic backgrounds, and update the model using parameter estimation techniques [2].

To overcome the problem of inaccurate background model caused by errors in parameter estimation for fast varying pixels, Elgammal, Harwood, et al. use a non-parametric model [3]. The model employs a kernel estimator to determine which type current pixel value belongs to, based on recently observed values at the same pixel.

In [4], the authors build a simple pixel-based statistical model using a codebook approach. By clustering pixels into thin cylinders in 3-D color spaces, the codebook model reduces 44% of memory used compared to the mixture of Gaussians model. Calculations of probabilities can also be avoided, which is highly computing-intensive.

Region- or frame-based models are developed by some researchers recently [5-7]. These models consider pixels as correlated random variables, and estimate probabilities based on neighborhood relationships. Compared with these models, our model does not explicitly use spatial and temporal relationships between pixels. Instead, we focus on the color similarities among them, and try to get rid of redundant information by clustering similar colors.

Compared with pixel-based models, our block-based model maintains pixel values collectively within 4x4 or 8x8 square blocks. Similar colors at different pixels can be represented as one cluster instead of many. Usually, the average number of clusters inside a block is limited, due to the coherence between neighboring pixels. Since the number of acquired samples increases, the clustering can act more accurately than on small sample volumes. The model also learns dynamic background faster, since most of the pixels of dynamic background move inside small regions.

Our model maintains a small array of weighted indexes for each pixel. Each weighted index points to a color cluster of the block. The weight is adaptively learned from the rate of that indexed color occurs.

In Section 2, we describe the data structure and the updating procedure of the proposed model, including the block-based color clustering and the weighted indexing. In Section 3, we will show the performances of our model, compared with other classic models. Finally, conclusion and discussion are presented in Section 4.

2. PROPOSED MODEL

Before the proceeding of background modeling, we assume the camera is static in the world coordinates. In order to handle moving cameras, estimation and compensation of the global motion must be done first, then use a static background model to deal with the processed video. There are also some integrated background models using build-in methods to separate background and foreground according to different moving characteristics [8].

For a video frame consisting of $W \times H$ pixels, $\{P_{x,y} \mid x \in (1,W), y \in (1,H)\}$, we segment it into square blocks $\{B_{m,n} \mid m \in (1,\frac{W}{S}), n \in (1,\frac{H}{S})\}$, S is the block width. The size of blocks is chosen according to picture size to efficiently utilize spatial coherence information. we choose 8x8 for CIF (352x288) and 4x4 for QCIF (176x144), for bigger or smaller frames, the choice of block size has an upper limit that depends on the speed of the machine.

For each block $B_{m,n}$, there is a color cluster array, consisting of L cluster: $C_{m,n} = \{c_1, c_2, ..., c_L\}$.

Our cluster array is similar to the codebook in [4], except the cluster array is defined for each block and the codebook is defined for each pixel. Since this, some pixel-wise codebook attributes such as maximum negative run-length (mnrl) are removed. We will see later how weighted indexing plays the pixel-wise role in our model.

Each cluster $c_i (i = 1,2,...,L)$ consists of 6 attributes:

$\overline{y}$, the mean value of Y color component,
$\overline{u}$, the mean value of U color component,
$\overline{v}$, the mean value of V color component,
f, frequency of total samples occurred,
p, the creation time of the cluster.

For each pixel $P_{x,y}$, there is a weighted index array consisting of N indexes: $I_{x,y} = \{i_1, i_2, ..., i_N\}$. Each index $i_k (k = 1,2,...,N)$ consists of 3 items:

idx, the index of the cluster in cluster array,
w, the weight of this cluster of this pixel,
q, the last time color appears in this cluster.

The general updating procedure of the proposed model works as follows: when new value of a pixel arrives, (1) lookup in the color cluster array of the block which the pixel belongs to, if hit, update the cluster, if not hit, create a new cluster, (2) lookup in the weighted index array of the current pixel, if hit, update the index, if not hit, create a new index, (3) determine the type of the pixel according to the weights. The whole procedure of updating the background model is shown in Figure 1.

When the size of the color cluster array L reaches a maximum, cluster with smallest f is replaced by a new coming color and all the old indexes pointing to that cluster must be removed. Because it is inefficient to search every pixel and delete invalid indexes, we use a later-delete approach. Replace cluster without deleting indexes first, then, when indexes are accessed later, use p and q to verify indexes: since q is the last time some color appears in this cluster, it must be larger than the color cluster creation time p, otherwise, the index is invalid.

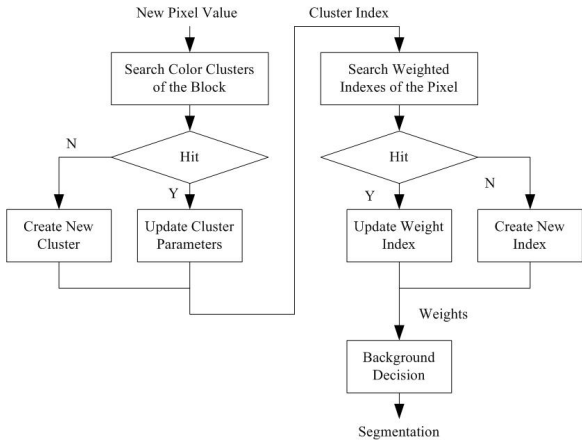

Figure 1. Background model updating procedure.

2.1. Block-based Color Clustering

Human eyes are more sensitive to intensity changes than color changes. In our model, we employ the YUV format which is widely used in video coding applications. In a small region of a frame, the U, V variations between pixels are very small, due to the color consistency of most object surfaces, and Y component varies greatly, see Figure 2. YUV format makes it convenient to cluster colors into cylinders in 3-D color space [4].

The color cluster array is initialized to be empty. When a new pixel $p_t = (y_t, u_t, v_t)$ arrives, search the array for a cluster with (u_t, v_t) differs from $(\overline{u}, \overline{v})$ less than a certain threshold ε_1, which is manually chosen between 5 and 15, determined by the color complexity in the video. Then, y_t must be within a region near $\overline{y}$, the dimension of the region is proportional to $\overline{y}$ by a linear factor ε_2, since the fact that intensity varies more greatly when the value increases. Typically, ε_2 is between 5/255 and 15/255. In the codebook model, the lower-bound and upper-bound of intensity are estimated dynamically according to deviations between current value and the mean value. However, with small variations at some pixels, the training process always tends to over-fit the data, which causes too many clusters, so we use a linear proportional threshold instead.

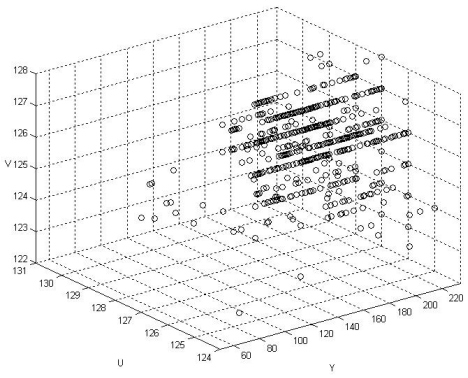

Figure 2. YUV distribution of an 8x8 block in 20 frames.

Then, the mean values of color components $\bar{y}$, $\bar{u}$, $\bar{v}$ and frequency f will be updated. If no proper cluster is found, create a new cluster and initialized it with current color values. The whole clustering and updating procedure is presented in Table 1.

Table 1. Block-based Color Clustering.

I. New pixel $p_t = (y_t, u_t, v_t)$ arrives, $p_t \in B_{m,n}$

II. For each $(\bar{y}, \bar{u}, \bar{v}, f, p)$ in $C_{m,n}$

If $diff((u_t, v_t), (\bar{u}, \bar{v})) \le \varepsilon_1$ and $y_t \in \bar{y} \times \delta(1, \varepsilon_2)$

$$(\bar{y}, \bar{u}, \bar{v}) = \frac{(\bar{y}, \bar{u}, \bar{v}) \times f + (y_t, u_t, v_t)}{f+1}$$

$$f = f + 1$$

III. If no proper cluster found, create a new cluster

$$(\bar{y}, \bar{u}, \bar{v}) = (y_t, u_t, v_t)$$

$$f = 1$$

$$p = t$$

2.2. Weighted Indexing

Compared with pixel-based background models, which record color variations at a pixel directly, block-based model must keep block-wise and pixel-wise information separately. In our model, we maintain an index array for each pixel, while each index in the array points to a color cluster used to appear at the pixel.

In models such as mixture of Gaussians [2], different colors are weighted according to the rate of occurrence automatically. Color weights perform a critical role in the decision of types which a pixel belongs to. Usually, the weight updating procedure uses an unsupervised learning. The simplest one adopted in mixture of Gaussians is to adjust the weight w according to a leaning parameter α :

$$w = (1-\alpha) \times w + \alpha \times M \quad (1)$$

M is 1 when the new pixel value is in the cluster to be weighted, and 0 otherwise, α is typically between 0.3 and 0.7. We employ the same method used by mixture of Gaussians, and merge the weight information with the color cluster index, called weighted index.

Finally, decision must be made to see whether a pixel belongs to the background, or the foreground. Since the weights of colors are adjusted every frame, rare colors' weights descend very quickly. On the other hand, Most of the frequent colors' weights become very large, and is in a dominant ratio β of the sum of weights of all colors, β is usually from 0.6 to 0.9. A pixel with color index of weight w is considered to be a background pixel if the sum of weights larger or equal to w exceeds the sum of all weights times β.

3. EXPERIMENTAL RESULTS

In order to validate the proposed model, we tested the software using standard MPEG-4 test sequences "Hall" and a road surveillance video captured from real road scenes. There's a car moving along the lane in this video, the background is almost static except moving tree leaves. Both formats of the videos are YUV 4:2:0 CIF (352x288).

We compared our model with the mixture of Gaussians (MOG) model (implemented by OpenCV [9]), the results are shown in Figure 3. Compared with results of MOG, our results show better object detection and a clean edge, with lower false alarm rate, due to the color coherence within blocks, and a faster learning speed of block-based model.

Since our model groups pixels into blocks, the average color cluster number respect to pixel number is small. For a CIF video and block size 8x8, we assign the maximum color clusters per block to 100, this is usually enough and won't cause data shuffling (in fact, most block's maximum color cluster number is 50-70). We set the index array size for each pixel to 10, because the weights of colors that do not often appear will degrading very fast and become negligible, a small index array will not be affected by unstable backgrounds and foregrounds. Compared with MOG, our model saves 62% of the memory, see Table 2. For a CIF video, the total number of bytes required is only 7.7Mbytes, which can be fitted into small memory devices also.

Table 2. Memory usage compared with MOG.

MOG	Our method	Memory saved
200 byte/pixel	76 byte/pixel	62%

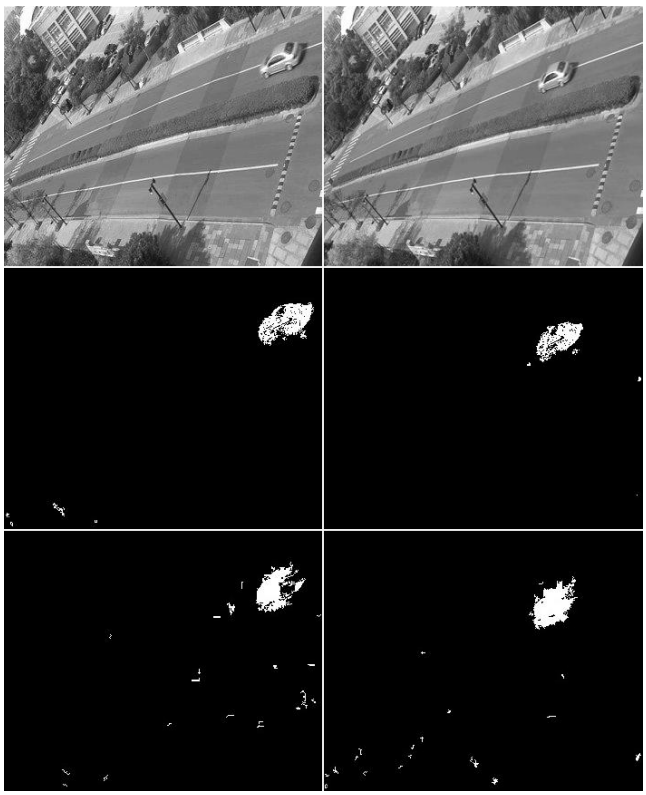

Figure 3. Proposed method compared with MOG (the first row is the original video in gray values, the second row is a binary mask produced by our method, the third row is the MOG method implemented by OpenCV)

We implemented our model in C++, compiled by Microsoft Visual C++ 2005, and runs on a Pentium M 1.8GHz laptop computer. We tested running speeds of our model and MOG, and made a comparison between the two, using standard MPEG-4 test sequence "Hall" and our surveillance video, which shows our model is generally much faster, see Table 3.

Table 3. Speed compared with MOG.

Sequence	MOG(fps)	Ours (fps)	Speed up
MPEG-4 Hall	13.2	18.9	43%
Surveillance	12.6	20.0	58%

4. CONCLUSION

In this paper, a novel block-based background model is proposed. The new model achieves the goal of reducing memory usage successfully without speed penalties. In scenes with dynamic background, the foreground detection result of our model is more accurate than the classic background model MOG, with effective foreground detection and low false alarm rate.

Currently, our model uses a full search method in block-based color clustering. In future studies, we will focus on developing faster clustering algorithms, utilizing weighted indexes of neighboring pixels as a clue to speedup the search. We will also explore more effective algorithms for color clustering and color weighting for more dynamic scenes.

5. REFERENCES

[1] C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland, "Pfinder: Real-Time Tracking of the Human Body", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, pp. 780-785, July 1997.

[2] C. Stauffer, W.E.L. Grimson, "Adaptive background mixture models for real-time tracking", IEEE Conference on Computer Vision and Pattern Recognition, 1999.

[3] A. Elgammal, D. Harwood, L. Davis, "Non-parametric model for background subtraction", Lecture Notes In Computer Science, Vol. 1843, pp. 751-767, 2000.

[4] K. Kim, T.H. Chalidabhongse, D. Harwood, L. Davis, "Real-time foreground-background segmentation using codebook model", Real-Time Imaging, Elsevier, No. 11, pp. 172-185, 2005.

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

[6] K. Toyama, J. Krumm, B. Brumitt, B. Meyers, "Wallflower: principles and practice of background maintaenance", IEEE International Conference on Computer Vision 1999.

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

[8] Y. Zhang, S.J. Kiselewich, W.A. Bauson, R. Hammoud, "Robust Moving Object Detection at Distance in the Visible Spectrum and Beyond Using A Moving Camera", IEEE Conference on Computer Vision and Pattern Recognition Workshop, 2006.

[9] OpenCV, http://www.intel.com/technology/computing/opencv/.