A Multi Resolution Algorithm for Real-Time Foreground Objects Detection based on Codebook

Mikaël Ange Mousse* — Bethel Atohoun**

* Institut Universitaire de Technologie
Université de Parakou, BENIN
mikael.mousse@univ-parakou.bj

** Ecole Supérieure de Gestion d'Informatique et des Sciences, BENIN
b.atohoun@esgis.bj

RÉSUMÉ. *A définir par la commande* `\resume{...}`

ABSTRACT. The object detection is first step for all automatic videosurveillance system. It is also one of the most interesting, well focused and well addressed but still challenging topic in computer vision. This paper introduced a new foreground- background segmentation method based on codebook. The main objective of our proposed method is to reduce the complexity of background modeling using codebook. For this, we proposed a background modeling based on superpixels. The number of superpixels is automatically update according to the size of the foreground objects detected. We use some frame-based metrics to evaluate the proposed method. Experimental results demonstrate the performance of our proposed approach.

MOTS-CLÉS : Mots clefs

KEYWORDS : Object detection, background modeling, Superpixels, Codebook

1. Introduction

In the field of computer vision, many algorithms about object detection exist with different purposes. These algorithms are subdivide in three categories : without background modeling, with background modeling and combined approach. The algorithm based on background modeling is recommended in case of dynamic background observed by a static camera. Many research works used this approach. One of them is Codebook based method proposed by Kim et al [1]. In this algorithm, Kim et al. represent each pixel p_t by using a codebook $\mathcal{C} = \{c_1, c_2, ..., c_L\}$. In this codebook, each codeword c_i, $i = 1, ..., L$ is represented by a RGB vector v_i and a 6-tuples $aux_i = \{\check{I}_i, \hat{I}_i, f_i, p_i, \lambda_i, q_i\}$ in where $\check{\text{I}}$ and $\hat{\text{I}}$ are the minimum and maximum brightness of all pixels assigned to this codeword c_i, f_i is the frequency at which the codeword has occurred, λ_i is the maximum negative run length defined as the longest interval during the training period that the codeword has not recurred, p_i and q_i are the first and last access times, respectively, that the codeword has occurred. The codebook model is created or updated using two criteria. The first criterion is based on color distortion (3) whereas the second is based on brightness distortion (4). Color distortion ($Colordist$) and brightness distortion (or pixel intensity I) are respectively computed by using expressions (1) and (2).

$$Colordist(p_t, c_i) = \sqrt{||p_t||^2 - C_p^2} \tag{1}$$

$$I = \sqrt{R^2 + G^2 + B^2} \tag{2}$$

$$Colordist(p_t, c_i) \leq \varepsilon_1 \tag{3}$$

$$I_{low} \leq I \leq I_{hi} \tag{4}$$

In (3), the autocorrelation value C_p^2 is given by expression (5) and $||p_t||^2$ is given by expression (6).

$$C_p^2 = \frac{(R_i R + G_i G + B_i B)^2}{R_i^2 + G_i^2 + B_i^2} \tag{5}$$

$$||p_t||^2 = R^2 + G^2 + B^2 \tag{6}$$

In relation (4), $I_{low} = \alpha \hat{I}_i$, $I_{hi} = min\{\beta \hat{I}, \frac{\check{I}}{\alpha}\}$.

After the training period, we extract the moving object based on obtained codebook model. If an incoming pixel matches to a codeword in the codebook, then this codeword is updated. If the pixel doesn't match, its information is put in cache word and this pixel is treated as a foreground pixel. The matched codeword is searched by using a condition based on color distortion (7) and brightness distortion (4).

$$Colordist(p_t, c_i) \leq \varepsilon_2 \tag{7}$$

The choice of parameters $(\varepsilon_1, \varepsilon_2, \alpha, \beta)$ is very important. A Discussion about its is made by Kim et al. [1] in their research paper. Due to the performance of this method, several researchers studies it in deeply.

Ilyas et al. [2] suggest the use of the maximum negative run length λ and the frequency f_i to decide whether to delete codewords or not. When the access frequency f_i is large, They propose to put cache codeword into the codebook. Their objective was to reduce

the size of the codebook while maintaining a good detection accuracy. Cheng et al. [3] suggest to convert pixels from RGB to YUV color space, and they use the V component to build a single Gaussian model. Shah et al. [4] propose a statistical parameter estimation method to control adaptation procedure whereas Pal et al. [5] spread codewords along boundaries of the neighboring layers. The purpose of their works [3, 5] was to improve the obtained background model. Doshi and Trivedi [6] propose a hybrid cone-cylinder model to build the background model. They use the V component in HSV representation of pixels to represent the brightness of these pixels. The purpose of the algorithm is to extract moving object in a sequence which contains shadow. Donghai et al. [7] propose an algorithm which overcomes the errors of the Gaussian mixture model, sphere model, and codebook cylinder model. Their background modeling algorithm is based on principal component analysis (PCA). Li et al. [10] suggest combining Gaussian mixture model and Codebook. Yu et al. [11] propose to use local binary pattern (LBP) for establishing of the first layer of background and combining it with codebook. Li et al. [16] build a texture-wise background model by LBP and used single Gaussian to model codewords. Mousse et al. [12] propose an algorithm based on combination of codebook with an edge detector. They use an edge detector to verify if foreground pixels detected by the codebook algorithm belong to an object or not. The goal of these algorithms [10, 16, 12] is to reduce the false positive pixels. They obtained a good performance but they can't be used in real time condition due to their complexities. Doshi and Trivedi [6] propose to convert pixel from RGB to HSV color space. Cheng et al. [3] suggest changing pixels information from RGB to YUV color space. Fang et al. [9] exploit HSL color space and use L component as brightness value to reduce amount of calculation. They also modeled the spatial dependencies between pixels. Mousse et al. [13] also use superpixels segmentation algorithm to model the spatial dependencies between pixels but they convert pixel from RGB to CIE L*a*b* color space and exploiting this color space specification. They also extended their work to reduce the complexity of the algorithm [14, 15].

This work also folows the logic of the reduction of the calculation of the process of moving object extraction. Firstly, we use SLIC algorithm to perfom superpixels segmentation. The main contribution of this paper is the automatic calculation of the optimal number of pixels clusters. This number is obtained in function of the size of detected objects.

2. Proposed Algorithm

In this section, we present our algorithm based on codebook for foreground object detection. Like all background modeling algorithms, the proposed algorithm works in two phases: the learning phase and the moving objects extraction phase. The contribution of this work is done in the second phase. The learning phase is same than the learning phase of our past work [13] whereas unlike other approaches (Fang et al. [9] and Mousse et al. [13]) of the state of the art which perform a native subtraction method, in this paper, the second phase proposes an adaptive method to extract foreground maps. The section consists of three subsections. The first subsection presents the superpixel segmentation strategy the second subsection presents our background modeling method and the third subsection presents our moving pixels extraction strategy.

2.1. Superpixel Segmentation using Improved Simple Linear Iterative Clustering

Superpixel segmentation has become a popular preprocessing step in computer vision. It allows to represent an image with only a couple of hundred segments that function as atomic building blocks instead of tens of thousands of pixels. We used Improved Simple Linear Iterative Clustering algorithm (SLIC) to create pixels clusters (superpixels).

There are few algorithms that output the desired number of regular, compact superpixels with a low computational overhead. But the Improved Simple Linear Iterative Clustering algorithm introduced by Schick et al. [**?**] have better performance than existing methods. In fact, Schick et al. [**?**] prove the efficacy of this superpixels segmentation in object category recognition and medical image segmentation. They obtain better quality and higher computational efficiency when they compared it to other state-of-the-art algorithms. The detailed SLIC algorithm is given by algorithm 1.

Algorithm 1: SLIC algorithm for superpixels segmentation

```
Initialize cluster centers C_k = [l_k, a_k, b_k, x_k, y_k]^T by sampling pixels at regular
  grid steps S.
Perturb cluster centers in an n * n neighborhood, to the lowest gradient position
  using expression (8).
repeat
    for each cluster center C_k do
        Assign the best matching pixels from a 2S × 2S square neighbourhood
          around cluster center according to the distance measure (using
          expression (9)).
    Compute new cluster centers and residual error E {L1 distance between
      previous centers and recomputed centers}.
until E <= threshold
Enforce connectivity.
```

In algorithm 1, we assumed that the size of video frame is $N \times M$, and we segment it into K superpixels. Each superpixel approximately has $\frac{N \times M}{K}$ pixels and the central region is approximately $S = \sqrt{\frac{N \times M}{K}}$.

$$\begin{aligned} G(x,y) &= ||I(x+1,y) - I(x-1,y)||^2 \\ &\quad + ||I(x,y+1) - I(x,y-1)||^2 \end{aligned} \tag{8}$$

$$\begin{aligned} d_{lab} &= \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \\ d_{xy} &= \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \\ D_s &= d_{lab} + \frac{m}{S} d_{xy} \end{aligned} \tag{9}$$

In (8), (9), d_{lab} is the lab distance, d_{xy} is the plane distance and $I(x,y)$ is the lab vector corresponding to the pixel at position (x,y). Schick et al. [**?**] prove the efficacy of SLIC algorithm in image segmentation. They obtained better quality and higher computational efficiency than other state-of-the-art superpixels segmentation algorithms.

The using of superpixels reduces the complexity of background modeling using codebook with respect to the amount of data. We segment the first input frame to obtain the frontiers of each superpixel. These frontiers are used on all input frames. Then we have the same number of superpixels in all frames of the sequence and each superpixel has the same number of pixels from a frame to another.

2.2. Background Modeling

After the extraction of the K superpixels, we build a codebook background model based on these superpixels. Let $P = \{s_1, s_2, ..., s_k\}$ represents these superpixels. Each superpixel $s_j, j \in \{1, 2, ..., k\}$ is composed approximately by m pixels. With each superpixel we built a codebook $C = \{c_1, c_2, ..., c_L\}$. The obtained codebook which contains L codewords $c_i, i \in \{1, 2, ..., L\}$ and each codewords c_i is composed by an vector $v_i = (\bar{a}_i, \bar{b}_i)$ and 6-tuples $aux_i = \{\check{L}_i, \hat{L}_i, f_i, p_i, \lambda_i, q_i\}$ in which $\check{L}_i$, $\hat{L}_i$ are the minimum and maximum of luminance value, f_i is the frequency at which the codeword has occurred, λ_i is the maximum negative run length defined as the longest interval during the training period that the codeword has not recurred, p_i and q_i are the first and last access times, respectively, that the codeword has occurred. $\bar{L}$, $\bar{a}$, $\bar{b}$ are respectively the average value of component L*, a* and b* of the pixels in a superpixel. We compute the color distortion by replacing (10) and (11) into (1). For the brightness distortion degree (2) we use $\bar{L}$ value as the intensity of the superpixel.

$$p_t = \bar{a}^2 + \bar{b}^2 \tag{10}$$

$$C_p^2 = \frac{(\bar{a}_i\bar{a} + \bar{b}_i\bar{b})^2}{\bar{a}_i^2 + \bar{b}_i^2} \tag{11}$$

For each superpixel, if we find a codeword c_i which respect these two criteria (brightness distortion criterion and color distortion criterion) then we update this codeword by setting v_i to $(\frac{f_i\bar{a}_i+\bar{a}}{f_i+1}, \frac{f_i\bar{b}_i+\bar{b}}{f_i+1})$ and aux_L to $\{min(\bar{L}, \check{L}_i), max(\bar{L}, \hat{L}_i), f_i + 1, max(\lambda_i, t - q_i), p_i, t\}$. If we don't find a matched codeword, we create a new codeword c_K. In this case, v_K is equal to $(\bar{a}, \bar{b})$ and aux_K is equal to $\{\bar{L}, \bar{L}, 1, t - 1, t, t\}$. The detailed algorithm is given by Algorithm 2. In this algorithm :

– N is the number of frame that we used to model background;

– The variable which is the subject of the condition in the loop "for" is automatically incremented at the end of the loop.

2.3. Foreground pixel extraction

After the building of the background model, we extract foreground pixels. Like the background modeling step, this operation is also based on the superpixels. Using the first incoming frame, we perform the native subtraction algorithm which is defined by Algorithm 3. In this algorithm, ϵ_2 is the detection threshold and the variable which is the subject of the condition in the loop "for" is automatically incremented at the end of the loop. If there is no acceptable matching codeword the superpixel is detected as foreground. Else if we find an acceptable matching codeword the superpixel is classified as background and the corresponding codeword is updated.

Algorithm 2: Background modeling

$l \leftarrow 0$
for $t = 1$ *to N* **do**
  Segment frame F_t into superpixels
  for *each superpixels* Su_k *of frame* F_t **do**
    $p_t(\bar{L}, \bar{a}, \bar{b})$
    Find the matched codeword c_i in codebook matching to Su_k based on two conditions (a) and (b).
      (a) colordist $(p_t, v_i) <= \epsilon_1$
      (b) (brightness ($\bar{L}$, $\check{L}_i$, $\hat{L}_i$)) = true
    if $l = 0$ *or there is no match* **then**
      $l \leftarrow l + 1$
      create codeword c_L by setting parameter
      $v_L \leftarrow (\bar{a}, \bar{b})$ and $aux_L \leftarrow \{\bar{L}, \bar{L}, 1, t-1, t, t\}$
    else
      update codeword c_i by setting $v_i \leftarrow (\frac{f_i\bar{a}_i+\bar{a}}{f_i+1}, \frac{f_i\bar{b}_i+\bar{b}}{f_i+1})$ and
      $aux_i \leftarrow \{min(\bar{L}, \check{L}_i), max(\bar{L}, \hat{L}_i), f_i + 1, max(\lambda_i, t - q_i), p_i, t\}$
for *each codeword* c_i **do**
  $\lambda_i \leftarrow max\{\lambda_i, ((m \times n \times t) - q_i + p_i - 1)\}$

3. Experimental Results and Performance Evaluation

In this part, we present the performance of the proposed approach by comparing it with other enhancement of codebook algorithm. The section consists on two subsections. The first subsection presents the experimental results, the second presents and analyzes the performance results.

3.1. Experimental Results

We have selected two public video sequences. These sequences are widely used to test moving object detection algorithm [17]. They are ?fall? and ?boats? datasets. The programming language is C++ with the framework OpenCV. In our implementation if the size of input frame is (M × N) then we construct firstly $\frac{M \times N}{50}$ superpixels. The foreground object segmentation results are presented in Fig. 1 and Fig. 2.

Algorithm 3: Foreground Pixel Extraction

$Su_k(\bar{L}, \bar{a}, \bar{b})$
for *all codewords* **do**
  find the codeword c_m matching to Su_k based on :
    (a) colordist $(p_t, v_m) <= \epsilon_2$
    (b) (brightness ($\bar{L}$, $\check{L}_m$, $\hat{L}_m$)) = true
  Update the matched codeword as in Step 11 in the
  algorithm of background modeling (Algorithm 2).

$$BGS(Su_k) = \begin{cases} \textbf{foreground} & \text{if there is no match} \\ \textbf{background} & \text{otherwise} \end{cases}$$

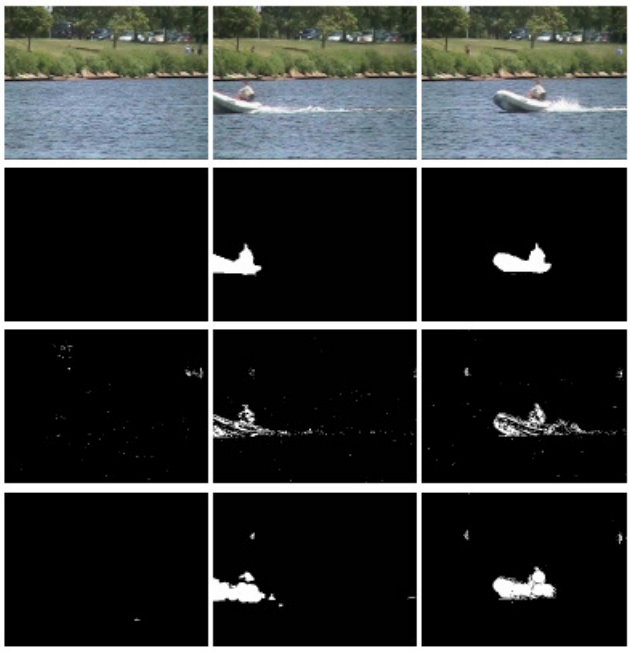

Figure 1. *?boats? video sequence segmentation results. The first row shows the original images. The second row shows the ground truth. The third row shows the detected results by [1], and the last row shows the detected results by our proposed algorithm.*

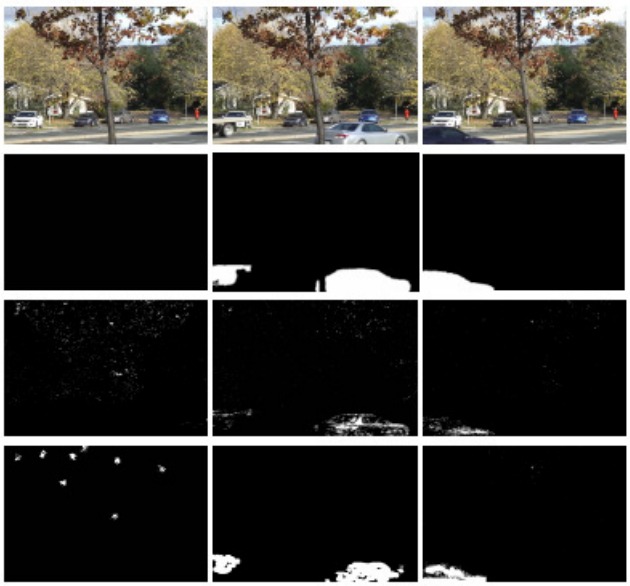

Figure 2. *?fall? video sequence segmentation results. The first row shows the original images. The second row shows the ground truth. The third row shows the detected results by [1], and the last row shows the detected results by our proposed algorithm.*

Tableau 1. *Different metrics according to experiments with "boats" dataset.*

	CB	CB_HSV	CB_HSL	CB_YUV	CB_LAB	Proposed
FPR	0.23	0.25	0.21	0.27	**0.21**	**0.21**
PR	0.87	0.86	0.89	0.80	**0.91**	**0.91**
FM	0.60	0.62	0.64	0.65	**0.66**	**0.66**

Tableau 2. *Different metrics according to experiments with "fall" dataset*

	CB	CB_HSV	CB_HSL	CB_YUV	CB_LAB	Proposed
FPR	0.31	0.33	0.25	0.38	**0.23**	**0.23**
PR	0.56	0.60	0.63	0.41	**0.67**	**0.67**
FM	0.41	0.47	0.51	0.43	**0.54**	**0.54**

3.2. Performance Evaluation

In order to evaluate the performance of our proposed method, we use an evaluation based on ground truth. The ground truth has been obtained by manually labeling foreground objects in the original frame. The ground truth based metrics are : true negative (TN), true positive (TP), false negative (FN) and false positive (FP). We use these metrics to compute other parameters for the evaluation of the algorithm. These parameters are false positive rate (FPR), true positive rate (TPR), precision (PR) and F-measure (FM). These parameters are respectively computed using expressions (12), (13), (14) and (15).

$$FPR = 1 - \frac{TN}{TN + FP} \tag{12}$$

$$PR = \frac{TP}{TP + FP} \tag{13}$$

$$TPR = \frac{TP}{TP + FN} \tag{14}$$

$$FM = \frac{2 \times PR \times TPR}{PR + TPR} \tag{15}$$

The obtained results are presented in Table 1 and Table 2. In these tables, we prove that the accuracy of the proposed system is the same (compared to Mousse et al. [15]) or close to the accuracy of other codebook based algorithm enhancements. In these tables, CB refers to the method proposed by Kim et al., and CB_HSV refers to the method suggested by Doshi and Trivedi, CB_HSL refers to the algorithm proposed by Fang et al., CB_YUV refers to the algorithm suggested by Cheng et al., and CB_LAB refers to Mousse et al. [15].

4. Conclusion

In this work, we propose a new method to segment moving objects using codebook background model. The proposed method models the background using a cluster of pixels. The number of cluster varies according to the time and size of the objects presented

on the sequence. The experiment results obtained using public sequences proove that our proposed approach outperforms other algorithms in calculation cost and has a competitive accuracy.

5. Bibliographie

K. Kim, T. H. Chalidabhonse, D. Harwood, and L. Davis, “Real-time foreground-background segmentation using codebook model”, Real-Time Imaging, 11(3):172?185, 2005.

A. Ilyas, M. Scuturici, S. Miguet, “Real time foreground-background segmentation using a modified codebook model”, International Conference on Advanced Video and Signal Based Surveillance, 2009, USA, pp. 454-459.

X. Cheng, T. Zheng, L. Renfa, “A fast motion detection method based on improved codebook model”, Journal of Computer and Development, 2010, vol. 47, pp. 2149-2156.

M. Shah, J. D. Deng, B. J. Woodford, “Enhanced codebook model for real-time background subtraction”, ICONIP(3) 2011, pp. 449-458.

A. Pal, G. Schaefer, M. E. Celebi. “Robust codebook-based video background subtraction”, ICASSP 2010,pp. 1146-1149.

A Doshi, M. M. Trivedi, “Hybrid cone-cylinder codebook model for foreground detection with shadow and highlight suppression”, AVSS, 2006, vol. 19, pp. 121-133.

H. Donghai, Y. Dan, Z. Xiaohong, H. Mingjian, “Principal Component Analysis Based Codebook Background Modeling Algorithm”, Acta Automatica Sinica, 2012, vol. 38, pp. 591-600.

M. Wu, X. Peng, “Spatio-temporal context for codebook- based dynamic background subtraction”, International Journal of Electronics and Communications, 2010, Vol. 64, pp. 739-747.

X. Fang, C. Liu, S. Gong, Y. Ji, “Object Detection in Dynamic Scenes Based on Codebook with Superpixels”, Asian Conference on Pattern Recognition, 2013, pp. 430-434.

Y. Li, F. Chen, W. Xu, Y. Du, “Gaussian-based codebook model for video background subtraction”, Lecture Notes in Computer Science, 2006, Vol. 4222, pp. 762-765.

W. Yu, D. Zeng, H. Li, “Layered video objects detection based on LBP and codebook”, ETCS, 2009, pp. 207-213.

M. A. Mousse, E. C. Ezin, and C. Motamed, “Foreground-background segmentation based on codebook and edge detector”, In Proceedings of the 10th International Conference on Signal Image Technology & Internet Based Systems, 2014.

M. A. Mousse, C. Motamed, and E. C. Ezin, “Fast moving object detection from overlapping cameras”, In Proceedings of the International Conference on Informatics in Control, Automation and Robotics, pages 296?303, 2015.

M. A. Mousse, C. Motamed, E. C. Ezin, "An Adaptive Algorithm for Fast Moving Object Detection from Dynamic Background based on Cobebook", 4th International Conference on Robotics and Mechatronics, 2016.

M. A. Mousse, C. Motamed, and E. C. Ezin, “Enhanced codebook algorithm for fast moving object detection from dynamic background using scene visual perception”, Journal of Electronic Imaging 25(6), pages : 061618-1:061618-9, DOI : 10.1117/1.JEI.25.6.061618, SPIE, 2016.

B. Li, Z. Tang, B. Yuan, Z. Miao, “Segmentation of moving foreground objects using codebook and local binary patterns”, CISP, 2008, 239-243.

N. Goyette, P. M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, ?Changedetection.net: A new change detection benchmark dataset,? in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2012, pp. 1–8.