Journal of
Electronic Imaging

JElectronicImaging.org

Enhanced codebook algorithm for fast moving object detection from dynamic background using scene visual perception

Mikaël A. Mousse
Cina Motamed
Eugène C. Ezin

Mikaël A. Mousse, Cina Motamed, Eugène C. Ezin, "Enhanced codebook algorithm for fast moving object detection from dynamic background using scene visual perception," *J. Electron. Imaging* **25**(6), 061618 (2016), doi: 10.1117/1.JEI.25.6.061618.

Enhanced codebook algorithm for fast moving object detection from dynamic background using scene visual perception

Mikaël A. Mousse,[a,b,*] **Cina Motamed,**[a] **and Eugène C. Ezin**[b]
[a]Université Littoral Côte d'Opale, EA 4491, Laboratoire d'Informatique Signal et Image de la Côte d'Opale (LISIC), 50 rue F. Buisson, BP 719, 62228 Calais, France
[b]Unité de Recherche en Informatique et Sciences Appliquées, Institut de Mathématiques et de Sciences Physiques, Université d'Abomey-Calavi, BP 613 Porto-Novo, Benin

Abstract. The detection of moving objects in a video sequence is the first step in an automatic video surveillance system. This work proposes an enhancement of a codebook-based algorithm for moving objects extraction. The proposed algorithm used a perceptual-based approach to optimize foreground information extraction complexity by using a modified codebook algorithm. The purpose of the adaptive strategy is to reduce the computational complexity of the foreground detection algorithm while maintaining its global accuracy. In this algorithm, we use a superpixels segmentation approach to model the spatial dependencies between pixels. The processing of the superpixels is controlled to focus it on the superpixels that are near to the possible location of foreground objects. The performance of the proposed algorithm is evaluated and compared to other algorithms of the state of the art using a public dataset that proposes sequences with a dynamic background. Experimental results prove that the proposed algorithm obtained the best the frame processing rate during the foreground detection. *© 2016 SPIE and IS&T* [DOI: 10.1117/1.JEI.25.6.061618]

Keywords: perceptual information; moving object detection; background model; codebook.

Paper 16372SS received May 1, 2016; accepted for publication Nov. 14, 2016; published online Dec. 1, 2016.

1 Introduction

In video surveillance, the video sequences are filmed by static or nonstatic cameras and they contain moving objects on a fixed or dynamic background. The first step of any intelligent video surveillance system is the extraction of moving objects from the background. Then, it is important to have a competitive foreground extraction module. Many techniques are used to extract moving objects. One of them is background modeling. Background modeling is a widely used technique to extract objects from a dynamic background. It consists of modeling the background using pixel values from previous frames. Every image pixel is matched with its corresponding component in the background model. Background modeling techniques are considered as a good choice for their better performance for an indoor and/or outdoor scene. One of these methods is the mixture of a Gaussian (MoG)-based approach. Stauffer and Grimson[1,2] proposed this method. They proposed to use multiple Gaussian distributions to represent each pixel in background modeling. According to their experiments, Stauffer and Grimson[2] proved that this method overcomes nonstationary background and thus provides a better adaptation for background modeling. Elgammal et al.[3] proposed an algorithm based on kernel density estimation (KDE). It is proved that KDE is closer to the real probability distribution than MoG. Rittscher et al.[4] proposed a probabilistic background modeling algorithm based on hidden Markov models (HMM). They discuss several state splitting criteria to model background. Stenger et al.[5] also suggested using HMM for background modeling process and proposed a nonadaptive three-state HMM to perform it. Kim et al.[6] proposed a real-time foreground–background segmentation using codebook model. The overview of this algorithm is presented in Fig. 1. This method has a competitive performance. In this work, we propose an extension of Codebook method. The main objective of this work is to propose a fast Codebook-based algorithm with competitive performance. The rest of this paper is organized as follows. In Sec. 2, we will present the moving object detection using Codebook model and will review some codebook-based approaches, whereas Sec. 3 presents the moving object detection approach proposed. The experimental results and analysis are presented, respectively, in Secs. 4 and 5. Finally, we conclude this paper in Sec. 6.

2 Moving Object Detection Using Codebook

This section presents the Codebook algorithm for moving object extraction. In this algorithm, Kim et al.[6] represented each pixel p_t by using a codebook $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$. In this codebook, each codeword c_i, $i = 1, \ldots, L$, is represented by an RGB vector v_i and a 6-tuples $\text{aux}_i = \{\check{I}_i, \hat{I}_i, f_i, p_i, \lambda_i, q_i\}$ in which $\check{I}$ and $\hat{I}$ 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, and 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 [Eq. (3)], whereas the

*Address all correspondence to: Mikaël A. Mousse, E-mail: mikael.mousse@lisic.univ-littoral.fr

1017-9909/2016/$25.00 © 2016 SPIE and IS&T

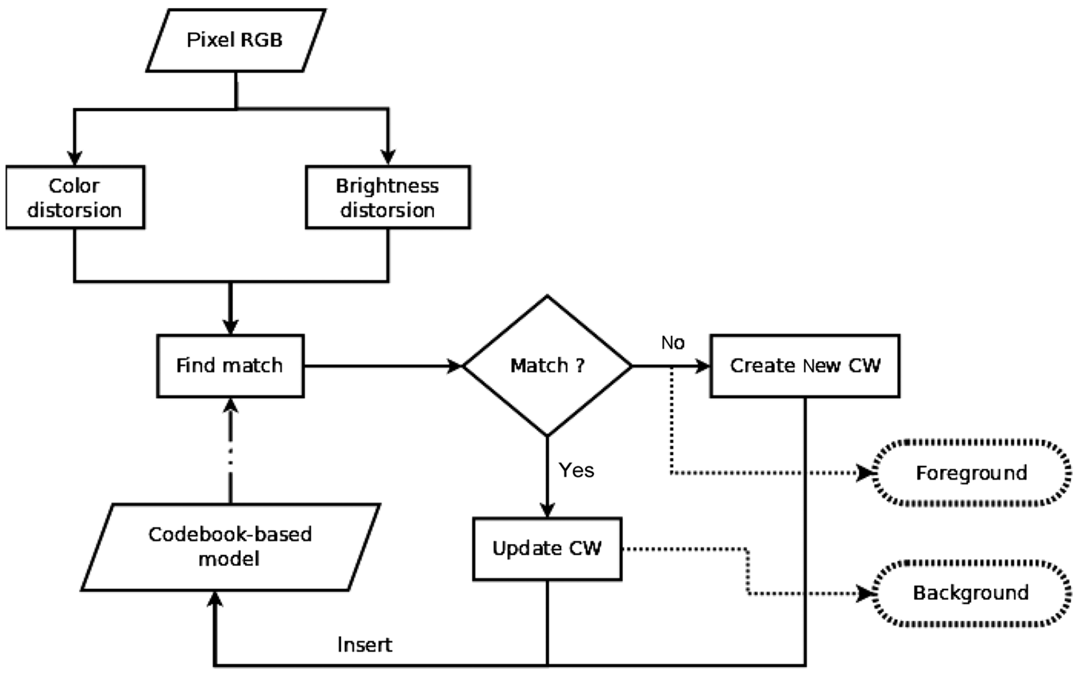

Fig. 1 Flow diagram of the codebook-based algorithm. (The solid lines correspond to the training phase and dashed lines correspond to the phase of moving objects detection).

second is based on brightness distortion [Eq. (4)]. Color distortion (Colordist) and brightness distortion (or pixel intensity I) are, respectively, computed by using the following equations:

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

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

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

$$I_{\text{low}} \le I \le I_{hi}. \tag{4}$$

In Eq. (3), the autocorrelation value C_p^2 is given by Eq. (5) and $\|p_t\|^2$ is given by Eq. (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 Eq. (4), $I_{\text{low}} = \alpha \hat{I}_i$, $I_{hi} = \min\{\beta \hat{I}, \frac{\check{I}}{\alpha}\}$.

After the training period, we extract the moving object based on the obtained codebook model. If an incoming pixel matches a codeword in the codebook, then this codeword is updated. If the pixel does not 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 [Eq. (7)] and brightness distortion [Eq. (4)]

$$\text{Colordist}(p_t, c_i) \le \varepsilon_2. \tag{7}$$

The choice of parameters $(\varepsilon_1, \varepsilon_2, \alpha, \beta)$ is very important. A discussion about it is included in a research paper by Kim et al.[6] Due to the performance of this method, several researchers studied it deeply. Guo et al.[7] extended the algorithm based on the codebook by proposing a hierarchical scheme to improve its efficiency. Ilyas et al.[8] suggested 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 proposed to put a cache codeword into the codebook. Their objective was to reduce the size of the codebook while maintaining a good detection accuracy. Cheng et al.[9] suggested to convert pixels from RGB to YUV color space, and they use the V component to build a single Gaussian model. Shah et al.[10] proposed a statistical parameter estimation method to control the adaptation procedure, whereas Pal et al.[11] spread codewords along boundaries of the neighboring layers. The purpose of their works[9,11] was to improve the obtained background model. Doshi and Trivedi[12] proposed 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 objects in a sequence that contains shadow. Donghai et al.[13] proposed an algorithm that 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.[14] suggested combining Gaussian the mixture model and codebook. Yu et al.[15] proposed to use a local binary pattern (LBP) for establishing the first layer of background and combining it with the codebook. Li et al.[16] built a texture-wise background model by LBP and used single Gaussian to model codewords. Mousse et al.[17] proposed an algorithm based on combination of the codebook with an edge detector. They used an edge detector to verify if foreground pixels detected by the codebook algorithm belong to an object or not. The goal of these algorithms[14,16,17] is to reduce the false positive pixels. They obtained a good performance but they cannot be used in a real-time condition due to their

Algorithm 1 SLIC algorithm for superpixels segmentation.

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

complexities. Doshi and Trivedi[12] proposed to convert pixels from RGB to HSV color space. Cheng et al.[9] suggested changing pixel information from RGB to YUV color space. Fang et al.[18] exploited HSL color space and used the L component as a brightness value to reduce the amount of calculation. They also used the improved simple linear iterative clustering (SLIC) algorithm to model the spatial dependencies between pixels. Mousse et al.[19] also used the SLIC algorithm to model the spatial dependencies between pixels but they converted the pixel from RGB to CIE $L * a * b*$ color space and exploited this color space specification. Due to the use of superpixels, Fang et al.[18] and Mousse et al.[19] proposed the fastest methods among all the improvements of the codebook.

The main objective of this work is to propose a foreground pixel extraction method based on the codebook. This algorithm must have competitive performance and less computational complexity. For this, we exploit our past research work presented in Mousse et al.[19] because it has an acceptable moving object detection accuracy and is used in a video surveillance application[20] with good results. So the objective of this work is to reduce the complexity of this algorithm while keeping its accuracy. Then, unlike our prior algorithm and Fang et al.[18] algorithms that perform a native superpixel subtraction, we suggest an adaptive approach to manage the distribution of superpixels. In this strategy, we exploit the perceptual image characteristic to reduce the number of superpixels that will be treated during the foreground objects extraction process. The purpose of this adaptive strategy is to manage clusters efficiently and thereby reduce the complexity of the detection algorithm while maintaining the performance of the previous algorithm.

3 Modified Codebook Model for Object Detection

In this section, we present our algorithm based on the 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 as the learning phase of our past work,[19] whereas unlike other approaches (Fang et al.[18] and Mousse et al.[19]) 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. Section 3.1 presents the superpixel segmentation strategy, Sec. 3.2 presents our background modeling method, and Sec. 3.3 presents our moving pixels extraction strategy.

3.1 *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 SLIC to create pixel clusters (superpixels).

There are few algorithms that output the desired number of regular, compact superpixels with a low computational overhead. But the improved SLIC algorithm introduced by Schick et al.[21] has better performance than existing methods. In fact, Schick et al.[21] proved the efficacy of this superpixels segmentation in object category recognition and medical image segmentation. They obtained 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.

In Algorithm 1, we assumed that the size of a 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 $S = \sqrt{\frac{N \times M}{K}}$.

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

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

In Eqs. (8) and (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.[21] proved the efficacy of the SLIC algorithm in image segmentation. They obtained better quality and higher computational efficiency than other state-of-the-art superpixels segmentation algorithms.

In this work, for each input image, we construct $\frac{M \times N}{50}$ superpixels. The using of superpixels reduces the complexity of background modeling using the 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 one frame to another.

3.2 *Background Modeling*

After the extraction of the K superpixels, we build a codebook background model based on these superpixels. Let

$P = \{s_1, s_2, \dots, s_k\}$ represent these superpixels. Each superpixel s_j, $j \in \{1,2,\dots,k\}$ is composed approximately by m pixels. With each superpixel, we built a codebook $C = \{c_1, c_2, \dots, c_L\}$. The obtained codebook contains L codewords c_i, $i \in \{1,2,\dots,L\}$ and each codeword c_i is composed by a vector $v_i = (\bar{a}_i, \bar{b}_i)$ and 6-tuples $\text{aux}_i = \{\check{L}_i, \hat{L}_i, f_i, p_i, \lambda_i, q_i\}$ 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, and 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 Eqs. (10) and (11) into Eq. (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 that respects 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 do not 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 frames that we used to model background.
- The variable that is the subject of the condition in the loop "for" is automatically incremented at the end of the loop.

3.3 *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 that 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. Otherwise, if we find an acceptable matching codeword, the superpixel is classified as background and the corresponding codeword is updated.

With the following frames, we use an adaptive strategy. To perform it, after the extraction of the superpixels, we classify the superpixels in two groups as shown in Fig. 2. The first group of superpixels is those that are on the border of the frame. For these superpixels, we extract foreground objects by subtracting them from the corresponding superpixels in the background model by using Algorithm 3. This part of the process allows us to detect objects that enter the scene.

After that, we focus on the superpixels of the second group. This group is composed of the superpixels that are not on the border of the frame. We also divide this group into two subgroups. The first subgroup contains the relevant superpixels, whereas the second contains the superpixels that are not important for our system. We consider that a superpixel is a relevant superpixel if it meets one of the three following conditions.

- The superpixel is adjacent to the background superpixels that belong to the first group.
- The superpixel belongs to the second group and is the foreground superpixel in the previous frame.
- The superpixel is adjacent to foreground superpixels in the previous frame.

All superpixels that do not meet the conditions listed in the last two sentences are considered as superpixels of the second subgroup. We use Algorithm 3 to check if the first subgroup superpixels are foreground superpixels or not. Using the results of this operation, we include in the first subgroup the superpixels that are adjacent to new foreground superpixels detected. These superpixels are also used to verify whether they were foreground superpixels or not. If any of these superpixels is a foreground superpixel, we will repeat the process of testing its adjacent superpixels. This repetition is done until the result of the foreground/background

Algorithm 2 Background modeling.

```
1 $l \leftarrow 0$
2 for $t = 1$ to $N$ do
3   Segment frame $F_t$ into superpixels
4   for each superpixels $Su_k$ of frame $F_t$ do
5     $p_t(\bar{L}, \bar{a}, \bar{b})$
6     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
7     if $l = 0$ or there is no match then
8       $l \leftarrow l + 1$
9       create codeword $c_L$ by setting parameter
        $v_L \leftarrow (\bar{a}, \bar{b})$ and $\text{aux}_L \leftarrow \{\bar{L}, \bar{L}, 1, t-1, t, t\}$
10  else
11    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
      $\text{aux}_i \leftarrow \{\min(\bar{L}, \check{L}_i), \max(\bar{L}, \hat{L}_i), f_i + 1, \max(\lambda_i, t - q_i), p_i, t\}$
12 for each codeword $c_i$ do
13 $\lambda_i \leftarrow \max\{\lambda_i, [(m \times n \times t) - q_i + p_i - 1]\}$
```

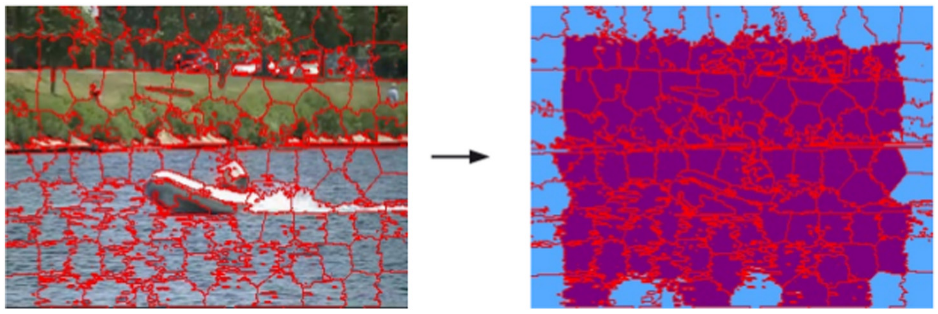

Fig. 2 Superpixels classification.

Algorithm 3 Foreground Pixel Extraction.

```
1 $Su_k(\bar{L}, \bar{a}, \bar{b})$
2 for all codewords do
3   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).
4   $\mathrm{BGS}(Su_k) = \begin{cases} \textbf{foreground} & \text{if there is no match} \\ \textbf{background} & \text{otherwise} \end{cases}$
```

checking does not provide foreground superpixels. This strategy is resumed in Algorithm 4.

Using this strategy, we do not use necessarily all superpixels but we only focus the process on the significant superpixels. The relevant superpixels depend on the evolution of the foreground visual information.

4 Experimental Results

In this section, we present the experimental environment and results. We have selected public benchmarking sequences for the validation of the proposed approach. These sequences have dynamic background and are covered under the work done by Goyette et al.[22] and available from Ref. [23]. For this purpose, we choose "boats," "canoe," "fountain01," "fountain02," "overpass," and "fall" sequences. The first two sequences represent boats on shimmering water. The sequences "fountain01" and "fountain02" show cars passing next to a fountain, whereas and the last two depict pedestrians, cars, and trucks passing in front of a tree shaken by the wind. These sequences present a sequence with dynamic background and are made available to researchers for the evaluation of moving object detection algorithms. Table 1 presents the number of frames (frame used for training and frame used for testing) of each sequence and the size of the frames of the sequences.

Algorithm 4 Proposed strategy.

```
1 $I \leftarrow \varnothing$
2 insert to $I$ the border superpixels
3 insert into $I$ the superpixels that are adjacent to foreground
  superpixels from the last frame.
4 while $I$ is not empty do
5   Use Algorithm 3 to verify if the superpixels that are stored in $I$ are
    foreground or background
6   $I \leftarrow \varnothing$
7   insert into $I$ the relevant superpixels that are adjacent to
    foreground superpixels detected at step 5.
```

The algorithm is implemented on a laptop that has the following characteristics: Intel Core i5 CPU L 640 at 2.13 GHz $\times$ 4 processor with 4 GB memory and the programming language is C++ with the OpenCv library. The parameters of superpixels segmentation algorithm are given by Schick et al.[21] In our implementation, if the size of input frame is $(M \times N)$ then we construct $\frac{M \times N}{50}$ superpixels as suggested by Mousse et al.[19] Some segmentation results are presented in Figs. 3 and 4.

5 Performance Evaluation and Discussion

In this part, we show the performance of the proposed approach by comparing with standard codebook algorithm proposed by Kim et al.[6] and with other extensions of the codebook algorithm based on an extension on pixels. All algorithms are implemented using the laptop described in

Table 1 Testing sequences.

	Number of frames		
Datasets	Training	Testing	Size
Boats	1900	6099	320×240
Canoe	800	389	320×240
Fountain01	400	784	432×288
Fountain02	500	999	432×288
Overpass	1000	2000	320×240
Fall	1000	3000	720×480

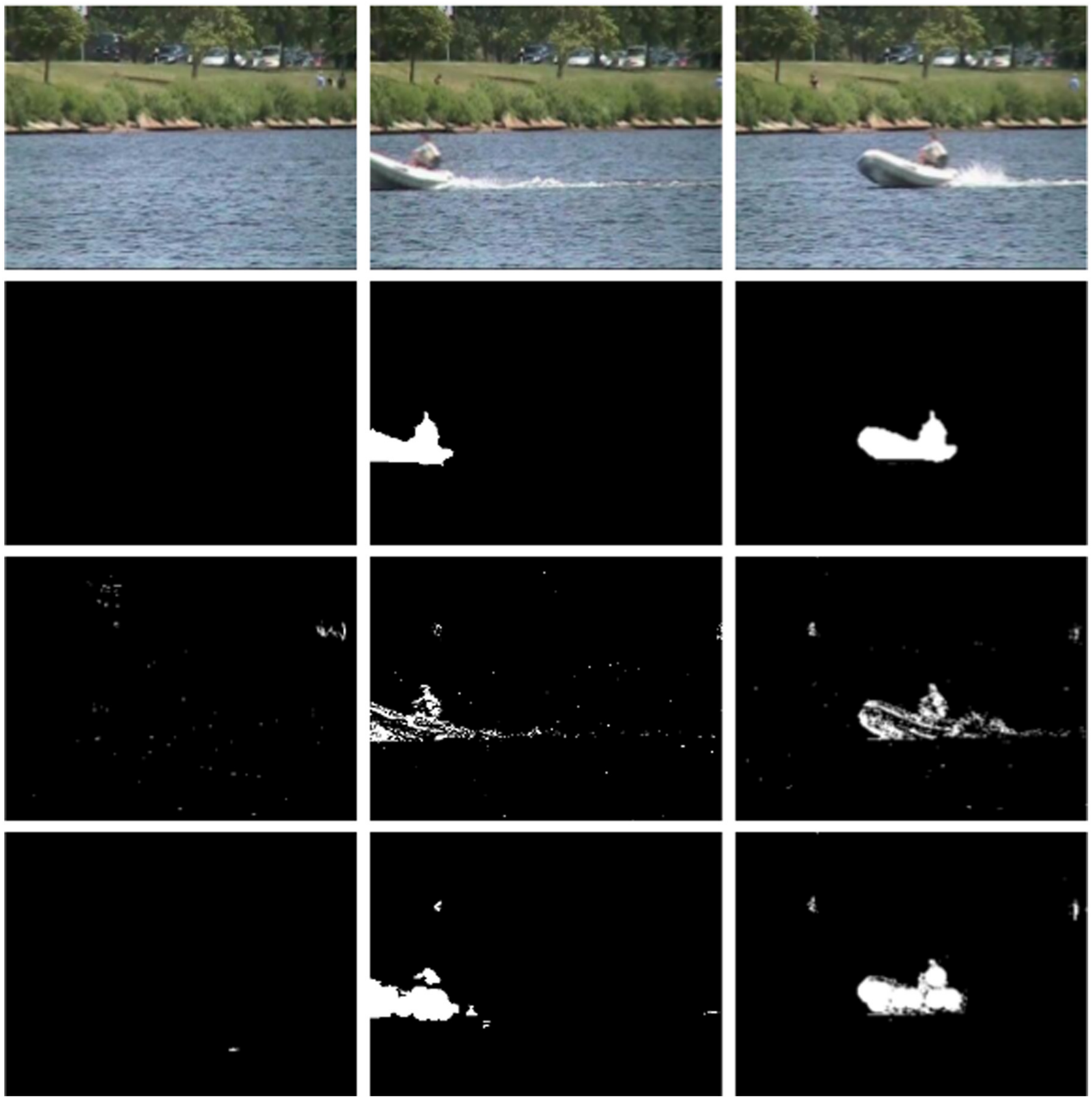

Fig. 3 “Boats” dataset segmentation results. The first row shows the original images. The second row shows the ground truth. The third row shows the detected results in Ref. 6 and the last row shows the detected results by our proposed algorithm.

Sec. 4. For the performance evaluation of our moving objects detection approach, we used some quantitative frame-based metrics. These metrics are based on true negative (TN), true positive (TP), false negative (FN), and false positive (FP) as shown in Table 2. According to this table:

- A pixel is a true negative pixel when both ground truth and system result agree on the absence of an object.
- A pixel is a true positive pixel when ground truth and system agree on the presence of an object.
- A pixel is a false negative (FN) when system result agrees on an absence of an object whereas ground truth agrees of the presence of an object.
- A pixel is a false positive (FP) when the system result agrees with the presence of an object whereas ground truth agrees with the absence of an object.

With these values, we calculate:

- False positive rate (FPR) using the following equation:

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

- True positive rate (TPR) using the following equation:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{13}$$

- F-measure (FM) using the following equation:

$$\mathrm{FM} = \frac{2 \times \mathrm{PR} \times \mathrm{TPR}}{\mathrm{PR} + \mathrm{TPR}}, \tag{14}$$

with

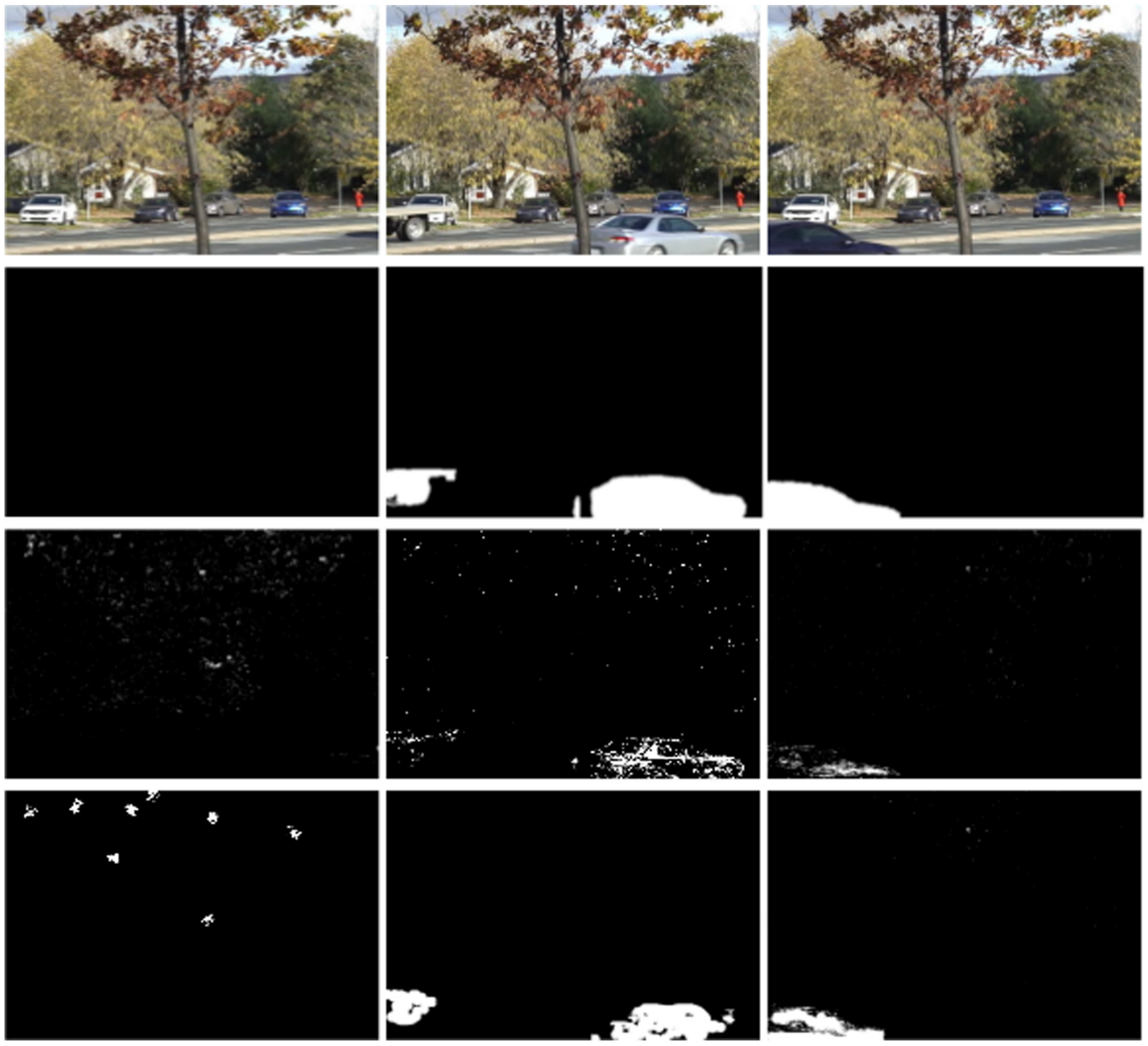

Fig. 4 "Fall" dataset segmentation results. The first row shows the original images. The second row shows the ground truth. The third row shows the detected results in Ref. 6 and the last row shows the detected results by our proposed algorithm.

$$PR = \frac{TP}{TP + FP}. \tag{15}$$

FPR allows us to know the number of false alarms that are sent by the algorithm. With TPR, we calculate the percentage of true positive pixels that have been detected. PR allows us to estimate the precision of object detection and FM represents a harmonic mean of precision and TPR. It measures the accuracy and the efficiency video segmentation algorithm. These criteria are frequently used to compare moving object detection algorithms.[24] According to our experiment, we conclude that our algorithm has the same performance as the algorithm proposed in Ref. 17 and according to the results presented in Tables 3–5, we prove that the accuracy of the proposed system is 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 our proposed algorithm. In Table 5,

Table 2 Pixel identification.

		System output	
		Foreground	Background
Ground	Foreground	TP	FN
Truth	Background	FP	TN

Table 3 Different metrics according to experiments with "boats" dataset.

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

Note: Bold values indicate the best values.

Table 4 Different metrics according to experiments with "fall" dataset.

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

Note: Bold values indicate the best values.

Table 5 Different metrics according to experiments with "canoe" dataset.

Metrics	CB	CB_HSV	CB_HSL	CB_YUV	CB_LAB
FPR	0.16	0.18	0.15	0.21	**0.13**
PR	0.41	0.46	0.46	**0.52**	0.48
FM	0.35	0.38	0.37	**0.41**	0.39

Note: Bold values indicate the best values.

Table 6 Execution times of mobile object detection process (frame/s).

Datasets	Mousse et al.[19]	Proposed approach
Overpass	75.82	**92.15**
Fall	63.15	**75.78**
Fountain02	73.15	**91.44**
Fountain01	70.52	**95.91**

Note: Bold values indicate the best values.

we notice the FM and the precision of the algorithm proposed by Cheng et al. were superior to the FM and the precision of our proposed method. This is explained by the fact that in this sequence there is not much variation in brightness. So in this case, the YUV color space allows obtaining a detection rate. This conclusion confirms the results of the work of Cheng et al. But note that the false alarm rate has decreased with the use of the CIE Lab color space. This confirms the interest of this color space in the realization of applications that have for an objective the reduction of the rate of false detections.

To assess the contribution of our approach, we calculate our execution time (in frame per second) and compare it to the execution time of Mousse et al.[19] (Fang et al.[18] and Mousse et al.[19] proposed the fastest methods among all the improvements of the codebook with practically the same execution time). The results are reported in Table 6. In this table, the speed is expressed in frames per second. According to values presented in Table 6, our approach improves processing time by at least 20%. The gain is more substantial when the size of the foreground objects is small. This is evidenced by experiments based on the sequence fountain01 where the speed was increased by 36%.

6 Conclusion

In this work, we investigate a method to detect moving objects in a video sequence based on a codebook background model. We propose an adaptive method to reduce the amount of calculation. The perceptual visual information is used to actively focus the attention on the relevant superpixels. By doing like this, the treatment is done only on the important parts of the image. The experiment results obtained using public sequences with dynamic scene verify that our proposed approach outperforms other algorithms in calculation cost and has an accuracy close to the state of art algorithms.

References

1. C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (1999).
2. C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 747–757 (2000).
3. A. Elgammal et al., "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proc. IEEE* **90**, 1151–1163 (2002).
4. J. Rittscher et al., "A probabilistic background model for tracking," in *Proc. of the 6th European Conf. on Computer Vision Dublin*, pp. 336–350 (2000).
5. B. Stenger et al., "Topology free hidden Markov models: application to background modeling," in *Proc. IEEE Int. Conf. on Computer Vision*, pp. 294–301 (2001).
6. K. Kim et al., "Real-time foreground-background segmentation using codebook model," *Real-Time Imaging* **11**(3), 172–185 (2005).
7. J.-M. Guo et al., "Hierarchical method for foreground detection using codebook model," *IEEE Trans. Circuits Syst. Video Technol.* **21**(6), 294–301 (2011).
8. A. Ilyas, M. Scuturici, and S. Miguet, "Real time foreground-background segmentation using a modified codebook model," in *Proc. of the IEEE Int. Conf. on Advanced Video and Signal Based Surveillance*, pp. 454–459 (2009).
9. C. Xu, Z. Tian, and R. F. Li, "A fast motion detection method based on improved codebook model," *J. Comput. Res. Dev.* **47**, 2149–2156 (2010).
10. M. Shah, J. D. Deng, and B. J. Woodford, "Enhanced codebook model for real-time background subtraction," in *Proc. of the 18th Int. Conf. ICONIP, Part III*, pp. 449–458 (2011).
11. A. Pal, G. Schaefer, and M. E. Celebi, "Robust codebook-based video background subtraction," in *Proc. of the IEEE Int. Conf. on Acoustics Speech and Signal Processing*, pp. 1146–1149 (2010).
12. A. Doshi and M. M. Trivedi, "Hybrid cone-cylinder codebook model for foreground detection with shadow and highlight suppression," in *Proc. of the IEEE Int. Conf. on Advanced Video and Signal Based Surveillance*, pp. 121–133 (2006).
13. H. Donghai et al., "Principal component analysis based codebook background modeling algorithm," *Acta Autom. Sin.* **38**(4), 591–600 (2012).
14. Y. Li et al., "Gaussian-based codebook model for video background subtraction," *Lect. Notes Comput. Sci.* **4222**, 762–765 (2006).
15. W. Yu, D. Zeng, and H. Li, "Layered video objects detection based on LBP and codebook," in *Proc. of First Int. Workshop on Education Technology and Computer Science*, Vol. 1, pp. 207–213 (2006).
16. B. Li et al., "Segmentation of moving foreground objects using codebook and local binary patterns," in *Proc. of the 2008 Congress on Image and Signal Processing*, Vol. 4, pp. 239–243 (2008).
17. M. A. Mousse, E. C. Ezin, and C. Motamed, "Foreground-background segmentation based on codebook and edge detector," in *Proc. of the 10th Int. Conf. on Signal Image Technology & Internet Based Systems* (2014).
18. X. Fang et al., "Object detection in dynamic scenes based on codebook with superpixels," in *Proc. of the Asian Conf. on Pattern Recognition*, pp. 430–434 (2013).
19. M. A. Mousse, C. Motamed, and E. C. Ezin, "Fast moving object detection from overlapping cameras," in *Proc. of the Int. Conf. on Informatics in Control, Automation and Robotics*, pp. 296–303 (2015).
20. M. A. Mousse, C. Motamed, and E. C. Ezin, "Percentage of human-occupied areas for fall detection from two views," in *The Visual Computer* (2016).
21. A. Schick, M. Fischer, and R. Stiefelhagen, "Measuring and evaluating the compactness of superpixels," in *Proc. Int. Conf. on Pattern Recognition*, pp. 930–934 (2012).
22. N. Goyette et al., "Changedetection.net: a new change detection benchmark dataset," in *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops*, pp. 1–8 (2012).
23. ChangeDetection, "ChangeDetection video database," (2012), http://www.changedetection.net/ (6 November 2016).

24. P. L. Rosin and E. Ioannidis, "Evaluation of global image thresholding for change detection," *Pattern Recognit. Lett.* **24**, 2345–2356 (2003).

Mikaël A. Mousse is currently pursuing his PhD in computer vision and pattern recognition conjointly at the Institut de Mathmatiques et de Sciences Physiques, Université du Littoral Côte d'Opale, France. As a computer vision foundation member, his research interests include signal processing, image processing, video processing, computer vision, and human behaviors analysis and recognition.

Cina Motamed is an associate professor in computer science at the University of Littoral Côte d'Opale, Calais, France. His research interests focus on the design of multicameras system for real-time multiobject tracking and human action recognition. He is recently focusing on the uncertainty management over the vision system by using graphical models, and beliefs propagation. He is also interested in unsupervised learning approaches for human activity recognition.

Eugène C. Ezin received his PhD with the highest level of distinction after research works carried out on neural networks and fuzzy systems for speech applications. Since July 2012, he has been an associate professor in computer science in the field of artificial intelligence. His research interests include machine learning, neural networks fuzzy systems, signal and image processing, cryptography, and human activities recognition through multisensor systems. He is an IEEE member in computer society.