A Benchmark for Background Subtraction Algorithms in Monocular Vision: a Comparative Study

Yoann Dhome¹, Nicolas Tronson², Antoine Vacavant³,⁴, Thierry Chateau⁵, Christophe Gabard¹, Yann Goyat² and Dominique Gruyer⁶

¹ CEA, LIST, Laboratoire, Gif-sur-Yvette, F-91191, France
e-mail: yoann.dhome@cea.fr, christophe.gabard@cea.fr
² LCPC, Bouguenais, F-44341, France
e-mail: nicolas.tronson@lcpc.fr, yann.goyat@lcpc.fr
³ Université de Lyon, CNRS
⁴ Université Lyon 2, LIRIS, UMR5205, F-69676, France
e-mail: antoine.vacavant@liris.cnrs.fr
⁵ LASMEA, Université Blaise Pascal, Aubière, F-63177, France
e-mail: thierry.chateau@lasmea.univ-bpclermont.fr
⁶ LIVIC, INRETS Satory, Versailles F-78000, France
e-mail: dominique.gruyer@inrets.fr

Abstract—Background subtraction of video sequences is mainly regarded as a solved problem. However, no complete benchmark about Background Subtraction Algorithms (BSA) has been established, with ground truth and associated quality measures. One of the reasons is that such comparative study needs annotated datasets. In this article, we propose a BSA evaluation dataset built from realistic synthetic image and we compare six BSA, according to several quality measures.

Keywords—Video processing, video surveillance, background subtraction dataset, benchmark.

I. INTRODUCTION

Many realtime vision-based applications devoted to video surveillance search to distinguish moving objects from an image sequence given by a static camera. This background/foreground segmentation stage could be addressed by simple approaches, e.g., by computing the difference between two successive frames, or by building a time-averaged background image. However, such simple algorithms are very limited in outdoor environment (because of global variation of luminance, shadows of objects, etc.).

Since this is a popular topic, many background subtraction algorithms (BSA) have been proposed since 90’s to tackle these problems. Although the evaluation of BSA is an important issue, relevant papers that handle with both benchmarks and annotated dataset are limited [9], [1]. In this article, we propose a BSA evaluation dataset build from realistic synthetic image and we compare six BSA, according to several quality measures.

In Section II, we describe in details the BSA evaluated using the dataset and methodology presented in section III. Comparative measures are given and discussed in section IV.

II. DESCRIPTION OF THE BSA

This section describes the six background/foreground algorithms proposed into this comparative study.

A. Gaussian Mixture Model (GMM)

One of the most popular BSA is based on a parametric probabilistic background model proposed by Stauffer and Grimson [11], and improved by Hayman and Eklundh [6]. In this algorithm, distributions of each pixel color is represented by a sum of weighted Gaussian distributions defined in a given colorspace: the Gaussian Mixture Model (or GMM). These distributions are generally updated using an online expectation-minimization algorithm. Even if this method is able to handle with low illumination variations, rapid variations of illumination and shadows are still problematic. Furthermore, the learning stage can be inefficient if it is realized with noisy video frames. To tackle these problems, many authors have extended the GMM. For example, Kaewtrakulpong and Bowden [7] propose to modify the updated equations in this model to improve the adaptation of the system to illumination variations.

Each pixel has a parametric distribution model given by a mixture of N Gaussians, \( P(Z_t|M_t) \) is correct. The likelihood that a pixel is a background pixel is:

\[
P(Z_t|M_t) = \sum_{n=1}^{N} \alpha_n \mathcal{N}(\mu_n, \Sigma_n)
\]

(1)

\[
\mathcal{N}(\mu_n, \Sigma_n) = \frac{1}{(2\pi)^{d/2}|\Sigma_n|^{1/2}} e^{-\frac{1}{2}(Z_t-\mu_n)^T \Sigma_n^{-1}(Z_t-\mu_n)}
\]

(2)

where \( d \) is the dimension of color space of the measures \( Z_t \).
To update the GMM, we first associate the measures \( Z_t \) to one Gaussian \( n' \) if
\[
||Z_t - \mu_n|| < K \sigma_n
\]  
where \( K \) is 2 or 3. The operator \(<\) is true if all vector components at the left are smaller than \( K \sigma_n \). This measure represents the background if the Gaussian \( \alpha_n \) explains the background of the scene. In fact, the weight \( \alpha_n \) is high. This Gaussian is then updated:
\[
\alpha'_n \leftarrow (1 - \delta) \alpha_n + \delta
\]
\[
\mu'_n \leftarrow (1 - \rho_n) \mu_n + \rho_n Z_t
\]
\[
\sigma'^2_n \leftarrow (1 - \rho_n) \sigma'^2_n + \rho_n (Z_t - \mu'_n)^T (Z_t - \mu'_n)
\]
\[
\rho'_n \leftarrow \delta N(Z_t | \mu'_n, \sigma'^2_n)
\]
where \( \delta \) is the learning coefficient. It represents adaptation time of the model. For all Gaussians \( n \neq n' \), mean and variance aren’t modified, but: \( \alpha'_n \leftarrow (1 - \delta) \alpha_n \). If the test 3 fails, pixel is associated to the foreground. The Gaussian with the smallest weight is reinitialized with current measure: \( \alpha_n = \delta, \mu_n = Z_t, \sigma^2_n = \sigma^2_n \), where \( \sigma^2_n \) is a high variance. We also apply those affectionation for GMM initialization.

B. Fast-learning Gaussian Mixture Model (FGMM)

The approach proposed by Kaewtrakulpong and Bowden [7] is strongly inspired from Stauffer and Grimson [11]. However, the authors considered that [11] suffers from slow learning at the beginning, especially in busy environments. By reinvestigating the update mechanism, the authors propose different equations at different phases. During the learning step, the authors use the following equations:
\[
\alpha'_n \leftarrow \left(1 - \frac{1}{(n + 1)}\right) \alpha_n + \frac{1}{(n + 1)} m_k(Z_t)
\]
\[
\mu'_n \leftarrow \left(1 - \frac{m_k(Z_t)}{\sum_{i=1}^{n} m_k(Z_i)}\right) \mu'_n + \frac{m_k(Z_t)}{\sum_{i=1}^{n} m_k(Z_i)} Z_t
\]
\[
\sigma'^2_n \leftarrow \left(1 - \frac{m_k(Z_t)}{\sum_{i=1}^{n} m_k(Z_i)}\right) \sigma'^2_n + \frac{m_k(Z_t)}{\sum_{i=1}^{n} m_k(Z_i)} (Z_t - \mu'_n)^T (Z_t - \mu'_n)
\]
\[
m_k(Z_t) = \begin{cases} 1 & \text{if } Z_t \text{ is the first match Gaussian component} \\ 0 & \text{otherwise} \end{cases}
\]
Then they switch to a \( L \)-recent window version when the first \( L \) samples are processed.
\[
\alpha'_n \leftarrow \left(1 - \frac{1}{L}\right) \alpha'_n + \frac{1}{L} m_k(Z_t)
\]
\[
\mu'_n \leftarrow \left(1 - \frac{1}{L} \alpha'_n \right) \mu'_n + \frac{1}{L} \alpha'_n Z_t
\]
\[
\sigma'^2_n \leftarrow \left(1 - \frac{1}{L} \alpha'_n \right) \sigma'^2_n + \frac{1}{L} (Z_t - \mu'_n)^T (Z_t - \mu'_n)
\]

C. Bayesian Background Model (BBM)

Tuzel et al. [12] introduced a method for modeling background using recursive Bayesian learning approach. Each pixel is modeled with layers of Gaussian distributions. Using recursive Bayesian learning, they estimate the probability distribution of the mean and covariance of each Gaussian. Here, we will consider that the update phase is called for each frame, and the system is speed up by making independence assumption on color channels. To update the layers, the following equations are used (\( m_k(Z_t) \) is the same as Equation 12):
\[
\nu'_n \leftarrow \nu'_n + m_k(Z_t)
\]
\[
k'_n \leftarrow k'_n + (2 m_k(Z_t) - 1)
\]
\[
\mu'_n \leftarrow \left(1 - \frac{k'_n + m_k(Z_t)}{k'_n + m_k(Z_t)}\right) \mu'_n + \frac{m_k(Z_t)}{k'_n + m_k(Z_t)} Z_t
\]
\[
\Theta'_n \leftarrow \Theta'_n + \frac{k'_n}{k'_n + m_k(Z_t)} (Z_t - \mu'_n)^T (Z_t - \mu'_n)
\]
\[
\Sigma'_n \leftarrow (\nu'_n - 4)^{-1} \Theta'_n
\]
Where \( k'_n \) is the absolute number of prior measurements, \( \nu'_n \) the relative number of prior measurements, \( \Sigma'_n \) the diagonal covariance matrix, \( \Theta'_n \) the scale matrix of inverse Wishart distribution.

The segmentation phase is organized as follows. The confidence score \( C \) is used to order and select the layers thanks to a threshold \( T_c \). Then, the Mahalanobis distance of observed color from the confident layers is computed. Pixels that are outside of 99% confidence interval of all confident layers of the background are considered as foreground pixels.

\[
C = \frac{k'_n \nu'_n - 2^4}{(\nu'_n - 4) ||\Theta'_n||}
\]
The system is started with the following initial parameters: \( k'_0 = 10, \nu'_0 = 10, \mu'_0 = Z_0, \theta'_0 = (\nu'_0 - 4)16^2 I \) where \( I \) is the three dimensional identity matrix.

D. Bloc-Level Gaussian Mixture Model (BLGMMM)

Chen et al. [2] proposed a hierarchical method inspired by Stauffer and Grimson [11]. Here, we will focus only on the bloc-level approach. Using the algorithm of [11], Chen et al. [2] replace the RGB pixel descriptor by a \( 8 \times 8 \) bloc texture one called contrast histogram.

After dividing an image into blocks, a descriptor is built for each block \( B_c \). Since the center pixel \( p_c \) in \( B_c \) does not exist, its value is estimated by averaging the four center pixels of \( B_c \). Each block is separated into four quadrant bins, until positive and negative contrast-value histograms for each quadrant bin \( q_i \) are computed.

Let \( j \in R, G, B \) and \( k \in R, G, B \) stand for the color channels of \( p \) and \( p_c \), respectively. The positive contrast histogram \( CH_{(j,k)}^{(i)}(p_c) \) and negative \( CH_{(j,k)}^{(i)}(p_c) \) one of \( q_i \) with respect to \( p_c \) are defined as follows:
For $i = 0,\ldots,3$, $\omega^i(q_i)$ is the number of pixels with positive contrast values in $q_i$, $\omega^{-i}(q_i)$ is the number of pixels with negative contrast values in $q_i$, and $C^{(j,k)}(p,p_c)$ is the contrast value between the $j$ channel of $p$ and the $k$ channel of $p_c$. There are nine combinations of the pair $(j,k)$, which introduces a $72(=9 \times 4 \times 2)$-dimensional description vector for each block. However, it is not necessary to use all the pairs. Instead, the authors only pick up the six pairs, $(j,k) \in \{(R,R),(R,G),(R,B),(G,G),(G,B),(B,B)\}$ and thus a $48(=6 \times 4 \times 2)$-dimensional color descriptor, $CH(pc) \in R^{48}$, is employed for efficient construction and matching.

E. Codebook 2 Layers (CB2)

The codebook 2 layers algorithm by Sigari and Fathy [10] is inspired by codebook by Kim et al. [8]. But in contrary to simple codebook, which contains an unique codebook per pixel, this method uses 2 codebooks.

Each codebook contains some codeword to model a cluster of samples that constructs a part of background and each codeword contains these informations: 1) $v_i$: value of mean pixel $(R,G,B)$, 2) $I_{max}$: high intensity bound of codeword, 3) $I_{min}$: low intensity bound of codeword, 4) $f$: frequency of codeword, 5) $\lambda$: $MNRL$ (maximum negative run length), represents the longest number of image where the codeword doesn’t occur in the sequence, 6) $p$: first occurrence of the codeword, and 7) $q$: last occurrence of the codeword. The principle is the same than simple codebook, but we have 2 codebook per pixel: a main codebook called $M$, and an hidden codebook called $H$. For each new pixel $x_t = (R,G,B)$, its intensity $I_t$ is calculated by $I_t = \sqrt{R^2 + G^2 + B^2}$. The color distortion $\delta$ between this pixel $x_t = (R,G,B)$ and a codeword $c_i$ where $v_i = (\hat{R}_i, \hat{G}_i, \hat{B}_i)$ can be calculated by:

$$\langle x_t, v_i \rangle^2 = (\hat{R}_i R + \hat{G}_i G + \hat{B}_i B)^2$$
$$\|v_i\|^2 = \hat{R}_i^2 + \hat{G}_i^2 + \hat{B}_i^2$$
$$\|x_t\|^2 = R^2 + G^2 + B^2$$
$$\text{colordist}(x_t, v_i) = \delta = \sqrt{\|x_t\|^2 - \langle x_t, v_i \rangle^2 / \|v_i\|^2}$$

A pixel $x_t$ with an intensity $I_t$ match to a codeword $c_i$ with a pixel value $v_i$ and $I_{min}, I_{max}$ if $I_t$ is in range $[I_{min}, I_{max}]$ and the color distortion $\delta$ corresponds to $\delta < \epsilon$. A new codeword is created with a pixel $x_t$ like this:

$$v_i \leftarrow (R, G, B)$$
$$I_{min} \leftarrow \max\{0, I_t - \alpha\}; I_{max} \leftarrow \min\{255, I_t + \alpha\}$$

where $\alpha$ is a value which represents a tolerance of intensity. During the training phase a codeword is updated by a pixel $x_t$ as follows:

$$\hat{R} \leftarrow \frac{R \times f + \hat{R}}{f + 1} \quad \text{(idem for G and B)}$$
$$I_{min} \leftarrow \frac{I - \alpha + f \times I_{min}}{f + 1}; I_{max} \leftarrow \frac{I + \alpha + f \times I_{max}}{f + 1}$$

In detection phase the codeword is updated like previously in training phase except for $I_{min}$ and $I_{max}$ which are updated as follow, with $\beta$ a coefficient to change adaptation speed:

$$I_{min} \leftarrow (1 - \beta)(I_t - \alpha) + \beta \cdot I_{min}$$
$$I_{max} \leftarrow (1 - \beta)(I_t + \alpha) + \beta \cdot I_{max}$$

After, $M$ and $H$ models are refined with these rules: 1) Remove all codewords in $H$ having $\lambda$ more than $T_H$; 2) Move the codewords from $H$ to $M$ who staying longer than $T_{add}$ in $H$; 3) Remove all codewords from $M$ not appearing longer than $T_{delete}$. This method keeps in $H$ many data that are not important at this moment but could be more important after. This is very useful to have a good detection of waving trees for example. However, adding data during the process causes a not stable memory occupation and can change the speed of the algorithm to find the corresponding codeword in a very large quantity of codewords.

F. VuMeter

The VuMeter method proposed by Goyat et al. [4] is a non parametric model, based on a discrete estimation of the probability distribution. It is a probabilistic approach to define the image background model. $I_t$ is an image at time $t$, and $y_t(u)$ gives the color vector Red Green Blue of pixel $u$. A pixel can take two states, $(\omega_1)$ if the pixel is background, $(\omega_2)$ if the pixel is foreground. This method try to estimate $p(\omega_1 \mid y_t(u))$. With 3 color component $i$ (Red, Green, Blue), the probability density function can be approximated by:

$$p(\omega_1 \mid y_t(u)) = \prod_{i=1}^{3} p(\omega_1 \mid y^i_t(u))$$

with

$$p(\omega_1 \mid y^i_t(u)) \approx K_i \sum_{j=1}^{N} \pi^i_j \delta(b^j_t(u) - j)$$

where $\delta$ is the Kronecker delta function, $b_t(u)$ gives the bin index vector associated to $y_t(u)$, $j$ is a bin index, and $K_i$ is
a normalization constant to keep at each moment

\[ \sum_{j=1}^{N} \pi_{tj}^i = 1 \quad (38) \]

\( \pi_{tj}^i \) is a discrete mass function which is represented by a bin. At first image (\( t = 0 \)), we set bins values, \( \pi_{0j}^i = 1/N \) to have a sum to 1 like in Equation 38. At each new pixel, its value match to a bin \( \pi_{tj}^i \). The level of this bin is updated as follow:

\[ \pi_{t+1j}^i = \pi_{tj}^i + \alpha \cdot \delta(b_{t+1j}(u) - j) \quad (39) \]

After a lot of images, the bins which are modeling the background have a high value. To choose at each moment if a pixel is background or not, a threshold \( T \) is set. Each new pixel with corresponding bins under this threshold will be detected as background. In RGB mode each pixel will be modeled by 3 VuMeter (1 for each color). To consider a pixel as background, it must be detected as background with each VuMeter. To improve background detection and reduce problems with edges between two bins, the value the classes in neighborhood of matched class are also updated, but less. To have a good learning and adaptation of algorithm, it’s necessary to have a good learning rate and a good threshold. Typically, the learning rate value is 0.01 and the threshold value is 0.2. But these values can change in particular scene with luminosity variations for example and different speeds of tracking object.

III. DATASETS AND EVALUATION MEASURES

In this section, we present the datasets and measures used to evaluate the six algorithms presented in the previous section and recall in Table 1.

Table 1. The BSA we compare, and their main reference in the literature.

<table>
<thead>
<tr>
<th>Id.</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Original GMM</td>
<td>Stoffer and Grimson [11]</td>
</tr>
<tr>
<td>2</td>
<td>Fast learning GMM</td>
<td>Kaeuwtrakulpong and Bowden [7]</td>
</tr>
<tr>
<td>3</td>
<td>BBM</td>
<td>Tuzel et al. [12]</td>
</tr>
<tr>
<td>4</td>
<td>Bloc-Level GMM</td>
<td>Chen et al. [2]</td>
</tr>
<tr>
<td>5</td>
<td>Codebooks 2 layers</td>
<td>Kim et al. [8]</td>
</tr>
<tr>
<td>6</td>
<td>VuMeter</td>
<td>Goyat et al. [4]</td>
</tr>
</tbody>
</table>

A complete realistic 3D urban environment renderer has been developed using the LIVIC SIVIC simulator [5]. This tool is able to simulate cars and humans, noisy acquisition, sun (and shadows) moves, etc. Moreover, since video sequences are synthetic, annotated ground truth is also given by the simulator. One can note that moving object shadows are considered as background in our framework. We describe in Table 2 the four videos we have chosen, with a brief description. Moreover, an illustration of each video is presented in Figure 2.

In this article, we first consider the background-foreground segmentation problem as a binary classification scheme. For a given video frame \( i \), we compute the number of true/false positives (denoted by \( TP_i \) and \( FP_i \), respectively) and the number of true/false negatives (\( TN_i \) and \( FN_i \)) according to the associated ground truth frame. We also define the sensibility and the specificity, for each frame \( i \):

\[ S_{ei} = TP_i/(TP_i + FN_i) \quad (40) \]
\[ Sp_i = TN_i/(TN_i + FP_i) \quad (41) \]

The first one indicates the rate of true detected foreground pixels, while the second one represents the quality of the detection of the background. For a given algorithm, we plot a sensibility/specificity point distribution by representing each video frame \( i \) by the point \( p_i = (1 - Sp_i, Se_i) \). In this coordinate system, the points belonging to the half-plane given by \( y \leq x \) represents the segmentations that were not able to distinguish the background from the foreground. If a video contains \( N \) frames, we define the following quality measure:

\[ \Delta = \sum_{i=1}^{N} 2 - Sp_i - Se_i, \quad (42) \]

where \( 2 - Sp_i - Se_i \) represents the distance between (1) the projection of the \( i^{th} \) point onto \( Y \) axis, with the vector \((-1, -1)\); (2) the point \((0, 1)\) of the system, which is considered as the perfect segmentation. This distance is able to handle with the property of the half-plane \( y \leq x \) that we have given above, and illustrated in Figure 1.

![Fig. 1. The distance between two points \( p_1, p_2 \) in our coordinate system. We can notice that \( 2 - Sp_1 - Se_1 = d_1 < d_2 = 2 - Sp_2 - Se_2 \), and that \( p_2 \) represents a very bad segmentation (since it is on the border of the half-plane \( y \leq x \) (dotted line).](image)

A background/foreground segmentation problem can also be seen as a statistical problem, where it is relevant to compute recall/precision point distribution from each class. We thus define the recall and the precision for the positive (foreground) and negative (background) detection, for a frame \( i \) of a video sequence:

\[ Re_i(P) = TP_i/(TP_i + FN_i) = Se_i \quad (43) \]
\[ Re_i(N) = TN_i/(TN_i + FN_i) = Sp_i \quad (44) \]
\[ Pr_i(P) = TP_i/(TP_i + FP_i) \quad (45) \]
\[ Pr_i(N) = TN_i/(TN_i + FN_i) \quad (46) \]

Table 2. The synthetic videos used for the evaluation, with their number of frames and a brief description.

<table>
<thead>
<tr>
<th>Id.</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>724</td>
<td>Very noisy, 5 vehicles, 2 pedestrians, 1 curt change of light</td>
</tr>
<tr>
<td>2</td>
<td>649</td>
<td>Dark scene, several vehicles and pedestrians, 1 curt change of light</td>
</tr>
<tr>
<td>3</td>
<td>980</td>
<td>One car, sun and trees move</td>
</tr>
<tr>
<td>4</td>
<td>575</td>
<td>Some noise, many traffic</td>
</tr>
</tbody>
</table>
And we use the $F$-measure given by

$$F = 2 \times \frac{P_{ri} \times Re_{i}}{(P_{ri} + Re_{i})}, \text{ with}$$

$$P_{ri} = \frac{(Pr_{i}(P) + Pr_{i}(N))}{2}$$

$$Re_{i} = \frac{(Re_{i}(P) + Re_{i}(N))}{2}$$

As previously, the recall/precision point distribution by considering for a frame $i$ its associated point $p_{i} = (1 - Re_{i}, Pr_{i})$ can be plotted. The half-plane $y \leq x$ represents the indiscriminate part of the plane, that is, the points where background and foreground are not well distinguished.

IV. EXPERIMENTAL RESULTS

In this section, we present the experimental measures achieved in order to evaluate the six BSA.

A. Results

We first show in Figure 3 the sensibility/specificity and recall/precision point distributions for two algorithms, according to two video sequences. In Figure 4, we present images obtained from the six BSA, for some videos of our benchmark. In Table 3, we sum up the quality measures $\Delta$ (according to Equation 42) and in Table 4 the $F$ measures (Equation 47) we obtained for each algorithm and each video of our tests.

Table 3. The quality measure $\Delta$ for the algorithms and the videos we have chosen in our tests. The rank is established according to the mean of the $\Delta$ values over the four video sequences. The less $\Delta$ is, the better is the background/foreground segmentation.

<table>
<thead>
<tr>
<th>$\Delta$</th>
<th>Vid. 1</th>
<th>Vid. 2</th>
<th>Vid. 3</th>
<th>Vid. 4</th>
<th>Rank (mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg. 1</td>
<td>0.512</td>
<td>0.559</td>
<td>0.351</td>
<td>0.368</td>
<td>6 (0.447)</td>
</tr>
<tr>
<td>Alg. 2</td>
<td>0.580</td>
<td>0.295</td>
<td>0.062</td>
<td>0.242</td>
<td>2 (0.295)</td>
</tr>
<tr>
<td>Alg. 3</td>
<td>0.470</td>
<td>0.202</td>
<td>0.078</td>
<td>0.562</td>
<td>3 (0.328)</td>
</tr>
<tr>
<td>Alg. 4</td>
<td>0.532</td>
<td>0.200</td>
<td>0.080</td>
<td>0.180</td>
<td>1 (0.248)</td>
</tr>
<tr>
<td>Alg. 5</td>
<td>0.481</td>
<td>0.360</td>
<td>0.299</td>
<td>0.391</td>
<td>5 (0.383)</td>
</tr>
<tr>
<td>Alg. 6</td>
<td>0.476</td>
<td>0.348</td>
<td>0.345</td>
<td>0.219</td>
<td>4 (0.346)</td>
</tr>
</tbody>
</table>

Table 4. The quality measure $F$ for the algorithms and the videos we have chosen in our tests. The rank is established according to the mean of the $F$ values over the four video sequences. The greater $F$ is, the better is the background/foreground segmentation.

<table>
<thead>
<tr>
<th>$F$</th>
<th>Vid. 1</th>
<th>Vid. 2</th>
<th>Vid. 3</th>
<th>Vid. 4</th>
<th>Rank (mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg. 1</td>
<td>0.578</td>
<td>0.721</td>
<td>0.702</td>
<td>0.847</td>
<td>5 (0.712)</td>
</tr>
<tr>
<td>Alg. 2</td>
<td>0.608</td>
<td>0.723</td>
<td>0.722</td>
<td>0.832</td>
<td>4 (0.721)</td>
</tr>
<tr>
<td>Alg. 3</td>
<td>0.584</td>
<td>0.675</td>
<td>0.722</td>
<td>0.626</td>
<td>6 (0.652)</td>
</tr>
<tr>
<td>Alg. 4</td>
<td>0.600</td>
<td>0.835</td>
<td>0.832</td>
<td>0.849</td>
<td>2 (0.779)</td>
</tr>
<tr>
<td>Alg. 5</td>
<td>0.642</td>
<td>0.767</td>
<td>0.751</td>
<td>0.859</td>
<td>3 (0.755)</td>
</tr>
<tr>
<td>Alg. 6</td>
<td>0.570</td>
<td>0.859</td>
<td>0.849</td>
<td>0.891</td>
<td>1 (0.792)</td>
</tr>
</tbody>
</table>

B. Discussion

The $\Delta$ and $F$ measures we have defined above are complementary, and point out two different phenomenas: $\Delta$ increases if a method hardly models the background (e. g. codebook based BSA 5), while $F$ decreases if an approach is too excessive (i. e. it leads to too much positive or negative detections (e. g. GMM-based BSA 3), according to the ground truth). If we consider the sum of the ranks from tables 3 and 4, the global ranking of the BSA is the following: 4, 6, 2, 5, 3, 1. For both measures, Algorithm 1 is an inefficient approach because it does not handle noise and very moving backgrounds (see Figure 4, video 3). An other GMM based BSA of our tests is the best approach: Algorithm 4. It is based on a $8 \times 8$ block process, which filter noise and efficiently treat moving backgrounds (Figure 4, video 1 for example). For example, the fact that Algorithm 4 overtake Algorithm 2 can be observed with Figure 3. In these plots, we can notice that the points associated with the BSA 4 are globally closer to the $(0,1)$ point, and they are globally farther from the $y \leq x$ half-plane. However, we obtain a coarse background/foreground representation with Algorithm 4. In the case where a fine description is required, Algorithm 6 should be chosen instead, which has the best $F$ measure, even if it hardly handles noise. The third algorithm in our global ranking is BSA 2, which is a GMM based method with a fast learning phase. Even if this phase is improved according to BSA 1, it is not enough to delete completely noise in the video sequences.

V. CONCLUSION AND FUTURE WORK

We have presented in this article a comparative study of background subtraction algorithms, where we compare their quality of segmentation with two different measures, inspired from classification problems. Moreover, we have proposed a complete benchmark that could be used by other authors to compare their contribution. We have shown that the approach introduced by Chen et al. [2] is the best BSA, then come the BSA developed by Goyat et al. [4] and by Kaewtrakulpong and Bowden [7]. As a future work, we first would like to add more BSA in our survey, and in particular other GMM algorithms [13], codebook based approaches [8], etc. Indeed, this latter type of algorithms may be a very interesting choice with a good parametrization. We also plan to evaluate execution times of the BSA, and to add more realistic synthetic video sequences, and outside videos acquired from real cameras.

REFERENCES


Fig. 2. A frame of each video from Table 2, with its associated ground truth.

Fig. 3. Examples of Sensibility/specificity and recall/precision point distributions from video sequence 2. (a) and (b) are generated with algorithm 2, while (c) and (d) are generated with algorithm 4.

Fig. 4. Some segmentations computed for the evaluated BSA, illustrating each video sequence.


