Pattern Recognition Letters
journal homepage: www.elsevier.com

Foreground detection by probabilistic modeling of the features discovered by stacked denoising autoencoders in noisy video sequences

Jorge García-González[a,**], Juan M. Ortiz-de-Lazcano-Lobato[a], Rafael M. Luque-Baena[a], Miguel A. Molina-Cabello[a], Ezequiel López-Rubio[a]

[a]*Department of Computer Languages and Computer Science. University of Málaga. Bulevar Louis Pasteur, 35. 29071 Málaga. Spain.*

ABSTRACT

A robust foreground detection system is presented, which is resilient to noise in video sequences. The proposed model divides each video frame in patches that are fed to a stacked denoising autoencoder, which is responsible for the extraction of significant features from each image patch. After that, a probabilistic model that is composed of a mixture of Gaussian distributions decides whether the given feature vector describes a patch belonging to the background or the foreground. In order to test the model robustness, several trials with noise of different types and intensities have been carried out. A comparison with other ten state of the art foreground detection algorithms has been drawn. The algorithms have been ranked according to the obtained results, and our proposal appears among the first three positions in most case and its the one that best performs on average.

Declarations of interest: none

1. Introduction

The detection of foreground objects is a hot topic which has been studied in the field of computer vision research in recent years. It is the first step to be carried out in video surveillance systems, and its viability determines the outcome of the following phases, from the tracking of detected objects to the object interpretation inside the scene. Thus, poor detection of moving objects means that their behaviour cannot be correctly ascertained. Most of the used sequences to test object detection algorithms are not entirely real and do not reflect the acquisition problems of real fixed cameras, especially in outdoor scenes. Camera jitter due to weather factors such as wind, loss of connection and data transmission that generate corrupted image blocks, or focusing problems that generate blurry images, are relevant aspects that practical foreground detection algorithms must deal with. Hence, robustness should be a key feature of a surveillance system which is designed to work continuously.

Most foreground detection algorithms model information at the pixel level. Among them, we can find those that analyze each pixel independently using its color intensity: *Wren* (Wren et al. (1997)), *Zivkovic* (Zivkovic and van der Heijden (2006)), *KDE* (Elgammal et al. (2000)) and *CL-VID* (López-Rubio et al. (2018a)). These methods present some limitations, such as their low tolerance to the appearance of noise (López-Rubio et al. (2018b)). Other proposals such as *SOBS* (Maddalena and Petrosino (2008)), *SC-SOBS* (Maddalena and Petrosino (2012)) and *FSOM* (López-Rubio et al. (2011)) use the information of their neighbors to provide greater robustness in the segmentation mask, greatly improving their noise resilience. On the other hand, other more complex techniques like *LOBSTER* (St-Charles and Bilodeau (2014)), *SuBSENSE* (St-Charles et al. (2015)) or *PAWCS* (St-Charles et al. (2016)), represent the information of each pixel not only by its intensity, but also by including texture patterns of the region to which they belong (Local Binary Similarity Patterns, LBSP), thereby increasing their noise robustness with regard to the previous ones. In the field of detection, new techniques are based on either capturing multiple view data latent structure (Wang et al. (2018)) or learning deep models. The latter have been successfully applied in video surveillance, from convolutional neural networks that are

**Corresponding author
e-mail: jorgegarcia@lcc.uma.es (Jorge García-González), jmortiz@lcc.uma.es (Juan M. Ortiz-de-Lazcano-Lobato), rmluque@lcc.uma.es (Rafael M. Luque-Baena), miguelangel@lcc.uma.es (Miguel A. Molina-Cabello), ezeqlr@lcc.uma.es (Ezequiel López-Rubio)

able to recognize different versions of the same object obtained from different cameras, problem known as re-identification (Wu et al. (2018a,b,d, 2019b,a)), to deep recurrent neural networks which manage to distinguish objects in an image with a high level of detail (Wu et al. (2018c)).

In this work, the combination of deep learning techniques and probabilistic modeling is proposed in order to detect foreground objects. The main objective is to develop a method with a higher robustness in presence of videos with noise, which improves the quality of the foreground detection. The proposed region-level based method divides each frame into patches, which are processed by a previously trained stacked denoising autoencoder that tries to generate noise-free regions. Next, the output vector of the innermost layer of the encoder, i.e., the representation of the input information in a space of reduced dimension, is modeled by a mixture of Gaussian distributions to determine whether the patch belongs to the foreground or background. This approach has been compared with the previously cited pixel-level techniques for video sequences with presence of noise, obtaining, on average, better segmentation results despite having aggregate data (region vs. pixel).

2. Methodology

Most previous approaches to background modeling in video sequences model each pixel of the video frame separately. Our model intends to model small patches of size $N \times N$ pixels, so that for each incoming video frame an estimation is made whether each patch belongs to the background of the scene. It turns out that stacked denoising autoencoders might find difficulties in modeling too small patches. Here we propose to overcome this limitation by augmenting the $N \times N$ pixel patch by $M < N$ pixels in each direction (up, down, left and right), so that an augmented patch of size $(N + 2M) \times (N + 2M)$ is supplied to the autoencoder, while the decision whether the patch belongs to the background only affects to the central $N \times N$ pixel section of the augmented patch. In this way, the augmented patches overlap with their neighbors, while the small patches do not.

Let $\mathbf{X} \in \mathbb{R}^H$ be an augmented patch of size $H = 3\,(N + 2M)^2$, where tristimulus pixel color values are assumed. The patch is processed by a stacked denoising autoencoder (Vincent et al. (2010)):

$$\tilde{\mathbf{X}} = g\,(f\,(\mathbf{X}))\,, \quad f\ :\ \mathbb{R}^H \to \mathbb{R}^L, \quad g\ :\ \mathbb{R}^L \to \mathbb{R}^H \tag{1}$$

where $\tilde{\mathbf{X}} \in \mathbb{R}^H$ is the reconstructed version of the input patch $\mathbf{X}$, f is the encoding part of the autoencoder, g is the decoding part of the autoencoder, and L is the number of neurons of the innermost layer of the neural architecture, i.e. the autoencoder reduces the high dimensional input of size H to a a low dimensional set of features of size L with $L < H$.

An autoencoder is trained to minimize the reconstruction error $\mathcal{E}$:

$$\mathcal{E} = \sum_{i=1}^{R} \left\| \mathbf{X} - \tilde{\mathbf{X}} \right\|^2 \tag{2}$$

where R is the overall number of patches existing in the training data set. In an attempt to enforce the invariance of the autoencoder to the diverse scene conditions, the training set is not generated from the input video sequence but from the Tiny Images data set (Torralba et al. (2008)) and comprises a huge amount of generic natural image patches that may be corrupted.

A probabilistic model can be learned for the features which are discovered by the autoencoder. This model aims to capture the main characteristics of the probability distribution of the feature vector $\mathbf{v} \in \mathbb{R}^L$:

$$\mathbf{v} = f\,(\mathbf{X}) \tag{3}$$

A probability model for $p\,(\mathbf{v})$ must be chosen. Here we propose to assume that $p\,(\mathbf{v})$ can be approximated by a probabilistic mixture of two components:

$$p\,(\mathbf{v}) = P\,(B)\,p\,(\mathbf{v} \mid B) + P\,(F)\,p\,(\mathbf{v} \mid F) \tag{4}$$

where B stands for the background part of the scene, and F stands for the foreground objects appearing in the scene.

We propose to model the probability distribution of the feature vectors $\mathbf{v}$ associated to foreground objects occurring at any position in the video frame with a multivariate L-dimensional Gaussian with diagonal covariance matrix. The parameters of the Gaussian are the mean $\mu_j = E\left[v_j\right]$ and the variance $\sigma_j^2 = E\left[\left(v_j - \mu_j\right)^2\right]$ of each component of $\mathbf{v}$. They are estimated offline by considering all the R image patches that were used to train the autoencoder:

$$\mu_j = \frac{1}{R} \sum_{i=1}^{R} v_{i,j} \tag{5}$$

$$\sigma_j^2 = \frac{1}{R-1} \sum_{i=1}^{R} \left(v_{i,j} - \mu_j\right)^2 \tag{6}$$

where $v_{i,j}$ stands for the j-th component of the i-th feature vector $\mathbf{v}_i$ in the training set of the autoencoder.

Once the parameters of the Gaussian are estimated, the log probability of observing $\mathbf{v}$ is given by:

$$\log p\,(\mathbf{v} \mid F) = -\frac{L}{2} \log 2\pi - \sum_{j=1}^{L} \log \sigma_j - \frac{1}{2} \sum_{j=1}^{L} \frac{\left(v_j - \mu_j\right)^2}{\sigma_j^2} \tag{7}$$

Another specific probabilistic model at each patch of the video frame is also learned. In this case, it models the feature vectors representing the background of the scene which appear in that patch. Here we assume that $p\,(\mathbf{v}_k \mid B)$, where k is the index of the patch of interest, can be approximated by a multivariate L-dimensional Gaussian with diagonal covariance matrix. The parameters of the Gaussian are the mean $\mu_{k,j} = E\left[v_{k,j}\right]$ and the variance $\sigma_{k,j}^2 = E\left[\left(v_{k,j} - \mu_{k,j}\right)^2\right]$ of each component of $\mathbf{v}_k$.

The means $\mu_{k,j}$ and the variances $\sigma_{k,j}^2$ are estimated online as the incoming video frames are processed. To this end, the Robbins-Monro stochastic approximation algorithm is employed (Robbins and Monro, 1951). Initially, $\mu_{k,j}$ is set to the

j component of the median reduced feature vector of the first video frames, while the initial value for $\sigma^2_{k,j}$ is obtained as shown in eq (8). Then the Gaussian distribution is updated as follows:

$$\mu_{k,j,t+1} = (1 - \alpha P(B \mid \mathbf{v}_{k,t}))\,\mu_{k,j,t} + \alpha P(B \mid \mathbf{v}_{k,t})\,v_{k,j,t} \quad (8)$$

$$\sigma^2_{k,j,t+1} = (1 - \alpha P(B \mid \mathbf{v}_{k,t}))\,\sigma^2_{k,j,t} + \alpha P(B \mid \mathbf{v}_{k,t})\,(v_{k,j,t} - \mu_{k,j,t})^2 \quad (9)$$

where t is the time instant (the frame index), k is the index of the patch of interest, $\mu_{k,j,t}$ and $\sigma_{k,j,t}$ are the estimations of $\mu_{k,j}$ and $\sigma_{k,j}$ at time t, $\mathbf{v}_{k,t}$ is the feature vector observed at time t in patch k, and α is the step size.

Then the log probability of observing $\mathbf{v}_k$ is given by:

$$\log p(\mathbf{v}_k \mid B) = -\frac{L}{2}\log 2\pi - \sum_{j=1}^{L}\log\sigma_{k,j} - \frac{1}{2}\sum_{j=1}^{L}\frac{\left(v_{k,j} - \mu_{k,j}\right)^2}{\sigma^2_{k,j}} \quad (10)$$

The Bayes theorem can be employed to compute the probability that the observed feature vector $\mathbf{v}_k$ belongs to the background of the scene:

$$P(B \mid \mathbf{v}_k) = \frac{P(B)\,p(\mathbf{v}_k \mid B)}{P(B)\,p(\mathbf{v}_k \mid B) + P(F)\,p(\mathbf{v}_k \mid F)} \quad (11)$$

$$P(F \mid \mathbf{v}_k) = 1 - P(B \mid \mathbf{v}_k) \quad (12)$$

where it can be assumed that the background and the foreground are equally probable:

$$P(B) = P(F) = \frac{1}{2} \quad (13)$$

Since (11) is prone to numerical overflows, it is recommended that the following formula is used whenever $\log p(\mathbf{v}_k \mid B) \geq \log p(\mathbf{v}_k \mid F)$:

$$P(B \mid \mathbf{v}_k) = \frac{1}{1 + \exp\left(\log p(\mathbf{v}_k \mid F) - \log p(\mathbf{v}_k \mid B)\right)} \quad (14)$$

On the other hand, if $\log p(\mathbf{v}_k \mid B) < \log p(\mathbf{v}_k \mid F)$, then the following formula is recommended:

$$P(F \mid \mathbf{v}_k) = \frac{1}{1 + \exp\left(\log p(\mathbf{v}_k \mid B) - \log p(\mathbf{v}_k \mid F)\right)} \quad (15)$$

where $\log p(\mathbf{v}_k \mid B)$ is computed from (10) and $\log p(\mathbf{v}_k \mid F)$ is computed from (7).

The computational complexity of the proposed method can be obtained as follows. The input to the stacked denoising autoencoder has size $H = 3(N + 2M)^2$, which means that each pixel window is processed by the autoencoder with complexity $O\left(N^2\right)$, since $N > M$. That is, the autoencoder processing module is linear in the number of pixels, because the window contains $N \times N$ pixels. The output of the encoding part of the autoencoder has size L, and the subsequent probabilistic model processes the L-dimensional feature vectors in $O(L)$, because diagonal covariance matrices are considered in the probabilistic model. Therefore the probabilistic modeling module is also linear in the number of pixels, since $L < H$. That is, our proposal has a computational complexity which is linear in the number of pixels of the incoming video frame.

3. Experimental Results

3.1. Noise description

Our goal is to show the resilience to noise of our method, as compared to others. In order to achieve this, original video frame sequences have been altered with different types of noise to generate new noisy sequences. The comparison must be as fair as possible, thus, videos with noise are generated only once and the same video is processed with all methods. The distinct types of noise that have been considered are described below:

- **Gaussian**: the effect of Gaussian noise has been tested by means of the addition of Gaussian noise with mean $\sigma = 0$, $\sigma = 0.1$, $\sigma = 0.2$ and $\sigma = 0.31$ respectively to the original video sequence.
- **Mask**: Black squares of different sizes have been inserted in order to cover 20% of each frame. Four square sizes have been used: 1x1 (black dots), 2x2, 3x3 and 4x4.
- **Salt and pepper**: Black and white pixels have been inserted. The probability for each one to appear is 10%.
- **Uniform**: Uniform noise ranging from -0.5 to 0.5 has been added after normalizing image pixels to values in $[0, 1]$.
- **Compression**: This noise has been simulated by forcing the saving of the data set images with lower quality than the originals. [1]

3.2. Evaluation

As a measure to compare the performance from a quantitative point of view, the well-known *F-score* (also noted as *F-measure* or F_1 score) has been considered. It is defined as a balanced harmonic mean of precision and recall and provides values in the interval $[0, 1]$, where higher is better. *F-score* has been calculated for each binary foreground mask (each segmented frame) in the region of interest that is specified by ChangeDetection.net. After that, for each method, the *F-score* values for all the frames with foreground objects corresponding to the same video have been averaged.

In order to evaluate the overall performance and get a ranking, the average *F-score* for all tested videos for each method is calculated. Then, the sum of ranks for each method is computed. The method which obtains the minimum sum is considered the best.

[1] The IMWRITE_JPEG_QUALITY parameter of the *imwrite* function from OpenCV library for Python was used to alter saving quality. It accepts values from 0 to 100 (the higher, the better quality) and its default value is 95. Sequences with values 10, 5 and 1 were generated.

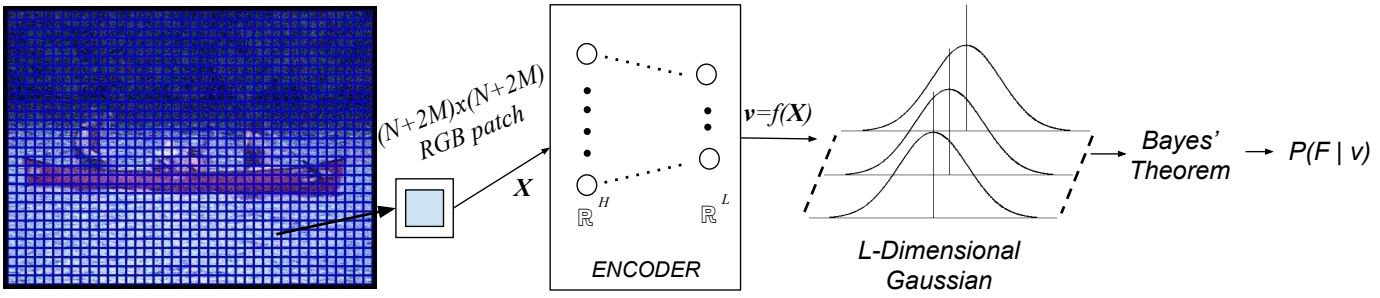
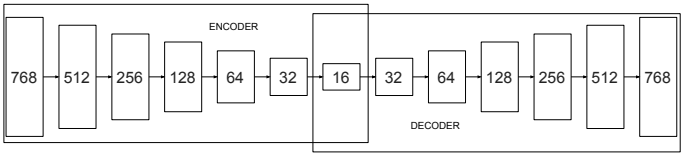
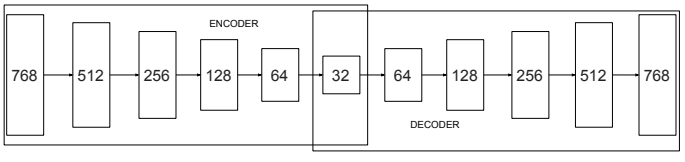


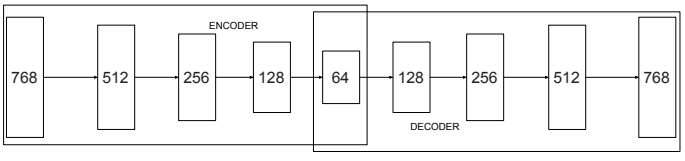
Fig. 1: Frame processing. For each NxN patch of the image a 2Mx2M margin is added. Then, these data X are inputted to the encoder and a feature vector v is obtained. Finally, v is compared to the L-dimensional Gaussian model for that patch using Bayes' Theorem and the probability of belonging to the foreground is obtained.



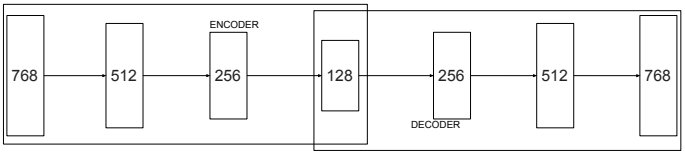
(a) $L = 16$.



(b) $L = 32$.



(c) $L = 64$.



(d) $L = 128$.

Fig. 2: Autoencoder structure with different innermost layer size (L).

3.3. Autoencoder architecture and parameter selection

We have created several architectures for the autoencoder by changing the number of layers and their sizes. As a result, the size of the encoder narrowest layer changes. This layer is crucial, because it determines the feature vector size L that our probabilistic model uses. Four autoencoder architectures with $L = 16$, $L = 32$, $L = 64$ and $L = 128$ have been tested in an attempt of knowing which feature vector size works better and seems to represent more faithfully the significant features from patches. We can observe those structures on Figure 2. In all cases, the patch size has been $16x16$, which is determined by setting $N = 8$ and $M = 4$.

Table 1 on page 5 shows method average *F-score* with each encoder for each sequence and noise type which we have tested. We can observe that the method using an encoder with 16 as narrowest layer size has much better results than any other tested encoder (it has the best score in 23 out of 42 cases). Therefore, and encoder with $L = 16$ is selected by default. The first column shows the probabilistic model performance when the patches are inputted directly to it and no autoencoder is used as a previous step of the processing. It can be noted that the results are clearly worse than those obtained by the proposed method when video sequences are noisy. Therefore, the autoencoder can be considered an essential part of the presented method and its effectiveness in finding significant features which contribute to robustness seems to be beyond doubt.

After fixing the encoder output layer size to $L = 16$, α is the only parameter which the user has to define. If it is set to a small value then the training process is more likely to converge to some of the minima. Thus, we have run our method with $\alpha \in \{0.001, 0.005, 0.01, 0.05\}$ for each video and the configuration showing the best score has been selected.

3.4. Methods

Ten methods have been selected to make a performance comparison with our proposal. Seven of these methods, namely, *Wren* (Wren et al. (1997)), *Zivkovic* (Zivkovic and van der Heijden (2006)), *SOBS* (Maddalena and Petrosino (2008)), *KDE* (Elgammal et al. (2000)), *SuBSENSE* (St-Charles et al. (2015)), *PAWCS* (St-Charles et al. (2016)) and *LOBSTER* (St-Charles and Bilodeau (2014)) have been obtained from BGS library (Sobral and Bouwmans (2014))[2]. *SC-SOBS* executable has been obtained from CVPRLAB web[3] The proposed approach has been implemented using Python. Specifically, the neural network implementation makes use of the high-level API Keras[4], which is based on TensorFlow[5]. The aforementioned four autoencoder architectures have been trained and tested using the same 100,000 random images from Tiny Images data set (Torralba et al. (2008)) [6]. Since each image has 32x32 pixels, we have divided each one to obtain four 16x16 images. We do not use any additional post processing in any of the methods.

[2]https://github.com/andrewssobral/bgslibrary

[3]http://cvprlab.uniparthenope.it/index.php/code/moving-object-detection-software-2.html. The selected parameters values are those indicated as default by the authors or used by default in BGS library and CVPRLAB code.

[4]https://keras.io/

[5]https://www.tensorflow.org/

[6]http://groups.csail.mit.edu/vision/TinyImages/

Table 1: Proposed method version comparison with different noises. The score shows the average F-score through α value.

Average F-score with no noise for each video					
	no autoencoder	L=16	L=32	L=64	L=128
overpass	**0.391 ± 0.19**	0.375 ± 0.19	0.353 ± 0.19	0.380 ± 0.19	0.277 ± 0.17
pedestrians	0.367 ± 0.20	0.477 ± 0.11	0.436 ± 0.11	0.465 ± 0.11	**0.485 ± 0.12**
canoe	0.321 ± 0.14	**0.707 ± 0.14**	0.342 ± 0.18	0.428 ± 0.22	0.377 ± 0.19
port-17fps	0.091 ± 0.14	0.108 ± 0.14	0.113 ± 0.14	**0.118 ± 0.14**	0.077 ± 0.08
boats	0.432 ± 0.19	**0.548 ± 0.10**	0.522 ± 0.12	0.510 ± 0.11	0.459 ± 0.10
fountain01	0.159 ± 0.15	0.190 ± 0.16	0.191 ± 0.17	0.244 ± 0.19	**0.286 ± 0.17**
fountain02	0.215 ± 0.21	0.469 ± 0.18	0.411 ± 0.20	0.493 ± 0.19	**0.535 ± 0.17**
Average F-score for each video with Gaussian noises					
	no autoencoder	L=16	L=32	L=64	L=128
overpass	0.350 ± 0.22	**0.499 ± 0.21**	0.474 ± 0.20	0.465 ± 0.19	0.397 ± 0.20
pedestrians	0.401 ± 0.22	**0.524 ± 0.10**	0.471 ± 0.15	0.491 ± 0.15	0.507 ± 0.11
canoe	0.258 ± 0.13	**0.625 ± 0.16**	0.301 ± 0.16	0.383 ± 0.18	0.356 ± 0.17
port-17fps	0.090 ± 0.14	0.125 ± 0.15	0.130 ± 0.15	**0.140 ± 0.15**	0.086 ± 0.10
boats	0.108 ± 0.09	**0.520 ± 0.14**	0.466 ± 0.17	0.433 ± 0.15	0.409 ± 0.10
fountain01	0.079 ± 0.12	0.190 ± 0.17	0.223 ± 0.20	**0.237 ± 0.21**	0.217 ± 0.15
fountain02	0.141 ± 0.16	**0.487 ± 0.17**	0.426 ± 0.20	0.448 ± 0.20	0.426 ± 0.18
Average F-score for each video with mask noises					
	no autoencoder	L=16	L=32	L=64	L=128
overpass	0.408 ± 0.22	0.411 ± 0.23	**0.439 ± 0.22**	0.417 ± 0.22	0.323 ± 0.21
pedestrians	0.218 ± 0.17	0.288 ± 0.08	**0.359 ± 0.13**	0.356 ± 0.13	0.201 ± 0.08
canoe	0.166 ± 0.09	**0.418 ± 0.19**	0.274 ± 0.16	0.249 ± 0.14	0.286 ± 0.15
port-17fps	**0.082 ± 0.14**	0.063 ± 0.08	0.080 ± 0.10	0.065 ± 0.08	0.021 ± 0.03
boats	0.032 ± 0.05	**0.228 ± 0.12**	0.216 ± 0.12	0.171 ± 0.11	0.112 ± 0.05
fountain01	0.111 ± 0.16	0.098 ± 0.11	**0.122 ± 0.14**	0.118 ± 0.14	0.054 ± 0.06
fountain02	0.074 ± 0.10	**0.288 ± 0.14**	0.256 ± 0.15	0.199 ± 0.12	0.090 ± 0.06
Average F-score for each video with salt and pepper noise					
	no autoencoder	L=16	L=32	L=64	L=128
overpass	0.024 ± 0.05	**0.391 ± 0.21**	0.331 ± 0.18	0.275 ± 0.18	0.258 ± 0.20
pedestrians	0.178 ± 0.21	**0.484 ± 0.15**	0.319 ± 0.21	0.422 ± 0.18	0.322 ± 0.12
canoe	0.130 ± 0.07	**0.280 ± 0.15**	0.149 ± 0.11	0.187 ± 0.12	0.100 ± 0.06
port-17fps	0.037 ± 0.10	0.105 ± 0.14	**0.113 ± 0.15**	0.084 ± 0.12	0.024 ± 0.04
boats	0.003 ± 0.01	**0.250 ± 0.16**	0.198 ± 0.15	0.182 ± 0.13	0.235 ± 0.10
fountain01	0.000 ± 0.00	**0.133 ± 0.17**	0.095 ± 0.14	0.079 ± 0.12	0.064 ± 0.07
fountain02	0.006 ± 0.02	**0.284 ± 0.19**	0.212 ± 0.20	0.173 ± 0.16	0.125 ± 0.09
Average F-score for each video with uniform noise					
	no autoencoder	L=16	L=32	L=64	L=128
overpass	0.252 ± 0.20	**0.512 ± 0.21**	0.443 ± 0.20	0.422 ± 0.19	0.338 ± 0.20
pedestrians	0.378 ± 0.24	**0.526 ± 0.10**	0.422 ± 0.19	0.434 ± 0.19	0.437 ± 0.13
canoe	0.181 ± 0.10	**0.534 ± 0.19**	0.246 ± 0.14	0.311 ± 0.16	0.295 ± 0.15
port-17fps	0.071 ± 0.13	0.130 ± 0.15	0.135 ± 0.16	**0.140 ± 0.16**	0.069 ± 0.09
boats	0.012 ± 0.03	**0.455 ± 0.18**	0.412 ± 0.20	0.341 ± 0.18	0.318 ± 0.11
fountain01	0.021 ± 0.06	0.177 ± 0.18	**0.190 ± 0.21**	0.178 ± 0.20	0.115 ± 0.13
fountain02	0.034 ± 0.09	**0.471 ± 0.18**	0.383 ± 0.21	0.379 ± 0.22	0.278 ± 0.16
Average F-score for each video with compression noise levels					
	no autoencoder	L=16	L=32	L=64	L=128
overpass	0.237 ± 0.17	**0.317 ± 0.18**	0.289 ± 0.18	0.280 ± 0.18	0.239 ± 0.17
pedestrians	0.360 ± 0.20	0.445 ± 0.12	0.398 ± 0.12	0.417 ± 0.12	**0.447 ± 0.13**
canoe	0.294 ± 0.16	**0.653 ± 0.17**	0.328 ± 0.21	0.406 ± 0.24	0.368 ± 0.20
port-17fps	0.074 ± 0.13	0.098 ± 0.14	0.101 ± 0.14	**0.108 ± 0.14**	0.069 ± 0.09
boats	0.342 ± 0.14	0.474 ± 0.11	**0.518 ± 0.12**	0.500 ± 0.11	0.427 ± 0.10
fountain01	0.071 ± 0.10	0.161 ± 0.15	0.159 ± 0.16	0.171 ± 0.16	**0.228 ± 0.16**
fountain02	0.136 ± 0.18	0.446 ± 0.18	0.394 ± 0.21	0.430 ± 0.20	**0.472 ± 0.20**

3.5. Video Sequences

A set of video sequences have been selected from the 2014 dataset (Goyette et al. (2012)) of the ChangeDetection.net website[7]. Five of the selected scenes are from Dynamic Background category, one from Low Frame Rate category and another one from the Baseline one. *Canoe* exhibits a river with water and forest background where a canoe goes across (320x240 pixels and 1189 frames). *Fountain01* shows a fountain with various vertical water springs next to a road (432x288 pixels and 1184 frames), whereas a road behind a fountain that spits water out (432x288 pixels and 1499 frames) is displayed in *Fountain02*. *Boats* shows a river next to a road. Two boats cross through the river while various vehicles move on the road (320x240 pixels and 7999 frames). *Overpass* shows a bridge traversed by a man with a river, forest and a road behind (320x240 pixels and 3000 frames). *Port_0_17fps* is a low frame rate video that exhibits a dock with boats constantly moving, water and clouds as background and some persons and boats crossing from time to time as foreground (640x480 pixels and 3000 frames). *Pedestrians* is a baseline video where several people walk over a pavement next to grass with sun and shadows (360x240 pixels and 1099 frames).

3.6. Results

The quantitative results of each method on the previous sequences affected by noise are displayed in tables from 2 to 3. It can be observed that our approach is able to deal with Gaussian noise without too much loss, leading to the best performance in 5 of 13 tests. The *mask* noise includes an additional difficulty, specially when the square size grows. On average the other methods deal with it better, although the proposed one is the best with 1x1 square mask noise and reaches the third position with squares of size 2x2 and 4x4. With respect to *uniform* and *salt and pepper* noises, the results show that our method performance is, on average, the highest one. Although our proposal is not the best for the *compression* noise, the final accuracy is between the best three techniques with similar results to the best approach (SUBSENSE).

4. Discussion

First, it can be said that the intrinsic dimension of the patches is near to 16, in view of the results reported in Table 1. This implies that we managed to reduce patches of 16x16 pixels, in total 256 values, down to 16 values, leaving only 6.25% of the input information. The performance achieved when that feature vector size is selected is the best in most cases. An architecture with more neurons in the last encoder layer determines a feature vector with more components, which makes the task of estimating the probability density harder. In addition to that, some of those extra components, if not all, could be representing irrelevant characteristics derived from the effect of noise.

Zivkovic, KDE and Wren seem to be immune to square size in mask noise, because they do not take into account pixel neighborhood information. Therefore, only the percentage of erroneous pixels affects them. The other methods, including the proposed one, present worse performance when the size of the square is higher. As the input to our system is composed of patches, bigger squares can make the patches differ greatly from the original ones, thus, making the probabilistic model to classify them incorrectly. Even though the presented algorithm is more sensitive to mask noise than to other types of noise, it is ranked among the first four positions according to its $F - score$ value in that case.

Focusing on the ranking score, we can observe that, apart from our method, LOBSTER, PAWCS and SuBSENSE achieve good performance as well. Each one of them works well with a particular kind of noise. SuBSENSE behaves very well with compression noise but suffers a fall in performance with salt-pepper and in a minor degree with mask noise. LOBSTER works fine on average, being mostly affected by Gaussian and Uniform noise. PAWCS excels at images with Gaussian, salt-pepper and uniform noise, but its performance decays when mask or compression noise is present. On the other hand, our method shows a more robust behavior on average. It also reaches the top of the rank in most of the tests, despite the fact that it is a region-level based method, thus, having less training samples from which to extract information than the competing methods.

From a qualitative point of view (Figure 3), the effect of noise addition results in the appearance of many more FP pixels (spurious foreground objects), except for our method and PAWCS, or in the loss of foreground objects, as happens with SubSENSE.

5. Conclusions

Noise resilience is an essential feature a segmentation algorithm must exhibit, due to the fact that a video sequence acquired by a real camera is influenced by multiple conditions that may affect the quality of the images. Therefore, an algorithm able to precisely detect foreground objects in noisy images is a key part in a video surveillance system, making the task of subsequent processes easier and allowing the overall system performance to remain high.

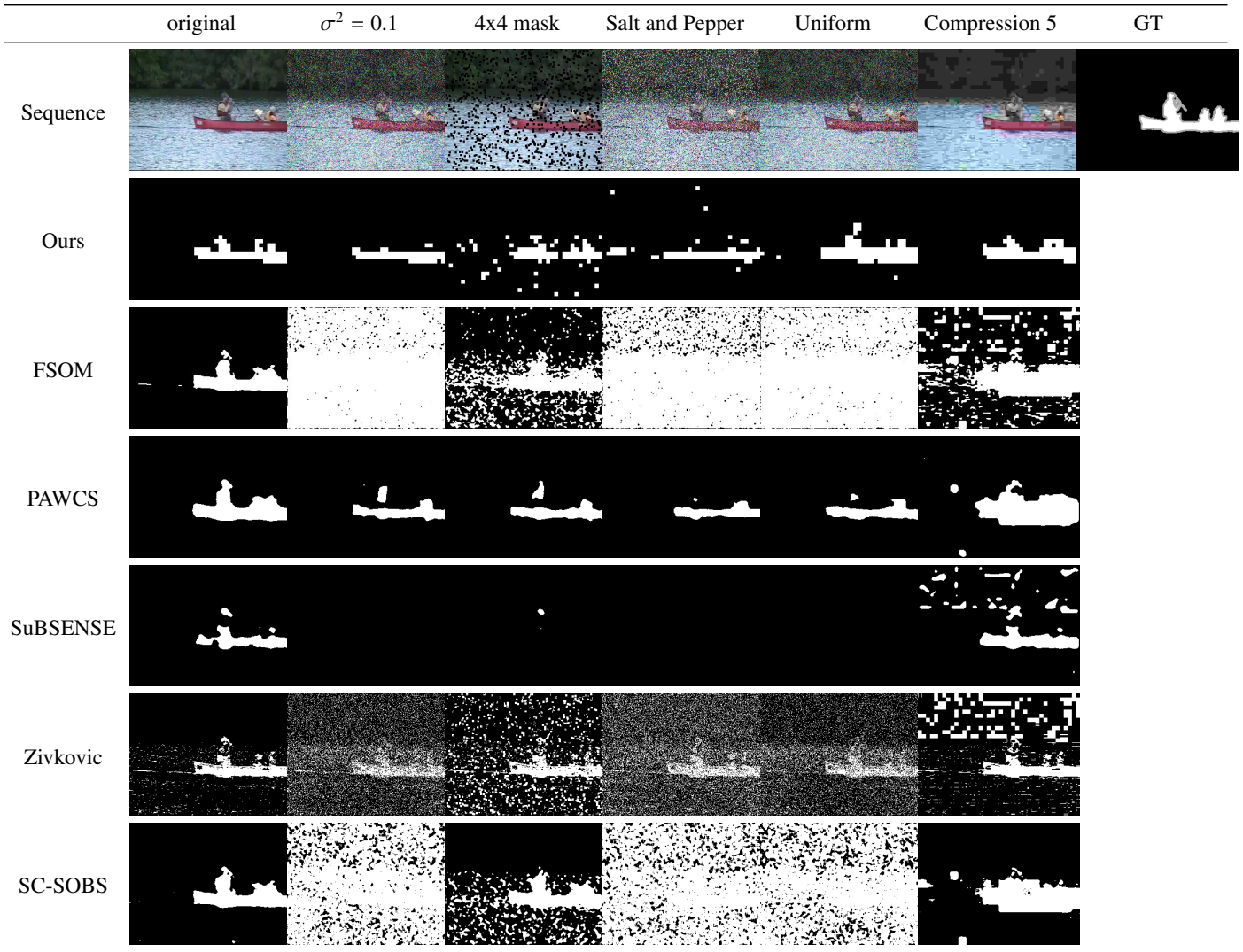
This paper presents a robust foreground detection method that combines a stacked denoising autoencoder with a probabilistic model consisting of a mixture of Gaussians. Tests on seven different images, affected by noise of distinct type and intensity, have been carried out. The proposal has been compared with other ten segmentation algorithms and a ranking has been made according to the yielded $F - score$ value. Not only does the proposed method show a high performance on average, but it can also be claimed the best one regarding the sum of ranks. Hence, the suitability of our approach has been experimentally validated.

Acknowledgments

This work is partially supported by the Ministry of Science, Innovation and Universities of Spain [grant number RTI2018-094645-B-I00], project name Automated detection with low

[7] http://changedetection.net/

Fig. 3: Qualitative results for canoe frame 1008. Type of noise added, from left to right: none, Gaussian with mean 0 and $\sigma^2 = 1$, 4x4 black squares, Salt and Pepper, uniform, and compression with IMWRITE_JPEG_QUALITY 5. First row displays the original input image with different noises and ground-truth. Other rows show foreground detected by each method.



cost hardware of unusual activities in video sequences. It is also partially supported by the Autonomous Government of Andalusia (Spain) [grant number TIC-657], project name Self-organizing systems and robust estimators for video surveillance. Both of them include funds from the European Regional Development Fund (ERDF). The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinformatics) center of the University of Málaga. They have also been supported by the Biomedic Research Institute of Málaga (IBIMA). They also gratefully acknowledge the support of NVIDIA Corporation with the donation of two Titan X GPUs.

References

Elgammal, A., Harwood, D., Davis, L., 2000. Non-parametric model for background subtraction, in: Computer Vision (ECCV), Springer. pp. 751–767.

Goyette, N., Jodoin, P.M., Porikli, F., Konrad, J., Ishwar, P., et al., 2012. Changedetection. net: A new change detection benchmark dataset., in: CVPR Workshops, pp. 1–8.

López-Rubio, E., Luque-Baena, R., Domínguez, E., 2011. Foreground detection in video sequences with probabilistic self-organizing maps. International Journal of Neural Systems 21, 225–246.

López-Rubio, E., Molina-Cabello, M.A., Luque-Baena, R.M., Domínguez, E., 2018a. Foreground detection by competitive learning for varying input distributions. International Journal of Neural Systems 28, 1750056.

López-Rubio, F., López-Rubio, E., Molina-Cabello, M., Luque-Baena, R., Palomo, E., Domínguez, E., 2018b. The effect of noise on foreground detection algorithms. Artificial Intelligence Review 49, 407–438.

Maddalena, L., Petrosino, A., 2008. A self-organizing approach to background subtraction for visual surveillance applications. IEEE Transactions on Image Processing 17, 1168–1177.

Maddalena, L., Petrosino, A., 2012. The sobs algorithm: What are the limits?, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pp. 21–26.

Robbins, H., Monro, S., 1951. A stochastic approximation method. The Annals of Mathematical Statistics 22, 400–407.

Sobral, A., Bouwmans, T., 2014. Bgs library: A library framework for algorithm's evaluation in foreground/background segmentation, in: Background Modeling and Foreground Detection for Video Surveillance. CRC Press, Taylor and Francis.

St-Charles, P., Bilodeau, G., 2014. Improving background subtraction using local binary similarity patterns, in: IEEE Winter Conference on Applications of Computer Vision, pp. 509–515.

Table 2: Average F-score over all videos for each method and noise. Number in parentheses shows each method ranking for that particular type of noise. Last line presents each method sum of ranks score (the lesser, the better).

Noise	Ours	Zivkovic	KDE	Wren	SOBS	FSOM
Gaussian 0	0.514 ± 0.20(7)	0.456 ± 0.23(8)	0.147 ± 0.11(11)	0.393 ± 0.21(9)	0.288 ± 0.17(10)	0.593 ± 0.24(3)
Gaussian 0.1	0.518 ± 0.20(2)	0.350 ± 0.20(8)	0.071 ± 0.05(11)	0.242 ± 0.15(9)	0.126 ± 0.09(10)	0.369 ± 0.22(7)
Gaussian 0.2	**0.509 ± 0.20**(1)	0.153 ± 0.11(5)	0.051 ± 0.04(11)	0.098 ± 0.07(6)	0.067 ± 0.05(8)	0.063 ± 0.05(9)
Gaussian 0.31	**0.486 ± 0.20**(1)	0.091 ± 0.08(5)	0.046 ± 0.04(9)	0.063 ± 0.05(6)	0.053 ± 0.04(7)	0.045 ± 0.04(10)
mask	**0.463 ± 0.18**(1)	0.125 ± 0.10(8)	0.122 ± 0.09(9)	0.068 ± 0.06(11)	0.121 ± 0.09(10)	0.440 ± 0.24(2)
mask 2x2	0.358 ± 0.18(3)	0.130 ± 0.10(8)	0.124 ± 0.09(9)	0.074 ± 0.06(11)	0.122 ± 0.09(10)	0.381 ± 0.22(2)
mask 3x3	0.263 ± 0.17(4)	0.131 ± 0.10(8)	0.124 ± 0.09(9)	0.076 ± 0.06(11)	0.122 ± 0.09(10)	0.301 ± 0.21(3)
mask 4x4	0.218 ± 0.16(3)	0.131 ± 0.10(8)	0.124 ± 0.09(9)	0.076 ± 0.06(11)	0.123 ± 0.09(10)	**0.226 ± 0.18**(1)
saltpepper	**0.387 ± 0.14**(1)	0.080 ± 0.07(4)	0.052 ± 0.04(7)	0.059 ± 0.05(5)	0.056 ± 0.04(6)	0.049 ± 0.04(9)
uniform	**0.487 ± 0.20**(1)	0.095 ± 0.08(4)	0.052 ± 0.04(8)	0.063 ± 0.05(6)	0.053 ± 0.04(7)	0.046 ± 0.04(10)
compression 10	0.505 ± 0.20(3)	0.412 ± 0.21(7)	0.189 ± 0.14(11)	0.380 ± 0.20(9)	0.241 ± 0.13(10)	0.440 ± 0.23(6)
compression 5	0.501 ± 0.21(3)	0.230 ± 0.15(8)	0.169 ± 0.12(9)	0.365 ± 0.20(5)	0.133 ± 0.09(11)	0.231 ± 0.13(7)
compression 1	0.416 ± 0.18(2)	0.184 ± 0.11(8)	0.157 ± 0.09(10)	0.351 ± 0.19(5)	0.162 ± 0.10(9)	0.247 ± 0.16(7)
Sum of Ranks	**32**	89	123	104	118	76

Table 3: Table 2 continues.Average F-score over all videos for each method and noise. Last line presents each method sum of ranks score (the lesser, the better)

Noise	Ours	CL-VID	LOBSTER	PAWCS	SC-SOBS	SuBSENSE
Gaussian 0	0.514 ± 0.20(7)	0.539 ± 0.28(6)	0.579 ± 0.19(5)	**0.662 ± 0.16**(1)	0.588 ± 0.28(4)	0.635 ± 0.16(2)
Gaussian 0.1	0.518 ± 0.20(2)	0.483 ± 0.28(5)	0.491 ± 0.24(4)	**0.610 ± 0.24**(1)	0.421 ± 0.23(6)	0.509 ± 0.19(3)
Gaussian 0.2	**0.509 ± 0.20**(1)	0.060 ± 0.05(10)	0.300 ± 0.29(3)	0.370 ± 0.31(2)	0.069 ± 0.06(7)	0.238 ± 0.25(4)
Gaussian 0.31	**0.486 ± 0.20**(1)	0.043 ± 0.04(11)	0.095 ± 0.09(4)	0.299 ± 0.29(2)	0.048 ± 0.04(8)	0.123 ± 0.22(3)
mask	**0.463 ± 0.18**(1)	0.414 ± 0.23(3)	0.339 ± 0.22(6)	0.409 ± 0.26(5)	0.411 ± 0.25(4)	0.314 ± 0.20(7)
mask 2x2	0.358 ± 0.18(3)	0.291 ± 0.17(5)	**0.450 ± 0.20**(1)	0.334 ± 0.27(4)	0.281 ± 0.20(6)	0.268 ± 0.18(7)
mask 3x3	0.263 ± 0.17(4)	0.175 ± 0.11(7)	0.313 ± 0.16(2)	**0.319 ± 0.27**(1)	0.201 ± 0.15(6)	0.208 ± 0.16(5)
mask 4x4	0.218 ± 0.16(3)	0.131 ± 0.09(8)	**0.226 ± 0.14**(1)	0.202 ± 0.26(5)	0.168 ± 0.14(6)	0.203 ± 0.13(4)
saltpepper	**0.387 ± 0.14**(1)	0.044 ± 0.04(11)	0.087 ± 0.08(3)	0.229 ± 0.25(2)	0.050 ± 0.04(8)	0.047 ± 0.10(10)
uniform	**0.487 ± 0.20**(1)	0.043 ± 0.04(11)	0.089 ± 0.08(5)	0.317 ± 0.29(2)	0.049 ± 0.04(9)	0.139 ± 0.23(3)
compression 10	0.505 ± 0.20(3)	0.492 ± 0.26(4)	0.561 ± 0.18(2)	0.472 ± 0.22(5)	0.404 ± 0.22(8)	**0.620 ± 0.14**(1)
compression 5	0.501 ± 0.21(3)	0.138 ± 0.10(10)	0.510 ± 0.20(2)	0.398 ± 0.19(4)	0.292 ± 0.19(6)	**0.517 ± 0.19**(1)
compression 1	0.416 ± 0.18(2)	0.126 ± 0.08(11)	0.407 ± 0.21(3)	0.397 ± 0.20(4)	0.285 ± 0.17(6)	**0.468 ± 0.15**(1)
Sum of ranks	**32**	102	41	38	84	51

St-Charles, P., Bilodeau, G., Bergevin, R., 2015. Subsense: A universal change detection method with local adaptive sensitivity. IEEE Transactions on Image Processing 24, 359–373.

St-Charles, P., Bilodeau, G., Bergevin, R., 2016. Universal background subtraction using word consensus models. IEEE Transactions on Image Processing 25, 4768–4781.

Torralba, A., Fergus, R., Freeman, W., 2008. 80 million tiny images: A large data set for nonparametric object and scene recognition. IEEE Trans. on Pattern Analysis and Machine Intelligence 30, 1958–1970.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P., 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research 11, 3371–3408.

Wang, Y., Wu, L., Lin, X., Gao, J., 2018. Multiview spectral clustering via structured low-rank matrix factorization. IEEE Transactions on Neural Networks and Learning Systems 29, 4833–4843.

Wren, C., Azarbayejani, A., Darrell, T., Pentl, A., 1997. Pfinder: Real-time tracking of the human body. IEEE Trans. on Pattern Analysis and Machine Intelligence 19, 780–785.

Wu, L., Hong, R., Wang, Y., Wang, M., 2019a. Cross-entropy adversarial view adaptation for person re-identification. IEEE Transactions on Circuits and Systems for Video Technology .

Wu, L., Wang, Y., Gao, J., Li, X., 2018a. Deep adaptive feature embedding with local sample distributions for person re-identification. Pattern Recognition 73, 275–288.

Wu, L., Wang, Y., Gao, J., Li, X., 2018b. Where-and-when to look: Deep siamese attention networks for video-based person re-identification. IEEE Transactions on Multimedia .

Wu, L., Wang, Y., Li, X., Gao, J., 2018c. Deep attention-based spatially recursive networks for fine-grained visual recognition. IEEE transactions on cybernetics , 1–12.

Wu, L., Wang, Y., Li, X., Gao, J., 2018d. What-and-where to match: deep spatially multiplicative integration networks for person re-identification. Pattern Recognition 76, 727–738.

Wu, L., Wang, Y., Shao, L., Wang, M., 2019b. 3-d personvlad: Learning deep global representations for video-based person reidentification. IEEE transactions on neural networks and learning systems .

Zivkovic, Z., van der Heijden, F., 2006. Efficient adaptive density estimation per image pixel for the task of background subtraction. Pattern Recognition Letters 27, 773–780.