Autoencoder-based background reconstruction and foreground segmentation with background noise estimation

Bruno Sauvalle Arnaud de La Fortelle
Centre de Robotique, Mines ParisTech PSL University
{bruno.sauvalle, arnaud.de-la.fortelle}@mines-paristech.fr

Abstract

Even after decades of research, dynamic scene background reconstruction and foreground object segmentation are still considered as open problems due to various challenges such as illumination changes, camera movements, or background noise caused by air turbulence or moving trees. We propose in this paper to model the background of a frame sequence as a low dimensional manifold using an autoencoder and compare the reconstructed background provided by this autoencoder with the original image to compute the foreground/background segmentation masks. The main novelty of the proposed model is that the autoencoder is also trained to predict the background noise, which allows to compute for each frame a pixel-dependent threshold to perform the foreground segmentation. Although the proposed model does not use any temporal or motion information, it exceeds the state of the art for unsupervised background subtraction on the CDnet 2014 and LASIESTA datasets, with a significant improvement on videos where the camera is moving. It is also able to perform background reconstruction on some non-video image datasets.

1. Introduction

We consider in this paper the tasks of dynamic background reconstruction and foreground/background segmentation, which can be described in the following way: The input is a sequence $X$ of consecutive frames $X_1, \ldots, X_N$ showing a scene cluttered by various moving objects, such as cars or pedestrians, and the expected output is a sequence $\hat{X} = \hat{X}_1, \ldots, \hat{X}_N$ of frames showing the backgrounds of each scene without those objects.

The foreground/background segmentation task similarly takes as input the same kind of frames sequence $X_1, \ldots, X_N$, but the expected output is a sequence $M$ of foreground masks $M_1, \ldots, M_N$ whose values at the pixel $p$ are equal to zero if this pixel shows the background in the considered frame, and equal to 1 if the background is masked by a foreground moving object at this pixel (Fig. 1). This task is often called background subtraction because the pointwise multiplication of the mask $M_k$ and the input image $X_k$ gives an image showing only the foreground moving objects present in $X_k$, the input image background being replaced by a black background.

Background subtraction is a fundamental tool in image analysis and has been studied for more than 30 years [67], but is still considered an open problem due to the various challenges appearing in real applications: illumination changes, high level of occlusion of the background, background motions caused by moving trees or water, challenging weather conditions, presence of shadows, etc. The applications of background subtraction are very diverse [19]: road, airport, store, maritime or military surveillance, observation of animals and insects, motion capture, human-computer interface, video matting, fire detection, etc.

The model presented in this paper starts from the classi-
Supervised models can reach very high accuracy results on a supervised semantic segmentation model [7, 71]. Although supervised by combining its results with the output of a supervised background subtraction model [50, 49]. A background subtraction model can be substantially improved by using the output of an unsupervised background subtraction model [62] to improve generalization. One can also use additional input to the deep learning model the output of an unsupervised background subtraction model [50, 49]. A background subtraction model can be substantially improved by combining its results with the output of a supervised semantic segmentation model [7, 71]. Although supervised models can reach very high accuracy results on a given video after labeling a significant number of frames of this video and training the model with these labeled data, their ability to generalize to new videos remain a major issue, and evaluations on unseen scenes lead to unfavorable results compared to unsupervised algorithms [43]. As a consequence, existing supervised models are not suited for real world applications where it is not possible to provide annotated data for each new input video.

One can classify unsupervised methods as statistical methods or reconstruction methods.

Statistical methods rely on a statistical modeling of the distribution of background pixel color values or other local features to predict whether a particular pixel is foreground or background. These statistical models can be parametric (univariate gaussian [67], mixture of gaussians [60], clusters [37], Student’s t-distributions [46], Dirichlet process mixture models [6], Poisson mixture models [18], asymmetric generalized gaussian mixture models [15], etc.) or non-parametric (pixel value histograms [72], kernel density estimation [14], codebooks [32], history of recently observed pixels [3, 24], etc.). The efficiency of these methods can be increased by using as input not only the pixel color values, but also features attached to superpixels [11] or local descriptors which are robust to illumination changes, such as SIFT [56], LBP or LBSP descriptors [58, 59]. If the camera is static, the segmentation of moving objects on a scene can also be performed by evaluating the motion associated to each pixel, using optical flow or flux tensor models. The blobs produced by these models are generally very fuzzy, but can be used as input to more complex models [8, 65].

Reconstruction methods use a background reconstruction model to predict the color (or other features) of the background at a particular pixel. The difference between the current image and the predicted background is then computed and followed by a thresholding to decide whether a pixel is background or foreground. Pixelwise reconstruction models try to predict the value of a background pixel at a particular frame from the sequence of values of the pixel of the last frames using a filter, which can be a Wiener filter [63], a Kalman filter [52] or a Chebychev filter [10]. A global prediction of the background can also be performed using the assumption that the background frames form a low dimensional manifold, which motivates the use of dimensionality reduction techniques such as principal component analysis (PCA) [48]. One can add to this approach a prior on the sparsity of the foreground objects by using a $L_1$ loss term applied to the foreground residuals, which leads to the development of models based on robust principal component analysis (RPCA) [68, 9]. More complex norms and additional regularizers have been proposed to improve the performance of this approach [42, 38, 70, 27, 26]. Non-linear dimensionality reduction using an autoencoder for background reconstruction has been proposed in [17, 51].
and is further developed in the proposed model. Several unsupervised models can be also combined to form a more accurate model, such as the IUTIS-5 models, which is an ensemble model combining 5 different unsupervised models [5].

**Background noise estimation** Explicit background noise estimation for foreground segmentation has been introduced in [25]. Estimating the prediction uncertainty of a deep learning model is usually implemented using a negative log-likelihood loss function associated to a probabilistic model which includes a variance or concentration parameter [47, 31, 2, 45, 54].

3. **Model description**

The proposed model is a reconstruction model and has a general structure similar to the DeepPBM model [17]: We assume that the background frames form a low dimensional manifold and train an autoencoder to learn this manifold from the complete video. We however observe that the DeepPBM model described in [17] is not really unsupervised since it requires a significant engineering and optimization work for each new video, which is incompatible with any real-world application: The structure of the autoencoder and the number of latent variables have to be defined and fine-tuned on a scene by scene basis, which can be considered as a form of supervision. One also remarks that if the number of latent variables is too high, the autoencoder quickly learns to reproduce the foreground objects, a phenomenon we call overfitting, and fails to generate a proper background.

The model proposed in this paper is fully unsupervised: It uses a constant set of hyperparameter, and the structure of the autoencoder, which depends on the size of the image and on the complexity of the background, is defined automatically without human supervision.

3.1. **Reconstruction loss using background bootstrapping**

We implement a reconstruction loss using background bootstrapping, adapted from [53]. In the case of dynamic background reconstruction, this loss function allows to reduce the risk of overfitting to the foreground objects by giving a higher weight to background pixels than to foreground pixels during the optimization process. This loss is more robust to outliers than the $L_1$ loss which gives the same weight to small and large errors. The proposed reconstruction loss can be described by the following formulae [53]: We note $x_{n,c,i,j}$ the pixel color value of the image $X_n$ for the channel $c$ at the position $(i, j)$ with $1 \leq c \leq 3$, $1 \leq i \leq h$ and $1 \leq j \leq w$, and $\hat{x}_{n,c,i,j}$ the pixel value of the reconstructed background $\hat{X}_n$ for the same channel and position. The local $L_1$ error associated to the pixel $(i, j)$ is

$$l_{n,i,j} = \sum_{c=1}^{3} |\hat{x}_{n,c,i,j} - x_{n,c,i,j}|.$$  \hspace{1cm} (1)

The soft foreground masks and spatially smoothed soft foreground masks are defined by the equations

$$m_{n,i,j} = \tanh \left( \frac{l_{n,i,j}}{\tau_1} \right)$$  \hspace{1cm} (2)

and

$$\tilde{m}_{n,i,j}(\hat{X}_n, X_n) = \frac{1}{(2k+1)^2} \sum_{l=-k}^{k} \sum_{p=-k}^{k} m_{n,i+l,j+p},$$  \hspace{1cm} (3)

where $\tau_1$ and $r$ are positive hyperparameters and $k = \lfloor w/r \rfloor$. The associated pixel-wise weight $w_{n,i,j}^{\text{bootstrap}}$ is then defined as

$$w_{n,i,j}^{\text{bootstrap}} = e^{-\beta \tilde{m}_{n,i,j}},$$  \hspace{1cm} (4)

where $\beta$ is another positive hyperparameter. The reconstruction loss of the auto-encoder is then computed by weighting the pixelwise $L_1$ losses $l_{n,i,j}$ using these bootstrap weights:

$$\mathcal{L}_{\text{rec}}(\hat{X}, X) = \frac{1}{Nh,w} \sum_{n=1}^{N} \sum_{i=1}^{h} \sum_{j=1}^{w} w_{n,i,j}^{\text{bootstrap}} l_{n,i,j}.$$  \hspace{1cm} (5)

The main differences between this loss function and the loss function defined in [53] is that it is a one-to-one loss, whereas the loss defined in [53] is one-to-many. It also does not use optical flow weights or abnormal image weights. Using optical flow weights would not allow to handle images taken from a moving camera, since it would give a low weight to all pixels associated to the moving background. We do not use abnormal image weights because we want the model to accurately reconstruct the background for each input image, which was not the case in [53], which is dedicated to fixed background reconstruction.

3.2. **Optimized thresholding using background noise estimation**

We remark that the bootstrap pixel weights $w_{n,i,j}^{\text{bootstrap}}$ can be used to get an estimate of the level of background noise of a frame sequence, considering that these weights are close to one when the associated pixel is a background pixel, and close to zero when this is not the case.

We therefore add a fourth output channel to the autoencoder, which is dedicated to give an estimate $\hat{l}_{n,i,j}$ of the value of the $L_1$ error $l_{n,i,j}$ for each pixel $(i, j)$ for the frame $X_n$ (Fig. 2).
The associated loss function is weighted using the bootstrap weights in order to limit its scope to background regions:

$$\mathcal{L}_{\text{noise}} = \frac{1}{3NHw} \sum_{n=1}^{N,h,w} \sum_{i=1,j=1}^{\text{bootstrap}} w_{n,i,j} |\hat{l}_{n,i,j} - l_{n,i,j}|$$  \hspace{1cm} (6)

When the background is very noisy, the autoencoder is not able to predict accurately the value of a background pixel color. As a consequence, the expectation of $l_{n,i,j}$ is large, which leads to a high value of $\hat{l}_{n,i,j}$. One could consider that a more principled method would be to model the background noise as a gaussian distribution and estimate the variance of this distribution by learning the weighted average $L_2$ error instead of the $L_1$ error, but we have empirically found that such an approach is not robust to the presence of foreground objects.

The autoencoder is trained using the sum of the reconstruction loss and the loss associated to the background noise estimation. The complete loss function is then

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{noise}}.$$  \hspace{1cm} (7)

The gradients of the weights $w_{n,i,j}^{\text{bootstrap}}$ are not computed during the optimization process [53]. We also do not use the gradient of $l_{n,i,j}$ in equation 6 because we do not want the quality of the background reconstruction be impacted by the background noise estimation optimization process.

In order to set the pixelwise threshold $\tau_{n,i,j}$ associated to the pixel $(i,j)$ of the frame $X_n$ and necessary to compute the background/foreground segmentation mask, we also take into account the average illumination $\hat{I}_n$ of the reconstructed background $\hat{X}_n$, as defined by the formula

$$\hat{I}_n = \frac{1}{3NHw} \sum_{c=1,i=1,j=1}^{3,h,w} |\hat{x}_{n,c,i,j}|.$$  \hspace{1cm} (8)

The threshold $\tau_{n,i,j}$ is then set according to the formula

$$\tau_{n,i,j} = \alpha_1 \hat{I}_n + \alpha_2 \hat{l}_{n,i,j},$$  \hspace{1cm} (9)

where $\alpha_1$ and $\alpha_2$ are two positive hyperparameters. The $\alpha_1$ hyperparameter can then be interpreted as the threshold applicable to a scene showing a noiseless white background. The motivation of the second term is that if the background noise is high at some pixel, we have to increase the associated threshold for background/foreground segmentation in order to prevent the misclassification of background pixels as foreground caused by background noise.

For a given frame sequence $X_1,...,X_n$ and a reconstructed background sequence $\hat{X}_1,...,\hat{X}_n$, we then compute the foreground mask $M_n$ before post-processing using the thresholding rule $M_{n,i,j} = 1$ if and only if $l_{n,i,j} > \tau_{n,i,j}$.

A post-processing is then applied in order to remove rain drops, snow flakes, and other spurious detections. It is composed of two morphological operations: a morphological closing using a $5 \times 5$ square structural element, followed by a morphological opening with a $7 \times 7$ square structural element.

### 3.3. Detecting significant background changes

The improved reconstruction loss function introduced in 3.1 reduces the risk of overfitting, but is not able to prevent it completely. We observe that the risk of overfitting increases when the number of optimization iterations and the number of parameters of the network increase. This is a significant issue because sequences showing background changes require a high number of training iterations and a model with a large number of parameters. In order to prevent overfitting, the number of training iterations and the complexity of the model are therefore adjusted to the complexity of the backgrounds sequence.

The main challenge here is to estimate without any human supervision whether the video shows substantial background changes or not. Such a task, which is very easy for a human, is far from trivial for a computer. For example, simply taking the variance of the various frames does not allow to estimate the complexity of the background changes because this variance will generally be dominated by foreground objects appearing in the video. More generally, it appears that in order to estimate the importance of the background changes, it is necessary to remove the foreground objects from the estimation process. We observe however that the proposed model can be used to perform this task. We then first train the model for a fixed small number $N_{\text{eval}}$ of iterations, which is however sufficient to
get a rough evaluation of the background changes. Using this trained model, we compute $B_{\text{eval}}$ reconstructed backgrounds $\hat{X}_n$ using frames $X_n$ sampled randomly from the sequence $\mathcal{X}$. Although these backgrounds estimates $\hat{X}_n$ are not accurate, we are confident that they do not show any foreground objects since a low number of iterations have been performed, so that the risk of overfitting is very low. We then compute the temporal median $\bar{X}$ of these backgrounds and compare this median background with the reconstructed backgrounds $\hat{X}_n$, computing soft masks $m_{n,i,j}$ following the same process as in formula 1 and 2. We then consider the average soft mask value over the $B_{\text{eval}}$ reconstructed backgrounds

$$\bar{m} = \frac{1}{B_{\text{eval}}h\sub{w}} \sum_{n,i,j} B_{\text{eval}}m_{n,i,j}. \quad (10)$$

If $\bar{m}$ is higher than a threshold $\tau_0$, we consider that the background is a complex background. The partially trained model is discarded, a new autoencoder is created with more parameters and the number of training iterations is set to $N_{\text{complex}}$ with a minimum of $E_{\text{complex}}$ epochs for very long sequences. If this ratio is lower than $\tau_0$, we consider that the background is a simple background, keep the partially trained model, and finish the training, with a total number of training iterations set to $N_{\text{simple}}$. The autoencoder structures for simple and complex backgrounds are described in the supplementary material.

4. Experimental results

4.1. Evaluation method

We consider the CDnet 2014, LASIESTA and BMC 2012 benchmark datasets for background subtraction. We use the public implementations of the algorithms PAWCS [59] and SuBSENSE [58] provided with the BGS library [57] to get baseline performance estimates for these methods when they are not available. We rely on published results for the other state of the art methods which do not provide public implementations.

We use the F-measure as main evaluation criteria. To compute the F-measure associated to a sequence of foreground masks predictions $M_1, ... , M_n$, we first compute the sums $TP, TN, FP, FN$ of the true positives, true negatives, false positives and false negatives associated to the sequence of masks $M_1, ... , M_n$, and then compute the F-measure associated to this sequence as the harmonic mean of precision and recall.

We provide in Figure 3 some samples of background reconstruction, with the associated predicted foreground mask, and a comparison with foreground masks obtained using PAWCS and SuBSENSE. Other samples are provided in the supplementary material.

4.2. CDnet 2014 dataset

The CDnet 2014 dataset [66] is composed of 53 videos, for a total of 153,278 frames, selected to cover the various challenges which have to be addressed for background subtraction: dynamic background (scenes with water or trees), camera jitter, intermittent object motion, presence of shadows, images captured by infrared cameras, challenging weather (snow, fog), images captured with a low frame rate, night images, images filmed by a pan-tilt-zoom camera, air turbulence. Ground truth foreground segmentation masks are provided for all frames of the dataset, with specific labels for shadow pixels which are not considered in the F-measure computation. We provide in Table 1 the F-measure results per category of the proposed model for each category of the CDnet 2014 dataset, with a comparison with the results obtained by other unsupervised models.

The proposed model gets a higher average F-measure on the CDnet 2014 dataset than all published unsupervised models, including ensemble models such as IUTIS-5, with an average F-measure of 0.784. One can observe a significant improvement in accuracy with the proposed model in the “pan-tilt-zoom” (PTZ) category with an average F-measure of 0.800 on this category. To our best knowledge, the proposed model is the first able to correctly handle videos taken from a moving camera.

4.3. LASIESTA dataset

The LASIESTA dataset [12] is composed of 48 videos grouped in 14 categories, for a total of 18,425 video frames. All frames are provided with ground truth pixel labels, with a specific label for pixels associated to stopped moving objects which are excluded from the F-measure computation. These videos are very short (The average number of frames per video is 383), which is challenging for the proposed deep-learning based model. We provide in Table 2 the average F-measure results of the proposed model for all 14 categories. Out of the 48 videos of the dataset, 4 videos are taken with a moving camera (categories IMC and OMC), and 24 videos include simulated camera motion (categories ISM and OSM). These 28 videos which include real or sim-
ulated camera motion are very difficult for existing background subtraction models and to our best knowledge, no paper has ever published category-wise evaluation results for these videos. In order to allow a comparison with these published results, we therefore also provide the average F-measure over the 10 categories showing only videos taken from a fixed camera. We observe that the proposed model performs better than available unsupervised algorithms on static scenes, and with a significant improvement on scenes where the camera is moving.

4.4. BMC 2012 dataset

The BMC dataset [64] contains 9 videos showing real scenes taken from static cameras and including the following challenges: shadows, snow, rain, presence of trees or big objects. Three of these sequences are very long (32 965, 117 149 and 107 815 frames). For fair comparison with other published results for this dataset, we provide the F-measure results for our model obtained using the usual F-measure definition described in 4.1, but also the results obtained using the executable evaluation tool provided with the dataset which does not use the same definition of the F-measure [64]. We compute SuBSENSE and PAWCS results on this dataset and provide published evaluation results for other models in Table 3.

We observe that the proposed model gets again a better average F-measure than PAWCS and SuBSENSE on this dataset using the standard definition of the F-measure.

4.5. Non-video image datasets: Clevr, ObjectsRoom, ShapeStacks

The proposed model, which does not use any temporal information, can be adapted to perform background reconstruction and foreground segmentation on some image datasets which are not extracted from video sequences. We have tested this approach on three synthetic image datasets: Clevr [30], ShapeStacks, [21] and ObjectsRoom[29]. We use on ShapeStacks and ObjectsRoom the same preprocessing as in [16]. Although each image of these datasets shows a different background, the model is able to recognize that all the backgrounds appearing in a given dataset lie in a low dimensional manifold, which is the case because they have been generated using the same method. These datasets are provided with segmentation annotations for each object appearing in the scenes, which we converted to binary foreground segmentation masks in order to compute the F-measure of the predicted foreground masks.

Considering that on these datasets the risk of overfitting is very low and the background complexity is very high, we substantially increased the number of iterations, which is set to 500 000. We do not use morphological post-processing on the ShapeStacks and ObjectsRoom datasets, because these images have a very low resolution (64 × 64).

We provide in Table 4 the average F-measure obtained on the test sets of these datasets after training on the associated training sets, and in Figure 4 some image samples. To our best knowledge, no other model is able to perform background reconstruction on these datasets.

4.6. Robustness to domain shift and fine-tuning

The proposed model is a batch model. In order to see whether it could be adapted for real-time applications, we studied whether a trained model could perform background reconstruction on new unseen images of the scene which do not belong exactly to the same distribution as the images used for training due to various possible domain shifts such as unseen illumination changes. We then have performed the following experiment: We have split each of the 53 videos provided in the CDnet dataset in two videos of equal lengths. The first half of each video is used to train the autoencoder, and the second half is used as a test dataset. The results of this experiment are provided in Table 5 and show stable results on three categories (baseline, bad weather, camera jitter) which do not show noticeable domain shifts, but a significant worsening on the other categories.

We then adopt the pretrain/fine-tune paradigm, consider the models trained on the first half of the videos as pretrained models, and study how many fine-tuning iterations using images randomly sampled from the second half of the videos are necessary to get competitive test results. We observe that the number of required iterations is very low compared to the number of iterations necessary for a full training, and conclude that a trained model is not robust to domain shifts, but can be quickly updated with a small number of fine-tuning iterations.
Table 2. Average per category of video F-measures on LASIESTA (sources: [12],[4], authors experiments for PAWCS and SuBSENSE)

<table>
<thead>
<tr>
<th>Method</th>
<th>ISI</th>
<th>ICA</th>
<th>IOC</th>
<th>IL</th>
<th>IMB</th>
<th>IBS</th>
<th>OCL</th>
<th>ORA</th>
<th>OSN</th>
<th>OSU</th>
<th>Average 10 cate.</th>
<th>Average 14 cate.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE-NE (ours)</td>
<td>0.91</td>
<td>0.88</td>
<td>0.91</td>
<td>0.81</td>
<td>0.92</td>
<td>0.79</td>
<td>0.94</td>
<td>0.80</td>
<td>0.82</td>
<td>0.91</td>
<td>0.83 0.79 0.86</td>
<td>0.89 0.85</td>
</tr>
<tr>
<td>PAWCS [59]</td>
<td>0.90</td>
<td>0.88</td>
<td>0.90</td>
<td>0.79</td>
<td>0.81</td>
<td>0.79</td>
<td>0.96</td>
<td>0.93</td>
<td>0.69</td>
<td>0.82</td>
<td>0.48 0.77 0.43</td>
<td>0.75 0.85</td>
</tr>
<tr>
<td>SubSENSE [58]</td>
<td>0.90</td>
<td>0.89</td>
<td>0.95</td>
<td>0.65</td>
<td>0.77</td>
<td>0.73</td>
<td>0.92</td>
<td>0.90</td>
<td>0.81</td>
<td>0.79</td>
<td>0.33 0.70 0.31</td>
<td>0.65 0.73</td>
</tr>
<tr>
<td>Cuevas [4]</td>
<td>0.88</td>
<td>0.84</td>
<td>0.78</td>
<td>0.65</td>
<td>0.93</td>
<td>0.66</td>
<td>0.93</td>
<td>0.87</td>
<td>0.78</td>
<td>0.72</td>
<td>n/a  n/a  n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Haines [22]</td>
<td>0.89</td>
<td>0.89</td>
<td>0.92</td>
<td>0.85</td>
<td>0.84</td>
<td>0.68</td>
<td>0.83</td>
<td>0.89</td>
<td>0.17</td>
<td>0.86</td>
<td>n/a  n/a  n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Maddalena [41]</td>
<td>0.95</td>
<td>0.86</td>
<td>0.95</td>
<td>0.21</td>
<td>0.91</td>
<td>0.90</td>
<td>0.47</td>
<td>0.90</td>
<td>0.81</td>
<td>0.88</td>
<td>n/a  n/a  n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Maddalena [40]</td>
<td>0.87</td>
<td>0.85</td>
<td>0.91</td>
<td>0.61</td>
<td>0.76</td>
<td>0.42</td>
<td>0.88</td>
<td>0.84</td>
<td>0.58</td>
<td>0.80</td>
<td>n/a  n/a  n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Table 3. Comparison of top unsupervised BGS algorithms according to the video F-measure on BMC 2012

<table>
<thead>
<tr>
<th>Method</th>
<th>Video 001</th>
<th>Video 002</th>
<th>Video 003</th>
<th>Video 004</th>
<th>Video 005</th>
<th>Video 006</th>
<th>Video 007</th>
<th>Video 008</th>
<th>Video 009</th>
<th>Average 9 videos</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-measure (standard definition)</td>
<td>0.81</td>
<td>0.72</td>
<td>0.78</td>
<td>0.78</td>
<td>0.60</td>
<td>0.73</td>
<td>0.32</td>
<td>0.84</td>
<td>0.77</td>
<td>0.71</td>
</tr>
<tr>
<td>AE-NE (ours)</td>
<td>0.90</td>
<td>0.86</td>
<td>0.89</td>
<td>0.89</td>
<td>0.80</td>
<td>0.87</td>
<td>0.51</td>
<td>0.92</td>
<td>0.89</td>
<td>0.84</td>
</tr>
<tr>
<td>PAWCS [59]</td>
<td>0.86</td>
<td>0.77</td>
<td>0.93</td>
<td>0.86</td>
<td>0.66</td>
<td>0.89</td>
<td>0.79</td>
<td>0.87</td>
<td>0.90</td>
<td>0.84</td>
</tr>
<tr>
<td>SubSENSE [58]</td>
<td>0.85</td>
<td>0.80</td>
<td>0.92</td>
<td>0.85</td>
<td>0.68</td>
<td>0.87</td>
<td>0.75</td>
<td>0.84</td>
<td>0.91</td>
<td>0.83</td>
</tr>
<tr>
<td>DeepLBM [17]</td>
<td>0.73</td>
<td>0.86</td>
<td>0.94</td>
<td>0.90</td>
<td>0.71</td>
<td>0.81</td>
<td>0.70</td>
<td>0.76</td>
<td>0.69</td>
<td>0.78</td>
</tr>
<tr>
<td>G-LBM [51]</td>
<td>0.73</td>
<td>0.85</td>
<td>0.93</td>
<td>0.91</td>
<td>0.71</td>
<td>0.85</td>
<td>0.70</td>
<td>0.76</td>
<td>0.63</td>
<td>0.79</td>
</tr>
<tr>
<td>MSCL-FL [27]</td>
<td>0.84</td>
<td>0.84</td>
<td>0.88</td>
<td>0.90</td>
<td>0.83</td>
<td>0.80</td>
<td>0.78</td>
<td>0.85</td>
<td>0.94</td>
<td>0.86</td>
</tr>
<tr>
<td>B-SSSR [26]</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>0.88</td>
</tr>
</tbody>
</table>

Table 4. F-Measure on the Clevrtex, ShapeStacks and Object-sRoom datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Image size</th>
<th>Number of frames training set</th>
<th>Number of frames test set</th>
<th>Average F-measure on test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clevrtex</td>
<td>128 x 128</td>
<td>40000</td>
<td>5000</td>
<td>0.78</td>
</tr>
<tr>
<td>ObjectsRoom</td>
<td>64 x 64</td>
<td>98000</td>
<td>20000</td>
<td>0.84</td>
</tr>
<tr>
<td>ShapeStacks</td>
<td>64 x 64</td>
<td>217888</td>
<td>46656</td>
<td>0.83</td>
</tr>
</tbody>
</table>

4.7. Implementation details

The proposed model is implemented using Python and the Pytorch framework. The associated code is available on the Github platform. Optimization is performed using the Adam optimizer with a learning rate of $5 \times 10^{-3}$ and batch size equal to 32. The learning rate is divided by 10 when the number of optimization or fine-tuning iterations reaches 80% of the total number of iterations. The most important hyperparameters $\beta$, $r$ and $\tau_1$, which are associated to the loss function, are set to the values recommended in [53] i.e. $\beta = 6$, $r = 75$, $\tau_1 = 0.25$. The other hyperparameter values, which are related to the segmentation threshold and the detection and management of complex background changes, were found empirically using manual hyperparameter tuning. We then set $\alpha_1 = 96/255$, $\alpha_2 = 7$, $N_{\text{eval}} = 2000$, $B_{\text{eval}} = 480$, $\tau_0 = 0.24$, $N_{\text{simple}} = 2500$, $N_{\text{complex}} = 24000$, $E_{\text{complex}} = 20$.

For non-video dataset experiments, which take small images ($64 \times 64$ and $128 \times 128$) as inputs, the batch size and learning rate are increased to 128 and $2 \times 10^{-3}$ and the number of iterations $N_{\text{complex}}$ is set to 500 000. The other hyperparameters remain the same. The autoencoder architecture is described in the supplementary material.

4.8. Computation time

We provide in Table 6 some computation time measurements, obtained using an AMD EPYC 7402 2.8 GHz CPU and a Nvidia RTX 3090 GPU. The inference and training times of the proposed model depend on the size of the image and the complexity of the background. The inference speed is between 50 frames per second and 240 frames per second. The time necessary to perform 100 training iterations is between 3.5 and 27 seconds.

4.9. Limitations

This model is not suited for night videos, considering the low score obtained on this category on the CDnet dataset. One also notes that although the model is able to handle correctly small objects staying still for a long time, as shown by the good results obtained the intermittent object category of the CDnet dataset, it suffers from overfitting when large foreground objects stay still (or appear to stay still) for a long time in a frame sequence. Out of the 110 tested videos contained in the datasets CDnet, LASIESTA and BMC, we observed this problem on 4 videos: "office", "library" and "canoe" in the CDnet dataset, and "video007" in the BMC dataset (Fig. 5). The proposed model should then not be used when the video is expected to show large objects staying still for a long time. This model is a batch model and adapting it to real-time applications requires further work in order to reduce the latency caused by the fine-tuning iterations described in section 4.6.

4.10. Ablation study

In order to assess the impact of the various model features described in this paper, we have implemented sev-
Overall also significant, as already observed for other unsupervised models. The improvement associated to post-processing is expected to have a substantial positive impact on the accuracy of the foreground object segmentation and the use of the background noise estimation layer (see Table 7). They show that the design of the loss function and the use of the background noise estimation layer improved object detection on real-world scenes with complex backgrounds.

The main strength of the proposed model is that it is able to perform unsupervised dynamic background reconstruction and foreground segmentation for real-time applications, and using it to perform unsupervised object detection on real-world scenes with complex backgrounds. 

5. Conclusion

We have proposed in this paper a new fully unsupervised dynamic background reconstruction and foreground segmentation model which does not use any temporal or motion information and is on average more accurate than available unsupervised models for background subtraction. The main strength of the proposed model is that it is able to perform background reconstruction on videos taken from a moving camera. Future works include adapting the model for real-time applications, and using it to perform unsupervised object detection on real-world scenes with complex backgrounds.

Acknowledgment We thank Sascha Hornauer for useful comments on the first draft of this paper.
References


[13] Farcas Diana and Thierry Bouwmans. Background modeling via a supervised subspace learning To cite this version : University of La Rochelle. 2010.


[23] Paul Henderson and Christoph H. Lampert. Unsupervised object-centric video generation and decomposi-


[72] Shengping Zhang, Hongxun Yao, Shaohui Liu, Shengping Zhang, Hongxun Yao, Shaohui Liu, Dynamic Background, Subtraction Based, Shengping Zhang, Hongxun Yao, and Shaohui Liu. Dynamic Background Subtraction Based on Local Dependency Histogram To cite this version. 2008.