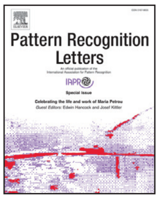
Pattern Recognition Letters 000 (2016) 1–10

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec



# Interactive deep learning method for segmenting moving objects

Yi Wang[a,*], Zhiming Luo[a,b], Pierre-Marc Jodoin[a]

[a] University of Sherbrooke, Sherbrooke QC, J1K 2R1, Canada
[b] Xiamen University, Xiamen Fujian, 361005, China

A R T I C L E I N F O

*Article history:*
Available online xxx

*MSC:*
41A05
41A10
65D05
65D17

*Keywords:*
Motion detection
Ground truthing
Convolutional neural network

A B S T R A C T

With the increasing number of machine learning methods used for segmenting images and analyzing videos, there has been a growing need for large datasets with pixel accurate ground truth. In this letter, we propose a highly accurate semi-automatic method for segmenting foreground moving objects pictured in surveillance videos. Given a limited number of user interventions, the goal of the method is to provide results sufficiently accurate to be used as ground truth. In this paper, we show that by manually outlining a small number of moving objects, we can get our model to learn the appearance of the background and the foreground moving objects. Since the background and foreground moving objects are highly redundant from one image to another (videos come from surveillance cameras) the model does not need a large number of examples to accurately fit the data. Our end-to-end model is based on a multi-resolution convolutional neural network (CNN) with a cascaded architecture. Tests performed on the largest publicly-available video dataset with pixel accurate groundtruth (changedetection.net) reveal that on videos from 11 categories, our approach has an average F-measure of 0.95 which is within the error margin of a human being. With our model, the amount of manual work for ground truthing a video gets reduced by a factor of up to 40. Code is made publicly available at: https://github.com/zhimingluo/MovingObjectSegmentation

## 1. Introduction

With millions of hours of videos recorded daily in the world, the need for efficient video analytic methods is becoming a glaring issue. Considering that a large number of videos are recorded by surveillance cameras, video analytics allows for multiple surveillance tasks including object tracking [38], scene understanding [16], anomaly detection [23], and traffic analytics [26] to name a few. In the last decade, a growing number of machine learning methods have been used to solve these issues [9,24]. Although different, machine learning methods all share a common denominator which is their need for large annotated datasets on which to train. Unfortunately, video annotation is a tedious task, especially when it comes to the annotation of foreground moving objects.

Of course, foreground moving objects can be outlined by fully automatic [mostly background-subtraction related [6]] methods. Although these methods are fast and widely available, they are far from being sufficiently accurate for their results to be used as ground truth [5]. As reported on the changedetection.net (CD-net) website, the only videos for which fully automatic motion detection methods are highly accurate are the so-called *Baseline* videos. *Baseline* videos are those for which the scene contains well-contrasted and well-illuminated macroscopic moving objects pictured in front of a fix background with a rigorously fixed camera, recording video at a high frame rate and without hard shadows. As reported on the website, whenever one of these conditions is violated, the F-measure of the segmented videos drops below 0.88 (and very often below 0.75) which is far too low for it to be used as ground truth.

As an alternative, one can manually annotate every foreground moving object and then use it as ground truth. Although very accurate, manual annotation is tedious and very time consuming. Extensive empirical evaluations led in our lab reveal that even with a well-design and ergonomic annotation software, manual segmentation may take up to 60 s per frame. Thus, the manual labeling of a 4 min video (∼ 7000 frames) may take several days for a single person.

In this letter, we propose a highly accurate semi-automatic method for segmenting foreground moving objects. The proposed solution has two main objectives: 1) produce segmentation maps sufficiently accurate to be used as ground truth and 2) require as little user intervention as possible. The proposed solution is based on a convolution neural network (CNN) model [19]. The main reason for using CNN comes with its ability to learn its own

* Corresponding author.
*E-mail address:* yi.wang@usherbrooke.ca (Y. Wang).

http://dx.doi.org/10.1016/j.patrec.2016.09.014
0167-8655/© 2016 Elsevier B.V. All rights reserved.

features which is far better than using hand-design features. CNN are also translation invariant which is the key feature for dealing with background motion. Furthermore, the convolution operation can be easily parallelized on a GPU which makes CNN a fast predictor.

The outline of our method is straightforward. Given a certain video, the user first outlines foreground objects from a small set of frames. The method then uses those manually annotated images as training data. Once training is over, the method generalizes by automatically labeling the remaining frames of the video. One important characteristic of our method is that it trains and generalizes on images *from the same video*. Since the video comes from a single (and usually fix) camera, its content if very redundant and so the number of manually segmented frames required to properly train our model does not need to be large. This is unlike other machine learning tasks which train and test on images containing very different content such as ImageNet [15] and CIFAR datasets [18]. Our approach also differs from traditional motion detection method as it processes each frame independently without motion features and without having to maintain a background model.

We explore various CNN configurations such as a multiresolution CNN, a cascaded architecture, the FCN-8s [24] model as well as various training configurations. Results obtained on the CDnet dataset shows that our approach is as accurate as a human being with an average F-measure of 0.95.

The contributions of this letter are two folds:

- We propose what we believe is the first machine learning method for ground truthing videos. The method is both highly accurate and requires a small number of user interactions.
- Following an extensive evaluation on the CDnet dataset, we identify the category of videos for which our method is effective as well as the number of frames that ought to be manually labeled for each category.

## 2. Related work

Video foreground detection methods can be classified into two large classes: the fully-automatic methods and those involving user interaction.

The fully-automatic video foreground segmentation methods are usually based on a background model which is updated as the video streams in. The foreground pixels are those whose color (or texture feature) deviates from the background model. The most widely used video foreground segmentation methods implement a parametric background model. This includes those using a per-pixel single Gaussian model [37], a mixture of Gaussians [32], generalized Gaussian mixture [1], and Bayesian models [28] to name a few. Parametric models can deal with videos with small background movement i.e. moving trees or water waves but are very sensitive to camera movement due to jitter or a pan-til-zoom camera motion.

In the past five years, various non-parametric models have achieved good performances. Barnich and Van Droogenbroeck [3] proposed a method called “Vibe” whose per-pixel background model is made of a collection of $N$ pixel values randomly selected over time. Furthermore, when a pixel is updated, its neighboring pixels are also updated which makes Vibe less sensitive to ghosting artifacts. Hofmann et al. [13] proposed an extension to Vibe by allowing the decision threshold and the learning rate to dynamically change over time. Another improvement of Vibe is the so-called “SubSCENE” method proposed by St-Charles et al. [31] which uses both color and local binary pattern features to improve the spatial awareness of the method. It also has a per-pixel feedback scheme that dynamically adjusts its parameters. From the same authors, the so-called “PAWCS” method [30] is an extension of SubSCENE that implements a real-time internal parameter updating strategy. It also adds a persistence indicator feature to the color and local binary patterns (LBP) feature as well as a visual word model.

Many other background subtraction methods have been proposed, some involving a one-class support vector machine (SVM), others a neural network, a Parzen window estimator, a principal component analysis (PCA) model, some fuzzy logic, and many more (refer to [6] for an extensive survey). However, none of these methods have been shown sufficiently accurate to produce groundtruth quality results.

As an alternative, some foreground segmentation methods rely on user interaction to improve accuracy. For these methods, the user provides information on the location of the foreground objects as well as the background. Manual annotation can be in the form of a bounding box around each foreground object or a series of brush strokes drawn on top of foreground and background areas. Approaches for semi-automatic segmentation often rely on graph-cut [21,29]. Unfortunately, these methods being oriented towards the segmentation of 2D images, segmenting a video would require the manual annotation of every frame. As a solution, Bai and Sapiro [2] proposed and extended 3D spatio-temporal graph cut method that implements a 6-pixel neighborhood (4 spatial and 2 temporal neighbors). In [34], users are asked to give interaction not only on each image, but also the x-t dimension to provide additional temporal information. The method by Gong et al. [9] ask the user to label the foreground and background in the first frame of the video and use this to train two one-class SVMs for each pixel. One important inconvenience of such algorithms comes with their way of segmenting the entire video as a whole. Although it works well for segmenting one or few objects seen thought the entire video, these methods cannot account for new moving objects. They are also ineffective on low-framerate videos or when the camera moves due to pan-till-zoom motion.

## 3. Proposed solution

The proposed method can be summarized as follows: based on a subset of frames in which foreground moving objects have been manually outlined, our method trains a foreground-background model that is then used to label the rest of the video. As mentioned earlier, the goal of our method is two fold: 1) get segmentation results sufficiently accurate to be used as ground truth and 2) get those results with as little user intervention as possible. To achieve these goals, we implemented a convolution neural network (CNN) model. The reason why our method gets to learn an accurate foreground-background model from a limited amount of training data comes from the very nature of surveillance videos. Being recorded from surveillance cameras, videos contain a highly redundant content (same background through the video with moving objects having similar orientation, look, and size). This lack of diversity allows for our method to quickly learn a foreground-background model from a very limited number of examples. Furthermore, since the goal is to generalize to other frames from the same video (and not to other videos) our method benefits from a certain level of overfiting which is typical when a limited number of samples are used for training.

As shown in Fig. 1, our approach implements a three-step procedure: (i) foreground moving objects are first manually delineated from a set of training frames, (ii) these frames are then used to train a foreground-background segmentation model, and (iii) once training is over, the model labels the remaining frames of the video. Note that, as will be shown in the results section, the resulting segmentation map is sufficiently accurate for not requiring any post-processing.

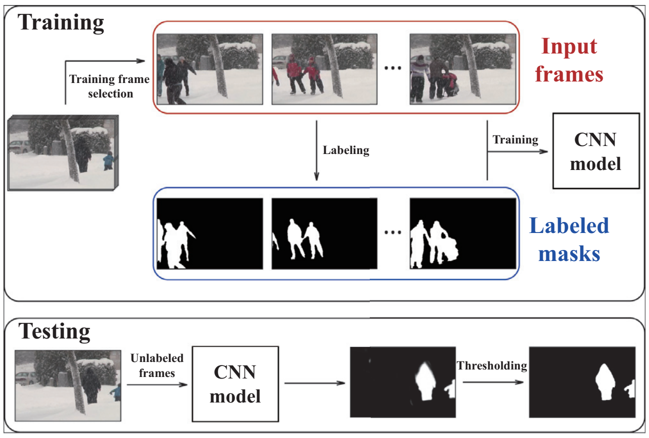


**Fig. 1.** The pipeline of our model.

### 3.1. Selecting and labeling training frames

Given an input video, the first step is to select and manually segment $N$ training frames. In that perspective, different selection strategies can be considered. One could select one frame out of $\frac{M}{N}$, where $M$ is the total number of frames in the video. One could also randomly select $N$ frames or manually select $N$ frames. Note that the latter approach requires extra user intervention which we look forward to minimize as much as possible. But as will be shown in the results section, the frame selection strategy is heavily correlated to the content of the video and in some cases, manual selection is unavoidable, especially for videos with sparse activity.

Once $N$ training frames have been selected, the user roughly outlines a region of interest (ROI) around the area where foreground moving objects are to occur. The reason for this ROI is to exclude regions (such as the sky or buildings) in which no moving objects are to appear. This allows to speed up the training phase and avoid false detections outside the ROI. As for manual delineation of moving objects, we use a custom-made software which greatly simplifies annotation.

### 3.2. CNN models used for training and testing

The method we used for learning the foreground-background model is a deep CNN. The main reasons for choosing CNN is twofold. First, a CNN has the sole ability of learning features that best fit a given set of data. This has a huge advantage over pre-existing approaches which banks on manually selected features such as histogram of oriented gradients (HOG) [7], scale-invariant feature transform (SIFT) [25], or local binary pattern (LBP) [11]. Furthermore, unlike conventional hand-design features, learned features come from multiple layers which focus on various level of details in the video. Second, since CNNs are based on an easily parallelized convolution operators, the prediction phase is very fast.

In this section, we propose three CNN models which we thoroughly test in the results section.

#### 3.2.1. Basic CNN model

A CNN is typically made of a series of convolutional layers, activation layers, pooling layers and fully-connected layers [10,22]. CNNs are generally used for classifying images and, as such, are usually fed with a 3-channel color image and outputs the most likely class label associated to that image [19]. In our case, the goal is to predict a class category (foreground/background) for each pixel instead of the entire image. In that perspective, we extract a $31 \times 31$ patch around each pixel and consider that patch as a small to-be-classified squared image [8].

The detailed configuration of our *basic* CNN model is provided in Table 1 and illustrated in Fig 3. As can be seen, our *basic* CNN model contains 4 convolutional layers and 2 fully connected (FC) layers. Each convolutional layer uses a filter size of $7 \times 7$ and rectified linear unit (ReLU) as the activation function. Also, the first 2 convolutional layers come with a $2 \times 2$ max pooling layer of a stride of 1 as well as a zero padding of one pixel at the bottom and right border. The first fully connected (FC) layer has an output of 64-dimension features while the second has a one-dimension output. For the second FC layer, a sigmoid function is used as an activation function to convert the output prediction between 0 and 1 which corresponds to the probability for a given pixel of being part of the foreground.

By considering the CNN output as a likelihood probability, we use a cross entropy loss function for training [4]:

$$Loss = -\frac{1}{K}\sum_{k=1}^{K}\left[C_k \log \hat{p}_k + (1 - C_k)\log\left(1 - \hat{p}_k\right)\right], \tag{1}$$

where $K$ is the number of training pixels, $C_k$ is the class label in the ground truth and $\hat{p}_n$ is the predicted foreground probability. Note that during training, each pixel is treated independently and without any motion features.

#### 3.2.2. Multi-scale CNN model

The *basic* CNN model is not void of limitations. One of its main drawback comes with from its fix input patch size. Since it processes patches of size $31 \times 31$, the *basic* CNN model is good for distinguishing foreground and background objects whose size is in the order of $31 \times 31$ or less. Unfortunately, videos often contain moving object significantly larger than that. This typically happens when foreground moving objects are close to the camera. As shown in Fig. 2, large moving objects often carry out large uniform textureless areas which can be miss-classified as background. Fig. 2 shows a large car which has been inappropriately segmented by the *basic* CNN model.

We can overcome this issue by implementing a *multi-scale* CNN model as illustrated in Fig. 4. Given a to-be-segmented 2D image $I$, we first resize it into two different scales $I_{scale1}$ and $I_{scale2}$. In this paper, we use 0.75, 0.5 for the two scales. Then $I$, $I_{scale1}$, and $I_{scale2}$ are fed to the *Basic* CNN network separately. This produces three outputs of three different sizes: $O$, $O_{scale1}$, and $O_{scale2}$. After that, $O_{scale1}$ and $O_{scale2}$ are resized back to input frame $I$ size. Note that since we use a stride of 1 at the pooling and convolution layers (cf. Table 1), $O$ has *de facto* the same size than $I$. The final foreground probability map $O_{final}$ is obtained with an average pooling across the upscaled maps (cf. the rightmost picture in Fig. 4). All three CNN share the same weights.

#### 3.2.3. Cascaded CNN model

Since both the *basic* CNN and the *multi-scale* CNN process each pixel independently based on the information contained in their local patch, they often produce isolated false positives and false negatives. Many image segmentation papers [14,17,36] use a conditional random field (CRF) with fixed weights as a way to enforce spatial coherence. However, while this CRF can be easily implemented with graph-cut, it produces in our case sub-optimal results, probably because of the fixed weights for all classes.

In order to model the dependencies among adjacent pixels and thus enforce spatial coherence, we implemented a *cascaded* CNN model. As shown in Fig. 6, the first CNN model (CNN-1) is used to compute a foreground probability map which is then concatenated with the original frame and fed to a second CNN model (CNN-2). The input of CNN-2 is thus an image with four channels: red, green, blue, and a foreground likelihood probability. CNN-2 computes a refined probability map for the input frame (cf. Fig. 2(e)).

**Table 1**
Architecture of our basic CNN model.

| Layer<br>Stage | 1<br>conv | 2<br>conv | 3<br>conv | 4<br>conv | 5<br>FC | 6<br>FC |
|---|---|---|---|---|---|---|
| Input size | 31 × 31 | 25 × 25 | 19 × 19 | 13 × 13 | 7 × 7 | 1 × 1 |
| Filter size | 7 × 7 | 7 × 7 | 7 × 7 | 7 × 7 | – | – |
| Conv stride | 1 × 1 | 1 × 1 | 1 × 1 | 1 × 1 | – | – |
| Pooling method | max | max | – | – | – | – |
| Pooling size | 2 × 2 | 2 × 2 | – | – | – | – |
| Pooling stride | 1 × 1 | 1 × 1 | – | – | – | – |
| Padding size | [0,1,0,1] | [0,1,0,1] | – | – | – | - |
| #Channels | 32 | 32 | 32 | 32 | 64 | 1 |

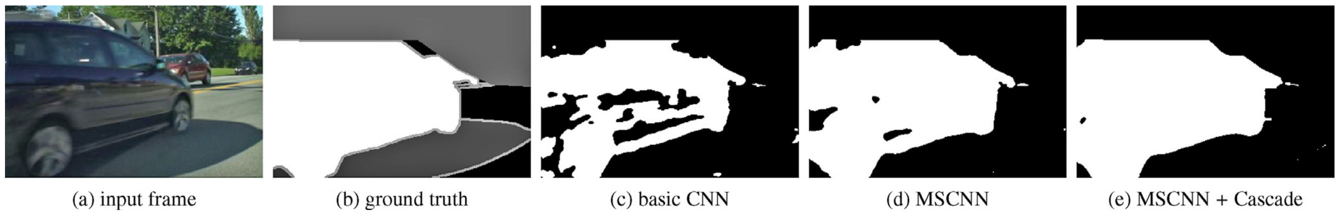


**Fig. 2.** Frame with large moving object (c) fooling our *basic CNN* method due to its large uniform area. (d) the multi-scale and (e) cascaded CNN models reduces greatly the number of false positives by making the system more scale invariant and improving spatial coherence.

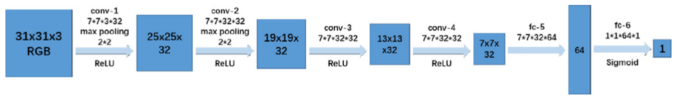


**Fig. 3.** The diagram of our basic CNN model which consist of 4 convolutional layers and 2 fully connected layer. Also the first 2 convolutional layers come with a 2 × 2 max pooling layer.

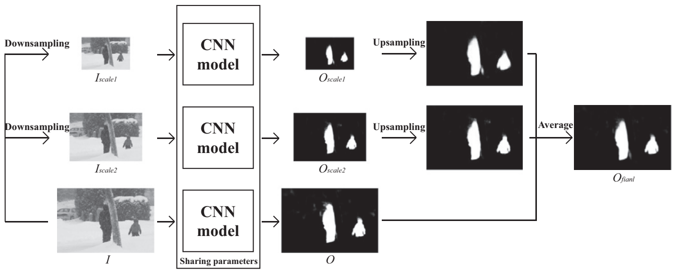


**Fig. 4.** The architecture of the proposed multi-scale CNN model.

Unlike CRF and Markov random fields (MRF) whose parameters need to be manually fine-tuned (e.g. kernel bandwidth, weights between unary and pair-wise terms, etc.), the parameters for our *cascaded* CNN model are learned from the data.

Note that CNN-1 and CNN-2 have the same architecture is showed in Table 1. The only difference between CNN-1 and CNN-2 is the number of input channels: 3 (RGB) for CNN-1 and 4 (RGB + probability map) for CNN2. For training the cascaded model, we fixed the parameters of CNN-1 and only updated the parameters of CNN-2. Note also that we tested our cascaded model with more than 2 CNNs but that did not improve significantly the results.

### *3.2.4. Training details*

All three models have been implemented with the MatConvNet deep learning toolbox which it a wrapper on top of Caffe [33]. Since we intend to train the models on a small number of annotated frames and that each CNN contains a large number of weights, we empirically observed that a CNN with well initialized weights always perform significantly better. So, instead of training from scratch the CNN models on the manually outlined frames, we pre-train the model on a larger dataset and use its weights to initialize our model [39]. Pretraining is done only once with the *Motorway* dataset [26], a dataset with pixel accurate groundtruth of video surveillance images. After transferring the weights to our models, we *finetune* the CNN parameters for each video based on the loss function in Eq. (1). The *Adadelta* optimization method [40] is used for updating parameters with an initial learning rate of 0.01. Models are trained along 20 epochs with a batch size of 5 frames. Besides, although the training can be done patch-wise, for more efficient approach, we train on the whole image by forcing to zero the energy gradient of pixels located outside the previously-selected ROI. Last but not least, to make sure the segmentation result has the same size than the input frame, we apply mirror padding on the original frame during testing.

## 4. Experiment and results

### *4.1. Dataset*

We tested our method on the CDnet 2014 dataset [35], the largest video dataset with pixel accurate groundtruth. CDnet contains 53 videos spanning across 11 categories corresponding to different challenging situations (camera jitter, background motion, pan-tilt-zoom cameras, night videos, etc.). This makes it a perfect dataset for evaluating our foreground labeling methods. Frames from the CDnet dataset are shown in Fig. 5.

### *4.2. Evaluation metrics*

In this paper, we evaluate results with the F-measure and the percentage of wrong classifications (PWC). The F-measure combines precision and recall into one metric. Given a number of true positives (TP), false positives (FP), and false negatives (FN), F-measure is defined as:

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}. \tag{2}$$

Although widely used, the F-measure must be interpreted with case. By its very nature, it does not consider the number of true negatives (TN) and thus is sensitive to very small moving objects. At the limit, missing a one-pixel-size moving objects may lead to a TP of 0 and a F-measure of 0. In order to compensate for this, we

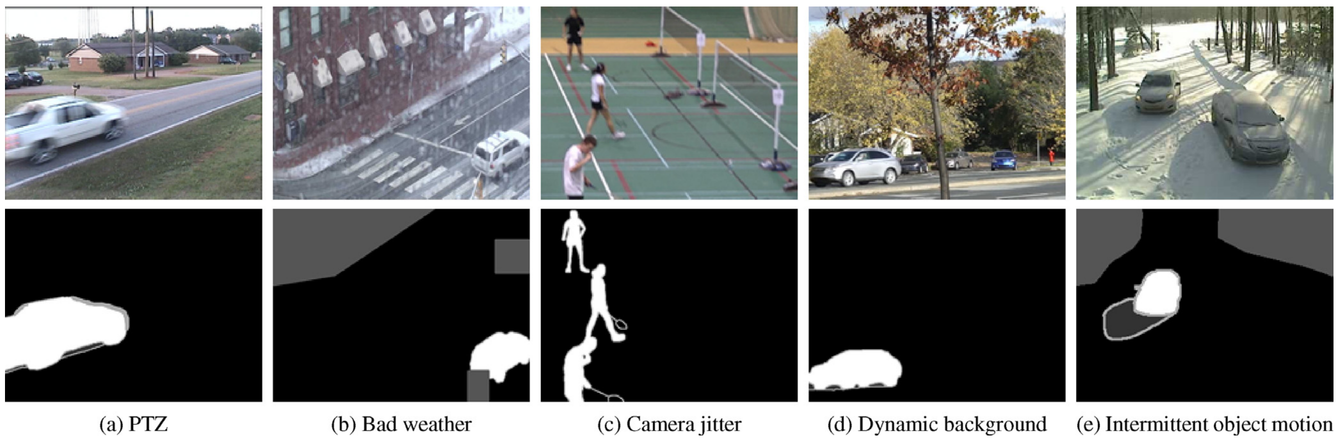


**Fig. 5.** A collection of CDnet color frames with their associated ground truth used in our experiments. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

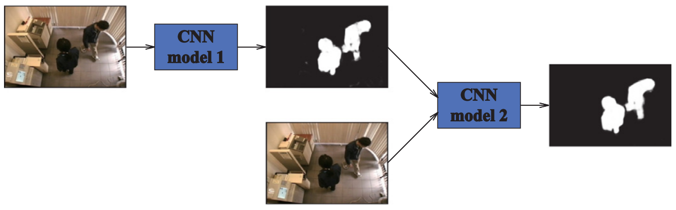


**Fig. 6.** The architecture of the proposed Cascaded model.

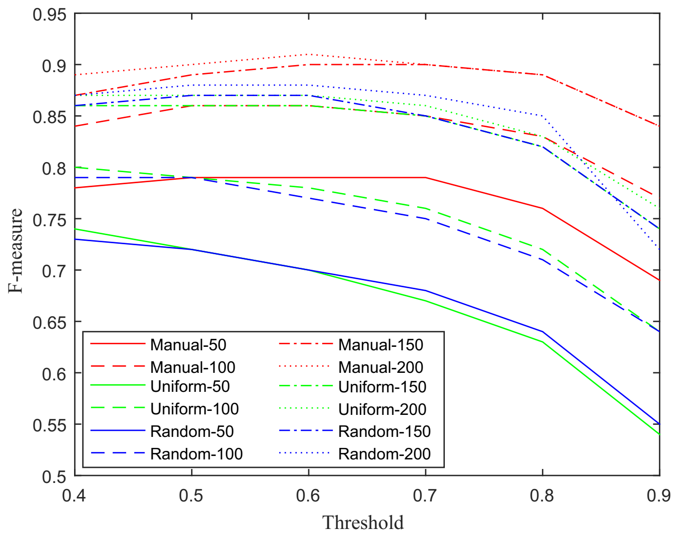


**Fig. 7.** F-measure of various training frame selection strategies, various training frames number and different thresholds.

also use the PWC metric which incorporates TN:

$$\text{PWC} = \frac{100 \times \ (\text{FN} + \text{FP})}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}. \quad (3)$$

The goal of our labeling method is thus to maximize the F-measure while minimizing the PWC. However, we also want to measure how far from the edges of the nearest foreground object the wrongly classified pixels are. Wrongly classified pixels located next to a foreground object is less of a problem than random noise. In that perspective, we use the false positive error distance (FPED) and the false negative error distance (FNED). whose goal is to measure how far from the nearest foreground objet a wrongly labeled pixel is located. The FPED and FNED are calculated as:

$$\text{FPED} = \frac{1}{|\text{FP}|} \sum_{x \in \text{FP}} \min_{y \in \text{FG}} Dist\,(x, y) \quad (4)$$

$$\text{FNED} = \frac{1}{|\text{FN}|} \sum_{x \in \text{FN}} \min_{y \in \text{BG}} Dist\,(x, y), \quad (5)$$

where FG is the set of foreground pixels and BG the set of background pixels.

## 4.3. Experiments

### 4.3.1. Different selection strategies

In this section, we first analyze the influence of the training-frame selection strategies. For each video, we selected 50, 100, 150 and 200 frames for training our *basic* CNN model. Note that those numbers are relatively small compared to the overall video size (CDnet videos contain between 1000 and 8000 frames). After applying different thresholds on our model's output foreground probability map, we get different F-measure values and plot it in Fig. 7. As can be seen, a threshold between 0.6 and 0.7 give the best performance in most cases. Furthermore, the manual strategy achieves higher F-measure than random and uniform strategy given the same number of training frames. This is because for videos with a low level of activity, the random and uniform training frame selection strategies get a much smaller number of foreground objects to train on.

**Table 2**
*Basic* CNN results for different training frame selection strategies.

| Strategy | Training frame # | F-measure | PWC | FPED | FNED |
|---|---|---|---|---|---|
| Random | 50 | 0.68 | 0.89 | 5.65 | 4.73 |
| | 100 | 0.75 | 0.67 | 3.57 | 4.42 |
| | 150 | 0.85 | 0.49 | 4.84 | 3.9 |
| | 200 | 0.87 | 0.40 | 4.68 | 3.71 |
| Uniform | 50 | 0.67 | 0.95 | 5.13 | 4.82 |
| | 100 | 0.76 | 0.64 | 4.93 | 4.35 |
| | 150 | 0.85 | 0.52 | 5.38 | 3.95 |
| | 200 | 0.86 | 0.41 | 4.18 | 3.8 |
| Manual | 50 | 0.79 | 0.90 | 16.17 | 4.12 |
| | 100 | 0.85 | 0.58 | 10.44 | 3.76 |
| | 150 | 0.90 | 0.46 | 12.11 | 3.54 |
| | 200 | 0.90 | 0.45 | 4.96 | 3.45 |

Results for every metric are provided in Table 2. As one can see, among the three training-frame selection strategies, the random and uniform achieve similar performances with a F-measure

**Table 3**
Results for our CNN models.

| Model | Training frames # | F-measure | PWC | FPED | FNED |
|---|---|---|---|---|---|
| CNN | 50 | 0.79 | 0.90 | 16.17 | 4.12 |
| | 100 | 0.85 | 0.58 | 10.44 | 3.76 |
| | 150 | 0.90 | 0.46 | 12.11 | 3.54 |
| | 200 | 0.90 | 0.45 | 4.96 | 3.45 |
| | 50 | 0.88 | 0.53 | 8.88 | 3.15 |
| CNN + Cascade | 100 | 0.90 | 0.47 | 7.80 | 2.82 |
| | 150 | 0.92 | 0.37 | 5.68 | 2.55 |
| | 200 | 0.93 | 0.37 | 5.68 | 2.37 |
| MSCNN | 50 | 0.87 | 0.51 | 5.80 | 3.51 |
| | 100 | 0.88 | 0.44 | 3.57 | 2.56 |
| | 150 | 0.91 | 0.35 | 4.27 | 2.86 |
| | 200 | 0.92 | 0.31 | 2.56 | 2.2 |
| | 50 | 0.88 | 0.49 | 10.52 | 2.05 |
| MSCNN + Cascade | 100 | 0.92 | 0.35 | 4.22 | 1.84 |
| | 150 | 0.94 | 0.28 | 3.25 | 1.65 |
| | 200 | 0.95 | 0.26 | 2.41 | 1.54 |

of at most 0.86 and 0.87. As for the manual selection, it reaches a F-measure of 0.9. That being said, by considering the PWC, all three selection strategies are roughly equivalent. One may also notice that the FPED and FNED are relatively large for all three selection strategies (more than 4 pixels on average). This is because the *basic* CNN model provides results with fuzzy edges, unfilled holes and producing some random noise. Also, since the manual strategy selects frames with large amount of foreground objects, the model trained on those frames have a tendency of producing slightly more false positives hence why the FPED is larger for manual and FNED lower.

#### *4.3.2. Evaluation of the proposed CNN models*

In this section, we evaluate the performances of our CNN models. Four different models have been trained, namely 1) the *basic* CNN model, 2) the *cascaded* CNN model (*CNN + Cascade*), 3) the *multi-scale* CNN model (*MSCNN*), and 4) a *multi-scale* with cascaded model (*MSCNN + Cascade*). Training frames have been manually selected and a fix threshold of 0.7 have been used.

As can be seen in Table 3, the multi-scale and cascaded architectures significantly improve results. We can also see that the more training frames one has, the more accurate the end result gets. The top performing method is the *MSCNN + Cascade* model whose PWC, FPED and FNED are 50% lower than for the *basic* CNN model. Qualitative inspection of results reveal that the *MSCNN* is both accurate on large and small foreground objects, it has very little isolated false positive and false negative pixels, and the boundaries of the foreground objects are very well defined.

We also show results for the *MSCNN + Cascade* model on each video category in Table 4. By using only 50 frames for training, our model gets to segment videos from 4 categories out of 11 with very high accuracy (F-measure $\geq$ 0.95). By increasing the number of training frames to 200, our model achieve outstanding performance for most of the categories; F-measure of 0.96, PWC of 0.06 FPED of 2.3 and FNED of 1.4 for pan-till-zoom (PTZ) videos and very good numbers of videos shot a night. For more difficult categories such as *Bad weather, Thermal* and *Turbulence*, we get F-measures above 0.94. Even for some pathological videos (especially *Low framerate* and *Intermittent object motion* in which foreground objects can be very small), our model achieves a good F-measure of 0.88.

#### *4.3.3. Comparison with other methods*

We implemented other deep learning methods but due to space limitation, we only report results of the most accurate one which is the fully convolutional network (FCN) [24] in this letter. The FCN model was designed to segment real images into different semantic categories and reached state-of-art performances on several benchmark datasets. The FCN model which we used is a re-implementation by the vlfeat team[1]. The only modification that we made to that model was the 2 class output (*Background*/*Foreground*). We trained the FCN the same way we did for our method.

Quantitative results for *MSCNN + Cascade* and FCN are presented in Table 5. From those results, *MSCNN + Cascade* outperforms FCN on every metric on all four training sets. Note that these results are average metrics across 11 categories including some extreme cases like night videos, camera motion, and low frame rate videos. Results for our method and FCN are also shown in Fig. 8. We noticed that FCN often underestimates the foreground objects which leads to several FN regions. Besides, because of the FCN upsampling strategy (please refer to [24] for more details on that), the foreground moving objects have a blobby shape, especially when the foreground and the background have similar color distributions.

We also present in Table 6 the results obtained with the top 3 automatic motion detection methods reported on the CDnet website, namely IUTIS-5 (a method which performs a smart majority vote of several motion detection methods), PAWCS [30], and SuBSENSE [31], two non-parametric methods. As one can see, these results are far less accurate that those obtained by our method in Tables 2–5.

### *4.4. Manual labeling accuracy*

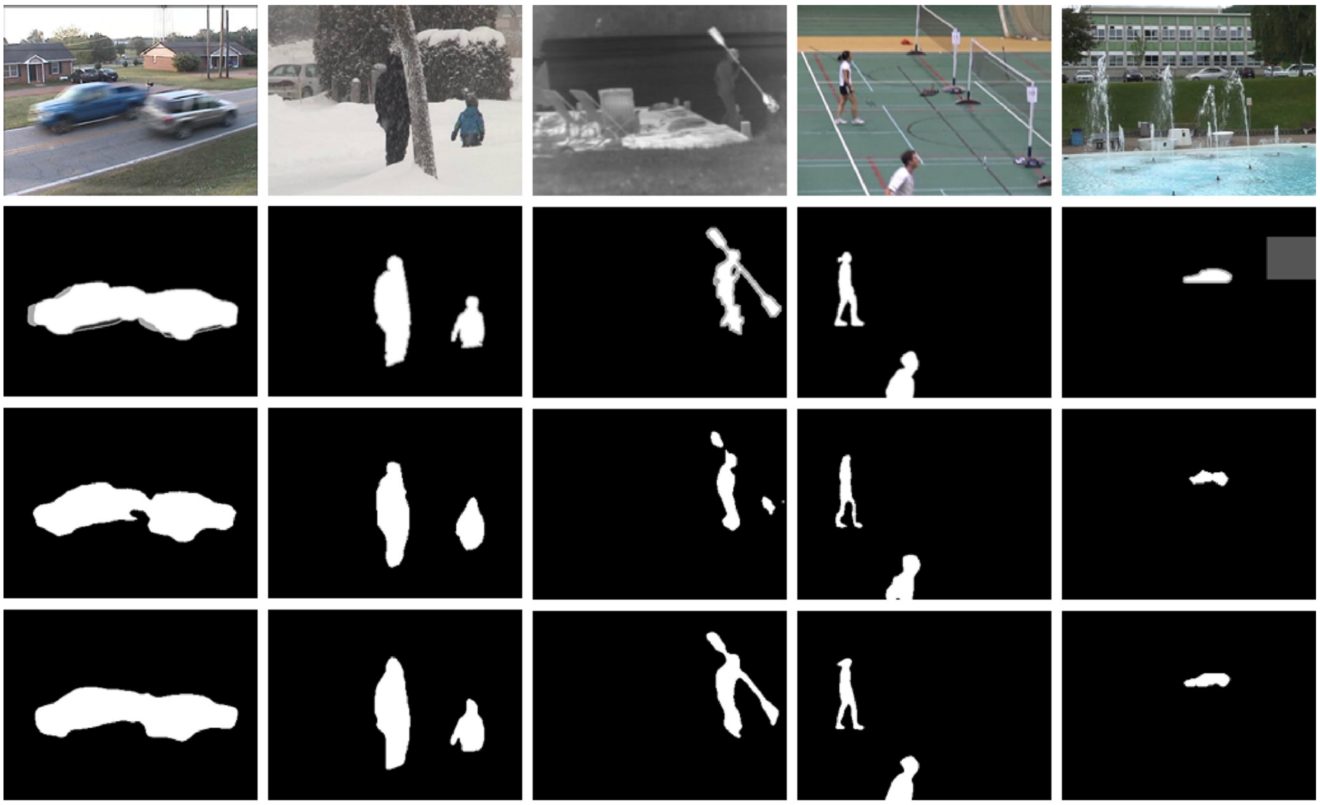
As mentioned at the beginning of the paper, our goal is to produce results sufficiently accurate to be used as ground truth. One may thus conclude that since our model does not reach a F-measure of 1 and a PWC of 0 in Tables 2–5 it is not accurate enough to be used as a reference. With the following experiments, we prove that those worries are baseless since human raters can hardly obtained a F-measure of more than 0.95 and that a F-measure of 0.94 is as precise as a 1 pixel erosion (or dilation) of the CDnet groundtruth.

Ground truthing is a subjective task as different persons may give different labeling results for the same video. To evaluate how results vary from one person to another, we selected 77 representative frames from the CDnet dataset and invited three persons to label it. We then compared their results with the CDnet ground truth (which has also been obtained by a person). Example is given in Fig. 9 and quantitative results are in Table 7.

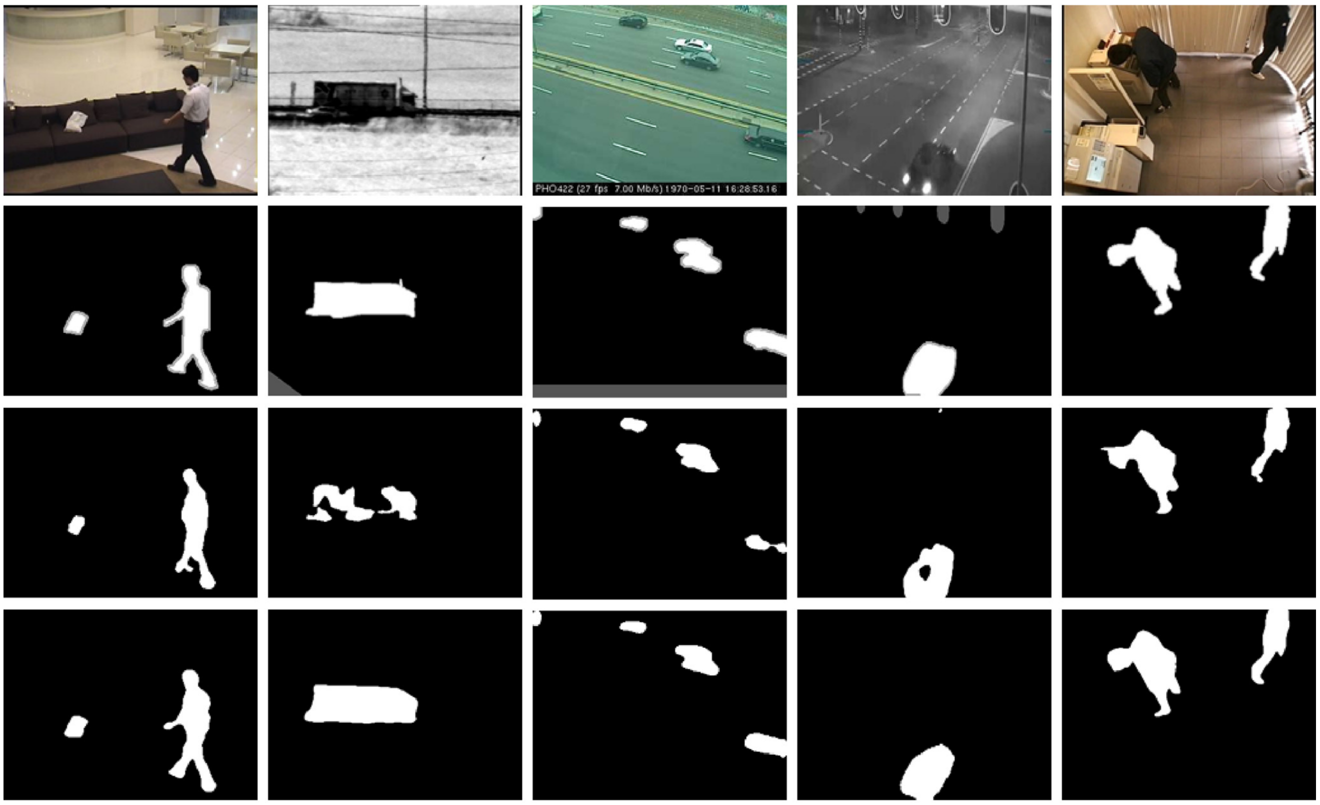
Interestingly, none of them got a F-measure above 0.96. On average, these persons got a F-measure of 0.948, a PWC of 0.87, and an error distance of 3.6 pixels. This leads us to believe that a method with a F-measure above 0.94, a PWC below 0.9 and an error distance of less than 3.6 pixels is within the error margin of a human annotation. As shown previously, it is the case for our method.

We also noticed that a F-measure variation between 0.93 and 1.0 may be caused by a very small number of wrongly classified pixels. In order to illustrate that claim, we simply dilated the CDnet ground truth by 1 and 2 pixels and measured the impact that operation had on the F-measure and the PWC (we did the same experiment with the erosion operator). Although a simple erosion (or dilation) of one pixel may not seriously affect the quality of the groundtruth (moving objects are only 1 pixel thinner or fatter), it nonetheless results into a F-measure of 0.94 and 0.93 (cf. Table 8).

[1] https://github.com/vlfeat/matconvnet-fcn.



(a) PTZ (b) Bad weather (c) Thermal (d) Camera jitter (e) Dynamic background



(f) Intermittent object motion (g) Turbulence (h) Low framerate (i) Night video (j) Shadow

**Fig. 8.** Results on CDnet videos. The first row shows input frames, the second row shows the ground truth, the third row the FCN results, and the fourth row the results by our method.

**Table 4**
Metrics for MSCNN + Cascade on each CDnet video category.

| Category | 50 training frames | | | | 200 training frames | | | |
|---|---|---|---|---|---|---|---|---|
| | F-measure | PWC | FPED | FNED | F-measure | PWC | FPED | FNED |
| Baseline | 0.97 | 0.19 | 1.6 | 0.7 | 0.99 | 0.08 | 2.0 | 0.4 |
| Dyn. back. | 0.95 | 0.08 | 1.9 | 2.0 | 0.98 | 0.03 | 1.7 | 1.7 |
| Camera jitter | 0.97 | 0.27 | 1.2 | 0.9 | 0.98 | 0.15 | 0.6 | 0.9 |
| Inter. obj. motion | 0.87 | 1.24 | 0.8 | 0.7 | 0.88 | 1.30 | 0.6 | 0.5 |
| Shadow | 0.95 | 0.42 | 5.2 | 2.7 | 0.98 | 0.18 | 2.8 | 2.1 |
| Thermal | 0.89 | 1.01 | 15.4 | 3.2 | 0.95 | 0.44 | 4.0 | 2.3 |
| Bad weather | 0.79 | 0.90 | 65.4 | 4.3 | 0.97 | 0.11 | 2.3 | 3.1 |
| Low framerate | 0.74 | 0.24 | 5.8 | 1.6 | 0.88 | 0.09 | 6.9 | 1.3 |
| Night video | 0.87 | 0.75 | 1.6 | 2.8 | 0.93 | 0.38 | 1.1 | 2.1 |
| PTZ | 0.88 | 0.17 | 8.0 | 1.9 | 0.96 | 0.06 | 2.3 | 1.4 |
| Turbulence | 0.84 | 0.09 | 8.8 | 1.69 | 0.94 | 0.05 | 2.1 | 1.2 |

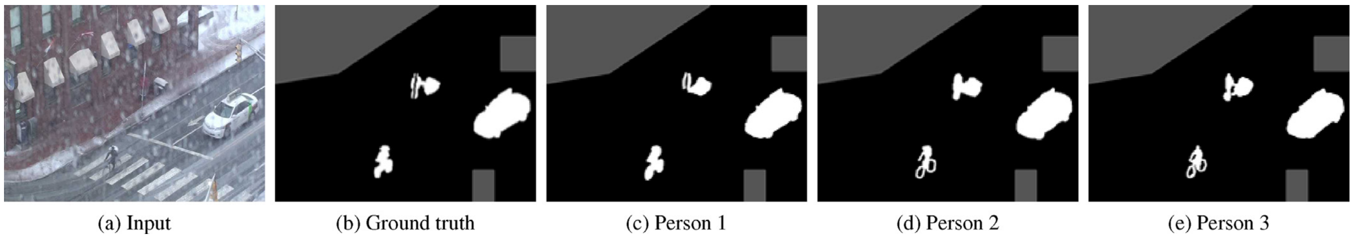


**Fig. 9.** Results showing the unavoidable variation between the ground truth and the manual labeling obtained by three independent persons.

**Table 5**
Results for our method and FCN for different training size.

| Training frames | Method | F-measure | PWC | FPED | FNED |
|---|---|---|---|---|---|
| 50 | FCN | 0.83 | 0.72 | 12.58 | 2.49 |
| | MSCNN + Cascade | 0.88 | 0.49 | 10.52 | 2.05 |
| 100 | FCN | 0.85 | 0.61 | 8.18 | 2.30 |
| | MSCNN + Cascade | 0.92 | 0.35 | 4.22 | 1.84 |
| 150 | FCN | 0.86 | 0.58 | 6.72 | 2.13 |
| | MSCNN + Cascade | 0.94 | 0.28 | 3.25 | 1.65 |
| 200 | FCN | 0.87 | 0.56 | 5.58 | 2.00 |
| | MSCNN + Cascade | **0.95** | **0.26** | **2.41** | **1.54** |

**Table 6**
Results from the most accurate motion detection methods.

| Model | F-measure | PWC | FPED | FNED |
|---|---|---|---|---|
| IUTIS-5 | 0.77 | 1.20 | 219.83 | 4.37 |
| PAWCS | 0.74 | 1.20 | 243.12 | 4.78 |
| SuBSENSE | 0.74 | 1.68 | 309.43 | 4.76 |

**Table 7**
Results from manual labeling.

| | F-measure | PWC | FPED | FNED |
|---|---|---|---|---|
| Person 1 | 0.93 | 1.18 | 3.7 | 5.3 |
| Person 2 | 0.96 | 0.72 | 2.1 | 2.5 |
| Person 3 | 0.96 | 0.72 | 4.4 | 3.5 |

**Table 8**
Metrics of dilating and eroding the ground truth.

| Method | #Pixel | F-measure | PWC |
|---|---|---|---|
| Dilate | 1 | 0.94 | 0.31 |
| | 2 | 0.88 | 0.73 |
| Erode | 1 | 0.93 | 0.33 |
| | 2 | 0.86 | 0.63 |

This shows again that a method with F-measure of 0.93 and above may be considered almost as good as the ground truth.

Let us also mention that our method comes without post-processing. After testing a series of post-processing operations including superpixels aggregation, median filter, open and closing morphological operations, we concluded that although post-processing may help under certain conditions, it always degrades our overall results. This is yet another indication that our method produces a very small number of false positives and false negatives.

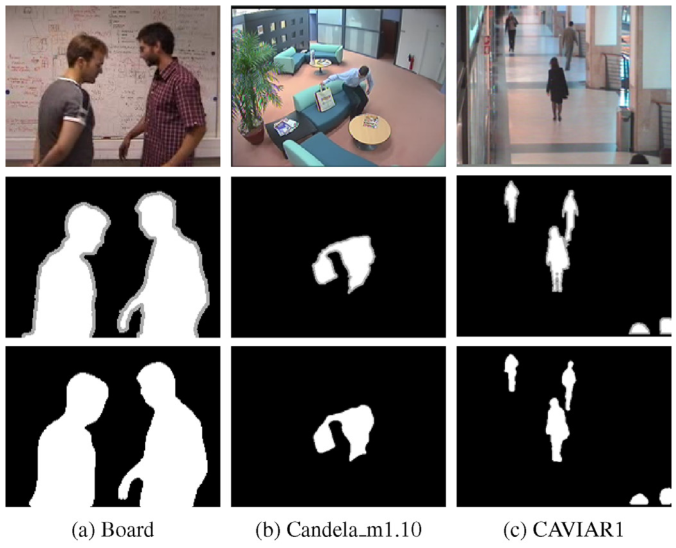
### *4.5. Experiments on SBI2015 dataset*

We also tested our method on the Scene Background Initialization 2015 (SBI2015) dataset [27] which contains 14 videos. Since this dataset does not contain any pixel-accurate groundtruth of background and foreground objects, we manually labeled each video of the dataset. Since the SBI2015 videos are relatively short (e.g. ”Toscana” contains only 6 frames), we randomly split each video into 20% frames for training and the remaining 80% for testing. Due to space limitation, we only report results of our best model (MSCNN + Cascade) in Table 9 and also plot some representative results in Fig. 10.

As can be seen from Table 9, our method achieves a F-measure of more than 0.95 for 12 out of the 14 SBI2015 videos. These results show again that our approach can be as accurate as a human being. That is especially true on the CAVIAR1 video for which the F-measure reaches 0.995. That said, we also noticed that our method performs poorly on two videos. It is the case for the ”Snellen” video which happens to be very difficult even for a human as there is no clear boundary between the foreground and background regions. As for the ”Toscana” video, since it contains only 6 frames, the system does not have enough training material to correctly learn the foreground and background distributions (here only 2 frames were used for training). We shall also mention that the main purpose of our approach is to reduce the burden of annotating long videos, which is obviously not a problem with the Toscana video.

**Table 9**
Results on SBI2015 dataset.

| | F-measure | PWC | FPED | FNED |
|---|---|---|---|---|
| Board | 0.99 | 0.30 | 1.84 | 3.69 |
| Candela_m1.10 | 0.98 | 0.12 | 1.80 | 3.97 |
| CAVIAR1 | 0.995 | 0.03 | 1.70 | 1.69 |
| CAVIAR2 | 0.95 | 0.04 | 2.03 | 1.48 |
| CaVignal | 0.97 | 0.58 | 1.33 | 1.42 |
| Foliage | 0.95 | 6.31 | 2.27 | 20.7 |
| HallAndMonitor | 0.97 | 0.16 | 1.93 | 2.04 |
| HighwayI | 0.98 | 0.30 | 3.36 | 5.62 |
| HighwayII | 0.98 | 0.10 | 2.10 | 7.20 |
| HumanBody2 | 0.96 | 0.77 | 2.57 | 5.40 |
| IBMtest2 | 0.95 | 0.48 | 2.58 | 4.53 |
| PeopleAndFoliage | 0.99 | 1.46 | 2.20 | 11.72 |
| Snellen | 0.33 | 45.84 | 13.74 | 30.23 |
| Toscana | 0.51 | 21.63 | 91.60 | 8.98 |



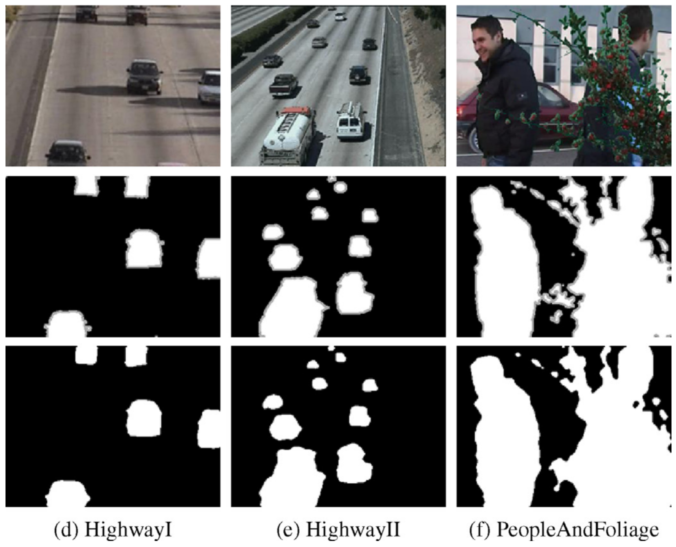


**Fig. 10.** Results on SBI2015 dataset The first row shows input frames, the second row shows the ground truth and the third row shows the results obtained by our method.

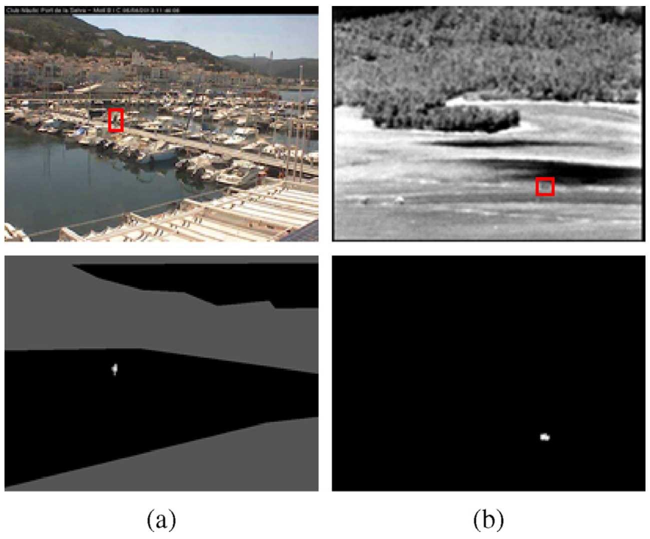


**Fig. 11.** Examples of videos for which our method does not perform well.

### 4.6. Processing time

All the experiments were conducted on a GTX970 GPU with a Matlab implementation. For a 1700 frames long video with frame size of 320 $\times$ 240, it takes roughly 14 min for our *MSCNN + Cascade* model to train 20 epochs with 200 frames and 2 min to segment the rest of the video. These 16 min are orders of magnitude smaller than the time required to manually label the remaining 1500 frames.

## 5. Discussion and conclusion

In this letter, we proposed a highly accurate semi-automatic method for segmenting foreground moving objects pictured in surveillance videos. With a small amount of user intervention, our model can provide ground truth accurate labeling results. Our model has shown to be successful in most video categories of the CDnet dataset, with an average F-measure of 0.95 and PWC of 0.26. The experiments reveal that:

- The best performing model involves a Multi-scale CNN with a cascaded architecture. Its results are systematically better than any other CNN model we have tested.
- For a given video, only 50–200 frames need to be manually labeled. This corresponds to a huge gain compared with the manual annotation of the entire video (i.e. a factor of up to 40 for CDnet videos containing 8000 frames).
- The number of training frames as well as the selection strategy depends on the complexity of the video. As a rule of thumb, videos with fix illumination showing a steady flow of well contrasted moving objects only require 50 training frames chosen at random. For more complex videos such as “Night Videos” which contains low-contrasted object and “PTZ” for which the camera moves in all directions, a larger number of training frames ($\approx$ 200) is required to reach good results. Also, videos with sparse activity usually require the manual selection of the training frames otherwise the system does not get enough foreground objects to train on.
- Our approach is not void of limitations as we noticed its difficulty (F-measure below 0.9) at dealing with very small foreground objects (cf. Fig. 11). Fortunately, such situations are relatively infrequent.

- Groundtruthing is a subjective task and we showed that a labeling result with a F-measure $\geq$ 0.94 and a PWC $\leq$ 0.8 can be considered within the error margin of a human.

Besides the model and results reported in this paper, we have tested many other CNN models. However, due to space limitation, we couldn't report all of it. We shall thus draw a short summary of these methods whose results have been systematically worse than our method.

In order to consider motion, we included a temporal gradient to the input RGB image and trained our CNN models accordingly. However, we noticed that temporal gradient is a poor indicator for low contrasted objects and produces ghosting artifacts in presence of intermittent motion (objects that stop for a short while and then leave). We also concatenated a collection of frames in order to process 3D video volumes instead of 2D images. The results ended up being equal or worst, especially for videos with intermittent motion object. Similar to [8], we segmented each image into superpixels and combined it with the CNN segmentation results with the hope of improving accuracy close to the borders. But that did not work out, especially for objects with a poorly contrasted silhouette (typical of night videos). Inspired by Hattori et al. [12], we tried to increase the training set by copy-pasting foreground objects on top of a background image. Unfortunately, we realized that adding fake foreground objects only helps when their color, size, shape and orientation is rigorously identical to that of the actual foreground objects. And finally, as in [20], we implemented an hysteresis thresholding procedure but again, it did not improve performance in any significant manner.

In the future, we will explore how to accommodate our method with a weakly-supervised training approach according to which users may only provide rough strokes on top of foreground and background regions. We shall also incorporate reinforcement learning in order for the system to account for users' corrections as well as 3D convolutional layers in order to integrate the temporal dimension of the video.

## References

[1] M.S. Allili, N. Bouguila, D. Ziou, A robust video foreground segmentation by using generalized gaussian mixture modeling, in: Proc. IEEE Conf. Comput. and Robot Vis., 2007, pp. 503–509.
[2] X. Bai, G. Sapiro, A geodesic framework for fast interactive image and video segmentation and matting, in: Proc. IEEE Int. Conf. Comput. Vis., 2007, pp. 1–8.
[3] O. Barnich, M. Van Droogenbroeck, Vibe: A universal background subtraction algorithm for video sequences, in: Proc. IEEE Conf. Image Process., vol. 20, 2011, pp. 1709–1724.
[4] Y. Bengio, Learning Deep Architectures for Ai, 2(1), Foundations and Trends® in Mach. Learning, 2009, pp. 1–127.
[5] T. Bouwmans, Recent advanced statistical background modeling for foreground detection-a systematic survey, Recent Pat. Comput. Sci. 4 (3) (2011) 147–176.
[6] T. Bouwmans, Traditional and recent approaches in background modeling for foreground detection: an overview, Comput. Sci. Rev. 11 (2014) 31–66.
[7] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Proc. IEEE Conf. Comput. Vis. Pattern Recog., vol. 1, 2005, pp. 886–893.
[8] C. Farabet, C. Couprie, L. Najman, Y. LeCun, Learning hierarchical features for scene labeling, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1915–1929.
[9] M. Gong, Y. Qian, L. Cheng, Integrated foreground segmentation and boundary matting for live videos, IEEE Trans. Image Process. 24 (4) (2015) 1356–1370.
[10] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[11] Z. Guo, L. Zhang, D. Zhang, A completed modeling of local binary pattern operator for texture classification, IEEE Trans. Image Process. 19 (6) (2010) 1657–1663.
[12] H. Hattori, V. Naresh Boddeti, K.M. Kitani, T. Kanade, Learning scene-specific pedestrian detectors without real data, in: Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2015, pp. 3819–3827.
[13] M. Hofmann, P. Tiefenbacher, G. Rigoll, Background segmentation with feedback: the pixel-based adaptive segmenter, in: Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshops, 2012, pp. 38–43.
[14] Q. Huang, M. Han, B. Wu, S. Ioffe, A hierarchical conditional random field model for labeling and segmenting images of street scenes, in: Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2011, pp. 1953–1960.
[15] ImageNet, http://www.image-net.org/, 2014,
[16] P.-M. Jodoin, Y. Benezeth, Y. Wang, Meta-tracking for video scene understanding, in: Proc. IEEE Conf. Advanced Video and Signal based Surveill., 2013, pp. 1–6.
[17] P. Krähenbühl, V. Koltun, Efficient inference in fully connected crfs with gaussian edge potentials, arXiv preprintarXiv:1210.5644(2012).
[18] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, 2009, pp. 1–60.
[19] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.
[20] P. Kumar, S. Ranganath, W. Huang, Queue based fast background modelling and fast hysteresis thresholding for better foreground segmentation, in: Proc. IEEE Conf. Informat., Commun. and Signal Process., vol. 2, 2003, pp. 743–747.
[21] V. Kwatra, A. Schödl, I. Essa, G. Turk, A. Bobick, Graphcut textures: image and video synthesis using graph cuts, in: ACM Trans. Graph., vol. 22, 2003, pp. 277–286.
[22] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[23] W. Li, V. Mahadevan, N. Vasconcelos, Anomaly detection and localization in crowded scenes, IEEE Trans. Pattern Anal. Mach. Intell. 36 (1) (2014) 18–32.
[24] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2015, pp. 3431–3440.
[25] D.G. Lowe, Object recognition from local scale-invariant features, in: Proc. IEEE Conf. Comput. Vis. Pattern Recog., vol. 2, 1999, pp. 1150–1157.
[26] Z. Luo, P.-M. Jodoin, S.-Z. Li, S.-Z. Su, Traffic analysis without motion features, in: Proc. IEEE Int. Conf. on Image Process., 2015, pp. 3290–3294.
[27] L. Maddalena, A. Petrosino, Towards benchmarking scene background initialization, in: International Conference on Image Analysis and Processing, Springer, 2015, pp. 469–476.
[28] F. Porikli, O. Tuzel, Bayesian background modeling for foreground detection, in: Proc. ACM conf. Video surveillance & sensor netw. Workshops, 2005, pp. 55–58.
[29] C. Rother, V. Kolmogorov, A. Blake, Grabcut: Interactive foreground extraction using iterated graph cuts, in: ACM Trans. Graph., vol. 23, 2004, pp. 309–314.
[30] P.-L. St-Charles, G.-A. Bilodeau, R. Bergevin, A self-adjusting approach to change detection based on background word consensus, in: Proc. IEEE Conf. Appl. of Comput. Vis., 2015b, pp. 990–997.
[31] P.-L. St-Charles, G.-A. Bilodeau, R. Bergevin, Subsense: a universal change detection method with local adaptive sensitivity, IEEE Trans. Image Process. 24 (1) (2015a) 359–373.
[32] C. Stauffer, W.E.L. Grimson, Adaptive background mixture models for real-time tracking, in: Proc. IEEE Conf. Comput. Vis. Pattern Recog., vol. 2, 1999.
[33] A. Vedaldi, K. Lenc, Matconvnet: convolutional neural networks for matlab, in: Proc. ACM Conf. Multimedia Conf., 2015, pp. 689–692.
[34] J. Wang, P. Bhat, R.A. Colburn, M. Agrawala, M.F. Cohen, Interactive video cutout, ACM Trans. Graphics 24 (3) (2005) 585–594.
[35] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, P. Ishwar, Cdnet 2014: an expanded change detection benchmark dataset, in: Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshops, 2014, pp. 393–400.
[36] C. Wojek, B. Schiele, A dynamic conditional random field model for joint labeling of object and scene classes, in: Proc. European Conf. Comput. Vis., 2008, pp. 733–747.
[37] C.R. Wren, A. Azarbayejani, T. Darrell, A.P. Pentland, Pfinder: Real-time tracking of the human body, IEEE Trans. Pattern Anal. Mach. Intell. 19 (7) (1997) 780–785.
[38] Y. Wu, J. Lim, M.-H. Yang, Object tracking benchmark, IEEE Trans. Pattern Anal. Mach. Intell. 37 (9) (2015) 1834–1848.
[39] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks? in: Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 3320–3328.
[40] M.D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701 (2012).