A Deep Convolutional Neural Network for Background Subtraction

M. Babaee[a], D. Dinh[a], G. Rigoll[a]

[a]*Institute for Human-Machine Communication, Technical Univ. of Munich, Germany*

Abstract

In this work, we present a novel background subtraction system that uses a deep Convolutional Neural Network (CNN) to perform the segmentation. With this approach, feature engineering and parameter tuning become unnecessary since the network parameters can be learned from data by training a single CNN that can handle various video scenes. Additionally, we propose a new approach to estimate background model from video. For the training of the CNN, we employed randomly 5% video frames and their ground truth segmentations taken from the Change Detection challenge 2014(CDnet 2014). We also utilized spatial-median filtering as the post-processing of the network outputs. Our method is evaluated with different data-sets, and the network outperforms the existing algorithms with respect to the average ranking over different evaluation metrics. Furthermore, due to the network architecture, our CNN is capable of real time processing.

Keywords: Background subtraction, change detection, deep learning

1. Introduction

With the tremendous amount of available video data, it is important to maintain the efficiency of video based applications to process only relevant information. Most video files contain redundant information such as background scenery, which costs a huge amount of storage and computing resources. Hence, it is necessary to extract the meaningful information, e.g. vehicles or pedestrians, to deploy those resources more efficiently. Background subtraction is a binary classification task that assigns each pixel in a video sequence with a label, for either belonging to the background or foreground scene [25, 21, 1].

Background subtraction, which is also called change detection, is applied to many advanced video applications as a pre-processing step to remove redundant data, for instance in tracking or automated video surveillance [17]. In addition, for real-time applications, like tracking, the algorithm should be capable of processing the video frames in real-time.

One simple example of the application of a background subtraction method is the pixel-wise subtraction of a video frame from its corresponding background image. After being compared with the difference threshold, pixels with a larger difference than a certain threshold value are labeled as foreground pixels, otherwise as background pixels. Unfortunately, this strategy will

yield poor segmentation due to the dynamic nature of the background, that is induced by noise or illumination changes. For example, due to lighting changes, it is common that even pixels belonging to the background scene can have intensities very different from their other pixels in the background image and they will be falsely classified as foreground pixels as a consequence. Thus, sophisticated background subtraction algorithms that assure robust background subtraction under various conditions must be employed.
In the following sections, the difficulties in this area, our proposed solution for background subtraction and our contributions will be illustrated.

1.1. Challenges

The main difficulties that complicate the background subtraction process are:

Illumination changes: When scene lighting changes gradually (e.g. moving clouds in the sky) or instantly (e.g. when the light in a room is switched on), the background model usually has a illumination different from the current video frame and therefore yields false classification.

Dynamic background: The background scene is rarely static due to movement in the background (e.g. waves, swaying tree leaves), especially in outdoor scenes. As a consequence, parts of the background in the video frame do not overlap with the corresponding parts in the background image, hence, the pixel-wise correspondence between image and background is no longer existent.

Camera jitter: In some cases, instead of being static, it is possible that the camera itself is frequently in movement due to physical influence. Similar to the dynamic background case, the pixel locations between the video and background frame do not overlap anymore. The difference in this case is that it also applies to non-moving background regions.

Camouflage: Most background subtraction algorithms work with pixel or color intensities. When foreground objects and background scene have a similar color, the foreground objects are more likely to be (falsely) labeled as background.

Night Videos: As most pixels have a similar color in a night scene, recognition of foreground objects and their contours is difficult, especially when color information is the only feature in use for segmentation.

Ghosts/intermittent object motion: Foreground objects that are embedded into the background scene and start moving after background initialization are the so-called *ghosts*. The exposed scene, that was covered by the ghost, should be considered as background. In contrast, foreground objects that stop moving for a certain amount of time, fall into the category of intermittent object motion. Whether the object should be labeled as foreground or background is strongly application dependent. As an example, in the automated video surveillance case, abandoned ob-

jects should be labeled as foreground.

Hard shadows: Dark, moving shadows that do not fall under the illumination change category should not be labeled as foreground.

In this work, we follow the trend of Deep Learning and apply its concepts to background subtraction by proposing a CNN to perform this task. We justify this approach with the fact that background subtraction can be performed without temporal information, given a sufficiently good background image. With such a background image, the task itself breaks down into a comparison of a image-background pair. Hence, the input samples can be independent among each other, enabling a CNN to perform the task with only spatial information. The CNN is responsible for extracting the relevant features from a given image-background pair and performs segmentation by feeding the extracted features into a classifier. In order to get more spatially consistent segmentation, post-processing of the network output is done by spatial-median filtering and or a fully connected CRF framework. Due to the use of a CNN, no parameter tuning or descriptor engineering is needed.

To train the network, a large amount of labeled data is required. Fortunately, due to the process of background subtraction, by comparing an image with its background, it is not necessary to use images of a full scene for training. It is also possible to train the network via subsets of a scene, i.e. with patches of image-background pairs, since the procedure also holds for image patches. As a consequence, we can extract enough training samples from a limited amount of labeled data.

To the best of our knowledge, background subtraction algorithms that use CNN are scene specific to this day, i.e. a CNN can only perform satisfying background subtraction on a single scene (that was trained with scene specific data) and also lacks the ability to perform the segmentation in real time. Our proposed approach yields a universal network that can handle various scenes without having to retrain it every time the scene changes. As a consequence, one can train the network with data from multiple scenes and hence increase the amount of training data for a network. Also, by using the proposed network architecture, it is possible to process video frames in real-time with conventional computing resources. Therefore, our approach can be considered for real-time applications.

The outline of this paper is as follows: In Section 2, early and recent algorithms for background subtraction are presented. In Section 3, we explain our proposed approach for background subtraction. Here, we first describe our proposed approach to estimate background image and next we illustrate our CNN for background subtraction. In Sections 4, we describe the experimental evaluation of the algorithm including the used datasets, the evaluation metrics and the obtained results followed by detailed discussion and analysis. Finally, in Section 5, we conclude our work and provide future work with some ideas.

2. Backround and Related work

The majority of background subtraction algorithms are composed of several processing modules which are explained in the following sections (see Figure 1).

Background Model: The background model is essential for the background subtraction algorithm. In general, the background model is used as a reference to compare with the incoming video frames. Furthermore, the initialization of the background model plays an important role since video sequences are normally not completely free of foreground objects during the bootstrapping phase. As a consequence, the model gets corrupted by including foreground objects into the background model, which leads to false classifications.

Background Model Maintenance: In reality, background is never completely static but changes over time. There are many strategies to adapt the background model to these changes by using the latest video frames and/or previous segmentation results. Trade-offs must be found in the adaption rate, which regulates how fast the background model is updated. High adaption rate leads to noisy segmentation due to the sensitivity to small or temporary changes. Slow adaption rate, however, yields an outdated background model and therefore false segmentation. Selective update adapts the background model with pixels that were classified as background. In that case, deadlock situations can occur by not incorporating misclassified pixels into the background model, i.e. once a background pixel is falsely classified as foreground, it would never be used to update the background and would always be considered as a foreground pixel. On the other hand, by using all pixels as in the blind update strategy, such deadlock situations can be prevented but will also distort the background model since foreground pixels are incorporated into the background model.

Feature extraction: In order to compare the background image with the video frames, adequate features that represent relevant information must be selected. Most algorithms use gray scale or RGB intensities as features. In some cases, pixel intensities along with other hand engineered features (e.g. [11] or [2]) are combined. Also, the choice of the feature region is important. One can extract the features over pixels, blocks or patterns. Pixel-wise features often yield noisy segmentation results since they do not encode local correlation, while block-wise or pattern-wise features tend to be insensitive to slight changes.

Segmentation: With the help of a background model, the respective video frames can be processed. Background segmentation is performed by extracting the features from corresponding pixels or pixel regions of both frames and using a distance measure, e.g. the Euclidean distance, to measure the similarity between those pixels. After being compared with the similarity threshold, each pixel is either labeled as background or foreground.

The combination of those building blocks forms an overall background subtraction system. The robustness of the system is always dependent and limited by the performance of each individual

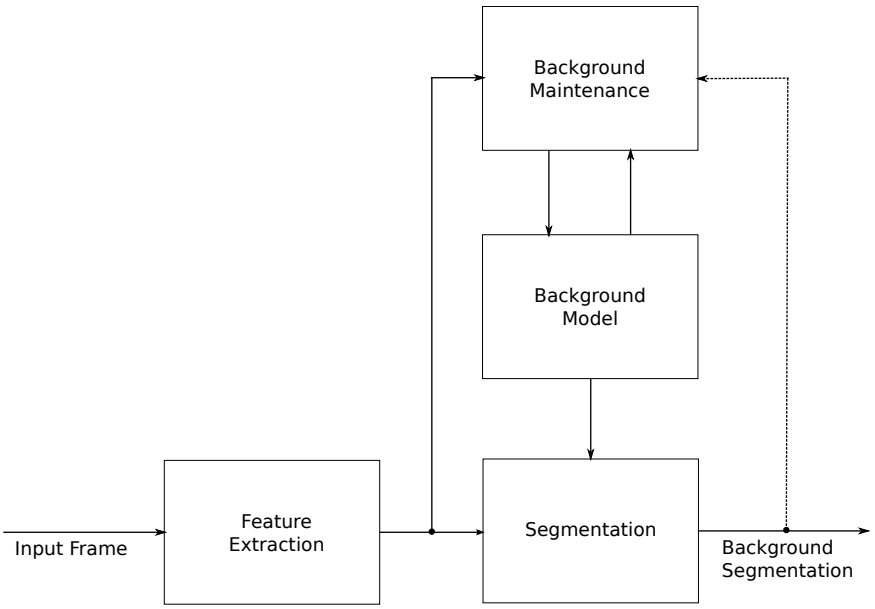


Figure 1: Block diagram of basic background subtraction algorithms. The dashed line is optional and not existent in some algorithms

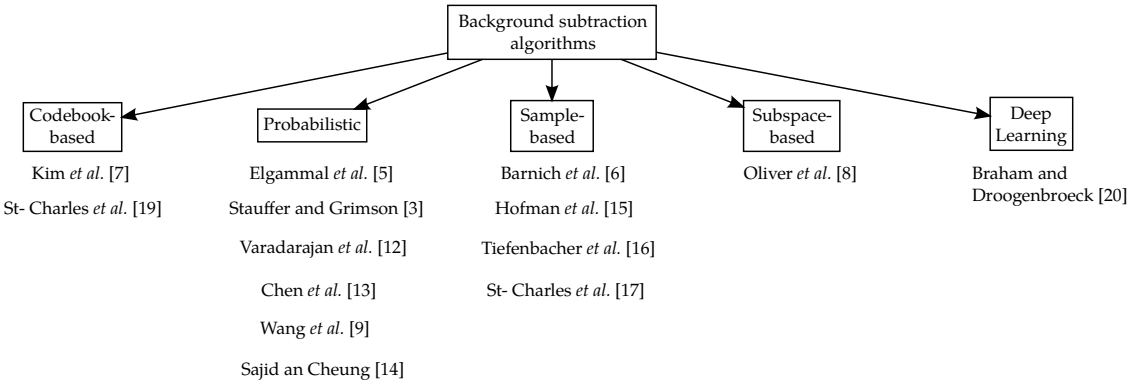


Figure 2: Categorization of the outlined background subtraction algorithms

block, i.e. it can not be expected to perform well if one module delivers poor performance.

Background subtraction is a well studied field, therefore there exists a vast number of algorithms for this purpose (see Figure. 2). Since most of the top performing methods at present are based on the early proposed algorithms, some of which are outlined in the beginning. Subsequently a few of the current methods for background subtraction will be introduced.

2.1. Early Approaches

Stauffer and Grimson [22] proposed a method that models the background scene with Mixture of Gaussian (MoG), also called Gaussian Mixture Model (GMM). It is assumed that each pixel in the background is drawn from a Probability Distribution Function (PDF) which is modeled by a GMM, also pixels are assumed to be independent from their neighboring pixels. Incoming pixels from video frames are labeled as background if there exists a Gaussian in the GMM, where the distance between its mean and the pixel lies within a certain bound. For learning the parameters, that maximize the likelihood, the authors proposed an online method that approximates the Expectation Maximization (EM) algorithm [6].
Elgammal *et al.* [7] introduced a probabilistic non-parametric method to model the background. Again it is assumed that each background pixel is drawn from a PDF. The PDF for each pixel is estimated with Kernel Density Estimation (KDE).
In contrast, Barnich *et al.* [1] propose a sample based method for background modeling. The background model consists of pixel samples from past video frames. For each pixel location, the number of past pixel samples is stored as N (e.g. N=20). To perform the background segmentation at each pixel position, the incoming pixel is compared with the N pixel samples from the background model. If there exist at least K samples in the model with a distance to the incoming pixel smaller than a global threshold R, the pixel is classified as background pixel, otherwise as foreground pixel.
Another approach is introduced by Kim *et al.* [15], where the background model for each pixel position is modeled by a codebook. During bootstrapping phase, incoming pixels build up codewords that consist of intensity, color and temporal features. After initialization, those codewords form a codebook for subsequent segmentation. The intensity and color of incoming pixels are compared with those of the codewords in the codebook. After calculating the distances between the pixels and the codewords and comparing them with the threshold value , either a foreground label (if no match was found) or a background label is assigned. In the latter case, the matching codeword is updated with the respective background pixel.
Oliver *et al.* [16] use subspace learning to compress the background, the so called eigenbackground. From N images, the mean and the covariance matrix are calculated. After a PCA of the covariance matrix, a projection matrix is set up with M eigenvectors, corresponding to the M largest eigenvalues. Incoming images are compared with their projection onto the eigenvectors. After calculating the distances between the image and the projection and comparing them with the corresponding threshold value, foreground labels are assigned to pixels with large distances.

2.2. Recent Approaches

Based on [22], Wang *et al.* [26] use GMM to model the background scene, in addition, single Gaussians are employed for foreground modeling. By computing flux tensors [4], which depict variations of optical flow within a local 3D spatio-temporal volume, blob motion is detected. With the combination of the different information from blob motion, foreground models and background models, moving and static foreground objects can be spotted. Also, by applying edge matching [8], static foreground objects can be classified as ghosts or intermittent motions.
Varadarajan *et al.* [25] introduce region-based MoG for background subtraction to tackle the

sensitivity to dynamic background . MoGs are used to model square regions within the frame, instead of pixel-wise feature modeling, the features are extracted from sub-blocks within the square regions.

With the same purpose to tackle dynamic background movement, Chen *et al.* [5] propose a method that applies MoG in a local region. Models for foreground and background are learned with pixel-wise MoG. To label a given pixel, a $N \times N$ region around the pixel is searched for a MoG that shows the highest probability for the center pixel. If a foreground model depicts the highest probability, the center pixel is labeled as foreground and vice versa.

To cope with sudden illumination changes, Sajid and Cheung [17] create multiple background models with single Gaussians and employ different color representations. Input images are pixel-wise clustered into K groups with the K-means algorithm. For each group and each pixel location a single Gaussian model is built by calculating the mean and variance for a cluster. For incoming pixels, the matching models are selected by taking the models among the K models that show the highest normalized cross-correlation to the pixels. The segmentation is done for each color channel for RGB and YCbCr color representations, which yields 6 segmentation masks. By combining all available segmentation masks the background segmentation is then performed.

Hofmann *et al.* [13] developed an algorithm that improved the method from Barnich *et al.* [1] by replacing the global threshold value R with an adaptive threshold R(x), which is dependent on the pixel location and a metric of the background model. The metric was named "background dynamics" by the authors. With the additional information from the background dynamics, the thresholds R(x) and the model update rate are adapted by a feedback loop. High variance in the background yields a larger threshold value and vice versa, therefore it can cope with dynamic background and highly structured scenes. Tiefenbacher *et al.* [23] improved this method by controlling the updates of the pixel-wise thresholds with a PID controller instead of a fixed value that was used before.

St-Charles *et al.* [21] also improved the method by using Local Binary Similarity Patterns (LBSP) [2] as additional features to pixel intensities and slight changes in the update heuristic of the thresholds and the background model.

They also proposed another method [20], which is similar [21], but works with codewords, or background words as they call it in their publication. Pixel intensities, LBSP features and temporal features are combined to a background word. To perform segmentation, incoming pixels' LBSP features together with their intensities are compared with the corresponding background words. After a two stage comparison, 1), color and structure threshold, 2), temporal feature threshold, the pixel is either labeled as foreground or background depending on the threshold results. All thresholds are dynamically updated in a feedback loop using a background dynamics measure.

A novel approach for background subtraction with the use of CNN was proposed by Braham and Droogenbroeck [3]. They used a fixed background model, which was generated from a temporal median operation over N video frames. Afterwards, a scene-specific CNN is trained with corresponding image patches from the background image, video frames and ground truth pixels, or alternatively, segmentation results coming from other background subtraction algorithms. After extracting a patch around a pixel, feeding the patch through the network and comparing

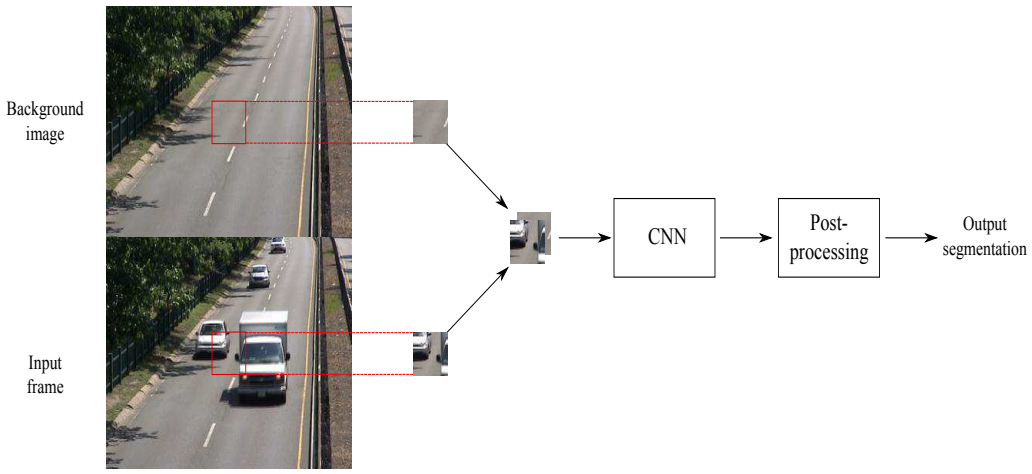


Figure 3: Background subtraction system overview: The input frame along with the corresponding background image are patch-wise processed. After merging the individual patches into a single output frame, the output frame is post-processed yielding the final output segmentation.

it with the score threshold, the pixel is assigned with either a background or a foreground label. However, due to the over-fitting that is caused by using highly redundant data for training, the network is scene-specific, i.e. can only process a certain scenery, and needs to be retrained for other video scenes (with scene-specific data).

3. Approach

In this section, we introduce our proposed method that consists of 1) a novel algorithm for background model (image) generation; 2) a novel CNN for background subtraction and 3) post-processing of the networks output using median filter. The complete system is illustrated in Figure 3. We use a background image to perform background subtraction from the incoming frames. Matching pairs of image patches from the background image and the video frames are extracted and fed into a CNN. After reassembling the patches into the complete output frame, it is post-processed, yielding the final segmentation of the respective video frame.
In the following sections, we introduce our algorithm to get the background images from the video frames. Furthermore, we illustrate our CNN architecture and the training procedure. At last, we discuss our employed post-processing strategies to improve the network output.

3.1. Background Image Generation

We propose a new approach to generate background model which is illustrated in Figure. 5. Here we combine the segmentation mask from SuBSENSE algorithm [21] and the output of Flux Tensor algorithm [26], which can dynamically change the parameters used in the background model based on the motion changes in the video frames. The block diagram of the robust

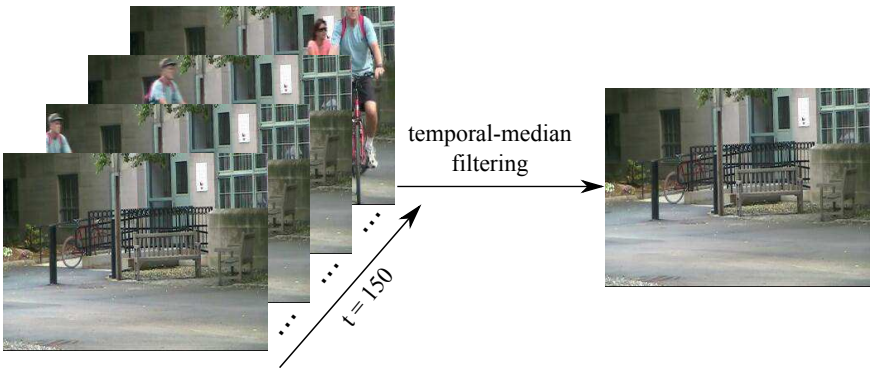


Figure 4: Temporal-median filtering over 150 frames of a video sequence. Since each background pixel is visible for at least 75 frames, the temporal-median filtering yields the background image without foreground objects.

background model algorithm is given in Figure. 5. The details of each block is explained in the following sections.

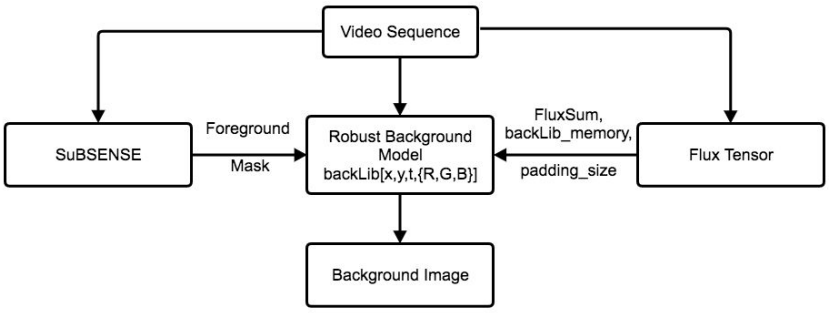


Figure 5: Block diagram of Robust Background Model Algorithm

3.1.1. Foreground Segmentation Using SuBSENSE

Perhaps, the simplest way to get background image from a video sequence is to perform pixel-wise temporal median filtering. However, using this method, the background model will be quickly corrupted if there are a lot of moving objects in some video frames, and hence, the pixel values of foreground object will eventually negatively affect the quality of the background model. This requires us to distinguish the foreground pixels and background pixels, and for the background model we only use the background pixel values to perform temporal median filter. To this end, we use the SuBSENSE algorithm. This method relies on spatio-temporal binary features as well as color information to detect changes. This allows camouflaged foreground objects to be detected more easily while most illumination variations are ignored. Besides, instead of using manually-set, frame-wide constants to dictate model sensitivity and adaptation speed, it uses pixel-level feedback loops to dynamically adjust the method's internal parameters without user intervention. The adjustments are based on the continuous monitoring of model fidelity and local

segmentation noise levels. The output of SuBSENSE algorithm will be a binary image which contains the classification information of the current video frame. The foreground objects are marked as white pixel, and black pixels represent the pixels belonging to background model.

(a) (b)

Figure 6: (a) Video input frame; (b) segmentation output of SuBSENSE

3.1.2. Background Pixel Library

Based on the foreground mask image from SuBSENSE, we build a background pixel library,BL, for each pixel location in the frame. The idea is that we only store the pixel values from the current frame in the background pixel library, when they are classified by SuBSENSE algorithm as background pixels. An illustration of background pixel library is shown in Figure. 7. Here,

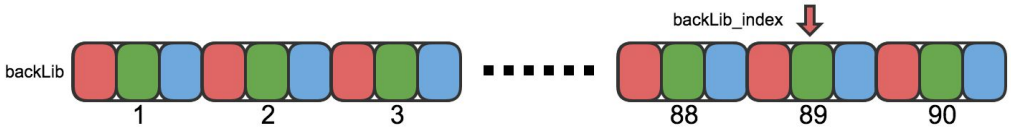


Figure 7: Background pixel library for one pixel location

we only keep the last 90 background pixel values from the video sequences. After the library is complete, the oldest background pixel value is replaced with the newest one. For this purpose, we have an indicator, *bi*, which is a pointer that points to the oldest background pixel value in the library. Each time, if a pixel value is classified by SuBSENSE as background, then it will be stored in the background library at the location where *bi* points to, and then we move *bi* to the next position in the library. To generate the background image, we calculate the average value over a certain memory length of the pixel values in the background pixel library. The memory length is defined using the variable *bm*. The background pixel at location (x, y) and at color channel k is defined by $BK(x, y, k)$, whose value is calculated using following equation

$$BK(x,y,k) = \frac{\sum_{i=bi-bm}^{bi} BL(x,y,i,k)}{bm}, \quad k = \{R,G,B\}. \tag{1}$$

But if we use a fixed memory length *bm* over the entire video sequence, we will get either blurry or outdated background model if the camera is moving or if there are some intermittent objects in the video sequence. These two cases are illustrated in Figure. 8. As we can see in the following figure, the first row shows the scenario that the camera is constantly rotating, the calculated background image is shown on the right hand side. In this case, the background image becomes very blurry if we use a fixed average length *bm*, or in the second row, the car on the left side stops at the same location for 80 percent of the frames in the video sequence, so the car will naturally be classified as background by SuBSENSE. Then if the car starts to move, the calculated background image using fixed memory length will not be updated because the pixel values of the car are still stored in the background library.

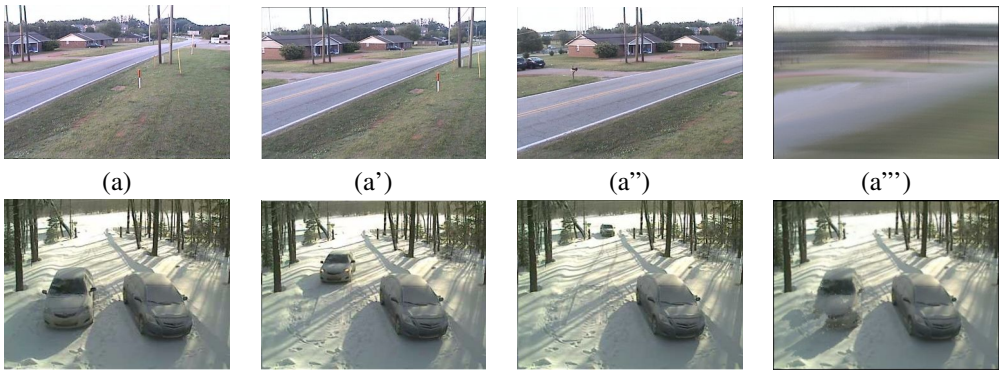


Figure 8: Bad Background Image Model When Using Fixed Average Length. On the first row, the first 3 images ($a - a''$) are the video input frames (Nr. 80,120,160) from continuousPan sequence of CDnet data-set. The first 3 images on the second row ($b - b''$) are the video input frames(Nr. 1800,1950,2100) from winterDriveaway sequence. The left images on the first (a''') and on the second row (b''') are the default background models from SuBSENSE with fixed memory length.

3.1.3. Motion Detection Using Flux Tensor

In order to have adaptive memory length based on the motion of the camera and objects in the video frames, we need to have a motion detector. A commonly used motion detector is the method using flux tensor. Compared with standard motion detection algorithms, the advantage of using flux tensor is that the motion information can be directly computed without expensive eigenvalue decompositions. Flux tensor represents the temporal variation of the optical flow field within the local 3D spatio-temporal volume [26], where in the expanded matrix form, flux tensor is written as

$$J_F = \begin{bmatrix} \int_\Omega \{\frac{\partial^2 I}{\partial x \partial t}\}^2 dy & \int_\Omega \frac{\partial^2 I}{\partial x \partial t}\frac{\partial^2 I}{\partial y \partial t} dy & \int_\Omega \frac{\partial^2 I}{\partial x \partial t}\frac{\partial^2 I}{\partial t^2} dy \\ \int_\Omega \frac{\partial^2 I}{\partial y \partial t}\frac{\partial^2 I}{\partial x \partial t} dy & \int_\Omega \{\frac{\partial^2 I}{\partial y \partial t}\}^2 dy & \int_\Omega \frac{\partial^2 I}{\partial y \partial t}\frac{\partial^2 I}{\partial t^2} dy \\ \int_\Omega \frac{\partial^2 I}{\partial t^2}\frac{\partial^2 I}{\partial x \partial t} dy & \int_\Omega \frac{\partial^2 I}{\partial t^2}\frac{\partial^2 I}{\partial y \partial t} dy & \int_\Omega \{\frac{\partial^2 I}{\partial t^2}\}^2 dy \end{bmatrix}. \quad (2)$$

The elements of the flux tensor incorporate information about temporal gradient changes which leads to efficient discrimination between stationary and moving image features. The trace of the

flux tensor matrix which can be compactly written and computed as

$$trace(J_F) = \int_\Omega \| \frac{\partial}{\partial t} \bigtriangledown I \|^2 dy, \tag{3}$$

can be directly used to classify moving and non-moving regions without eigenvalue decompositions. In our approach, the output of Flux Tensor algorithm is a binary image, which contains the motion information of the video frames. An example is shown in Figure. 9. The white pixel in the binary image indicates that the pixel at this location is moving either temporally or spatially.

(a) (b)

Figure 9: (a) input video; (b) the output of Flux Tensor

Next, we define a new variable called F_s as:

$$F_s = \frac{N_w}{W \times H}, \tag{4}$$

where N_w represents the number of white pixels in Flux tensor while W and H represent the width and height of the image respectively. The variable F_s presents how many percent of the pixels in the current video frame are moving. Large F_s means either the camera is moving or there is a large object in the frame which starts to move. In this case, we need to decrease the memory length bm. If F_s is relative small, then it means the background is steady and we can use a large memory length to suppress the noise. Using F_s, we dynamically increase or decrease the value of bm. The relation between bm and F_s is given as follows

$$bm = \begin{cases} 5 & if\ F_s \geq 0.25 \\ 5 + \dfrac{F_s - 0.02}{0.25 - 0.02} \cdot 85 & if\ 0.02 < F_s < 0.25 \\ 90 & if\ F_s \leq 0.02 \end{cases}. \tag{5}$$

In order to avoid the noise in F_s and resulting noise in bm, there is a low pass filter structure applied to the value of F_s, the low pass filter is defined as follows

$$F_s(t) = \alpha \cdot F_s(t) + (1 - \alpha) \cdot F_s(t-1) \tag{6}$$

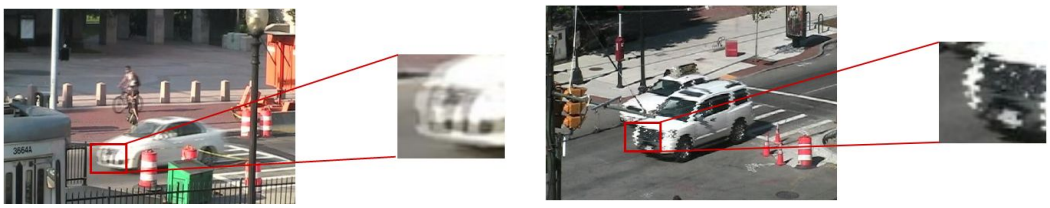
where

$$\alpha = \begin{cases} 0.2 & if\ F_s(t) < F_s(t-1) \\ 0.01 & if\ F_s(t) > F_s(t-1) \end{cases} \tag{7}$$

Here, $F_s(t)$ means the value of F_s at time t. Note that the different value of α is based on the fact whether F_s is increasing or decreasing. The reason for this is that after a dramatic decrease of F_s we want to increase F_s slower, in order to let B be updated with new background pixel values.

3.1.4. Foreground Mask Padding

Due to the low quality of some surveillance cameras, there will be some undesired pixel values around moving objects in the form of semi-transparency and motion blur, which are illustrated in Figure. 10. In these two cases, the pixels near segmentation mask of SuBSENSE



(a) (b)

Figure 10: Semi-transparency and motion blur around moving objects

will probably be false negatives. This means that even the pixels near the segmentation mask are classified as background, but they are actually corrupted by foreground moving pixels with semi-transparency and motion blur. In this case, we should not add this background pixel to the background library. For this purpose we need to perform a padding around the foreground mask with the size defined by using variable P_w, which means if a pixel value is classified as background but the pixels around it within the radius of path size P_s are classified as foreground, then this background pixel will be disregarded. The P_s should also be dynamically adjusted using the output of flux tensor. For instance, if F_s is large, then we need to increase P_s, because with more moving pixels the phenomenon of semi-transparency and motion blur will also increase. Figure. 11 shows the comparison between the background model from SuBSENSE and the robust background model obtained by the proposed approach.

3.2. Network Architectures and Training

We train the proposed CNN with background images obtained by the SuBSENSE algorithm [21]. Both networks are trained with pairs of RGB image patches from video and background frames and the respective ground truth segmentation patches. Before introducing the network architecture, we outline the data preparation step for the network training. Afterwards, we will illustrate our architecture for our CNN and explain the training procedure in detail.

3.2.1. Data preparation

For the training of the CNN, we use random data samples (around 5 percent) from the CDnet 2014 data-set [27], which contains various challenging video scenes and their ground truth segmentation. We prepare one set of background images that is obtained by the proposed algorithm.

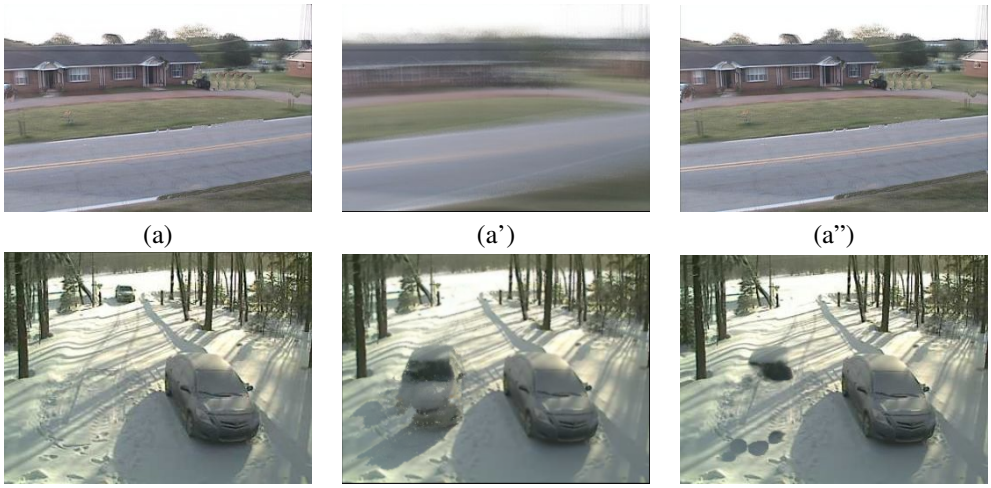


Figure 11: Comparison of background models obtained by SubSENSE and the propose approach. Each row represents the results of one sample. The first column shows the input video, the second column shows the background model from SubSENSE, and the theirs column shows the background image from proposed approach.

Since we want the network to learn representative features, we only utilize video scenes that do not challenge the background model, i.e. the background scene should not change significantly in a video. Therefore, we exclude all videos from certain categories for training (see Table 1).
We work with RGB images for background and video frames. Before we extract the patches, all employed frames are resized to the dimension 240×320 and the RGB intensities are rescaled to $[0, 1]$. Furthermore, zero padding is applied before patch extraction to avoid boundary effects. The training data consist of triplets of matching patches from video, ground truth and background frames of size 37×37 that are extracted with a stride of 10 from the employed training frames. An example mini-batch of training patches is shown in Figure. 12.
As widely recommended, we perform mean subtraction on the image patches before training, but we discard the division by the standard deviation, since we are working with RGB data and therefore each channel has the same scale.

3.2.2. Network Architecture and Optimization

The architecture of the proposed CNN is illustrated in Figure 13. The network contains 3 convolutional layers and a 2-layer Multi Layer Perceptron (MLP). We use the Rectified Linear Unit (ReLU) as activation function after each convolutional layer and the Sigmoid function after the last fully connected layer. In addition, we insert batch normalization layers [14] before each activation layer. A batch normalization layer stores the running average of the mean and standard deviation from its inputs. The stored mean is subtracted from each input of the layer and also division by the standard deviation is performed. It has been shown that by applying batch normalization layers, over-fitting is decreased and also higher learning rates for training

CDnet 2014 [27] categories	
badWeather	X
baseline	X
cameraJitter	X
dynamicBackground	X
intermittentObjectMotion	
lowFramerate	
nightVideos	X
PTZ	
shadow	X
thermal	X
turbulence	X

Table 1: Categories of CDnet 2014 [27]: Crosses indicate categories including all their video sequences that were considered for network training.

are achieved.
We train the networks with mini batches of size 150 via RMSProp [12] and a learning rate of $\alpha = 2.5 * 10^{-3}$. For the loss function, we choose the Binary Cross Entropy (BCE), which is defined as follows:

$$BCE(x, y) = -(x * \log y + (1 - x) * \log(1 - y)). \tag{8}$$

The BCE is calculated between the network outputs and the corresponding vectorized ground truth patches of size $37 * 37 = 1369$. Boundaries of foreground objects and pixels that do not lie in the region of interest are marked in the ground truth segmentations. These pixels are ignored in the cost function.

3.3. Post-Processing

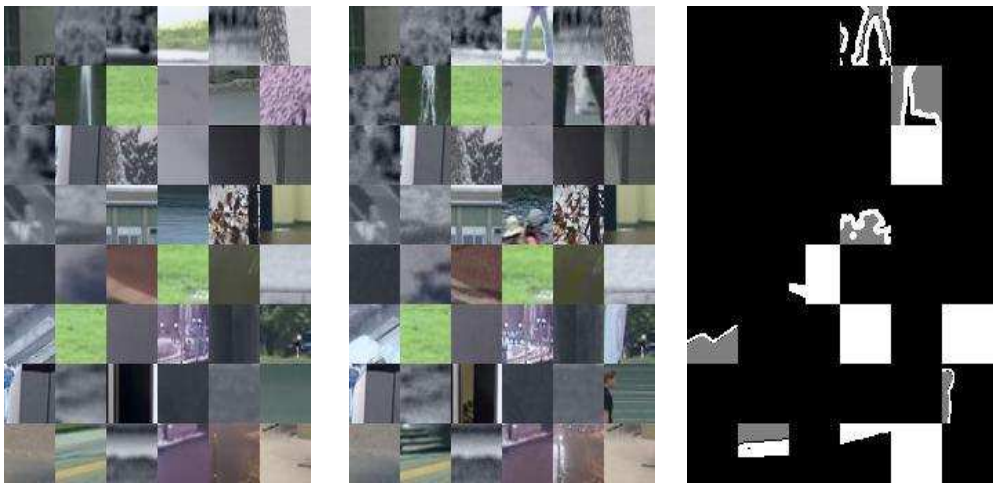
The spatial-median filtering, which is a commonly used post-processing method in background subtraction, returns the median over a neighborhood of given size (the kernel size) for each pixel in an image. As a consequence, the operation gets rid of outliers in the segmentation map and also performs blob smoothing (see Figure 14). After applying the spatial-median filter on the network output, we globally threshold the values for each pixel in order to map each pixel to $\{0, 1\}$. The threshold function is given by

$$g(z; R) = \begin{cases} 1 & \text{if } z > R \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

where R is the threshold level.

4. Experiments

In order to evaluate our approach, we conducted several experiments on various data-sets. At first, we introduce the utilized data-sets for performance testing. Afterwards, the evaluation



(a) (b) (c)

Figure 12: Visualization of a mini batch of size 48: (a) Background patches, generated with temporal-median filtering. (b) Corresponding input patches from video frames. (c) Ground truth patches, black pixels are background pixels, gray pixels are foreground pixels and white pixels are not of interest, i.e. those are not incorporated into the loss.

metrics are presented, which measure the quality of the segmentation outputs. The results on the evaluation data are subsequently reported. Furthermore, we analyze the network behavior during training. Additionally, we visualize the convolutional filters and generated feature maps at the end.

4.1. Data-sets

We employ multiple data-sets to perform our tests. The CDnet 2014 [27], the Wallflower [24] and the PETS 2009 data-set [9]. The CDnet 2014 and also the Wallflower data-set were specifically designed for the background subtraction task. These data-sets contain video sequences from different categories, which correspond to the main challenges in background subtraction and also hand segmented ground truth images are also provided. The PETS 2009 [9] data-set was designed for other purposes, such as the evaluation of tracking algorithms, and therefore no ground truth segmentations are available for its video sequences. The employed data-sets will be described in the following.

4.1.1. CDnet 2014

The CDnet 2014 [27] that was used for training of our CNNs will also be used for performance evaluation. With additional video sequences under new challenge categories, it is an extension of the CDnet 2012 [10] data-set, which is the predecessor of the CDnet 2014 [27].
For each video sequence in the data-set, corresponding ground truth segmentation images are available. For the newly added categories in CDnet 2014 [27] (see Table 2), only half of the ground truth segmentations were provided to avoid parameter tuning of background subtraction

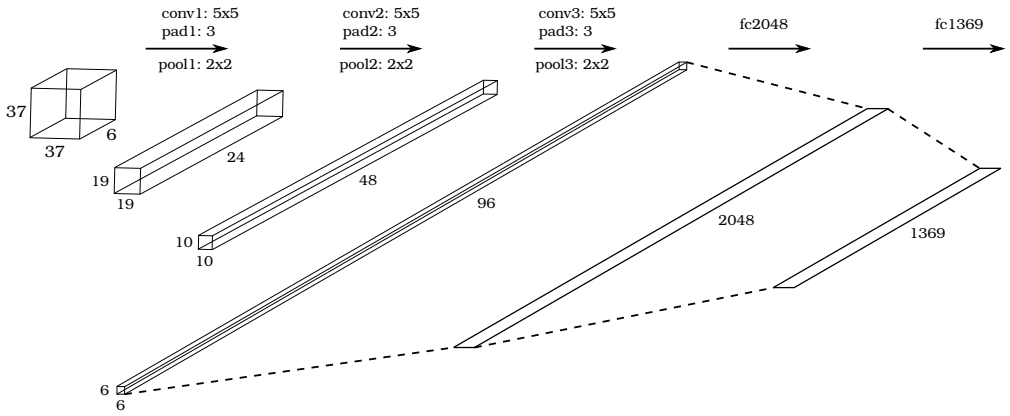
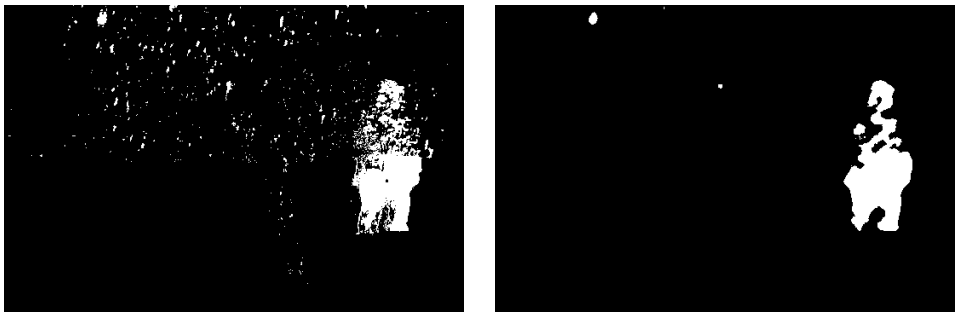


Figure 13: Illustration of proposed CNN: The network contains 3 convolutional layers and a 2 layer MLP.



(a) (b)

Figure 14: Effect of spatial-median filtering: (a) Raw segmentation result of the GMM algorithm [22]. (b) Spatial-median filtered output with a 9×9 kernel

algorithms on the benchmark data-set. One has to refer to the online evaluation[1] in order to get the results over all ground truth segmentations.

4.1.2. Wallflower

Another data-set that we employ for evaluation purpose is the Wallflower data-set [24]. For each category in the data-set, there exists a single video sequence (see Table 3). Also, for each video sequence, a hand segmented ground truth segmentation image is provided. Hence, when evaluating background subtraction algorithms on the Wallflower data-set [24], only a single ground truth segmentation is considered.

4.1.3. PETS 2009

The PETS 2009 data-set [9] is a benchmark for tracking of individual people within a crowd. It consists of multiple video sequences, recorded from static cameras, and different crowd activities. Since the data-set is not designed for background subtraction evaluation, there are no ground

[1] `http://changedetection.net`

Categories	Video Sequences	CDnet 2012 [10]	CDnet 2014 [27]
badWeather	4		X
baseline	4	X	X
cameraJitter	4	X	X
dynamicBackground	6	X	X
intermittent-ObjectMotion	6	X	X
lowFramerate	4		X
nightVideos	6		X
PTZ	4		X
shadow	6	X	X
thermal	5	X	X
turbulence	4		X

Table 2: Comparison of CDnet 2012 [10] and CDnet 2014 [27]: Categories that are marked with a cross are contained in the respective data-set.

Video sequences/categories	Number of frames	Groundtruth frame number
Bootstrap	3055	353
Camouflage	294	252
ForegroundAperture	2113	490
LightSwitch	2715	1866
MovedObject	1745	986
TimeOfDay	5890	1850
WavingTrees	287	248

Table 3: Overview of the Wallflower data-set [24]

truth segmentation images for this purpose. Thus, only the qualitative segmentation results will be evaluated without calculating any evaluation metric. A sample image from each category is represented in Fig. 15.

4.2. Evaluation Metrics

In order to measure the quality of a background subtraction algorithm, we evaluate the performance by comparing the output segmentations with the groundtruth segmentations to get the following statistics:

True Positive (TP): Foreground pixels in the output segmentation that are also foreground pixels in the ground truth segmentation.

False Positive (FP): Foreground pixels in the output segmentation that are not foreground pixels in the ground truth segmentation.

True Negative (TN): Background pixels in the output segmentation that are also background

Figure 15: Sample images of each video sequence. (a) baseline; (b) bad weather; (c) thermal; (d) camera jitter; (e) night videos; (f) shadow; (g) dynamic background; (h) Camouflage [24]; (i) WavingTrees [24]; (j) PETS2009 [9]

pixels in the ground truth segmentation.

False Negative (FN): Background pixels in the output segmentation that are not background pixels in the ground truth segmentation.

Using these statistics, different evaluation metrics are calculated that are outlined in the following:

- Recall (Re):

$$Re = \frac{TP}{TP + FN} \tag{10}$$

- Specificity (Sp):

$$Sp = \frac{TN}{TN + FP} \tag{11}$$

- Precision (Pr):

$$Pr = \frac{TP}{TP + FP} \tag{12}$$

- F Measure (FM):

$$FM = \frac{2 * Pr * Re}{Pr + Re} \tag{13}$$

- False Positive Rate (FPR):

$$FPR = \frac{FP}{FP + TN} \tag{14}$$

- False Negative Rate (FNR):

$$FNR = \frac{FN}{TP + FN} \tag{15}$$

- Percentage of Wrong Classifications (PWC):

$$PWC = 100 * \frac{FN + FP}{TP + FN + FP + TN} \tag{16}$$

We are especially interested in the FM metric since most state-of-the-art algorithms in background subtraction typically exhibit higher FM values than worse performing background subtraction algorithms. This is due to the combination of multiple evaluation metrics for the calculation of the FM. Thus, the overall performance of a background subtraction algorithm is highly coupled with its FM performance.

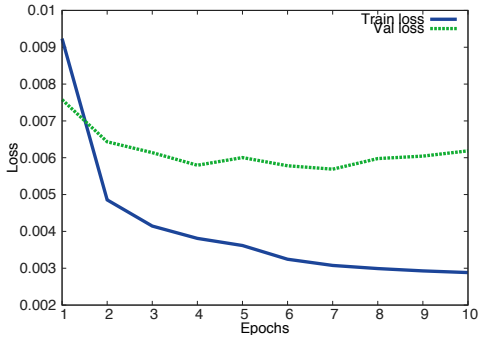
4.3. Evaluation Process

In Section 3, we have introduced our background subtraction system, consisting of a CNN, a background image retrieval and a post-processing module. For the background image retrieval and the post-processing technique, respectively two methods were proposed. The novel background image generation based on the SuBSENSE [21] and Flux tensor was proposed and for the post-processing, spatial-median filtering is considered. In order to get the best performing setup, we calculate the evaluation metrics, presented in Section 4.2, over the CDnet 2014 [27], for each setup of our background subtraction system. Also, we compare the category-wise FM and the overall FM for those data-sets with the FMs from other background subtraction algorithms. Afterward, we will select the best performing setup for further comparison. For the Wallflower data-set [24], we calculate the FM for the best setup and again, compare the values with those from other algorithms. Due to the missing ground truth images for the videos in the PETS 2009 data-set [9], we can not derive the numeric values for the FM and therefore only evaluate the segmentation outputs. In order to compare our approach with other algorithms on multiple data-sets, we need to be able to generate the corresponding segmentation, using different algorithms. For the CDnet 2014 [27], all algorithms that are listed in the online ranking[2], their evaluation metrics and segmentation results are available. For the other data-sets, namely the PETS 2009 [9] and the Wallflower data-set [24], we need to explicitly generate the segmentation images for those video sequences. For this purpose, we employed the BGSLibrary [19] as for the background image generation. As a consequence, we compare our method with certain algorithms that were online evaluated for the CDnet 2014 [27], as well as implemented in the BGSLibary [19].

4.3.1. Network Training

As already mentioned in Section 3, we train the network with mini-batches of size 150, a learning rate $\alpha = 2.5 * 10^{-3}$ over 10 epochs. The training data comprises of 150 images per video sequence in the categories listed in Table 1. The validation data contains 20 frames per video sequence. We train the network with RMSprop [12] using the BCE for the loss function. In Figure 16, the training plot is illustrated. Both networks yield a similar behavior and performance during training, mainly due to the identical network architecture and training setup.

[2] `http://changedetection.net`



(a)

Figure 16: Training and validation loss of the network using background images generated using the proposed background model.

4.4. Results

In order to measure the quality of our background subtraction algorithm, we compute the different evaluation metrics for our background subtraction system. The metrics are reported for the CDnet 2014 [27]. Also, we compare our FMs with the FMs from other background subtraction algorithms over the complete evaluation data. For this purpose, we compare our algorithm with the GMM [22], PBAS [13] and the SuBSENSE [21] algorithm. In addition, for both CDnet data-sets, we employ further background subtraction algorithms for the FM comparison. Also, we compare the segmenation outputs among the algorithms. For the comparison, we select further video sequences from the PETS 2009 data-set [9].

4.4.1. CDnet 2014

In the following, for each setup of our background subtraction system, the evaluation metrics for the CDnet 2014 are listed in Tables 4. The outputs are post-processed with a median filter with a size of 9×9. The outputs of the CNN are compared with the threshold at $R = 0.3$. As we can see in Table 4, the CNN yields very good results. In addition, the CNN yields the highest FM among the other algorithms in 6 out of 11 categories (see Table 5 and Table 6).

4.4.2. Wallflower Evaluation

For the the Wallflower data-set [24], we compare the FMs among the considered algorithms. For each video in the data-set, only a single groundtruth image is given, therefore, the FM is calculated only for this ground truth image.
The different FMs are reported in Table 7. Please note that we do not consider the "MovedObject" video for comparison since the corresponding ground truth image contains no foreground pixels and therefore, the FM can not be calculated.
From the results we can see that the CNN yields the best overall FM among the considered background subtraction algorithms.

Category	Re	Sp	FPR	FNR	PWC	FM	Pr
Baseline	0.9517	0.9991	0.0013	0.0483	0.2424	0.9580	0.9660
Camera jitter	0.8788	0.9957	0.0043	0.1212	0.8994	0.8990	0.9313
Dynamic Background	0.8543	0.9988	0.0012	0.1457	0.2067	0.8761	0.9083
Intermittent Object Motion	0.5735	0.9949	0.0051	0.4265	4.1292	0.6098	0.8251
Shadow	0.9584	0.9942	0.0058	0.0416	0.7403	0.9304	0.4844
Thermal	0.6637	0.9956	0.0044	0.3363	3.5773	0.7583	0.9257
Bad Weather	0.7517	0.9996	0.0004	0.2483	0.3784	0.8301	0.9494
Low Framerate	0.5924	0.9975	0.0025	0.4076	1.3564	0.6002	0.9677
Night Videos	0.5315	0.9959	0.0041	0.4685	2.5754	0.5835	0.8366
PTZ	0.7549	0.9248	0.0752	0.2541	7.7228	0.3133	0.2855
Turbulence	0.7979	0.9998	0.0002	0.2021	0.0838	0.8455	0.9082
Overall	0.7545	0.9905	0.0095	0.2455	1.9920	0.7548	0.8332

Table 4: Evaluation of CNN on the CDnet 2014

Method	FM_{BSL}	FM_{DBG}	FM_{CJT}	FM_{IOM}	FM_{SHD}	FM_{THM}
CNN	**0.9580**	**0.8761**	**0.8990**	0.6098	**0.9304**	0.7583
PBAS [13]	0.9242	0.6829	0.7220	0.5745	0.8143	0.7556
PAWCS [20]	0.9397	0.8938	0.8137	**0.7764**	0.8913	**0.8324**
SuBSENSE [21]	0.9503	0.8177	0.8152	0.6569	0.8986	0.8171
MBS [17]	0.9287	0.7915	0.8367	0.7568	0.8262	0.8194
GMM [22]	0.8245	0.6330	0.5969	0.5207	0.7156	0.6621
RMoG [25]	0.7848	0.7352	0.7010	0.5431	0.7212	0.4788
Spectral-360 [18]	0.9330	0.7872	0.7156	0.5656	0.8843	0.7764

Table 5: FM metric comparison of different background subtraction algorithms over the 6 categories of the CDnet2014, namely 1) Baseline (BSL), Dynamic background (DBG), Camera jitter (CJT), Intermittet objectmotion (IOM), Shadow (SHD), Thermal (THM).

Method	FM_{BDW}	FM_{LFR}	FM_{NVD}	FM_{PTZ}	FM_{TBL}
CNN	0.8301	0.6002	**0.5835**	0.3133	**0.8455**
PBAS [13]	0.7673	0.5914	0.4387	0.1063	0.6349
PAWCS [20]	0.8152	**0.6588**	0.4152	0.4615	0.6450
SuBSENSE [21]	**0.8619**	0.6445	0.5599	0.3476	0.7792
MBS [17]	0.7730	0.6279	0.5158	**0.5118**	0.5698
GMM [22]	0.7380	0.5373	0.4097	0.1522	0.4663
RMoG [25]	0.6826	0.5312	0.4265	0.2400	0.4578
Spectral-360 [18]	0.7569	0.6437	0.4832	0.3653	0.5429

Table 6: FM metric comparison of different background subtraction algorithms over the 5 categories of the CDnet2014, namely Bad weather (BDW), Low framerate (LFM), Night videos (NVD), PTZ (PTZ), Turbulence (TBL)

FM_{Video}	CNN	SuBSENSE [21]	PBAS [13]	GMM [22]
$FM_{Bootstrap}$	**0.7479**	0.4192	0.2857	0.5306
$FM_{Camouflage}$	**0.9857**	0.9535	0.8922	0.8307
$FM_{ForegroundAperture}$	0.6583	**0.6635**	0.6459	0.5778
$FM_{LightSwitch}$	**0.6114**	0.3201	0.2212	0.2296
$FM_{TimeOfDay}$	0.5494	0.7107	0.4875	**0.7203**
$FM_{WavingTrees}$	0.9546	0.9597	0.8421	**0.9767**
Overall	**0.7512**	0.6711	0.5624	0.6443

Table 7: F-Measures for the Wallflower dataset [24]

4.5. Discussion

The output segmentations are already precise and not very prone to outliers (e.g. through dynamic background regions), the main problems are the camouflage regions within foreground objects. In the first 6 categories of CDnet 2014, where mainly the feature extraction and segmentation are challenging, the CNN gives the best results, compared with other algorithms, in 6 out of 11 categories. For categories such as the "PTZ (Pan Tilt Zoom)" or "low framerate" category, the CNN yields poor results since the background images provided by the proposed algorithm are insufficient for performing good background subtraction. Therefore, our method gives poor segmentation for these categories and as a consequence, the average FM drops significantly.
However, for Wallflower data-set [24], where in most cases the background model is not challenged, our algorithm outperforms all other algorithms in terms of segmentation quality and overall FM.
To sum up, our system outperforms the existing algorithms when the challenge does not lie in the background modeling/maintenance, but on the other hand, due to the corruption of the background images, our method performs poorly once there are large changes in the background scenery.

4.5.1. Network Analysis

For further understanding and intuition, we visualize the convolutional filters and the feature maps when an image pair is fed into the network. In order to replace feature engineering, filters are employed which are learned during training. Since the learning is performed with ground truth data, highly application specific features will be extracted. Our network contains 3 convolutional layers which perform the feature extraction, especially the convolutional filters, the so-called kernels, are responsible for that task. Some of those will be illustrated in the following. All convolutional filters in our network are of size 5×5. The filters that are directly applied to the RGB image pair of input and background image are illustrated in Figure 19.
Due to the large number of filters, we only show the filter-sets that output the first feature map of the respective convolutional layer. Those filter-sets from the remaining convolutional layers are

shown in Figure 20. By applying the filters onto the input images or feature maps, the convolutional layer scans the input for certain patterns, that are defined by the filters, that again generate new feature maps which capture task-specific characteristics.

4.5.2. Feature Analysis

By applying the learned filters to the network input, passing the intermediate values through the activation and subsampling functions, feature maps are generated. Those are again the input of the subsequent convolutional layer or classifier. In order to analyze the generated feature maps, we first need to feed an input into the network. An example input pair consisting of background and input frame is shown along with the CNN output in Figure 18. The output feature maps from all convolutional layers in our network are illustrated in Figures. 19 and 20. At the end, the vectorized feature maps of the last convolutional layer in our CNN are fed into a MLP that performs the pixel-wise classification.

5. Conclusion

We have proposed a novel approach based on deep learning for background subtraction task. The proposed approach includes three processing steps, namely background model generation, CNN for feature learning and post-processing. It turned out that the CNN yielded the best performance among all other algorithms for background subtraction. In addition, we analyzed our network during training and visualized the convolutional filters and generated feature maps. Due to the fact that our method works only with RGB data, the potential of Deep Learning approaches and the available data, one could think of modeling the background with Deep Learning techniques, for example with RNN. Furthermore, as we use a global threshold for our network outputs at the moment, one could use adaptive pixel-wise thresholds, as in the PBAS algorithm, employing a kind of ”background dynamics” measure for the feedback loop. With this adaption, one could increase the sensitivity in static background areas and decrease it for areas with dynamic background. At last, when combining our method with an existing background subtraction algorithm, in our case with the SuBSENSE algorithm, one could use the information of both output segmentations and combine them to get a refined output and employ this output for improving the updates of the background model.

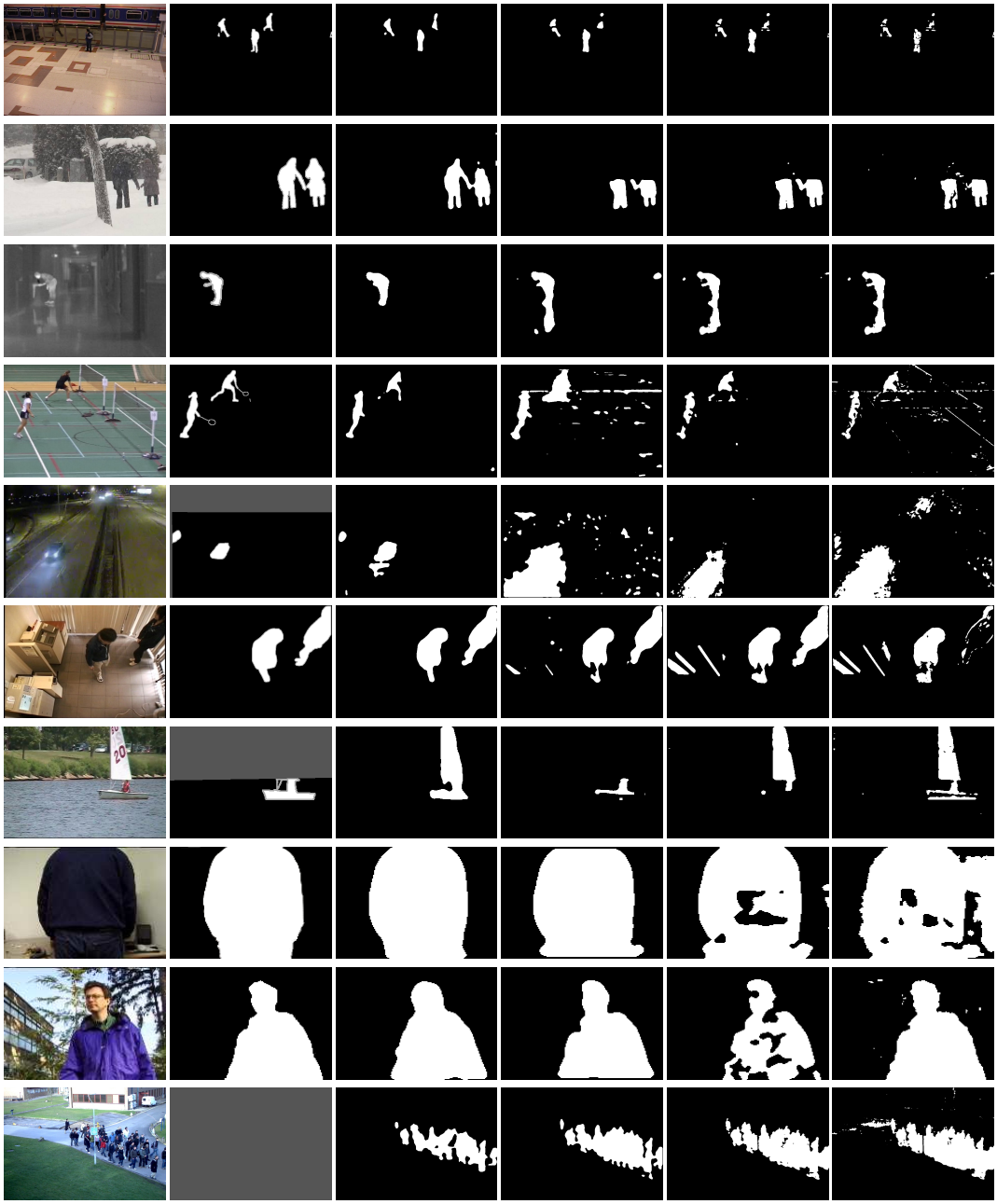
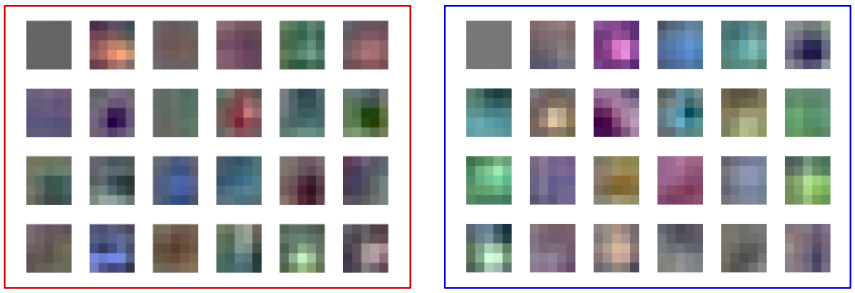


Figure 17: Comparison of the segmentation outputs: The first column is the input image, the second column is its ground truth image, the third column represents the output of the CNN. The fourth column is the output of the SuBSENSE [21], the fifth column represents the output of the PBAS [13] and the last column shows the output of the GMM [22] segmentation. Gray pixels in the ground truth segmentation indicate pixels which are not of interest

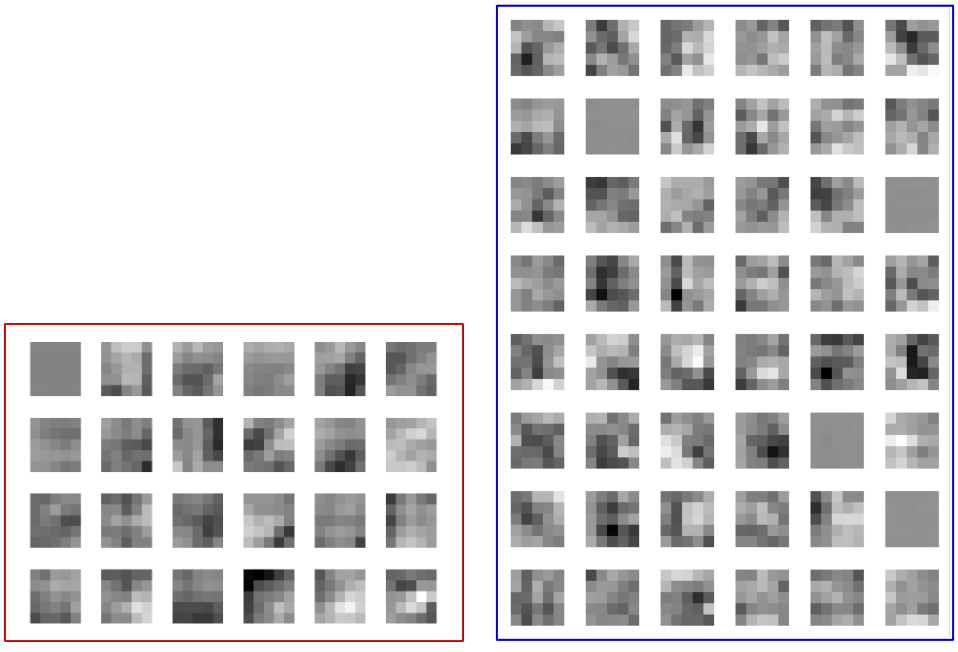
(a) (b) (c)

Figure 18: Network input and output: (a) Background image of the input pair. (b) Input frame. (c) Network output.



(a) (b)

Figure 19: Visualization of the kernels in the first convolutional layer: (a) Kernels for processing the input image. (b) Kernels that process the background image.



(a) (b)

Figure 20: Kernels of the second and third convolutional layer: (a) 24 kernels to generate the first output feature map in the second convolutional layer. (b) 48 kernels that output the first feature map in the third convolutional layer.

References

[1] O. Barnich and M. Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image processing*, 20(6):1709–1724, 2011.

[2] G.-A. Bilodeau, J.-P. Jodoin, and N. Saunier. Change detection in feature space using local binary similarity patterns. In *Computer and Robot Vision (CRV), 2013 International Conference on*, pages 106–112. IEEE, 2013.

[3] M. Braham and M. Van Droogenbroeck. Deep background subtraction with scene-specific convolutional neural networks. In *International Conference on Systems, Signals and Image Processing, Bratislava 23-25 May 2016*. IEEE, 2016.

[4] F. Bunyak, K. Palaniappan, S. K. Nath, and G. Seetharaman. Flux tensor constrained geodesic active contours with sensor fusion for persistent object tracking. *Journal of Multimedia*, 2(4):20, 2007.

[5] Y. Chen, J. Wang, and H. Lu. Learning sharable models for robust background subtraction. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2015.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[7] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *European conference on computer vision*, pages 751–767. Springer, 2000.

[8] R. H. Evangelio and T. Sikora. Complementary background models for the detection of static and moving objects in crowded environments. In *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, pages 71–76. IEEE, 2011.

[9] J. Ferryman and A. Shahrokni. An overview of the pets 2009 challenge. *Eleventh IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2009.

[10] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection. net: A new change detection benchmark dataset. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2012.

[11] M. Heikkila and M. Pietikainen. A texture-based method for modeling the background and detecting moving objects. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):657–662, 2006.

[12] G. Hinton, N. Srivastava, and K. Swersky. *Lecture 6a Overview of mini–batch gradient descent*, 2012.

[13] M. Hofmann, P. Tiefenbacher, and G. Rigoll. Background segmentation with feedback: The pixel-based adaptive segmenter. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 38–43. IEEE, 2012.

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[15] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground–background segmentation using codebook model. *Real-time imaging*, 11(3):172–185, 2005.

[16] N. M. Oliver, B. Rosario, and A. P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE transactions on pattern analysis and machine intelligence*, 22(8):831–843, 2000.

[17] H. Sajid and S.-C. S. Cheung. Background subtraction for static & moving camera. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 4530–4534. IEEE, 2015.

[18] M. Sedky, C. Chibelushi, and M. MONIRI. Image processing: Object segmentation using full-spectrum matching of albedo derived from colour images, 2010.

[19] A. Sobral. BGSLibrary: An opencv c++ background subtraction library. In *IX Workshop de Viso Computacional (WVC'2013)*, Rio de Janeiro, Brazil, Jun 2013.

[20] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin. A self-adjusting approach to change detection based on background word consensus. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 990–997. IEEE, 2015.

[21] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin. Subsense: A universal change detection method with local adaptive sensitivity. *IEEE Transactions on Image Processing*, 24(1):359–373, 2015.

[22] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[23] P. Tiefenbacher, M. Hofmann, D. Merget, and G. Rigoll. Pid-based regulation of background dynamics for foreground segmentation. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 3282–3286. IEEE, 2014.

[24] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 255–261. IEEE, 1999.

[25] S. Varadarajan, P. Miller, and H. Zhou. Region-based mixture of gaussians modelling for foreground detection in dynamic scenes. *Pattern Recognition*, 48(11):3488–3503, 2015.

[26] R. Wang, F. Bunyak, G. Seetharaman, and K. Palaniappan. Static and moving object detection using flux tensor with

split gaussian models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[27] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar. Cdnet 2014: an expanded change detection benchmark dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 387–394, 2014.