MÉTHODES DE VISION À LA MOTION ET LEURS APPLICATIONS

MOTION BASED VISION METHODS AND THEIR APPLICATIONS

par

Yi Wang

Thèse présentée au Département d'informatique
en vue de l'obtention du grade de philosophiæ doctor (Ph.D.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 8 Juin 2017

Le 08 Juin 2017

Le jury a accepté le mémoire de Monsieur Yi Wang dans sa version finale.

Membres du jury

Professeur Pierre-Marc Jodoin
Directeur de recherche
Département d'informatique, Université de Sherbrooke

Professeur Maxime Descoteaux
Membre interne
Département d'informatique, Université de Sherbrooke

Professeur Christian Desrosiers
Membre externe
Département de génie logiciel et des TI
École de technologie supérieure

Professeur Jean-Pierre Dussault
Prśident-rapporteur
Département d'informatique, Université de Sherbrooke

Sommaire

La détection de mouvement est une opération de base souvent utilisée en vision par ordinateur, que ce soit pour la détection de piétons, la détection d'anomalies, l'analyse de scènes vidéo ou le suivi d'objets en temps réel. Bien qu'un très grand nombre d'articles ait été publiés sur le sujet, plusieurs questions restent en suspens. Par exemple, il n'est toujours pas clair comment détecter des objets en mouvement dans des vidéos contenant des situations difficiles à gérer comme d'importants mouvements de fonds et des changements d'illumination. De plus, il n'y a pas de consensus sur comment quantifier les performances des méthodes de détection de mouvement. Aussi, il est souvent difficile d'incorporer de l'information de mouvement à des opérations de haut niveau comme par exemple la détection de piétons.

Dans cette thèse, j'aborde quatre problèmes en lien avec la détection de mouvement :

1. Comment évaluer efficacement des méthodes de détection de mouvement ? Pour répondre à cette question, nous avons mis sur pied une procédure d'évaluation de telles méthodes. Cela a mené à la création de la plus grosse base de données 100% annotée au monde dédiée à la détection de mouvement et organisé une compétition internationale (CVPR 2014). J'ai également exploré différentes métriques d'évaluation ainsi que des stratégies de combinaison de méthodes de détection de mouvement.

2. L'annotation manuelle de chaque objet en mouvement dans un grand nombre de vidéos est un immense défi lors de la création d'une base de données d'analyse vidéo. Bien qu'il existe des méthodes de segmentation automatiques et semi-automatiques, ces dernières ne sont jamais assez précises pour produire des résultats de type "vérité terrain". Pour résoudre ce problème, nous avons proposé une méthode interactive de segmentation d'objets en mouvement basée sur l'apprentissage profond. Les résultats

obtenus sont aussi précis que ceux obtenus par un être humain tout en étant 40 fois plus rapide.

3. Les méthodes de détection de piétons sont très souvent utilisées en analyse de la vidéo. Malheureusement, elles souffrent parfois d'un grand nombre de faux positifs ou de faux négatifs tout dépendant de l'ajustement des paramètres de la méthode. Dans le but d'augmenter les performances des méthodes de détection de piétons, nous avons proposé un filtre non linéaire basée sur la détection de mouvement permettant de grandement réduire le nombre de faux positifs.

4. L'initialisation de fond (background initialization) est le processus par lequel on cherche à retrouver l'image de fond d'une vidéo sans les objets en mouvement. Bien qu'un grand nombre de méthodes ait été proposé, tout comme la détection de mouvement, il n'existe aucune base de donnée ni procédure d'évaluation pour de telles méthodes. Nous avons donc mis sur pied la plus grosse base de données au monde pour ce type d'applications et avons organisé une compétition internationale (ICPR 2016).

Mots-clés: Détection de mouvement, détection de piétons, estimation d'image de fond.

Summary

Motion detection is a basic video analytic operation on which many high-level computer vision tasks are built upon, *e.g.*, pedestrian detection, anomaly detection, scene understanding and object tracking strategies. Even though a large number of motion detection methods have been proposed in the last decades, some important questions are still unanswered, including : (1) how to separate the foreground from the background accurately even under extremely challenging circumstances ? (2) how to evaluate different motion detection methods ? And (3) how to use motion information extracted by motion detection to help improving high-level computer vision tasks ?

In this thesis, we address four problems related to motion detection :

1. How can we benchmark (and on which videos) motion detection method ? Current datasets are either too small with a limited number of scenarios, or only provide bounding box ground truth that indicates the rough location of foreground objects. As a solution, we built the largest and most objective motion detection dataset in the world with pixel accurate ground truth to evaluate and compare motion detection methods. We also explore various evaluation metrics as well as different combination strategies.

2. Providing pixel accurate ground truth is a huge challenge when building a motion detection dataset. While automatic labeling methods suffer from a too large false detection rate to be used as ground truth, manual labeling of hundreds of thousands of frames is extremely time consuming. To solve this problem, we proposed an interactive deep learning method for segmenting moving objects from videos. The proposed method can reach human-level accuracies while lowering the labeling time by a factor of 40.

3. Pedestrian detectors always suffer from either false positive detections or false ne-

gative detections all depending on the parameter tuning. Unfortunately, manual adjustment of parameters for a large number of videos is not feasible in practice. In order to make pedestrian detectors more robust on a large variety of videos, we combined motion detection with various state-of-the-art pedestrian detectors. This is done by a novel motion-based nonlinear filtering process which improves detectors by a significant margin.

4. Scene background initialization is the process by which a method tries to recover the RGB background image of a video without foreground objects in it. However, one of the reasons that background modeling is challenging is that there is no good dataset and benchmarking framework to estimate the performance of background modeling methods. To fix this problem, we proposed an extensive survey as well as a novel benchmarking framework for scene background initialization.

Keywords: Motion detection, pedestrian detection, background image estimation.

Remerciements

I would like to express my sincere gratitude to my supervisor Prof. Pierre-Marc Jodoin. In the last four years, he not only helped me with my research with his immense professional knowledge, his patience, enthusiasm and motivation, he is also a good example for me in work and life.

I would also like to thank Dr. Lucia Maddalena and Dr. Alfredo Petrosino for their help with the scene background modeling dataset.

And my sincere thanks also goes to my labmates of the VITALlab at the Université de Sherbrooke, for all the discussions, encouragements and joys in the last four years.

In the end, I want to thank my family, who love me, trust me, understand me, support me, encourage me for all these years. I would not go this far without you.

Table des matières

Table des figures

Liste des tableaux

Introduction

Motion detection is a basic and important task in computer vision and video processing. A large number of high level computer vision tasks such as object tracking [248], scene understanding [107], anomaly detection [133], and traffic analytics [148] rely on motion detection. Its importance can be gauged by the large number of algorithms that have been developed to-date and the even larger number of articles that have been published on this topic. A quick search for "motion detection" on IEEE Xplore returns over 20,000 papers. This shows that motion detection is a fundamental topic for a wide range of video analytic applications. And it also proves that the number of motion detection methods proposed so far is impressively large.

Even though hundreds of papers have been published in the past decade, motion detection is still not considered as a solved problem. It appears that no single method can provide good performance in all challenging circumstances. Such challenges include sudden illumination variations, night scenes, background movements, illumination changes, low frame rate, shadows, camouflage effects (photometric similarity of object and background), ghosting artifacts (delayed detection of a moving object after it has moved away), *etc*. In this work, we aim to answer four questions related to motion detection.

This thesis starts with a short introduction of motion detection methods. Methods are classified into seven categories. The features, updating strategies and post-processing technologies that are usually used in motion detection are also described. Popular datasets used to evaluate motion detection methods are also listed and compared in that chapter.

Unfortunately, all these datasets have their limits, which makes it impossible to compare motion detection methods objectively. To solve this problem, in Chapter 2 we propose a more objective dataset : changedetction.net 2014 (CDnet 2014) [237], an extension of the previously released 2012 version of that dataset. With 75 videos in 11 challenging categories and seven evaluation metrics, CDnet 2014 is the largest and most objective motion detection dataset in the world. More details about the dataset are described in Chapter 2. In order to understand motion detection methods more deeply, a series of benchmarking experiments have been done, which are also mentioned in Chapter 2.

While building a motion detection dataset, ground truthing is always an important but time-consuming task. In each frame of a video, the ground truth is a binary mask that indicates whether a pixel in the frame belongs to the foreground or the background. It may also have other labels to indicate the region of interests (ROI), shadow, and uncertain areas. An accurate ground truth for a dataset may require several months of manual labeling. Because of that, most datasets either only label a very small number of frames for each video [219, 131], or only provide a bounding box around each moving object [261, 224]. To solve this problem, in Chapter 3, we propose an interactive deep learning method for segmenting moving objects [239]. By labeling only a small number of frames, our model can learn the appearance of the background and the foreground moving objects, and generalize the segmentation to the rest of the video frames. As will be shown, the average F-measure of our method is within the error margin of a human being.

Motion detection methods can be used to improve the performance of other higher level computer vision tasks, *e.g.* pedestrian detection. By considering motion information, we proposed a method to filter out the background of a video in order to decrease the false detection rate of pedestrian detectors [240]. Our method is robust and easy to combine with state-of-the-art pedestrian detectors. This part is discussed in Chapter 4.

The most straight forward way of detecting motion in a video is to separate the foreground from the background with a simple background subtraction. For this method, motion is defined as any significant pixel-wise difference between a frame in the video and a background image. But to do this, we need a background image void of foreground objects. Although many background initialization papers have been published, it is not easy to objectively estimate how good these background initialized images are.

In Chapter 5, we propose an extensive survey of scene background initialization methods as well as a novel benchmarking framework involving seven evaluation metrics, 14 different state-of-the-art methods, as well as the largest video dataset ever made for this purpose [108].

Accomplished Work

The follows are the projects that I am involved and their correlating publications during my Ph.D.

1. I was involved in a motion-tracking-based scene understanding project. With the help of optical flow, we proposed a method to estimate the motion patterns of traffic. Our method works both with sparse and crowded video scenes. A conference paper was published based on the subject :
   — Jodoin, P.M., Benezeth, Y., **Wang, Y.**, "*Meta-tracking for video scene understanding*", Advanced Video and Signal Based Surveillance, 2013, pp. 1-6.

   This paper is not mentioned in the rest of the thesis.
2. I worked on a motion detection benchmarking project. I reviewed a large number of motion detection methods, features, updating strategies, and post-processing methods and tested it on the CDnet 2012 dataset. I also extended the 2012 dataset which led to the CDnet 2014 dataset. This also led to the organization of a CVPR challenge in 2014. This work allowed me to publish the following two papers :
   — Jodoin, P.M., Pierard, S., **Wang, Y.**, Droogenbroeck, V., "*Overview and benchmarking of motion detection methods*", Background Modeling and Foreground Detection for Video Surveillance, 2014.
   — **Wang, Y.**, Jodoin, P.M., Porikli, F., Konrad, J., Benezeth, Y., and Ishwar, P., "*CDnet 2014 : An Expanded Change Detection Benchmark Dataset*", in Proc. IEEE Workshop on Change Detection (CDW-2014) at CVPR-2014, pp. 387-394.

   Chapter 2 is a combination of those two papers.
3. I proposed a multi-scale cascaded convolutional neural network used to segment foreground objects in videos. The model achieves human-level accuracies while

being 40 times faster than manual labeling. This led to the publication of the following journal paper :

— **Wang, Y.**, Luo, Z. M., and Jodoin, P. M. "*Interactive Deep Learning Method for Segmenting Moving Objects*", Pattern Recognition Letters, 2016.

This paper is the content of Chapter 3.

4. I worked on a method used to combine motion information and pedestrian detectors to improve the performances of state-of-the-art pedestrian detectors. A workshop paper and a journal paper were published on that topic :

   — **Wang, Y.**, Piérard, S., Su, S. Z., and Jodoin, P. M. "*Nonlinear Background Filter to Improve Pedestrian Detection*", in New Trends in Image Analysis and Processing–ICIAP 2015 Workshops, pp. 535-543.

   — **Wang, Y.**, Piérard, S., Su, S. Z., and Jodoin, P. M. "*Improving pedestrian detection using motion-guided filtering*", Pattern Recognition Letters, 2016.

   The journal paper is the content of Chapter 4.

5. I Co-organized the Scene Background Modeling Contest in conjunction with ICPR 2016. We provided the largest scene background modeling dataset and an online evaluation system (scenebackgroundmodeling.net). This led to a journal paper accepted by IEEE Transactions on Image Processing :

   — Jodoin P. M., Maddalena, L., Petrosino, A. and **Wang Y.** "*Extensive Benchmark and Survey of Modeling Methods for Scene Background Initialization*", Accepted by IEEE Transactions on Image Processing, 2017.

   This paper is the content of Chapter 5. Note that for this paper, **the author order is in alphabetical order**.

Chapter 1

Previous Work

1.1 Motion Detection Methods

Motion detection is often achieved by first building a representation of the scene, called "the background model", and then observing deviations from this model for each incoming frame. Sufficient changes from the background model are assumed to indicate the presence of moving objects. In this chapter, we report the most commonly-used models which we refer to as the *basic*, *parametric*, *non-parametric*, *data-driven* and *matrix decomposition* models. Other models for motion detection are also accounted for in this section such as the *prediction* model, the *motion segmentation* model, and the *machine learning* approaches, including *deep learning* models. All these motion detection methods are summarized in Table 1.1.

Together with these eight families of methods, we review commonly-used features, spatial aggregation techniques, updating scheme as well as post-processing methods.

1.1.1 Basic Models

The simplest strategy to detect motion is to subtract the pixel's color in the current frame from the corresponding pixel's color in the background model [22]. Given a background

Table 1.1 – Overview of eight families of motion detection methods.

Motion detection families	**References**
Basic	Running average [269, 22, 118, 105, 99] Temporal median [156] Motion history image [27, 162, 170]
Parametric	Single Gaussian [246] Gaussian mixture model (GMM) [210, 110, 274, 106, 65, 66, 191, 87, 255, 249] Background clustering [38, 101, 115] Generalized Gaussian model [9, 114] Bayesian [131, 183, 231] Chebyshev inequality [162]
None-parametric	Kernel density estimation (KDE) [64, 161, 168, 258, 266, 109]
Data-driven	Cyclostationary [184] Stochastic K-nearest neighbors (KNN) [16, 93] Deterministic KNN [274] Hidden Markov model (HMM) [212]
Matrix decomposition	Principal component analysis (PCA) [169, 254, 134, 59, 194] Sparsity and dictionary learning [182, 267]
Prediction model	Kalman filter [112, 270] Weiner filter [219]
Motion segmentation	Optical flow segmentation [244, 158, 147] GMM and optical flow segmentation [101, 271]
Machine learning	SVM [138, 91, 89] 1-class SVM [47] Neural networks [152, 154, 196]
Deep learning	Convolutional neural network (ConvNets or CNN) [123, 239]

B, the foreground of frame I at time t is estimated as:

$$F^t = \begin{cases} 1, & \text{if } |I^t - B^t| \geq tr \\ 0, & \text{otherwise}, \end{cases} \tag{1.1}$$

where tr is a threshold for the binarization.

The basic models usually generate the background model with a statistical method. For example, A temporal median filter can be used to estimate a color-based background model [156]. Denote N as the number of the frames in the training set of a video, a temporal median filter calculates the background as:

$$B_{(x,y)} = \text{median}(I^1_{(x,y)}, I^2_{(x,y)}, I^3_{(x,y)}, ..., I^N_{(x,y)}). \tag{1.2}$$

One can also generalize to other features such as color histograms [118, 269] and local self-similarity features [105]. Avola *et al.* [11] developed the pixel-by-pixel subtraction into a blob subtraction filter, which is more robust for noisy videos. Beyond that, a key points matching stage is also added for the background updating to increase the model stability. However, these temporal filtering methods are sensitive to compression artifacts, global illumination changes, and are incapable to detect moving objects once they become stationary.

Frame differencing [4] is another basic motion detection method. It aims to detect changes in the state of a pixel by subtracting the pixel's intensity (or color) in the current frame from its intensity (or color) in the previous frame. Although this method is computationally inexpensive, it cannot detect a moving object once it stops moving or when the object motion becomes small. Instead, it typically detects object boundaries, covered and exposed areas due to object motion. Frame differencing detects the edge of the foreground between two continuous frames I_{t-1} and I_t as:

$$F^t_{(x,y)} = \begin{cases} 1, & \text{if } |I^{t-1}_{(x,y)} - I^t_{(x,y)}| \geq tr \\ 0, & \text{otherwise}. \end{cases} \tag{1.3}$$

Motion history image (MHI) [27, 162, 170] is also used as a basic model for motion detection. It is obtained by successive layering of frame differences. For each new frame, the current motion history image is scaled down in amplitude, subject to some threshold, on which, the new motion label field is overlaid using its full amplitude range. In consequence, image dynamics ranging from two consecutive frames to several dozen frames can be captured in a single image. There is also a simplified way to calculate motion history image, which initializes the MHI with a fixed intensity τ and reduces it by 1 when no motion is detected:

$$\mathrm{MHI}^{t}_{(x,y)} = \begin{cases} \tau, & \text{if } |I^{t}_{(x,y)} - I^{t-1}_{(x,y)}| \geq tr \\ \max(0, \mathrm{MHI}^{t-1}_{(x,y)} - 1), & \text{if } |I^{t}_{(x,y)} - I^{t-1}_{(x,y)}| < tr. \end{cases} \tag{1.4}$$

1.1.2 Parametric Models

In order to improve robustness to noise, parasite artifacts, and background motion, the use of a per-pixel Gaussian model has been proposed [246]. In a first step, the mean μ and standard deviation σ are computed for each pixel. Then, for each frame, the likelihood p of each pixel color is determined and pixels whose probability is below a certain threshold tr are labeled as foreground pixels:

$$F^{t}_{(x,y)} = \begin{cases} 1, & \text{if } p(I^{t}_{(x,y)}|\mu, \sigma) < tr \\ 0, & \text{otherwise.} \end{cases} \tag{1.5}$$

Since pixels in noisy areas are given a larger standard deviation, a larger intensity variation is needed in those areas to detect motion. This is fundamentally different from the basic models for which the tolerance is fixed for every pixel. As shown by Kim *et al.* [114], a generalized Gaussian model can also be used and Morde *et al.* [162] have shown that a Chebychev inequality can further improve the performance.

Single Gaussian Model, however, is not a good model for dynamic scenes [76] as multiple colors may be observed at a location due to repetitive object motion, shadows, or reflectance changes. A substantial improvement is achieved by using multiple statistical models to describe background color. A Gaussian mixture model (GMM) [210] was proposed to represent each background pixel. Given Θ the mixture parameters, for a mixture of K (usually between 3 to 10) Gaussian models, the probability to observe a pixel value at (x, y) is calculated as:

$$p(I^t_{(x,y)}|\Theta) = \sum_{i=1}^{K} w^{i,t}_{(x,y)} * \mathcal{N}(I^t_{(x,y)}; \mu^{i,t}_{(x,y)}, \Sigma^{i,t}_{(x,y)}), \tag{1.6}$$

where $w^{i,t}$ is the weight of the ith Gaussian model at time t; $\mu^{i,t}$ and $\sum^{i,t}$ are the mean and the covariance matrix of the same Gaussian respectively; while $\eta(\cdot)$ is the probability density function of the Gaussian model defined as:

$$\eta(I^t_{(x,y)}, \mu^t, \Sigma^t) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma^t|^{\frac{1}{2}}} e^{-\frac{1}{2}(I^t_{(x,y)}-\mu^t)^T \Sigma^{t-1}(I^t_{(x,y)}-\mu^t)}. \tag{1.7}$$

GMM compares each pixel in the current frame with every model in the mixture until a matching Gaussian is found.

If a new pixel value matches the kth Gaussian models, *i.e.*, the pixel value is within 2.5 standard deviation of the Gaussian distribution, the weight, the mean and the standard deviation of that Gaussian will be updated as:

$$w^{k,t}_{(x,y)} = (1-\alpha)w^{k,t-1}_{(x,y)} + \alpha, \tag{1.8}$$

$$\mu^{k,t}_{(x,y)} = (1-\rho)\mu^{k,t-1}_{(x,y)} + \rho(I^t_{(x,y)}), \tag{1.9}$$

$$\Sigma^{k,t}_{(x,y)}{}^2 = (1-\rho)\Sigma^{k,t-1^2}_{(x,y)} + \rho(I^t_{(x,y)} - \mu^{k,t}_{(x,y)})^T(I^t_{(x,y)} - \mu^{k,t}_{(x,y)}), \tag{1.10}$$

where α and ρ are two learning parameters.

If no Gaussian distribution is matched, the one with the lowest weight in the current mixture will be replaced by a new Gaussian distribution with its mean equal to the current pixel value. The weight of it will be initialized to be very low and the standard deviation will be set to be very high.

After that, the K Gaussians will be ordered according to their $w^{k,t}/\sigma^{k,t}$. The first b Gaussians which satisfy the following equation will be chosen to be the background model BM for that pixel:

$$BM^t_{(x,y)} = \arg\min_b(\sum_{k=1}^{b} w^{k,t}_{(x,y)} > tr). \tag{1.11}$$

If the new pixel matches any of the b Gaussians, it will be classified as background, else foreground. To be notice, to classify a pixel, instead of calculating a probability function and thresholding it, GMM tries to find the most significant Gaussians to represent the background:

$$F^t_{(x,y)} = \begin{cases} 1, & \text{if } I^t_{(x,y)} \text{ does not match } BM^t_{(x,y)} \\ 0, & \text{otherwise.} \end{cases} \tag{1.12}$$

Instead of relying on only one pixel, GMM can be trained to incorporate extended spatial information [106]. Several papers [110] improved the GMM approach to add robustness when shadows are present and to make the background models more adaptive to parasitic background motion. A recursive method with an improved update of the Gaussian parameters and an automatic selection of the number of modes was presented in [274]. Haines *et al.* [87] also proposed an automatic mode selection method, but with a Dirichlet process. A splitting GMM that relies on a new initialization procedure and a mode splitting rule was proposed in [65, 66]. The splitting strategy helped to avoid over-dominating modes and resolve problems due to newly static objects and moved away background objects. A multi-resolution block-based GMM model was introduced in [191]. Yadav *et al.* [255] built GMM for both pure ground truth frames and the entire video, and used Kullback–Leibler divergence to auto adjust for the final thresholding. The GMM approach can also be expanded to include the generalized Gaussian model [9]. One limit of GMM is that it only considers the temporal information of each pixel without using any spacial knowledge. To solve this problem, Xia *et al.* [249] combined GMM with a spacial model. When the new frame arrives, each pixel is classified by a spacial model, *i.e.*, if the pixel appears in its neighborhood less than a certain times, it will be classified as foreground. If not, the traditional temporal GMM is then applied.

As an alternative to mixture models, Bayesian approaches have also been proposed for motion detection. In [183], each pixel is modeled as a combination of layered Gaussians. Recursive Bayesian update instead of the conventional expectation maximization fitting is performed to update the background parameters and better preserve the multi-modality of the background model. A similar Bayesian decision rule with various features and a learning method that adapt to both sudden and gradual illumination changes is used in [131]. In [231] heterogeneous features such as brightness variation, chromaticity variation, and texture variation are extracted to estimate the conditional probability densities for both foreground and background, pixels are then labeled with Bayes rule.

Another alternative to GMM is background clustering. In this case, each background pixel is assigned a certain number of clusters depending on the intensity variation observed in the training video sequence. Then, each incoming pixel whose color is close to a background cluster is considered part of the background. The clustering can be done using K-means (or a variant of it) [38, 101] or codebook [115].

1.1.3 Non-Parametric Methods

In contrast to parametric models, non-parametric kernel density estimation (KDE) tries to estimate the probability of observing a pixel value by accumulating the kernel probability density function temporally [64]. For KDE, the probability of observing a pixel value equal to $I^t_{(x,y)}$ is estimated as:

$$p(I^t_{(x,y)}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{K}(I^t_{(x,y)} - I^i_{(x,y)}), \tag{1.13}$$

where $I^i_{(x,y)}$ is the pixel value at time i, N is the total number of pixels in the sample, and $\mathcal{K}$ is a kernel function. If $\mathcal{K}$ is defined as a normal Gaussian function $G(0, \Sigma)$, and assuming the three color channels are independent from each other, the probability density function can be rewritten as:

$$p(I^t_{(x,y)}) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{3}\frac{1}{\sqrt{2\pi\sigma_j^2}}e^{-\frac{1}{2}\frac{(I^{tj}_{(x,y)} - I^{ij}_{(x,y)})^2}{\sigma_j^2}}, \tag{1.14}$$

in which, the standard deviation for each channel is estimated as a function of the difference between the pixel values of the current frame and the previous frame:

$$\sigma = \frac{|I^t_{(x,y)} - I^{t-1}_{(x,y)}|}{0.68\sqrt{2}}. \tag{1.15}$$

Once the probability density function is estimated, the probability is calculated when a new pixel $I^t_{(x,y)}$ arrives. The pixel is considered to be foreground if its probability is lower than a threshold tr, vice versa:

$$F^t_{(x,y)} = \begin{cases} 1, & \text{if } p(I^t_{(x,y)}) < tr \\ 0, & \text{otherwise.} \end{cases} \tag{1.16}$$

Mittal and Poggio [161] have shown that robustness to background motion can be increased by using variable-bandwidth kernels. Zhang *et al.* [266] proposed the "PAWCS" method which uses word consensus models to separate the foreground and background. The importance of each background word is calculated based on its recurrence. A frame-level dictionary and a local feedback are also used in the model to improve the performance. Liao *et al.* [109] maintained a background pool for each pixel. The background pool is updated with a random strategy when the new frame comes.

Although non-parametric models are robust against small changes, they are expensive both computationally and in terms of memory use. Moreover, extending the support causes small foreground objects to disappear. As a consequence, several authors worked to improve the KDE model. For instance, a multi-level method [168] makes KDE computationally independent of the number of samples. A trend feature can also be used to reliably differentiate periodic background motion from illumination changes [258].

1.1.4 Data-driven Methods

Recently, pixel-based data-driven methods using random samples for background modeling have shown robustness to several types of error sources. For example, in ViBe [16, 221], the background at location (x, y) is modeled by a collection $M(x, y)$ which contains N previous pixel values v in the neighborhood S_r of the pixel (including the location of the pixel) with a radius r:

$$M(x, y) = \{v_1, v_2, ..., v_N\}. \tag{1.17}$$

The collection will be used to compare with the new pixel value at the same location. In the neighborhood of (x, y), if the number of the pixels that match with the values in the collection $M(x, y)$ is smaller than a threshold tr, the pixel will be considered to be foreground:

$$F_{(x,y)} = \begin{cases} 1, & \#S_r(I_{(x,y)}) \cap M(x,y) < tr \\ 0, & \text{otherwise.} \end{cases} \tag{1.18}$$

ViBe not only shows robustness to background motion and camera jitter but also to ghosting artifacts. Hofmann [93] improved the robustness of ViBe on a variety of difficult scenarios by automatically tuning its decision threshold and learning rate based on previous decisions made by the system. By combining local binary similarity patterns (LBSP) features with the ViBe random sampling strategy, [209], St-Charles *et al.* proposed "SuBSENSE", a model which is more robust to noise and illumination changes and can detect camouflaged foreground objects more easily. In [16, 93, 209], a pixel is declared as foreground if it is not close to a sufficient number of background samples from the past. A deterministic K nearest neighbor approach has also been proposed by Zivkovic and van der Heijiden [274].

A shortcoming of the above methods is that they do not account for any "temporal correlation" within video sequences, thus they are sensitive to periodic (or near-periodic) background motion. For example, alternating light signals at an intersection, a flashing advertisement board, the appearance of rotating objects, *etc*. A cyclostationary background generation method based on frequency decomposition that explicitly harnesses the scene dynamics is proposed in [184]. In order to capture the cyclostationary behavior at each pixel, spectral coefficients of temporal intensity profiles are computed in temporal windows and a background model that is composed of those coefficients is maintained and fused with distance maps to eliminate trail effects.

An alternative approach is to use a hidden Markov model (HMM) with discrete states to model the intensity variations of a pixel in an image sequence. State transitions can then be used to detect changes [212]. The advantage of using HMMs is that certain events, which may not be modeled correctly by unsupervised algorithms, can be learned using the provided training samples.

1.1.5 Matrix Decomposition

Instead of modeling the variation of individual pixels, the whole image can be vectorized and used in background modeling. In [169], a holistic approach using eigenspace decomposition is proposed. For a certain number of input frames, a background matrix (called eigenbackground) is formed by arranging the vectorized representations of images in a matrix where each vectorized image is a column. An eigenvalue decomposition via principal component analysis (PCA) is performed on the covariance of this matrix. The background is then represented by the most descriptive eigenvectors that encompass all possible illuminations to decrease sensitivity to illumination. More specifically, PCA first does singular value decomposition to find the eigenvalues and the eigenvectors of the video. The eigenvectors are ranked according to their correlated eigenvalues. Given N frames of a video with the mean μ_b the covariance matrix C_b of it, this part can be formalized as:

$$L = \Phi C_b \Phi^T, \tag{1.19}$$

where Φ is the eigenvector matrix of the covariance of the frames and L is the corresponding diagonal matrix of its eigenvalues.

The eigenvalues are sorted and the top M eigenvalues' correlating eigenvectors are selected to be the eigenbackground space Φ_M. When a new frame arrives, it will be first mean normalized as:

$$I^t_{normalized} = I^t - \mu_b, \tag{1.20}$$

and then projected to the eigenbackground space to calculate the background image:

$$B^t = \Phi_M I^t_{normalized}. \tag{1.21}$$

And the foreground will be the thresholding results of the subtraction between the input image and the background image:

$$F^t = \begin{cases} 1, & \text{if } |I^t - B^t| \geq tr \\ 0, & \text{otherwise.} \end{cases} \tag{1.22}$$

Several improvements of the PCA approach have been proposed. To name a few, Xu *et al.* [254] proposed a variation of the eigenbackground model which includes a recursive error compensation step for more accurate detection. Others [194, 134] proposed PCA methods with a computationally efficient background updating scheme, while Doug *et al.* [59] proposed an illumination invariant approach based on a multi-subspace PCA, each subspace representing different lighting conditions.

Based on PCA theory, robust principal component analysis (RPCA) [39] has also been proposed. To separate the foreground from the background, RPCA tries to separate the data matrix into two components, namely a low-rank component for the background, and a sparse component for the foreground. Tepper *et al.* [215] used an online RPCA framework to increase the RPCA optimization speed.

Instead of the conventional background and foreground definition, Porikli [182] decomposes an image into "intrinsic" background and foreground images. The multiplication of these images reconstructs the given image. Inspired by the sparseness of the intensity gradient, it applies a spatial derivative filter in the log domain to a subset of the previous video frames to obtain the intensity gradient. Since the foreground gradients of natural images are Laplacian distributed and independent, the maximum likelihood (ML) estimate of the background gradient can be obtained by a median operator and the corresponding foreground gradient is computed. The computed gradient is used to reconstruct the background and foreground intensity images using a reconstruction filter and inverse log operator. This intrinsic decomposition is shown to be robust against sudden and severe illumination changes, but it is computationally expensive.

Another background subtraction approach based on the theory of sparse representation and dictionary learning is proposed by Zhao *et al.* [267]. This method makes the following two important assumptions: (1) the background of a scene has a sparse linear representation over a learned dictionary; (2) the foreground is sparse in the sense that a majority of the pixels of the frame belong to the background. These two assumptions enable handling both sudden and gradual background changes.

1.1.6 Other Methods

Prediction Models

Some methods use filters to predict background pixel intensities (or colors). Given the last N values of a pixel and their correlating prediction coefficients w, the prediction models try to predict what the pixel value would be at time t:

$$I^t_{predict} = -\sum_{i=1}^{N} w^i I^{t-i}. \tag{1.23}$$

If the distance between the predict pixel value and the real pixel value is larger than a threshold, the pixel will be classified to be a foreground:

$$F^t = \begin{cases} 1, & \text{if } |I^t - I^t_{predict}| \geq tr \\ 0, & \text{otherwise.} \end{cases} \tag{1.24}$$

In [112] and [270], a Kalman filter is used to model background dynamics. In [257], Kalman filter is used to smooth the trajectories of moving objects detected by three frames difference, the trajectories are then used to calculate the RPCA weights. Similarly, in [219] Wiener filtering is used to make a linear prediction at pixel level. The main advantage of these methods is their ability to cope with background changes (whether it is periodic or not) without having to assume any parametric distribution.

Motion Segmentation

Motion segmentation refers to the assignment of groups of pixels to various classes based on the speed and direction of their movements [147]. Most approaches to motion segmentation first seek to compute optical flow from an image sequence. Discontinuities in the optical flow can help in segmenting images into regions that correspond to different objects. In [244], temporal consistency of optical flow over a narrow time window is estimated; areas with temporally-consistent optical flow are deemed to represent moving objects and those exhibiting temporal randomness are assigned to the background. Before using optical flow to detect foreground, Hu *et al.* [96] use Harris corner detector and epipolar geometry to align the key points in each continues frames. In that case, their method can even detect motions in a video captured by moving cameras.

Optical flow-based motion detection methods work with strict assumptions such as the brightness of the video is constant, the velocity of the moving object is smooth, and the frame rate of the video is high enough to extract the optical flow. Optical flow-based methods will be erroneous if any of these assumptions is violated. In reality, such violations are quite common. Typically, optical flow methods fail in low-texture areas, around moving object boundaries, at depth discontinuities, *etc*. Due to the commonly imposed regularization term, most optical flow methods produce an over smooth optical flow near boundaries. Although solutions involving a discontinuity preserving optical flow function and object-based segmentation have been proposed [158], motion segmentation methods usually produce a halo artifact around moving objects. The resulting errors may propagate across the entire optical flow solution. As a solution, some authors [101, 271] use motion segmentation and optical flow in combination with a color-based GMM model. At the same time, optical flow with Canny edge detection is also proposed to improve the robustness [203].

Machine Learning

Motion detection methods in this category use machine learning discriminative tools such as support vector machine (SVM) and neural networks to decide whether or not a pixel is in motion. The parameters of these functions are learned given a training video. Lin *et al.* [138] use a probabilistic SVM to initialize the background model. They use the magnitude of optical flow and inter-frame image difference as features for classification. Han and Davis [89] model the background with kernel density approximation with multiple features (RGB, gradient, and Haar) and use a Kernel-SVM as a discriminative function. A somewhat similar approach has also been proposed by Hao [91]. These approaches are typical machine learning methods that need positive and negative examples for training. This is a major limitation for any practical implementation since very few videos come with manually labeled data. As a solution, Chen *et al.* [47] proposed a GPU-based 1-class SVM method called SILK. This method does not need pre-labeled training data, but also allows for online updating of the SVM parameters.

SOBS [152, 154] models the background of a video with the weights of a neural network. A very similar approach but with a post-processing Markov random field (MRF) stage has been proposed by Schick *et al.* [196]. Results reported in the paper show great compromise between processing speed and robustness to noise and background motion.

Deep Learning Methods

Recently, deep learning methods have achieved excellent results in many computer vision fields, including motion detection. The biggest advantage of deep learning methods is that instead of using handcrafted features, the model can learn both high level features and low level features directly from the data. Convolutional neural networks (ConvNets or CNN) [123] were first proposed for object classification, but are now used for motion detection tasks. Braham *et al.* [35] designed a CNN model similar to LeNet-5 [129] network. The network is composed of two feature stages and two fully connected layers. Wang *et al.* [239] proposed a multi-scale CNN model with cascade structure to learn the appearance of the video background and the foreground moving objects with a small amount human interaction. This paper is the topic of Chapter 3.

1.1.7 Features

Several features can be used to detect moving objects. The simplest one is certainly grayscale (or luminance) which is easy to interpret and has a well founded physical meaning [74]. Grayscale motion detection methods are normally used on mono-channel cameras like depth cameras, thermal cameras, or older grayscale surveillance cameras.

Nowadays, most motion detection methods rely on color. A color image consists of three channels per pixel (typically red (R), green (G), blue (B)) that can be processed separately or simultaneously. However, the physical meaning of these channels is less obvious than the one for mono-channel sensors. Ideally, color images are acquired using three spatially aligned sensors. But since this configuration increases the size and cost of the sensor and requires pixel registration, most color cameras use a single image sensor with a color filter array in front of it. The most widely implemented array is the Bayer color filter array [18]. Each location on the sensor measures one color and missing colors are interpolated from neighboring pixels. Suhr [213] proposes a GMM variant that conducts background modeling in a Bayer-pattern domain and foreground classification in an interpolated RGB domain. The authors argue that since performance is similar to that of the original GMM on RGB images, RGB video streams captured with one sensor are not three times more informative than their grayscale counterpart.

In practice though, most techniques exhibit a small performance increase for the classification task when using RGB instead of grayscale features [21]. Thus, from a classification perspective and despite that the computation time is more or less tripled, it is beneficial to use color images, even when colors have been interpolated in the image. In their survey paper, Benezeth *et al.* [21] compare six RGB color distance functions used for background subtraction, including the Euclidean distance, the L1 distance, and the Mahalanobis distance. They conclude that four of the six metrics had globally similar classification performances; only the simplest zero and first order distances were less precise.

Several motion detection techniques use other color spaces such as normalized color [161], cylindric color model [115], HSV [53], HSI [234], YCbCr [121], and normalized RGB [59]. From an application perspective, those color spaces are believed to be more robust to shadows and illuminations changes than RGB or grayscale [53].

RGB color space can be transferred to other color spaces:

1. RGB to normalized RGB:

$$\begin{aligned} R_{normalized} &= \tfrac{R}{R+G+B} \\ G_{normalized} &= \tfrac{G}{R+G+B} \\ B_{normalized} &= \tfrac{B}{R+G+B} \end{aligned} \tag{1.25}$$

2. RGB to HSV:

$$\begin{aligned} Min &= \min(R, G, B) \\ Max &= \max(R, G, B) \end{aligned} \tag{1.26}$$

$$H = \begin{cases} 0, & \text{if } Min = Max \\ 60 \times \frac{G-B}{Max-Min} + 0, & \text{if } Max = R \text{ and } G \geq B \\ 60 \times \frac{G-B}{Max-Min} + 360, & \text{if } Max = R \text{ and } G < B \\ 60 \times \frac{B-R}{Max-Min} + 120, & \text{if } Max = G \\ 60 \times \frac{R-G}{Max-Min} + 240, & \text{if } Max = B \end{cases}$$

$$S = \begin{cases} 0, & \text{if } Max = 0 \\ 1 - \frac{Min}{Max}, & \text{otherwise} \end{cases}$$

$$V = Max \tag{1.27}$$

3. RGB to HSL:

$$H = \begin{cases} 0, & \text{if } Min = Max \\ 60 \times \frac{G-B}{Max-Min} + 0, & \text{if } Max = R \text{ and } G \geq B \\ 60 \times \frac{G-B}{Max-Min} + 360, & \text{if } Max = R \text{ and } G < B \\ 60 \times \frac{B-R}{Max-Min} + 120, & \text{if } Max = G \\ 60 \times \frac{R-G}{Max-Min} + 240, & \text{if } Max = B \end{cases}$$

$$S = \begin{cases} 0, & \text{if } L = 0 \text{ or } Min = Max \\ \frac{Max-Min}{2L}, & \text{if } 0 < L < 0.5 \\ 1 - \frac{Max-Min}{2-2L}, & \text{if } L > 0.5 \end{cases}$$

$$L = \frac{1}{2}(Max + Min) \tag{1.28}$$

4. RGB to YCbCr:

$$\begin{aligned} Y &= 0.299 \times R + 0.587 \times G + 0.114 \times B \\ Cb &= 0.169 \times R + 0.331 \times G + 0.5 \times B \\ Cr &= 0.5 \times R + 0.419 \times G + 0.081 \times B \end{aligned} \tag{1.29}$$

Other features, like edges [102], texture [136], and optical flow [138, 161, 216], PCA-based features [169] are also used. Like the color space features, these features seem more robust to illumination changes and shadows than RGB features. Texture and optical flow features are also robust to noise and background motion. Since texture features integrate spatial information which often happens to be constant, a slight variation of in the background does not lead to spurious false positives. For example, a bush with a uniform texture will be undetected when shaken by the wind. As for optical flow, since moving objects are assumed to have a smooth and coherent motion distribution [216], noise and random background motion can be easily decorrelated from actual moving objects.

In general, it seems like adding features improves performances. Parag *et al.* [175] even propose to select the best combination of features at each pixel. They argue that different parts of the image may have different statistics and thus require different features. But this comes at the price of both a complexity and a computation time increase.

1.1.8 Updating Strategies

In order to produce consistent results over time, background models need to be updated as the video streams in. From a model point of view, there are two major updating techniques [178]: the *recursive* and *non-recursive* techniques. The recursive techniques maintain a single background model that is updated with each new video frame:

$$B^t = (1 - \beta)B^{t-1} + \beta I^t, \tag{1.30}$$

where β is the background updating ratio.

Non-recursive techniques, on the other hand, maintain a buffer L of n previous video frames and estimate a background model based solely on the statistical properties of these frames. This includes median filtering and eigenbackgrounds [169]. The major limitation of non-recursive approaches is that computing the basis functions requires video clips void of foreground objects. As such, it is not clear how the basis functions can be updated over time if foreground objects are continuously present in the scene.

As mentioned by Elgammal *et al.* [63], other updating strategies use the output of the segmentation process. The conditional approach (also called selective or conservative) updates only background pixels in order to prevent the background model from being corrupted by foreground pixels.

$$B^t_{(x,y)} = \begin{cases} (1-\beta)B^{t-1}_{(x,y)} + \beta I^t_{(x,y)}, & \text{if } I^t_{(x,y)} \text{ is foreground} \\ B^{t-1}_{(x,y)}, & \text{if } I^t_{(x,y)} \text{ is background.} \end{cases} \tag{1.31}$$

However, this approach is incapable of eliminating false positives as the background model will never adapt to it. Wang *et al.* [228] propose to operate at the blob level and define a mechanism to incorporate pixels in the background after a given period of time. As an alternative, the unconditional (or blind) approach updates every pixel whether it is identified as being active or not. This approach has the advantage of integrating new objects in the background and compensating for false detections caused, say, by global illumination changes or camera jitter. On the other hand, it can allow slowly moving objects to corrupt the background which leads to spurious false detections. Both conditional and unconditional techniques can be used, depending on the appropriateness to the model or on the requirements of the application.

Some authors introduce more nuances. For example, Porikli *et al.* [183] define a GMM method and a Bayesian updating mechanism, to achieve accurate adaptation of the models. A somewhat similar refinement method is proposed by Van Droogenbroeck *et al.* [221]. Both [183] and [221] distinguish between a segmentation mask, the binary output image which corresponds to the background/foreground classification result, and the updating mask. The updating mask corresponds to locations indicating which pixels have to be updated. The updating mask differs from the segmentation map in that it remains unknown to the user and depends on updating strategies. For example, one can decide not to update the model inside of static blobs or, on the contrary, decide to erode foreground mask to progressively remove ghosts. Another recent updating strategy consists in spatial diffusion; it was introduced with ViBe [16]. Spatial diffusion is a mechanism wherein a background value is diffused in a neighboring model. This diffusion mechanism can be modulated to help remove ghosts or static objects.

1.1.9 Spatial Aggregation, Markovian Models and Post-processing

Most motion detection techniques are local processes that focus on pixel-wise statistics ignoring neighboring pixels (at least during the modeling phase). This is a well-founded approach from a statistical point of view since neighboring pixels might have very different underlying feature probability density functions. Nevertheless, there exist techniques that aggregate information from neighboring pixels into regular blocks or so-called superpixels. Block-based aggregation is a coherent approach for video encoder, as blocks and macroblocks are the fundamental spatial units in encoders.

Grouping pixels into blocks is motivated by several factors. First, statistics averaged over a rectangular region increases the robustness to non-stationary backgrounds, despite the fact that it blurs the object silhouette and that a post-processing method might be needed to refine edges as in [46]. Second, if sharp edges are not mandatory, processing blocks speeds up the motion detection process. Hierarchical methods, as proposed by Park *et al.* [177] or Chen *et al.* [43], are typical examples of methods that play with different levels of pixel aggregation.

Pixels aggregation can also be achieved with the help of a Markovian model. Typical Markovian models are based on a maximum a posteriori formulation that is solved through an optimization algorithm such as iterative optimization scheme (ICM) or graph cut [2, 157] which are typically slow. In [22], it was shown that simple Markovian methods (typically those using the Ising prior) produce similar results as simple post-processing filters.

Other Markovian methods have been proposed. In [98], Markov random fields are used to re-label pixels. First, a region-based motion segmentation algorithm is developed to obtain a set coherent regions. This serves to define the statistics of several Markovian random fields. The final labeling is obtained by maximizing the a posteriori energy of the Markov random fields, which can be seen as a post-processing step. The approach by Schick *et al.* [196] relies on similar ideas. A first segmentation is used to define a probabilistic superpixel representation. Then a post-processing is applied on the statistical framework to provide an enhanced segmentation map. It is interesting to note that Schick *et al.* [196] have successfully applied their post-processing technique to several motion detection techniques.

A more classical, simpler and faster way to re-label pixels is throughout a post-processing filter. For example, Parks and Fels [178] consider a number of post-processing techniques to improve the segmentation map. Their results indicate that the performance is improved by morphological filters (closings), noise removal filter (such as median filters), and area filters. Morphological filters are used to fill internal holes and small gaps, while area filters are used to remove small objects.

In Section 2.1, we present the results of some post-processing operations. It appears that simple post-processing operations, such as the median or close/open morphological operations always improve the segmentation map. It is thus recommended to include post-processing operations, even when comparing techniques. This was also the conclusion of Brutzer *et al.* [37] and Benezeth *et al.* [20]. Note that other filters can be used such as temporal filters, shadow filters [37], and complex spatio-temporal filtering techniques to relabel the classification results.

1.2 Previous Motion Detection Datasets

Without aiming to be exhaustive, we list below 15 of the most widely used datasets for motion detection validation (see Table 1.2). Additional details regarding some of these datasets can be found on a web page of the European CANTATA project [1]. A dataset should always come with ground truth. For a motion detection dataset, the ground truth is provided by the dataset publisher to indicate where are the moving objects in the video. A ground truth can be accurate binary map which indicate the label of each pixel in a frame (*i.e.* foreground or background). It can also be bounding boxes approximately indicate the locations of the moving objects.

Table 1.2 – 15 of the Most Popular Motion Detection Datasets

Dataset	**Description**	**Ground truth**
CDnet 2012	31 videos in six categories: baseline, dynamic background, camera jitter, shadow, intermittent motion, and thermal.	Pixel-based labeling of 71,000 frames.
Wallflower [219]	Seven short video clips, each representing a specific challenge such as illumination change, background motion, *etc.*	Pixel-based labeling of one frame per video.
PETS [261]	Many videos aimed at evaluating the performance of tracking algorithms.	Bounding boxes.

[1] www.hitech-projects.com/euprojects/cantata/datasets_cantata/

CAVIAR	80 staged indoor videos representing different human behaviors such as walking, browsing, shopping, fighting, *etc*.	Bounding boxes.
i-LIDS	Very long videos meant for action recognition showing parked vehicle, abandoned object, people walking in a restricted area, and doorway.	Not fully labeled.
ETISEO	More than 80 videos meant to evaluate tracking and event detection methods.	High-level label such as bounding boxes, object class, event type, *etc*.
ViSOR 2009 [224]	Web archive with more than 500 short videos (usually less than 10 seconds).	Bounding boxes.
BEHAVE 2007	Seven videos shot by the same camera showing human interactions such as walking in group, meeting, splitting, *etc*.	Bounding boxes.
VSSN 2006	Nine semi-synthetic videos composed of a real background and artificially-moving objects. The videos contain animated background, illumination changes, and shadows. However the videos do not contain any frames void of activity.	Pixel-based labeling of each frame.
IBM	15 videos taken from PETS 2001 plus additional videos.	Bounding box around each moving object in one frame out of 30.
Karlsruhe	Four grayscale videos showing traffic scenes under various weather conditions.	10 frames per video have pixel-based labeling.
Li *et al.* [131]	10 small videos (usually 160×120) containing illumination changes and dynamic backgrounds.	10 frames per video have pixel-based labeling.
Karaman *et al.* [111]	Five videos coming from different sources (the web, the "art live" project, *etc*.) with various illumination conditions and compression artifacts.	Pixel-based labeling of every frame.
cVSG 2008 [217]	15 Semi-synthetic videos with various levels of textural complexity, background motion, moving object speed, size, and interaction.	Pixel-based labeling obtained by filming moving objects (mostly humans) in front of a blue-screen and then pasted on top of background videos.

Brutzer *et al.* [37]	Computer-generated videos showing one 3D scene representing a street corner. The sequences include illumination changes, dynamic background, shadows, and noise.	Pixel-based labeling.

Out of these 15 datasets, seven were initially made to validate tracking and pattern recognition methods (namely PETS, CAVIAR, i-LIDS, ETISEO, ViSOR 2009, BEHAVE 2007, and IBM). Although challenging for these applications, those datasets mostly contain day-time videos with fixed background, constant illumination, few shadows, and no camera jitter. As a consequence, it is difficult to evaluate how robust motion detection methods are when looking at benchmarking results reported on these seven datasets.

In parallel of these datasets, a number of survey papers have been written on the topic of motion detection. In Table 1.3, we list survey papers devoted to the comparison and ranking of motion detection algorithms. Note that some of these surveys use datasets mentioned previously while others use their own datasets.

Table 1.3 – List of the most important motion detection survey papers.

Survey	**Description and Benchmark**
Bouwmans *et al.* 2016 [34]	Survey of matrix decomposition methods, the paper carefully compares PCA family foreground detection methods. Performances are compared on the CDnet 2014 dataset.
Bouwmans *et al.*, 2016 [32]	Survey of features used in background modeling and motion detection.
Goyette *et al.*, 2012 [79]	Survey paper written in the wake of the CVPR 2012 Change Detection workshop. It surveys several methods and reports benchmark results obtained on the CDnet 2012 dataset.
Bouwmans *et al.*, 2011 [29]	Probably the most complete surveys to date with more than 350 references. The paper reviewed methods spanning six motion detection categories and the features used by each method. The survey also listed a number of typical challenges and gave insights into memory requirements and computational complexity. Benchmark on the Wallflower dataset.
Brutzer *et al.*, 2011 [37]	Report benchmarking results for eight motion detection methods on the computer generated Brutzer dataset.

Benezeth *et al.*, 2010 [20]	Used a collection of 29 videos (15 camera-captured, 10 semi-synthetic, and four synthetic) taken from PETS 2001, the IBM dataset, and the VSSN 2006 dataset.
Bouwmans *et al.*, 2008 [31]	Survey of GMM methods. Benchmarking had been done on the Wallflower dataset.
Parks and Fels, 2008 [178]	Benchmark results for seven motion detection methods and evaluation of the influence of post-processing on their performance. They used seven outdoor and six indoor videos different challenges such as dynamic backgrounds, shadows, and various lighting conditions.
Bashir and Porikli, 2006 [17]	Performance evaluation of tracking algorithms using the PETS 2001 dataset by comparing the detected bounding box locations with the ground truth.
Nascimento and Marques, 2006 [167]	Report benchmarks obtained on a single PETS 2001 video with pixel-based labeling.
Radke *et al.* [188]	Extensive survey of several motion detection methods. Most of the discussion in the paper was related to background subtraction methods, pre- and post-processing, and methodologies to evaluate performances. Contains no quantitative evaluation.
Piccardi [181]	Reviewed seven background subtraction methods and highlighted their strengths and weaknesses. Contains no quantitative evaluation.
Rosin and Ioannidis, 2003 [192]	Report results for eight methods. Videos used for validation show two lab scenes with balls rolling on the floor.
Prati *et al.*, 2001 [185]	Used indoor sequences containing one moving person. 112 frames were labeled.

These survey papers often contain good overviews of state-of-the-art motion detection methods. However, readers shall keep in mind that statistics reported in some of these papers were not computed on a well-balanced dataset composed of real (camera-captured) videos [192, 167]. Typically, synthetic videos, real videos with synthetic moving objects pasted in it, or real videos out of which only one frame was manually segmented for ground truth were used. Also, some survey papers report results from fairly simple and old motion detection methods [185, 192].

Chapter 2

Motion Detection Benchmarking on the CDnet 2012 Dataset and the Extended CDnet 2014 Dataset

2.1 Benchmarking Experiments

In Chapter 1, we introduced nine families of motion detection methods, presented different features, several updating schemes, and many spatial aggregation and post-processing methods. To provide empirical results to validate which configuration performs best, we designed a series of experiments to test their performances. Note that since the number of combinations of motion detection methods, features, updating schemes, and post-processing methods is intractable, we provide benchmarks for each aspect independently.

The goal of this section is to underline unsolved issues in motion detection and identify complementary methods whose combination can further improve results. Empirical results are obtained with the CDnet 2012 dataset. As mentioned previously, this dataset includes 31 videos divided in six categories namely dynamic background, camera jitter, shadow, intermittent motion, baseline, and thermal.

But prior to presenting benchmarking results, we first describe and explain the evaluation metrics used in this section.

2.1.1 Metric Evaluation

As stated by Goyette *et al.* [79], it is not a trivial task to find the right metric to accurately measure the ability of a method to detect motion. If we consider background subtraction as a classification process, we can recover the following four quantities for a processed video: the number of true positives (*TP*) and false positives (*FP*), which accounts for the number of foreground pixels correctly and incorrectly classified, and the number of true negatives (*TN*) and false negatives (*FN*), which are similar measures but for background pixels. With these values, one can come out with the following seven metrics, which are often used to rank background subtraction methods.

1. The *True Positive Rate* (*TPR*), also named *sensitivity* and *recall*:

$$Re = \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.1}$$

2. The *False Negative Rate* (*FNR*):

$$\text{FNR} = 1 - \text{TPR} \tag{2.2}$$

3. The *True Negative Rate* (*TNR*), also named *specificity*:

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.3}$$

4. The *False Positive Rate* (*FPR*):

$$\text{FPR} = 1 - \text{TNR} \tag{2.4}$$

5. The *precision*:

$$Pr = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.5}$$

6. The *Probability of Wrong Classification* (also named *Error Rate*):

$$\mathrm{PWC} = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{2.6}$$

7. The *Accuracy*:

$$\mathrm{A} = 1 - \mathrm{PWC} \tag{2.7}$$

In the upcoming subsections, we will try to answer the question of which metric(s) should be used to rank methods.

Limitation of Metrics Combining TPR and TNR

For obvious reasons, *TPR*, *TNR*, *FPR*, and *FNR* cannot be used alone to rank methods. In fact, methods are typically adjusted to prevent *FPR* and *FNR* from being large simultaneously. Such trade-offs can be interpreted by showing a Receiver Operating Characteristic (ROC) graph. But ranking methods based on ROC curves is rather inconvenient due to the large number of results that need to be generated and which can be prohibitive in the context of large videos. Therefore, most often, only a single point is known in the ROC space. Summarizing *TPR* and *TNR* into a single value remains difficult and this is highlighted by the following discussion.

Since most surveillance videos exhibit a low amount of activity (5% on average for the CDnet 2012 video sequences), the *TNR* value will always dominate *A* and *PWC*. Actually, as one can see in Table 2.1, nearly all methods have a very low *FPR* (except for [105]) and a large *FNR*. As a consequence, when used alone, the accuracy *A* and the probability of wrong classification *PWC* will always favor methods with a low *FPR* and a large *FNR*. At the limit, a method that would detect no moving object at all would have a not-so-bad ranking score according to *A* and *PWC* alone. That is because only a small fraction of the pixels would be wrongly classified on average. Another way of underlying the limitation of *A* and *PWC* is by rewriting the accuracy equation. If we denote the probabilities of observing a foreground pixel and a background pixel by p_{FG} and p_{BG} respectively, then one can show that the accuracy can be computed as:

$$\mathrm{A} = p_{\mathrm{FG}}\mathrm{TPR} + p_{\mathrm{BG}}\mathrm{TNR}. \tag{2.8}$$

Table 2.1 – Overall results for 22 methods. These results correspond to the average FPR, FNR and F-measure obtained on all 31 videos of the CDnet 2012 dataset.

Method	Description	FPR	FNR	F-measure
Spectral-360 [197]	Patent	0.008	0.22	0.77
DPGMM [87]	GMM + Dirichlet Process	0.014	0.17	0.77
SGMM-SOD [66]	Improved version of SGMM [65]	0.006	0.23	0.76
PBAS [93]	Data-driven and stochastic method	0.010	0.21	0.75
PSP-MRF [196]	Probabilistic super-pixels + neural maps	0.017	0.19	0.73
SC-SOBS [154]	Improved version of SOBS [152]	0.016	0.19	0.72
SOBS [152]	Neural maps	0.018	0.21	0.71
SGMM [65]	GMM + new initialization, updating and splitting rule	0.009	0.29	0.70
Chebyshev Inequality [162]	Multistage method + Chebyshev inequality + tracking	0.011	0.28	0.70
KNN [274]	Data-driven KNN	0.009	0.32	0.67
KDE|Elgammal [62]	Original KDE	0.024	0.25	0.67
GMM|Stauffer-Grimson [210]	Original GMM	0.014	0.28	0.66
GMM|Zivkovic [274]	GMM + automatic mode selection	0.015	0.30	0.65
KDE|Yoshinaga *et al.* [259]	Spatio-temporal KDE	0.009	0.34	0.64
KDE|Nonaka *et al.* [168]	Multi-level KDE	0.006	0.34	0.64
Bayesian Multi layer [183]	Bayesian layers + EM	0.017	0.39	0.62
Mahalanobis distance [22]	Basic background subtraction	0.040	0.23	0.62
Euclidean distance [22]	Basic background subtraction	0.030	0.29	0.61
GMM|KaewTraKulPong [110]	Self-adapting GMM	0.005	0.49	0.59
Histogram over time [269]	Basic method + color histograms	0.065	0.23	0.54
GMM|RECTGAUSS-Tex [191]	Multi-resolution GMM	0.013	0.48	0.52
Local-Self similarity [105]	Basic method + self-similarity measure	0.148	0.06	0.50

Thus, with a low p_{FG}, the *TNR* ends up having an overwhelming importance when computing *A*. As an alternative, one could consider the *Balanced Accuracy*:

$$\text{BA} = \frac{1}{2}\text{TPR} + \frac{1}{2}\text{TNR}. \tag{2.9}$$

However, since that metric is uncommon in the motion detection community, we decided not to use it.

So in conclusion, although the accuracy A and the probability of wrong classification PWC can be used to evaluate methods, they should not be used alone and one should keep in mind that these two metrics favor methods with a low *FPR*.

Metrics Derived from *Pr* and *Re*

Another trade-off for motion detection methods is to prevent *Pr* and *Re* from being large simultaneously, which is shown on a precision-recall curve.

But using precision-recall curves to rank methods is inconvenient for the same reasons as for ROC curves. In practice, precision and recall must be combined into one metric. The most frequent way of doing so is through the F-measure F_1, which is the harmonic mean between *Pr* and *Re*:

$$F_1 = \frac{1}{2}Pr^{-1} + \frac{1}{2}Re^{-1} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}. \tag{2.10}$$

When both *Pr* and *Re* are large, F_1 is approximately equal to the arithmetic mean of Pr and Re. Otherwise, it is approximately equal to min(Pr, Re).

The balanced accuracy and the F-measure, although similar at the first glance, are not equivalent. Let us consider the case shown in Figure 2.1 for which it is difficult to identify the human silhouette based on the first two results. In that example, all three results have the same balanced accuracy but a much higher F-measure for method 3. This is a strong indication that the F-measure is a better metric than the balanced accuracy in the context of motion detection and thus why we use it in our validation. Another reason for F_1 to be larger for method 3 is the fact that it does not take into account *TN*. As a consequence, F_1 is a metric that focuses more on the foreground than on the background which is good in the context of motion detection.

Influence of Noise

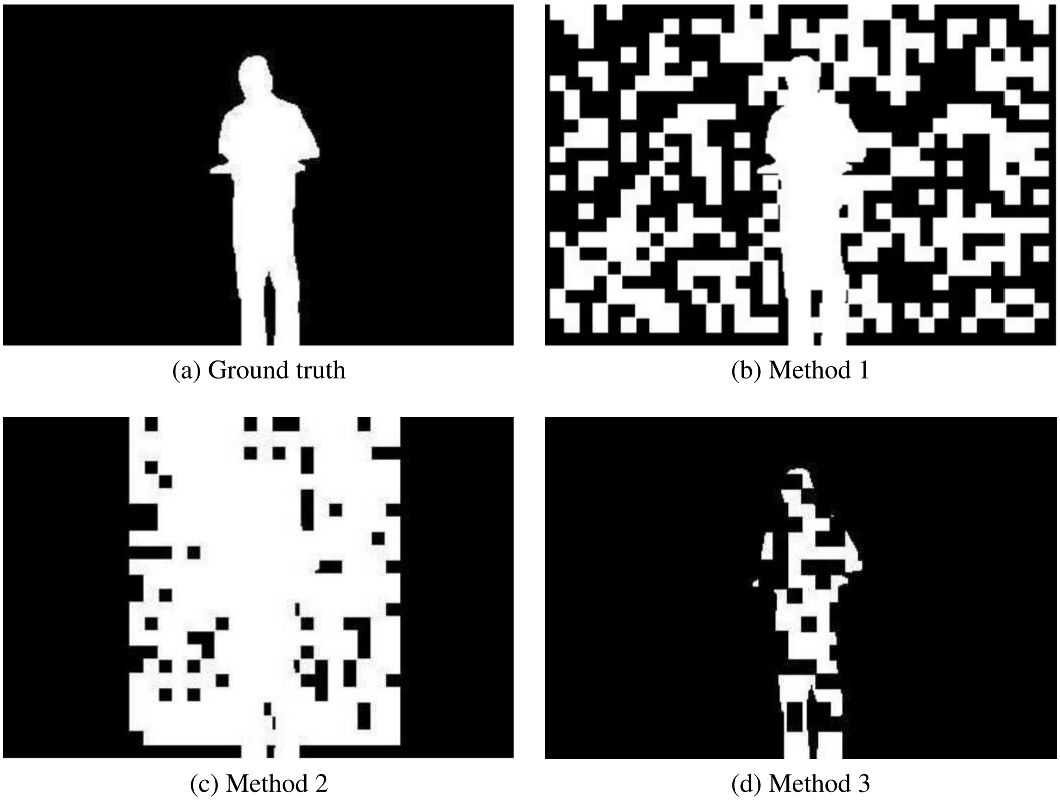
The F-measure is not void of limitations. As will be shown in this section, it is sensitive to noise and thus should be used with care. In order to illustrate the impact of noise and the importance of post-filtering operations, let us add a "salt and pepper" noise to a segmentation map. Let α be the probability to switch the class of a pixel while TPR$'$ and TNR$'$ the estimates on the noisy segmentation maps. In that case, we have

$$\text{TP}' = (1-\alpha)\text{TP} + \alpha\text{FN}, \tag{2.11}$$

$$\text{FN}' = (1-\alpha)\text{FN} + \alpha\text{TP}, \tag{2.12}$$

$$\text{TN}' = (1-\alpha)\text{TN} + \alpha\text{FP}, \tag{2.13}$$

$$\text{FP}' = (1-\alpha)\text{FP} + \alpha\text{TN}. \tag{2.14}$$


(a) Ground truth

(b) Method 1

(c) Method 2

(d) Method 3

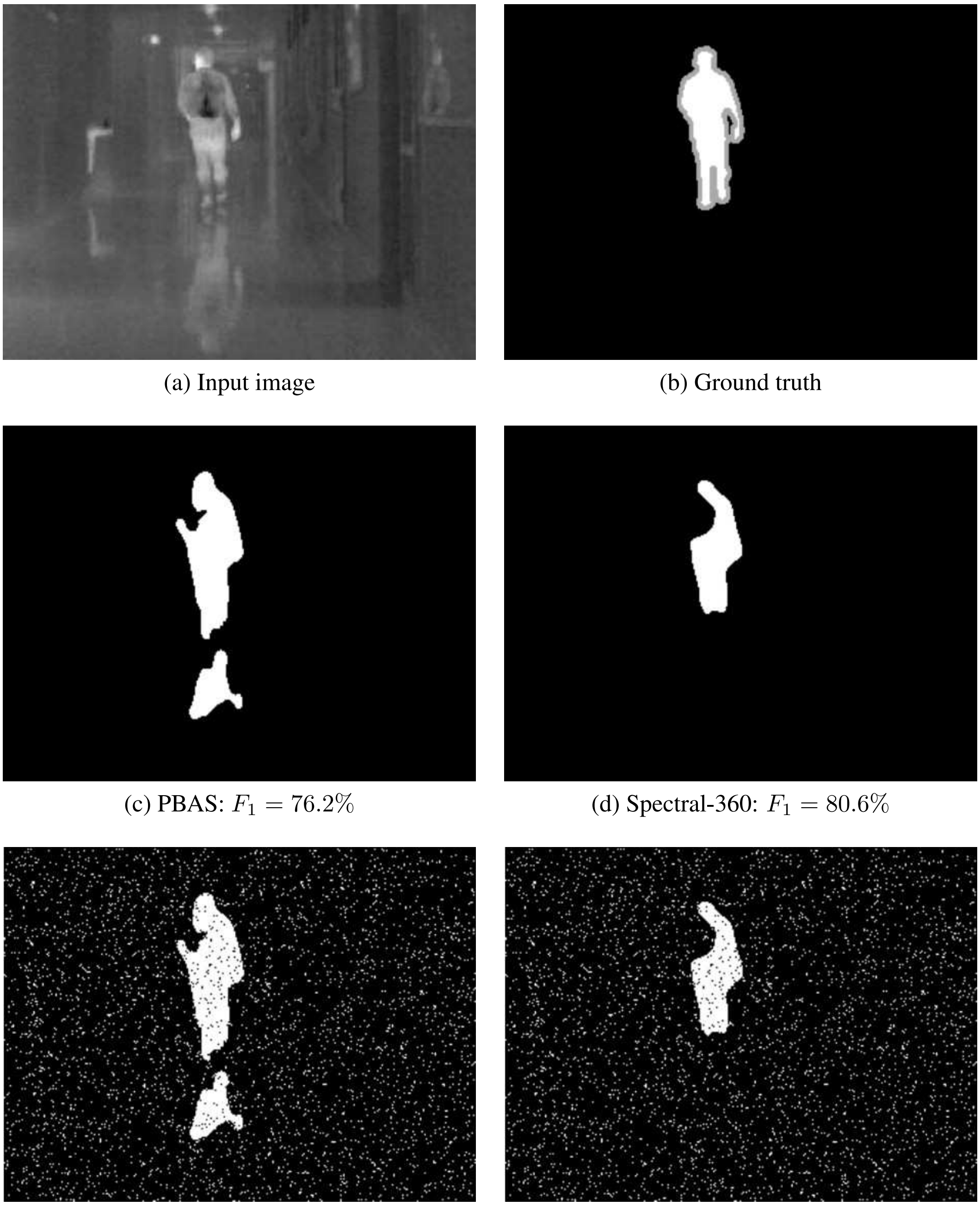
Figure 2.1 – Three methods with the same balanced accuracy (0.8) but with different F-measures. For methods 1 and 2, F_1 = 0.35 while for method 3 F_1 = 0.73.

Following some algebric manipulations, one can show that the relative ranking between two methods can change depending on the amount of noise in the data. This is illustrated in Figure 2.2 where F_1 for Spectral-360 goes below PBAS after noise has been added.

This sensitivity to noise leads us to conclude that it is preferable to filter noise with a post-processing filter before ranking background subtraction techniques according to F_1. This is what has been done for every method reported in Section 2.1.2.

Evaluation and Ranking of Methods

The previous discussion made it clear that summarizing the performance of a background subtraction algorithm with a single metric is restrictive. Several metrics like *FNR* and *FPR* are complementary and cannot be used independently whereas others



(a) Input image

(b) Ground truth

(c) PBAS: $F_1 = 76.2\%$

(d) Spectral-360: $F_1 = 80.6\%$

(e) PBAS + noise: $F_1 = 41.6\%$

(f) Spectral-360 + noise: $F_1 = 37.6\%$

Figure 2.2 – Ranking of the methods obtained according to F-measure is sensitive to noise. It is therefore important to filter out noise from the results before ranking methods with F_1.

like *PWC* and *A* give an overwhelming importance to *TNR*. As for the F-measure, although widely used, it is sensitive to noise. This leads us to conclude that no metric is perfect and should thus be used with care.

The last question that we ought to answer before presenting benchmarking results, is how to compute evaluation metrics when considering more than one video sequence. Naively, one could add up the total number of *TP*, *TN*, *FP* and *FN* across all videos out of which metrics could be computed. But unfortunately, since videos have different sizes in space and time, large videos would end up having more influence on the ranking than the smaller ones. As explained by Goyette *et al.* [79], a better solution is to compute the metrics for each video (that is *Re*, *FNR*, *FPR*, *Specificity*, *Pr*, *PWC*, *A*, and F_1) and then average it across videos. The CDnet 2012 dataset also has a multi-criteria ranking which we do not retained in this chapter for the sake of simplicity.

In this chapter, we rank methods according to the average F_1 computed across all videos and categories of the CDnet 2012 dataset. Although sensitive to noise, the result of every method has been post-processed with a median filter to prevent the previously-mentioned ranking problems. Also, Goyette *et al.* [79] mentioned that the F_1 score is correlated with this multi-criteria ranking which is a good indication that F_1 is a well balance metric.

And last, let us mentioned that the benchmarking results presented in Section 2.1.2 do not entirely capture the pros and cons of each method. Obviously, the complexity of an algorithm together with its processing speed and memory usage are to be considered for real-time applications.

2.1.2 Benchmarks on the CDnet 2012 Dataset

Motion Detection Methods

In this section, we report the results of motion detection benchmarking experiment based on the CDnet 2012 dataset. The results are an extension of those reported by Goyette [79]. Compare with it, this benchmarking not only contains more motion detection methods, but also tested the features, updating strategies, over-processing methods, and motion detection results merging technologies which are not mentioned in [79].

From the CDnet 2012 dataset, we retained results from 22 motion detection methods. Five methods are relatively simple as they rely on plain background subtraction, of which two use color features (Euclidean and Mahalanobis distance methods described in [126, 22]), one uses RGB histograms over time [269], and one uses local self-similarity features [105].

We also report results for eight parametric methods, seven of which use a GMM model. This includes the well-known methods by Stauffer and Grimson [210], a self-adapting GMM by KaewTraKulPong [110], the improved GMM method by Zivkovic and Heijden [274], the multiresolution block-based GMM (RECTGAUSS-Tex) by Dora *et al.* [191], GMM method with a Dirichlet process (DPGMM) that automatically estimated the number of Gaussian modes [87] and the SGMM and SGMM-SOD methods by Evangelio *et al.* [65, 66] which rely on a new initialization procedure and novel mode splitting rule. We also included a recursive per-pixel Bayesian approach by Porikli and Tuzel [183] which shows good robustness to shadows according to [79]. We also report results on three KDE methods. The original method by Elgammal *et al.* [62], a multilevel KDE by Nonaka *et al.* [168], and a spatio-temporal KDE by Yoshinaga *et al.* [259]. Results for data-driven methods and machine learning methods are also reported. That is Hofmann's stochastic and self-adaptive method (PBAS) [93], a simple K-nearest neighbor method [274] and neural maps methods (SOBS and SC-SOBS) by Maddalena *et al.* [152, 154] and a neural network method with a region-based Markovian post-processing methods (PSP-MRF) by Schick et al [196]. We also have results for two commercial products. One that does pixel-level detection using the Chebyshev inequality and peripheral and recurrent motion detectors by Morde et. al. [162] and one which has only been published in a pending patent so far and whose description is not available [197]. The false positives rate (*FPR*), false negative rate (*FNR*) and F-measure (F_1) for these 22 methods are reported in Table 2.1. Note that, as mentioned in [79], these are the average *FPR*, *FNR* and F_1 across all videos.

From these results, one can conclude that the top performing methods are mostly GMM methods (DPGMM, SGMM-SOD, SGMM), data-driven methods (KNN and PBAS) and machine learning methods (SOBS and PSP-MRF). As shown in Table 2.2, GMM methods (particularly DPGMM and SGM-SOD) seem robust to background motion, camera jitter, and intermittent motion. This can be explained by the fact that these GMM methods come with a mode initialization (and updating) procedures that reacts swiftly to changes in the background. Table 2.2 also shows that there is space for improvement on jittery sequences and intermittent motion which are the categories with the lowest F-measure. Another unsolved issue is robustness to shadows. Although the F-measure of the most effective methods is above 0.86, the *FPR* on shadows of the three best methods is above 58%. This means that even the most accurate methods wrongly classify hard shadows.

Table 2.2 – Three highest ranked methods for each category together with their F-measure obtained on the CDnet 2012 dataset.

Category	1^{st}		2^{nd}		3^{rd}	
Baseline	SC-SOBS	0.93	Spectral-360	0.93	PSP-MRF	0.92
Dynamic background	DPGMM	0.81	Spectral-360	0.79	Chebyshev Inequality	0.77
Shadows	Spectral-360	0.88	SGMM-SOD	0.86	PBAS	0.86
Camera jitter	PSP-MRF	0.78	DPGMM	0.75	SGMM	0.75
Thermal	DPGMM	0.81	Spectral-360	0.78	PBAS	0.76
Intermittent object motion	SGMM-SOD	0.72	SC-SOBS	0.59	PBAS	0.58

Features

Here, we report results for eight of the most commonly-used features *i.e.*: grayscale, RGB, Normalized RGB, HSL, HSV, norm of the gradient, RGB+gradient, and YCbCr. We tested these features with two different methods. The first one is a basic background subtraction method with a forgetting constant of 0.002 [22]. The second is a version of ViBe [16] (a stochastic data-driven method) that we adapted to the various color spaces and removed its post-processing stage.

Results in Table 2.3 lead us to two main conclusions. First, using all three RGB color channels when possible instead of grayscale only always improves results. Second, out of the "illumination-robust" features N-RGB, HSL, HSV and gradient (grad), only HSV seems to provide good results globally. That being said, combining gradient with RGB helps improving results, especially for the basic method. As mentioned by several authors, this suggests that for some methods, combining color and texture is a good way of improving results.

Table 2.3 – F-measure obtained on the CDnet 2012 dataset for eight different features and two motion detection methods.

Method	Gray	RGB	N-RGB	HSL	HSV	grad	RGB+grad	YCbCr
Basic Method	0.48	0.53	0.49	0.56	0.58	0.3	0.59	0.59
ViBe [16]	0.72	0.75	0.60	0.65	0.74	0.11	0.74	0.71

Updating Scheme

In this section, we tested different updating schemes on one method. We tested the blind, conservative, "soft" conservative and "edge" conservative updating schemes. Again, we implemented a simple background subtraction method with RGB color feature. The difference from one implementation to another is the forgetting constant α in the formular

$$B^t = (1-\alpha)B^{t-1} + \alpha I^t. \tag{2.15}$$

For the blind scheme, $\alpha = 0.002$; for the conservative $\alpha = 0.002$ only for background pixels; the soft conservative $\alpha = 0.002$ for foreground pixels and $\alpha = 0.008$ for background pixels; and edge conservative, $\alpha = 0.007$ for background pixel, $\alpha = 0.002$ for foreground edge pixels and $\alpha = 0$ for the other foreground pixels.

Results in Table 2.4 show that the edge-conservative strategy is the most effective one while the conservative strategy is the least effective, although by a small margin. The reason for this small difference between results comes from the fact that videos in the CDnet 2012 dataset are all relatively short (at most six minutes) and thus do not exhibit major changes in the background as is the case when dealing with longer videos. Longer videos would certainly stretch the difference between each strategy.

Table 2.4 – F-measure for different background updating strategies obtained on the CDnet 2012 dataset.

Blind	Conservative	Soft-conservative	Edge-conservative
0.52	0.5	0.53	0.55

Post-processing

In this section, we compared different post-processing filters on the output of three methods. These methods are a basic background subtraction with a forgetting constant of 0.002, ViBe [16] and ViBe+ [221]. Note that ViBe+ is a method which already has a post-processing stage. The post-processing methods are three median filters (3×3, 5×5, and 7×7), a morphological opening and closing operation, a closing operation followed by a region filling procedure (as suggested by Parks and Fels [178]) and a connected component analysis. The latter removes small isolated regions, whether they are active regions or not.

Results in Table 2.5 show that all post-processing filters improved the results of all three methods. From our experiment, the post-processing improved the results of ViBe+, a method which already had a post-processing stage! Of course, the improvement rate is more significant for a low ranked method than for a precise one. Given its positive impact on performance and noise removal, we recommend to use at least a 5×5 median, but also other filtering operations to fill gaps, smooth object shapes, or remove small regions.

Table 2.5 – F-measure obtained for six different post-processing filters on the output of three motion detection methods.

Method	**No Post -processing**	**Med** 3×3	**Med** 5×5	**Med** 7×7	**Morph**	**Close +fill**	**Connected Component**
Basic Method	0.53	0.56	0.63	0.60	0.55	0.54	0.58
ViBe [16]	0.67	0.68	0.68	0.69	0.70	0.70	0.68
ViBe+ [221]	0.71	0.72	0.73	0.73	0.74	0.74	0.72

Combining Methods

So far, we analyzed and compared the behavior of individual motion detection techniques. A further step consists in combining methods. From that point, at least two questions arise: how should methods be combined, and which methods should be combined?

There are two strategies to combine methods: (1) consider every available method, regardless of its own performance, or (2) select a small subset of methods, based on their performances or on an optimization criterion. Here, we explore three different combination rules: two involving all $n = 22$ methods and one involving a subset of methods. Because it is difficult to model the correlation between individual methods and to take it into account, the combination rules considered here are based on the assumption that individual classifiers are independent from each other. An alternative would be to learn the combination rule [60], but this is out of the scope of this chapter.

The results obtained with the three different combination rules are shown on precision recall graphs with F_1 contour lines. It should be noted that the conclusions that can be drawn from the receiver operating characteristic space are different from those of the precision recall space. In this chapter, we only focus on the latter, and aim at maximizing the F_1 score. The following observations should therefore be interpreted with care.

Combination rule 1: majority vote among all methods.

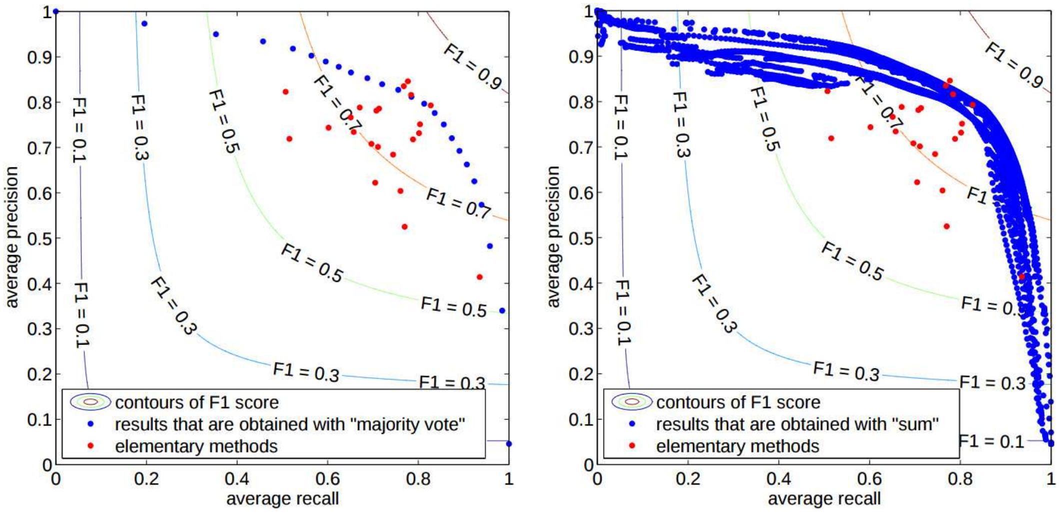
We define a decision thresholding function $\mathcal{F}_{th}$ as follows:

$$\mathcal{F}_{th}(x) = \begin{cases} 1, & \text{if } x \geq th \\ 0, & \text{otherwise .} \end{cases} \tag{2.16}$$

Let us denote the output of the i^{th} background subtraction method by $\hat{y}_i \in \{0,1\}$, the combined output by $\hat{y}_c \in \{0,1\}$, the ground truth by $y \in \{0,1\}$, and probabilities by $p(\cdot)$. The first combination rule considered in this chapter is

$$\hat{y}_c = \mathcal{F}_{th}(\frac{1}{n}\sum_{i=1}^{n}\hat{y}_i), \text{with } th \in [0,1]. \tag{2.17}$$

We refer to this technique as the "majority vote" rule, since it extends the classical unweighted majority vote (this one is obtained when n is odd, and the decision threshold is set to $th = 0.5$). This combination rule supposes that individual background subtraction algorithms are independent. Limits of what can be expected from such a combination are discussed in [125]. The results, obtained for every decision threshold, are shown in Figure 2.3(a).



(a) Majority vote

(b) Summation

Figure 2.3 – Results obtained from the combination of all 22 background subtraction methods, with two combination rules. For the purpose of comparison, the precision and the recall of the 22 individual methods are displayed in red. The blue dots correspond to different decision thresholds (th) as well as different estimations of the priors (Υ).

Combination rule 2: summation.

Another combination rule which is often encountered in the literature is the summation rule [119], which is also known as the mean rule [214], or the averaged Bayes' classifier [251]. Adapted to our framework, the summation rule can be formalized as:

$$\hat{y}_c = \mathcal{F}_{th}(\frac{1}{n}\sum_{i=1}^{n} p(y=1|\hat{y}_i,\Upsilon)), \text{with } th \in [0,1], \tag{2.18}$$

where

$$p(y=1|\hat{y}_i=0,\Upsilon) = \frac{\text{FNR}_i p(y=1|\Upsilon)}{\text{TNR}_i p(y=0|\Upsilon) + \text{FNR}_i p(y=1|\Upsilon)}, \tag{2.19}$$

$$p(y=1|\hat{y}_i=1,\Upsilon) = \frac{\text{TPR}_i p(y=1|\Upsilon)}{\text{FPR}_i p(y=0|\Upsilon) + \text{TPR}_i p(y=1|\Upsilon)}. \tag{2.20}$$

Here Υ represents the knowledge about the context (Υ is sometimes named the environment, as in [251]). The context is, for example, an indoor video-surveillance application, a particular video stream, the other pixels in the same image, or some information related to the past. However, the choice should be carefully made, since it can have an important impact on the performance of the combination. The priors $p(y = 0|\Upsilon)$ and $p(y = 1|\Upsilon)$ are usually estimated on the basis of the decisions taken by the individual methods on the whole image, in order to adapt dynamically to the context. But in some video-surveillance settings, some video regions are more likely to contain movement than others. In this case, it makes sense to estimate the priors on a neighborhood around the considered pixel, and also to take the history into account. This is somehow equivalent to the atlas used in [242], but in a dynamic setting.

The results for this combination rule are shown in Figure 2.3(b). We have considered the whole range of decision thresholds, and four ways of estimating the priors: (1) fixed priors ($p(y = 1) \in \{4\%, 8\%, 12\%, 16\%, 20\%\}$); (2) priors estimated on the whole image; (3) priors estimated on the whole image, with a temporal exponential smoothing applied on the estimated priors (with a smoothing factor $\alpha \in$ { 0.90, 0.37, 0.21, 0.14, 0.09, 0.05, 0.02 }); (4) priors estimated per pixel, on a square neighborhood of size $s \in \{1, 7, 31, 127, 511\}$. Note that estimating the priors for a combination is an ill-posed problem since false positives (false negatives) tend to increase (reduce) the estimated prior of the foreground, and therefore to encourage a higher number of positives (negatives) in the combined output. Obviously, the opposite behavior is wanted.

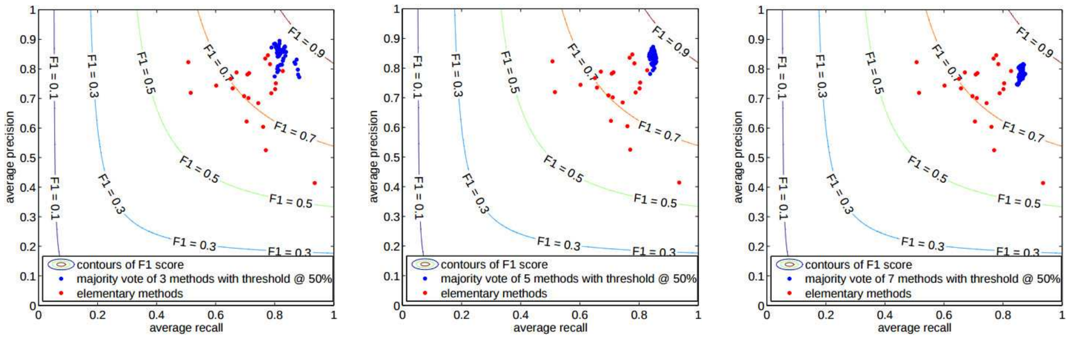
We observe some similarities between the majority vote and the summation. However, the majority vote only permits to reach $n = 22$ points in the precision recall space, whereas the summation permits a fine tuning. The optimal threshold for the majority vote and the summation varies significantly from one video to another (this is not represented on the graphs). Thus, there is a trade-off when choosing the threshold. The best overall threshold is about 0.4 for the majority vote and the sum. We have obtained our best results when estimating the priors on a neighborhood of 31×31 pixels.

Combination rule 3: majority vote of a predefined subset of methods.

It turns out that no combination of the 22 methods is able to beat significantly the best individual methods. Carefully selecting a subset of methods is therefore necessary. Note that an alternative would be to assign a "good" weight to each individual background subtraction method.

Our third combination rule is the same as the previous majority vote, except that it is applied on a subset of three, five and seven methods. Since computing the majority vote on every possible combination of methods is extremely time consuming, we first determined the 50 most promising subsets of methods. A prediction of the F_1 score has been obtained for every combination of three, five and seven methods, without the need to try them on the video sequences.

The results obtained with the third combination rule are depicted in Figure 2.4. We used a decision threshold of 0.5. Whereas a blind combination of all methods together does not permit to beat significantly the best individual methods (see Figure 2.3), combining carefully selected subsets of methods leads to a higher performance than the methods independently (see Figure 2.4).



(a) 50 subsets of three methods (b) 50 subsets of five methods (c) 50 subsets of seven methods

Figure 2.4 – Real precision and recall of the majority vote combination rule (at the neutral decision threshold). The predicted performance is shown, in blue, for 50 combinations of three, five, and seven methods, selected theoretically. The precision and the recall of the 22 individual methods are shown in red.

We have also observed how many times each method appears in the selected subsets of three, five, and seven methods. We have noticed that, as expected, the methods which have the highest F_1 score are often taken into account, even if the ranking in Table 2.1 is not strongly correlated with the occurrences. The results show that about one third of the methods are never selected. What is even more surprising is that the Local-Self similarity method [105], which has the worst ranking according to F_1 in Table 2.1, appears often in the selected combinations for three methods, and is systematically used in the top 50 subsets of five and seven methods, with no exception. Note that it is not a side effect of the independence assumption, as taking this method into account does not harm to the performance when the errors are positively correlated, as the results shown in Figure 2.4 illustrate. What should be noted about the Local-Self similarity method [105] is that it behaves differently from the other methods: it has the highest *TPR*, but also the highest *FPR*. Intuitively, a method that behaves differently may be useful in a combination, even if it has a bad ranking when evaluated alone, thanks to its complementarity with the other methods. This effect has already been observed by Duin *et al.* [61]. Therefore, if combining multiple background subtraction methods is possible, designing methods that are top-ranked, when they are evaluated alone, should not be the primary focus. Instead, designing complementary methods is preferable.

2.1.3 Benchmarking Conclusion

In this section, we presented different features, several updating schemes and many spatial aggregation and post-processing methods for motion detection. We also provided several benchmarking results based on the CDnet 2012 dataset. These results lead us to the following conclusions :

1. **Methods**: As of today, GMM (DPGMM, SGMM-SOD, SGMM), data-driven methods (KNN and PBAS) and machine learning methods (SOBS and PSPMRF) and among the most effective ones. On the other hand, every method has its limit. No method can perform best on all the categories.
2. **Remaining challenges**: Intermittent motion, camera jitter, and hard shadows are among the most glaring issues.
3. **Features**: HSV and RGB + gradient are the most effective features.
4. **Updating scheme**: The edge-conservative approach is the most effective scheme while the conservative approach is the least effective.
5. **Post-processing**: Every post-processing method that we have tested improved the results of our motion detection methods, especially for the simple low-ranked method. Post-processing should thus always be used.
6. **Combining methods**: One can beat the best performing methods by combining the output of several methods. The best results have been obtained with a majority

vote of three and five methods and with a threshold of 50$\%$. The best results are obtained by not only combining top ranked methods, but by combining methods which are complementary by nature.

2.2 The CDnet 2014 Dataset

According to Google Analytics, the CDnet 2012 website was visited by more than 12, 000 individual users after it was released, and results of 34 different methods were uploaded to our system (see the "2012 DATASET RESULTS" section on the CDnet website). However, the videos in the CDnet 2012 dataset still have not covered all the challenges of motion detection. To make CDnet dataset more objective, we prepared a new set of videos representing five additional categories incorporating challenges not addressed in the CDnet 2012 dataset. In total, more than $70,000$ frames have been captured, and then manually segmented and annotated by a team of 13 researchers from seven universities. Besides, ground truths for all frames were made publicly available for the CDnet 2012 dataset for testing and evaluation, thus users can use them to over-tune their algorithm parameters to achieve better results. For the new videos in the CDnet 2014 dataset, ground truths of only the first half of every video are made publicly available for testing. The evaluation, however, is across all frames for all the videos (both new and old) as in the CDnet 2012 dataset. This helps to reduce the possibility of over-tuning algorithm parameters.

The CDnet 2014 dataset provides realistic, camera-captured (without Computer generated imagery (CGI)), diverse set of indoor and outdoor videos like the CDnet 2012 dataset. These videos have been recorded using cameras ranging from low-resolution IP cameras, higher resolution consumer grade camcorders, commercial Pan–tilt–zoom (PTZ) cameras to near-infrared cameras. As a consequence, spatial resolutions of the videos in the CDnet 2014 dataset vary from 320×240 to 720×486. Due to the diverse lighting conditions present and compression parameters used, the level of noise and compression artifacts significantly varies from one video to another. Duration of the videos are from 900 to 7,000 frames. Videos acquired by low-resolution Internet protocol (IP) cameras suffer from noticeable radial distortion. Different cameras have different hue bias due to different white balancing algorithms employed. Some cameras apply automatic exposure adjustment resulting in global brightness fluctuations in time. Frame rate also varies from one video to another, often as a result of limited bandwidth. Since these videos have been captured under a wide range of settings, the extended CDnet 2014 dataset does not favour a certain family of change detection methods over others.

2.2.1 Video Categories

The CDnet 2014 dataset contains 53 videos grouped in 11 categories as shown in Fig. 2.5 (including six categories of the CDnet 2012 dataset, namely *Baseline*, *Dynamic Background*, *Camera Jitter*, *Shadow*, *Intermittent Object Motion*, and *Thermal*). Similarly to the CDnet 2012 dataset, the change detection challenge in a category is unique to that category. Such a grouping is essential for an unbiased and clear identification of the strengths and weaknesses of different methods. The categories in the CDnet 2014 dataset are:

1. **Baseline**: contains four videos with a mixture of mild challenges of the next four categories. These videos are fairly easy and are provided mainly as reference.
2. **Dynamic Background**: contains six videos depicting outdoor scenes with strong background motion.
3. **Camera Jitter**: represents four videos captured with unstable cameras.
4. **Shadow**: is composed of six videos with both strong and soft moving and cast shadows.
5. **Intermittent Object Motion**: contains six videos with scenarios known for causing ghosting artifacts (*e.g.* contains still objects that suddenly start moving).
6. **Thermal**: is composed of five videos captured by far-infrared cameras.
7. **Challenging Weather**: This category contains four outdoor videos showing low visibility winter storm conditions. This includes two traffic scenes in a blizzard, cars and pedestrians at the corner of a street and people skating in the snow. These videos present a double challenge: in addition to snow accumulation, the dark tire tracks left in the snow have potential to cause false positives.

8. **Low Frame Rate**: All four videos in this category are recorded with IP cameras. The frame rate varies from 0.17 fps to 1 fps due to limited transmission bandwidth. By nature, these videos show "erratic motion patterns" of moving objects that are hard (if not impossible) to correlate. Optical flow might be ineffective for these videos. One sequence is particularly challenging (*port_0_17fps*), which shows boats and people coming in and out of a harbour, as the low frame rate accentuates the wavy motion of moored boats causing false detections.

9. **Night**: This category has six motor traffic videos. The main challenge is to cope with low-visibility of vehicles yet their very strong headlights that cause over saturation. Headlights cause halos and reflections on the street.

10. **PTZ**: We included four videos in this category: one video with a slow continuous camera pan, one video with an intermittent pan, one video with a 2-position patrol-mode PTZ, and one video with zoom-in/zoom-out. The PTZ category by itself requires different types of change detection techniques in comparison to static camera videos.

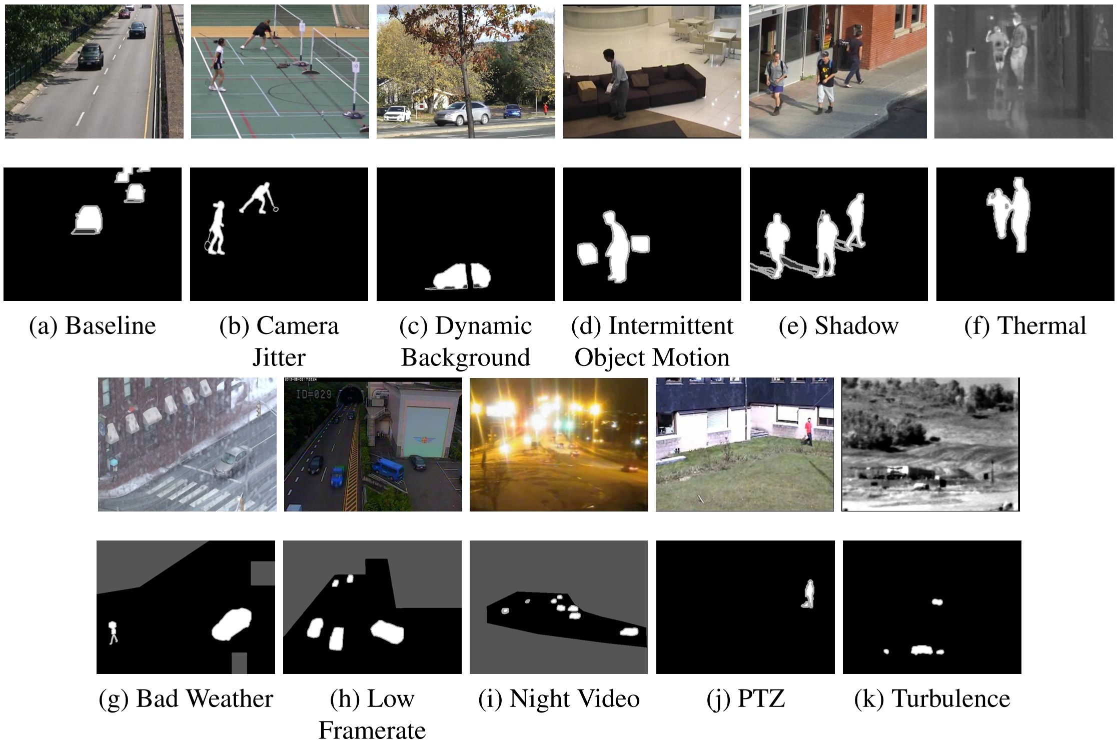
11. **Air Turbulence**: This category contains four videos showing moving objects depicted by a near-infrared camera at noon during a hot summer day. Since the scene is filmed at a distance (5 to 15 km) with a telephoto lens, the heat causes constant air turbulence and distortion in frames. This results in false positives. The size of the moving objects also varies significantly from one video to another. The air turbulence category presents very similar challenges to those arising in long-distance remote surveillance applications.

2.2.2 Ground Truth Labels

For consistency, we use the same labeling procedure as for the CDnet 2012 dataset. Each frame has been manually annotated at pixel level, with the following five labels:

1. **Static** pixels are assigned grayscale value of 0.

2. **Shadow** pixels are assigned grayscale value of 50. The *Shadow* label is associated with hard and well-defined moving shadows such as the one in Fig. 2.6.



(a) Baseline

(b) Camera Jitter

(c) Dynamic Background

(d) Intermittent Object Motion

(e) Shadow

(f) Thermal

(g) Bad Weather

(h) Low Framerate

(i) Night Video

(j) PTZ

(k) Turbulence

Figure 2.5 – Sample video frames of all 11 categories in the CDnet 2014 dataset.

3. **Non-ROI** [1] pixels (*i.e.* outside of the ROI) are assigned grayscale value of 85. The first few hundred frames of each video sequence are also labelled as *Non-ROI* to prevent the corruption of evaluation metrics due to errors during initialization. At the same time, the *Non-ROI* label can also prevent the metrics from being corrupted by activities unrelated to the category considered.
4. **Unknown** grayscale value of 170 assigned to pixels that are half-occluded or corrupted by motion blur.
5. **Moving** pixels are assigned grayscale value of 255.

Please note that the evaluation metrics discussed in Section 2.2.3 consider the *Shadow* pixels as *Static* pixels.

[1] ROI stands for Region of Interest.

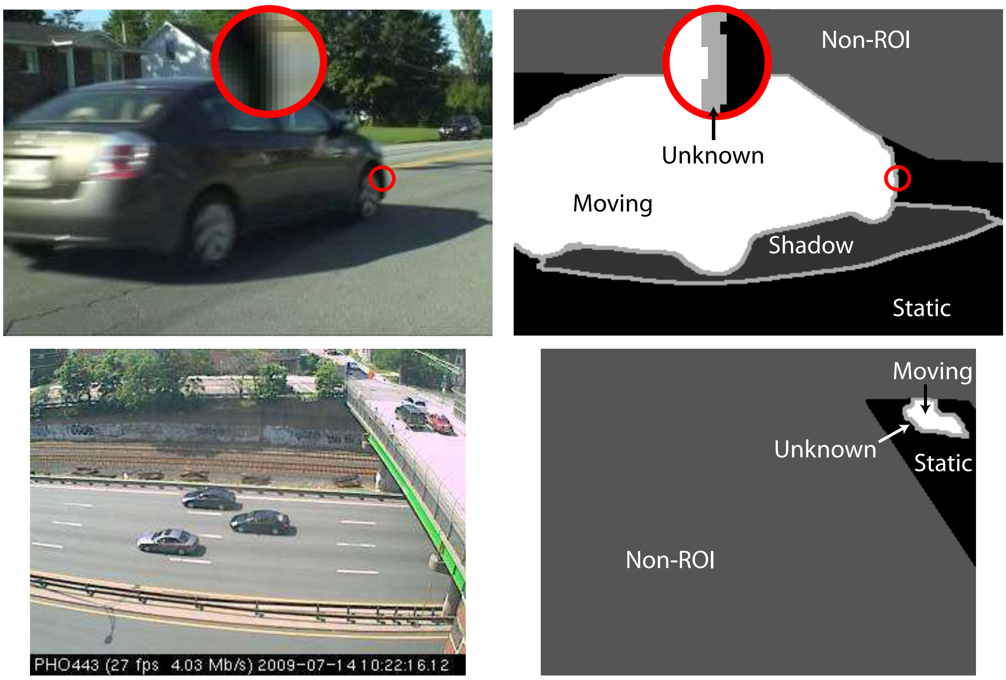


Figure 2.6 – 5-class ground truth label fields in the CDnet 2014 dataset.

2.2.3 Evaluation Metrics

We use seven metrics for the evaluation in the CDnet 2014 dataset, namely Recall (Re), Specificity (Sp), False Positive Rate (FPR), False Negative Rate (FNR), Percentage of Wrong Classifications (PWC), Precision (Pr), and F-measure (or F_1 score). For the *Shadow* category, we also provide an average FPR that is confined to the hard-shadow areas (*FPR-S*). The metrics are discussed and tested in Section 2.1.1.

In order to easily assess the various change detection methods, these metrics are then combined into two metrics *R* and *RC* [79]. *R* represents an average ranking computed across all overall-average metrics. *RC* is an average ranking computed across all categories. The scores of the seven metrics plus the *R* and *RC* for all methods submitted to the CDnet 2014 dataset are presented in Table 2.6.

Table 2.6 – Overall results across all categories (RC: average ranking across categories, R: average overall ranking).

Method	RC	R	Re	Sp	FPR	FNR	PWC	F-measure	Pr
Cascade CNN (supervised) [239]	1.45	1.00	0.95	0.99	0.0032	0.05	0.41	0.92	0.90
IUTIS-5 [26]	3.73	4.14	0.78	0.99	0.0052	0.22	1.20	0.77	0.81
IUTIS-3 [26]	7.27	6.43	0.78	0.99	0.0060	0.22	1.30	0.76	0.79
DeepBS (supervised) [12]	7.64	12.57	0.75	0.99	0.0095	0.25	1.99	0.75	0.83
PAWCS [208]	8.36	6.43	0.77	0.99	0.0051	0.23	1.20	0.74	0.79
SuBSENSE [207]	10.09	9.86	0.81	0.99	0.0096	0.19	1.68	0.74	0.75
WeSamBE	10.18	7.86	0.80	0.99	0.0076	0.20	1.51	0.74	0.77
SharedModel [45]	10.82	8.57	0.81	0.99	0.0088	0.19	1.50	0.75	0.75
FTSG [233]	11.36	11.14	0.77	0.99	0.0078	0.23	1.38	0.73	0.77
SaliencySubsense	12.09	12.71	0.77	0.99	0.0086	0.23	1.90	0.72	0.76
M4CD Version 2.0	12.09	15.29	0.79	0.98	0.0159	0.21	2.30	0.70	0.74
SSBS	12.36	13.14	0.74	0.99	0.0077	0.26	1.89	0.71	0.78
CwisarDRP	12.64	13.29	0.71	0.99	0.0053	0.29	1.72	0.71	0.79
M4CD Version 1.0	15.18	17.14	0.78	0.98	0.0151	0.23	2.36	0.69	0.73
C-EFIC [7]	15.27	14.86	0.80	0.98	0.0218	0.20	2.63	0.73	0.75
MBS-v1 [195]	15.45	12.57	0.74	0.99	0.0073	0.26	1.26	0.73	0.74
CwisarDH [83]	17.27	15.57	0.66	0.99	0.0052	0.34	1.53	0.68	0.77
MBS-v0 [195]	17.36	14.14	0.72	0.99	0.0071	0.28	1.39	0.71	0.74
EFIC [6]	17.73	18.57	0.79	0.98	0.0221	0.21	2.79	0.71	0.72
Spectral-360 [198]	21.55	20.43	0.73	0.99	0.0139	0.27	2.27	0.67	0.71
SBBS	21.73	22.86	0.71	0.98	0.0173	0.29	2.43	0.67	0.72
IUTIS-2 [26]	22.36	25.86	0.66	0.98	0.0162	0.34	3.15	0.60	0.71
BMOG	22.45	24.00	0.73	0.98	0.0187	0.27	2.98	0.65	0.70
AMBER [225]	22.91	24.86	0.70	0.98	0.0206	0.30	2.90	0.66	0.72
IUTIS-1 [26]	24.18	28.57	0.77	0.95	0.0501	0.23	5.75	0.58	0.59
AAPSA [189]	24.45	23.57	0.65	0.99	0.0095	0.35	2.07	0.62	0.69
GraphCutDiff [160]	26.45	31.00	0.63	0.98	0.0220	0.37	3.68	0.57	0.67
SC-SOBS [149]	26.64	26.71	0.76	0.95	0.0453	0.24	5.15	0.60	0.61
Mahalanobis distance [20]	27.09	24.86	0.16	0.99	0.0069	0.84	3.48	0.23	0.74
SOBS-CF [153]	27.36	27.43	0.78	0.94	0.0558	0.22	6.07	0.59	0.58
RMoG [222]	27.64	27.29	0.59	0.99	0.0135	0.41	2.96	0.57	0.70
KDE-ElGammal [62]	28.91	30.29	0.74	0.95	0.0481	0.26	5.63	0.57	0.58
CP3-online [135]	30.91	28.86	0.72	0.97	0.0295	0.28	3.43	0.58	0.56
GMM|Stauffer-Grimson [210]	31.09	29.86	0.68	0.98	0.0250	0.32	3.77	0.57	0.60
DCB	31.73	28.43	0.39	0.99	0.0103	0.61	2.88	0.40	0.63
GMM|Zivkovic [273]	32.36	32.29	0.66	0.97	0.0275	0.34	4.00	0.56	0.60
MSTBGM [146]	34.09	33.86	0.66	0.95	0.0458	0.34	5.55	0.51	0.55
Euclidean distance [20]	35.18	34.71	0.68	0.94	0.0551	0.32	6.54	0.52	0.55

2.2.4 Methods Tested and Experimental Results

A total of 14 change detection methods were evaluated for the IEEE Change Detection Workshop 2014 [1]. Until now, 38 methods submitted their results to the CDnet 2014 dataset. Among these methods, some are simple methods relying on a plain background subtraction, such as the Euclidean and Mahalanobis distance methods as described in [20]. Classical and frequently-cited methods such as KDE-based estimation by Elgammal *et al.* [62] and GMM by Stauffer and Grimson [210] are also included. There are also different variations of GMM, such as a region-based GMM model [222], and a shareable GMM model [45], and a recursive GMM method with an improved update of the Gaussian parameters and an automatic selection of the number of modes [273].

Among the more recent methods, two deep learning methods, *i.e.* Cascade CNN (a supervised method presented in chapter 3) [239] and DeepBS (a supervised method) [12] achieve very good performances. This clearly shows that deep learning methods have the ability to learn the most useful features of the video and model the background accurately even under challenging circumstances. Beyond that, instead of modeling the background, IUTIS-5, IUTIS-3 and IUTIS-2 [26] try to combine the results of the other motion detection methods to improve their performances. Among which, the combination of five (IUTIS-5) and three (IUTIS-3) methods achieve good results. GraphCutDiff [160] uses the optical flow and GMM model to detection motion. Then the graphcut is applied to improve the results. FTSG [233] is a method that detects motion with a three-step procedure. More specifically, the three steps are: (1) detecting moving object with two complementary pixel-level motion detection models, one model is based on the trace of a flux tensor while the other is based on a variant of the conventional GMM; (2) combing the two motion detection results; and (3) removal of ghosting artifacts. In SuBSENSE [207], color and local binary similarity patterns are used to make pixel-level decisions. The SC-SOBS method [149] is a machine learning method with a self-organizing neural map. The SOBS-CF [153] is the fuzzy version of the SOBS algorithm. Spectral-360 [198] detects motions by calculating the correlation between the diffuse spectral reflectance components of a new video frame with an evolving background model derived from recent training frames. The method of [225] compares the current pixel value with one long-term and several short-term adaptive templates. For CP3-online [135], instead of modeling the background for each pixel individually, it models the color distribution of pixels with strong spatial correlation. The authors argue that such spatial model is robust to sudden illumination changes. [146] uses a 3-scale spatio-temporal color/luminance Gaussian pyramid background model to model the background.

For each of these methods, only one set of parameters was used for all the videos. These parameters were selected according to the authors' recommendations or, when not available, were adjusted to enhance the overall results. All parameters are available on the CDnet 2014 website.

In order to give the reader an intuitive understanding of the overall performance of all 14 methods, we put in Fig. 2.7 the mean and standard deviation of the F-measure for all methods within each category of the CDnet 2014 dataset. Without much surprise, the "*PTZ*" category has the lowest performance. The second most difficult category is the "*night*" category, followed by the "*low framerate*" category. Surprisingly, most methods performed relatively well on the "*bad weather*" category. We also report the median metrics obtained by all methods in the new five categories as shown in Table 2.7.

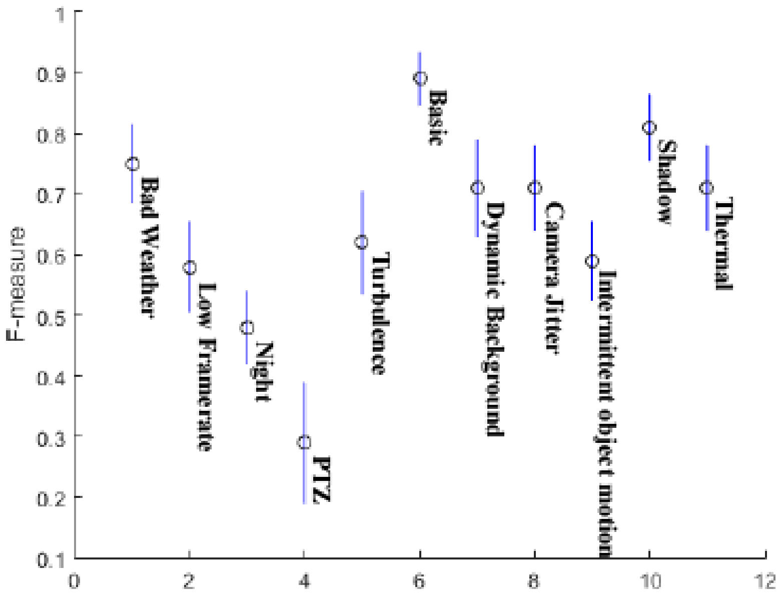


Figure 2.7 – Mean and standard deviation of the F-measure over all methods within each category of the CDnet 2014 dataset.

In order to identify where the methods fail, we integrated the error at each pixel of each frame and for every method. This leads to error maps shown in Fig 2.8. In these images, red, green, white and black stand for the false negative (FN), false positive (FP), true positive (TP), and true negative (TN) respectively. Pixels with saturated red and green indicate that every method failed at those pixels. After analyzing these error maps, we came to identify the most glaring issues that no single method handles well:

1. **PTZ**: any camera motion (pan, tilt or zoom) causes major false positives.

Table 2.7 – Median F-measure, FPR, FNR, and PWC obtained by all methods for each category.

Category	F-measure	FPR	FNR	PWC
Bad Weather	0.78	0.0015	0.26	0.66
Low Framerate	0.61	0.0050	0.26	1.00
Night	0.49	0.0227	0.42	3.86
PTZ	0.32	0.0505	0.25	5.50
Turbulence	0.66	0.0012	0.27	0.25
Basic	0.92	0.0021	0.07	0.49
Dynamic Background	0.74	0.0025	0.20	0.49
Camera Jitter	0.74	0.0096	0.22	2.08
Intermittent Object Motion	0.59	0.0131	0.39	4.73
Shadow	0.82	0.0082	0.14	1.59
Thermal	0.75	0.0053	0.33	1.97

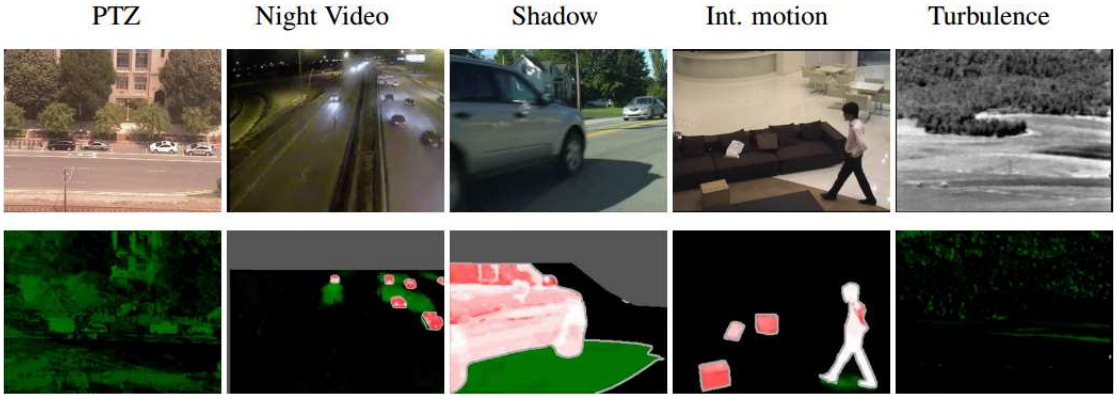


Figure 2.8 – Error maps showing systematic errors. Green: false positives, red: false negatives, gray: out of ROI, white: true positives, and black: true negatives.

2. **Night Videos**: the lack of illumination causes numerous false negatives while headlight reflections cause systematic false positives.
3. **Shadow**: Hard shadows are still a challenge for every method.
4. **Intermittent Object Motion**: Any object which stops moving for some time, eventually ends up being mis-detected. A similar situation occurs when a background object is removed from the scene.
5. **Turbulence**: Air turbulence causes the systematic occurrence of false positives.

2.2.5 Conclusion

In this section, we introduced the CDnet 2014 dataset, including the video categories, the ground truth labels, and the evaluation metrics. We also presented the experimental results of the submitted methods and statistic analysis of them. With all the discussions before, we can draw the conclusions:

1. Categories: State-of-the-art motion detection methods have no problem to separate foregrounds of a video if the background of the video is stable. Challenges such as thermal videos, video with jitters, and bad weather are slightly more difficult for most state-of-the-art methods. However, some challenges, *e.g.* videos shot by PTZ cameras, videos shot at night with over saturation cased by the light, and videos with low framerate, are extremely difficult for most of the methods.
2. Combining methods: Using a smart strategy to combine the results of different methods can always improve the performances. However, two things should be noticed while combining different methods: (1) It is recommended to combine methods that are good at different challenges, this can help to overcome each method's limit and achieve better performance. (2) Even though combining methods can improve the result, the improvement can be limited.
3. Deep learning methods: As a new approach of motion detection method, deep learning methods achieve very good results. However, as a supervised method, deep learning method requires a large number of labeled data to train the model.

As shown by the results, there is no single traditional motion detection method can perform excellent results for all the challenges. Merging results of different methods can help to improve the motion detection performance. However, the F-measures (*e.g.* IUTIS-5: 0.77 IUTIS-3: 0.76) are still not high enough to make the results directly be used as ground truths. In Section 3, we will introduce the "Cascade CNN" method, this CDnet 2014 top first ranked method produces extremely accurate results with F-measure of 0.95 which is within the error margin of a human being.

Chapter 3

Interactive Deep Learning Method for Segmenting Moving Objects

Résumé

As mentioned in the previous chapter, most motion detection dataset are relatively small, while most of them do not provide pixel-wise labeled ground truth. One of the reasons is that automatic labeling methods are not accurate enough to generate ground truth, while labeling videos manually is accurate but extremely time consuming. In this chapter, we propose a semi-automatic method for segmenting foreground moving objects pictured in surveillance videos. By manually labeling only a small number of frames, our model can label the rest of the video. The accuracy of our method is similar to that of a human being, but only take 1/50 of the manually labeling time. The model is fully convolutional, and thus the training and inference is done on the whole image. This chapter is published as a paper with title **Interactive Deep Learning Method for Segmenting Moving Objects** [239] in international journal of Pattern Recognition Letter in 2016.

Commentaires

The initial idea was proposed by the Ph.D. candidate's supervisor. The experiments are mostly designed and implemented by the Ph.D. candidate. The empirical evaluation was also made by the Ph.D. candidate. The article was mostly written and organized by the Ph.D. candidate.

Interactive Deep Learning Method for Segmenting Moving Objects

Yi Wang
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
yi.wang@usherbrooke.ca

Zhiming Luo
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
School of Information Science and Technology, Xiamen University,
Xiamen, Fujian, 361005 China
zhiming.luo@usherbrooke.ca

Pierre-Marc Jodoin
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
pierre-marc.jodoin@usherbrooke.ca

Keywords: Motion detection, Ground truthing, Convolutional neural network

Abstract

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 ground truth (changdetection.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

3.1 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 [248], scene understanding [107], anomaly detection [133], and traffic analytics [148] to name a few. In the last decade, a growing number of machine learning methods have been used to solve these issues [144, 77]. 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 [28]] 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 [29]. As reported on the changedetection.net (CDnet) 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 seconds per frame. Thus, the manual labeling of a 4 minute video ($\sim$ 7,000 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 [123]. The main reason for using CNN comes with its ability to learn its own 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 the 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 is very redundant, 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 [100] and CIFAR datasets [122]. Our approach also differs from traditional motion detection method as it processes each frame independently without considering motion features and maintaining a background model.

We explore various CNN configurations such as a multiresolution CNN, a cascaded architecture, the FCN-8s [144] model as well as various training configurations. Results obtained on the CDnet 2014 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 highly accurate and only requires a small number of user interactions.

— Following an extensive evaluation on the CDnet 2014 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.

3.2 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 [246], a mixture of Gaussians [211], generalized Gaussian mixture [8], and Bayesian models [183] 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. [16] 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. [93] 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 [209] 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 [208] 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 involve a one-class support vector machine (SVM), others involve a neural network, a Parzen window estimator, a principal component analysis (PCA) model, some fuzzy logic, and many more (refer to [28] for an extensive survey). However, none of these methods have been shown sufficiently accurate to produce ground truth quality results.

As an alternative, some foreground segmentation methods rely on user interaction to improve the 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 [126, 193]. 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, [13] proposed and extended 3D spatio-temporal graph cut method that implements a 6-pixel neighborhood (four spatial and two temporal neighbors). In [230], users are asked to give interaction not only on each image, but also on the x-t dimension to provide additional temporal information. The method by [77] 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.

3.3 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 goals of our method are two folds: (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, these 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. 3.1, 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.

3.3.1 Selecting and Labeling Training Frames

Given an input video, the first step of our method is to select and manually segment N training frames. In that perspective, different selection strategies can be considered. One could uniformly 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

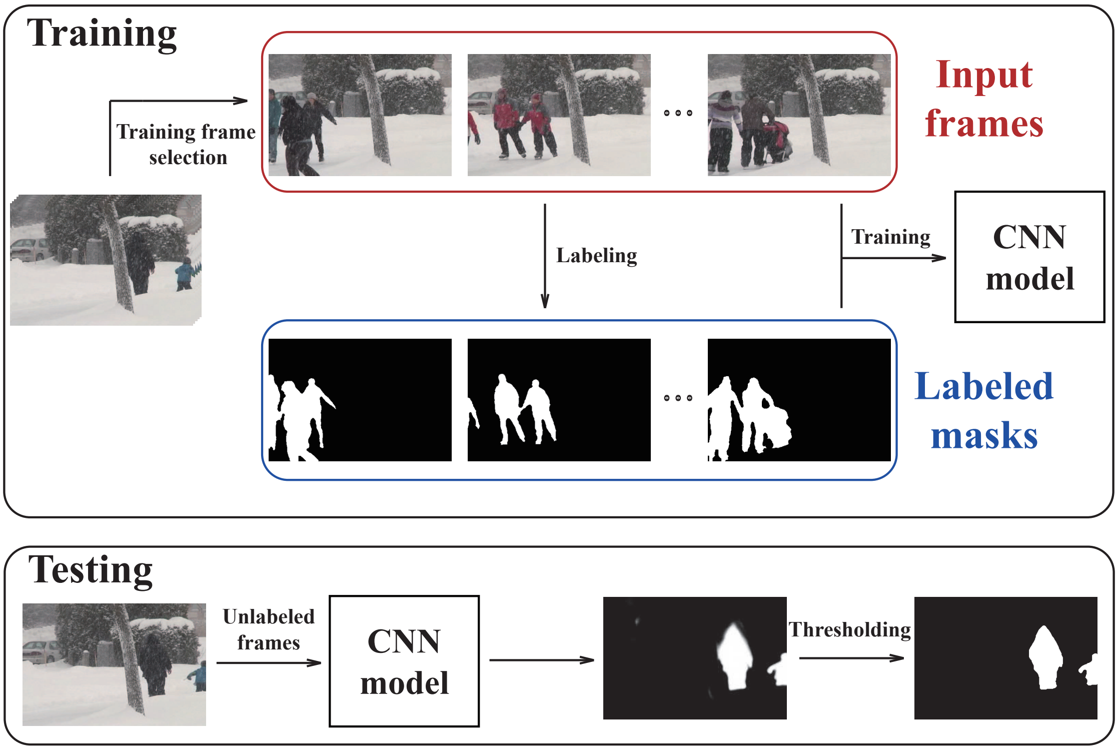


Figure 3.1 – The pipeline of our model.

(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 the annotation.

3.3.2 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 are two folds. 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) [55], scale-invariant feature transform (SIFT) [145], or local binary pattern (LBP) [85]. Furthermore, unlike conventional hand-design features, learned features which come from multiple layers of the network focus on various levels 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.

Basic CNN Model

A CNN is typically made of a series of convolutional layers, activation layers, pooling layers, and fully-connected layers [129, 78]. 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 [123]. 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 × 31 patch around each pixel and consider that patch as a small to-be-classified squared image [67].

The detailed configuration of our *basic* CNN model is provided in Table 3.1 and illustrated in Fig 3.2. As can be seen, our *basic* CNN model contains four convolutional layers and two fully connected (FC) layers. Each convolutional layer uses a filter with size of 7 × 7 and rectified linear unit (ReLU) as the activation function. Also, the first two convolutional layers come with a 2 × 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.

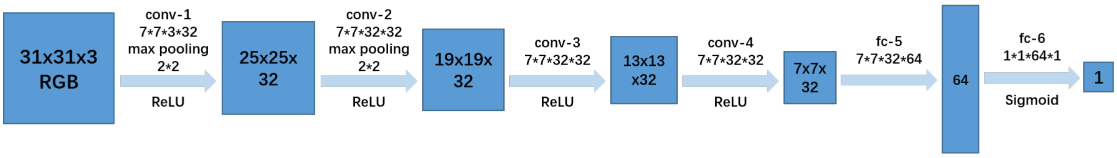


Figure 3.2 – The diagram of our basic CNN model which consist of four convolutional layers and two fully connected layers. Also the first two convolutional layers come with a 2 × 2 max pooling layer.

Table 3.1 – The architecture of our basic CNN model.

Layer	1	2	3	4	5	6
Stage	conv	conv	conv	conv	FC	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

By considering the CNN output as a likelihood probability, we use a cross entropy loss function for training [24]:

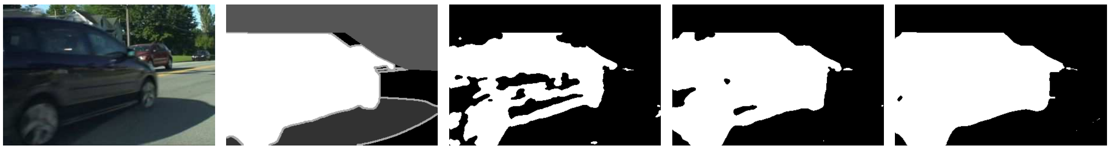
$$Loss = -\frac{1}{K} \sum_{k=1}^{K} \left[C_k \log \hat{p}_k + (1 - C_k) \log (1 - \hat{p}_k) \right], \tag{3.1}$$

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 the training, each pixel is treated independently and no motion features are extracted.

Multi-scale CNN Model

The *basic* CNN model is not void of limitations. One of its main drawback comes from its fix input patch size. Since it processes patches with a fixed size 31×31, the *basic* CNN model is good for distinguishing foreground and background objects whose sizes are in the order of 31×31 or less. Unfortunately, videos often contain moving objects significantly larger than that. This typically happens when foreground moving objects are close to the camera. As shown in Fig. 3.3, large moving objects often carry out large uniform textureless areas which can be miss-classified as background. Fig. 3.3 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. 3.4. Given a to-be-segmented 2D image I, we first resize it into two different



(a) Input frame (b) Ground truth (c) basic CNN (d) MSCNN (e) MSCNN + Cascade

Figure 3.3 – A video frame with large moving object (c) fooling our *Basic CNN* method due to its large uniform area; (d) the *multi-scale* and (e) the *cascaded CNN* models greatly reduce the number of false positives by making the system more scale invariant and improving spacial coherence.

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, I_{scale1} and I_{scale2} are resized back to the size of the input frame I. Note that since we use a stride of 1 at the pooling and convolution layers (*cf.* Table 3.1), O has *de facto* the same size as I. The final foreground probability map O_{final} is obtained with an average pooling across the upscaled maps (*cf.* the rightmost picture in Fig. 3.4). All three CNNs share the same weights.

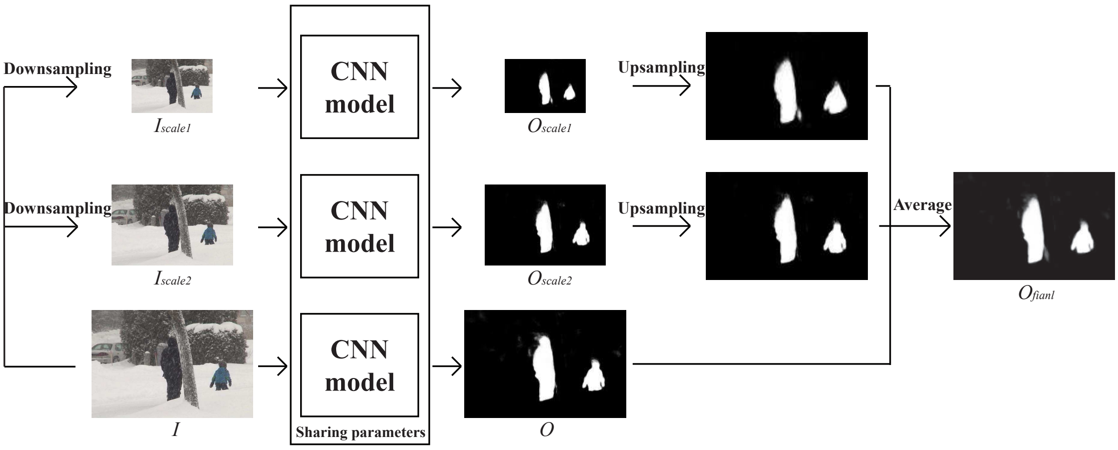


Figure 3.4 – The architecture of the proposed multi-scale CNN model.

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 [245, 120, 97] use a conditional random field (CRF) with fixed weights as a way to enforce spatial coherence. Even though 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.

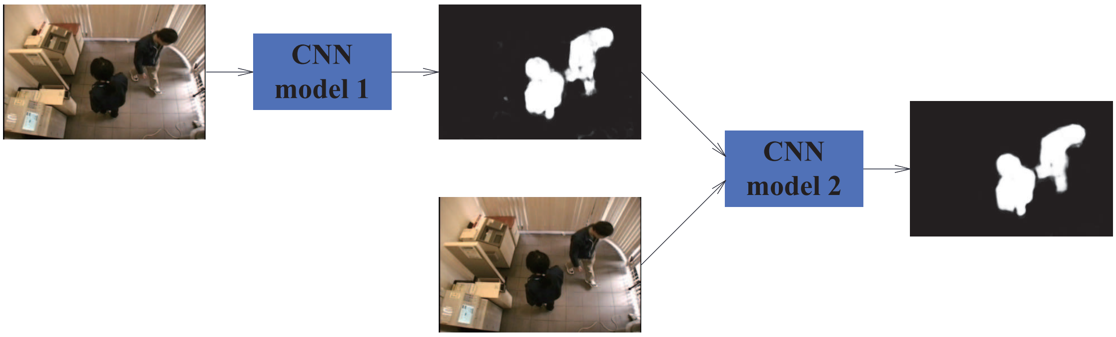


Figure 3.5 – The architecture of the proposed Cascaded model.

In order to model the dependencies among adjacent pixels and thus enforce spatial coherence, we implemented a *cascaded* CNN model. As shown in Fig. 3.5, 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.3.3(e)). 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 which is showed in Table 3.1. 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. While training the cascaded model, we first trained the CNN-1 model, and then fixed the parameters of CNN-1 and only updated the parameters of CNN-2. Note that we also tried to increase the number of CNNs in the cascade model, however, with more CNNs in a cascade model, the performance was rarely improved.

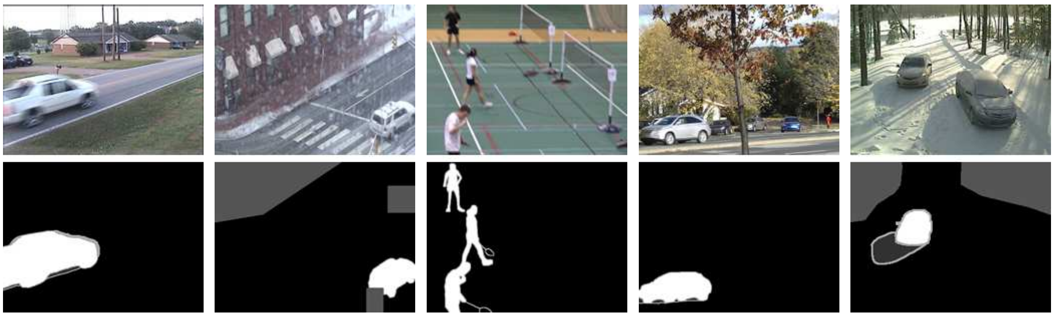
Training Details

All three CNN models have been implemented with the MatConvNet deep learning toolbox which it a wrapper on top of Caffe [223]. Since we intend to train the models on a small number of annotated frames while 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 the CNN models on the manually outlined frames from scratch, we pre-trained our model on a larger dataset as initialization [260]. The pre-training was done only once with the *Motorway* dataset [148], a dataset with pixel accurate ground truth of video surveillance images. After transfering the weights to our models, we *fine-tuned* the CNN parameters for each video based on the loss function in Eq.(3.1). The *Adadelta* optimization method [262] was used for updating parameters with an initial learning rate of 0.01. Our models were trained for 20 epochs with a batch size of 5 frames. Besides, although the training could be done patch-wise, for more efficient approach, we did it on whole images, in which, the energy gradient of the pixels located outside the previously-selected ROI were forced to zero. Last but not least, in order to keep the size of the segmentation result the same as the input frame, we applied mirror padding on the original frame during the testing.

3.4 Experiment and Results

3.4.1 Dataset

We tested our method on the CDnet 2014 dataset [237], the largest video dataset with pixel accurate ground truth. The CDnet 2014 dataset 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 2014 dataset are shown in Fig. 3.6.



(a) PTZ (b) Bad weather (c) Camera jitter (d) Dynamic background (e) Intermittent object motion

Figure 3.6 – A collection of video frames of the CDnet 2014 dataset with their associated ground truths used in our experiments.

3.4.2 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:

$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}. \tag{3.2}$$

Although is 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 videos with only 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 also use the PWC metric which incorporates TN:

$$\text{PWC} = \frac{100 * (\text{FN} + \text{FP})}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}. \tag{3.3}$$

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 object a wrongly labeled pixel is located. The FPED and FNED are calculated as:

$$\mathrm{FPED} = \frac{1}{|\mathrm{FP}|} \sum_{x \in \mathrm{FP}} \min_{y \in \mathrm{FG}} \mathrm{Dist}(x, y), \tag{3.4}$$

$$\mathrm{FNED} = \frac{1}{|\mathrm{FN}|} \sum_{x \in \mathrm{FN}} \min_{y \in \mathrm{BG}} \mathrm{Dist}(x, y), \tag{3.5}$$

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

3.4.3 Experiments

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 to train our *basic* CNN model. Note that those numbers are relatively small comparing with the overall video size (CDnet 2014 videos contain between 1,000 and 8,000 frames). After applying different thresholds on our model's output foreground probability map, we get different F-measure values and plot it in Fig. 3.7. As can be seen, a threshold between 0.6 and 0.7 give the best performance in most cases. Furthermore, given the same number of training frames, the manual strategy achieves higher F-measure than random and uniform strategy. 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 comparing with the manual training frame selection strategy.

Results for every metric are provided in Table 3.2. As one can see, among the three training frame selection strategies, the random and uniform achieve similar performances with a F-measure of at most 0.86 and 0.87. As for the manual selection, the F-measure reaches 0.9. While 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 with large amount of

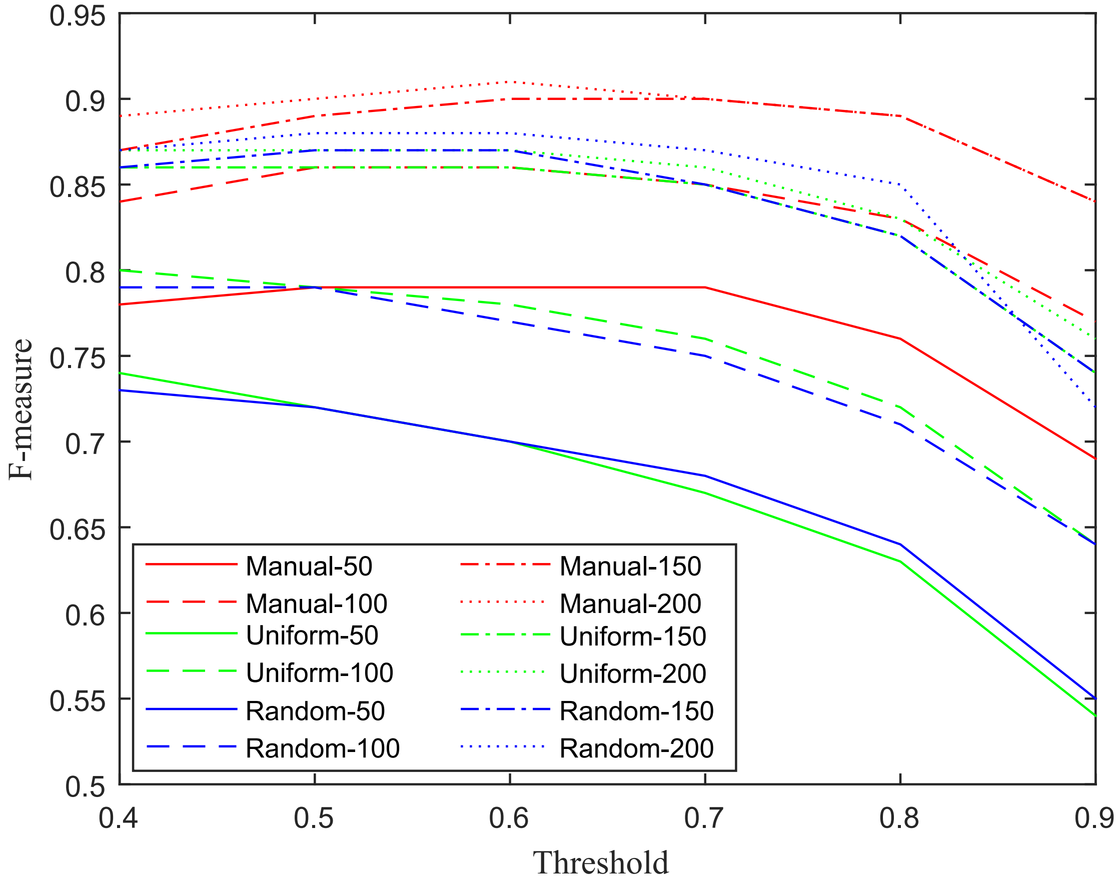


Figure 3.7 – F-measure of various training frame selection strategies, various training frames number and different thresholds.

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.

Evaluation of the Different 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 are manually selected, and the binarizing threshold is set to 0.7 for all the models.

Table 3.2 – Results of the *Basic* CNN model with different training frame selection strategies.

Strategy	Training	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

As can be seen in Table 3.3, the multi-scale and the cascaded architectures significantly improve the performance. We can also see that the more training frames a model has, the more accurate the end result it gets. The best performance is achieved by the *MSCNN + Cascade* model, whose PWC, FPED and FNED are about 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 a small number of 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 3.4. By using only 50 frames for training, our model gets to segment videos from four categories out of 11 with very high accuracy (F-measure $\geqslant 0.95$). By increasing the number of training frames to 200, our model achieve outstanding performance for most of the categories, *e.g.*, 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.

Table 3.3 – Results of four different 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
CNN+Cascade	100	0.90	0.47	7.80	2.82
	50	0.88	0.53	8.88	3.15
	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
MSCNN+Cascade	50	0.88	0.49	10.52	2.05
	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

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) [144] 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 as we did for our method.

Quantitative results for *MSCNN + Cascade* and FCN are presented in Table 3.5. As shown in the table, *MSCNN + Cascade* outperforms FCN on every metric with different number of training frames. Note that these results are averaged across 11 categories including some extreme cases like night videos, camera motion, and low frame rate videos. Examples of results given by our method and FCN are shown in Fig. 3.8. As shown in Fig. 3.8, FCN often mis-detects the foreground objects which leads to FN

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

Table 3.4 – Metrics for MSCNN + Cascade on each CDnet video category.

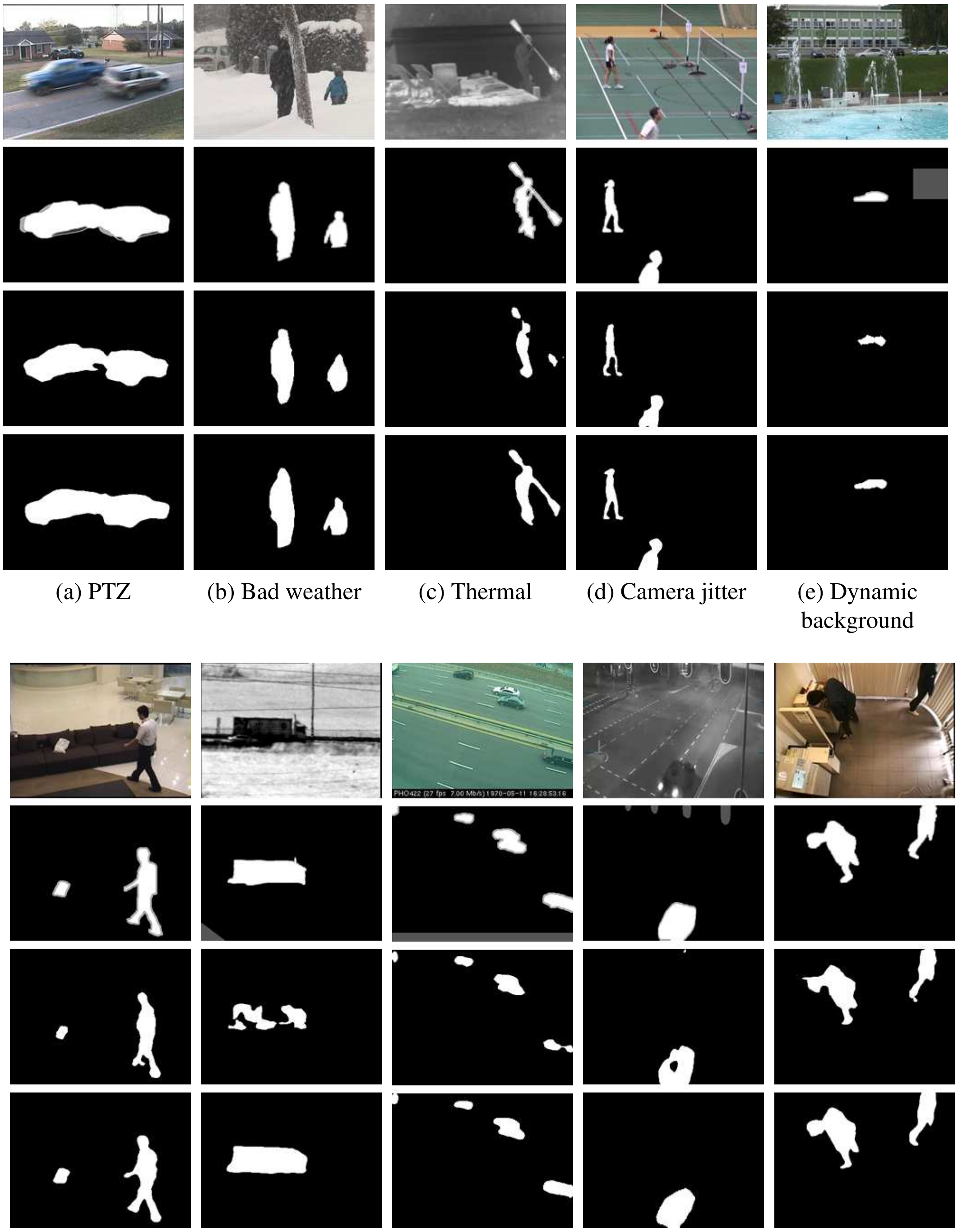
Category	**50 training frames**				**200 training frames**			
	F-measure	**PWC**	**FPED**	**FNED**	**F-measure**	**PWC**	**FPED**	**FNED**
Baseline	0.97	0.19	1.6	0.7	0.99	0.08	2.0	0.4
Dynamic backackground	0.95	0.08	1.9	2.0	0.98	0.03	1.7	1.7
Camera jitter	0.97	0.27	1.2	0.9	0.98	0.15	0.6	0.9
Intermittent Object 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

regions, while our method rarely has this problem. On the other hand, due to the FCN upsampling strategy (please refer to [144] for more details on that), the foreground objects it detected usually have a blobby shape, especially when the foreground and the background have similar color distributions.

Table 3.5 – Results of our method and FCN with different number of training frames.

#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**

We also present in Table 3.6 the results obtained with the top 3 automatic motion detection methods reported on the CDnet 2014 website, namely IUTIS-5 [26] (a method which performs a smart majority vote of several motion detection methods), PAWCS [208] (a non-parametric method), and SuBSENSE [209] (a non-parametric method). As one can see, these results are far less accurate that those obtained by our method in Table 3.2, 3.3, and 3.5.



(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

Figure 3.8 – Results on the CDnet 2014 dataset. row 1: input frames, row 2: the ground truth, row 3: the FCN results, and row 4: the results by our method.

Table 3.6 – Results of the top 3 motion detection methods on the CDnet 2014 dataset.

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

3.4.4 Manual Labeling Accuracy

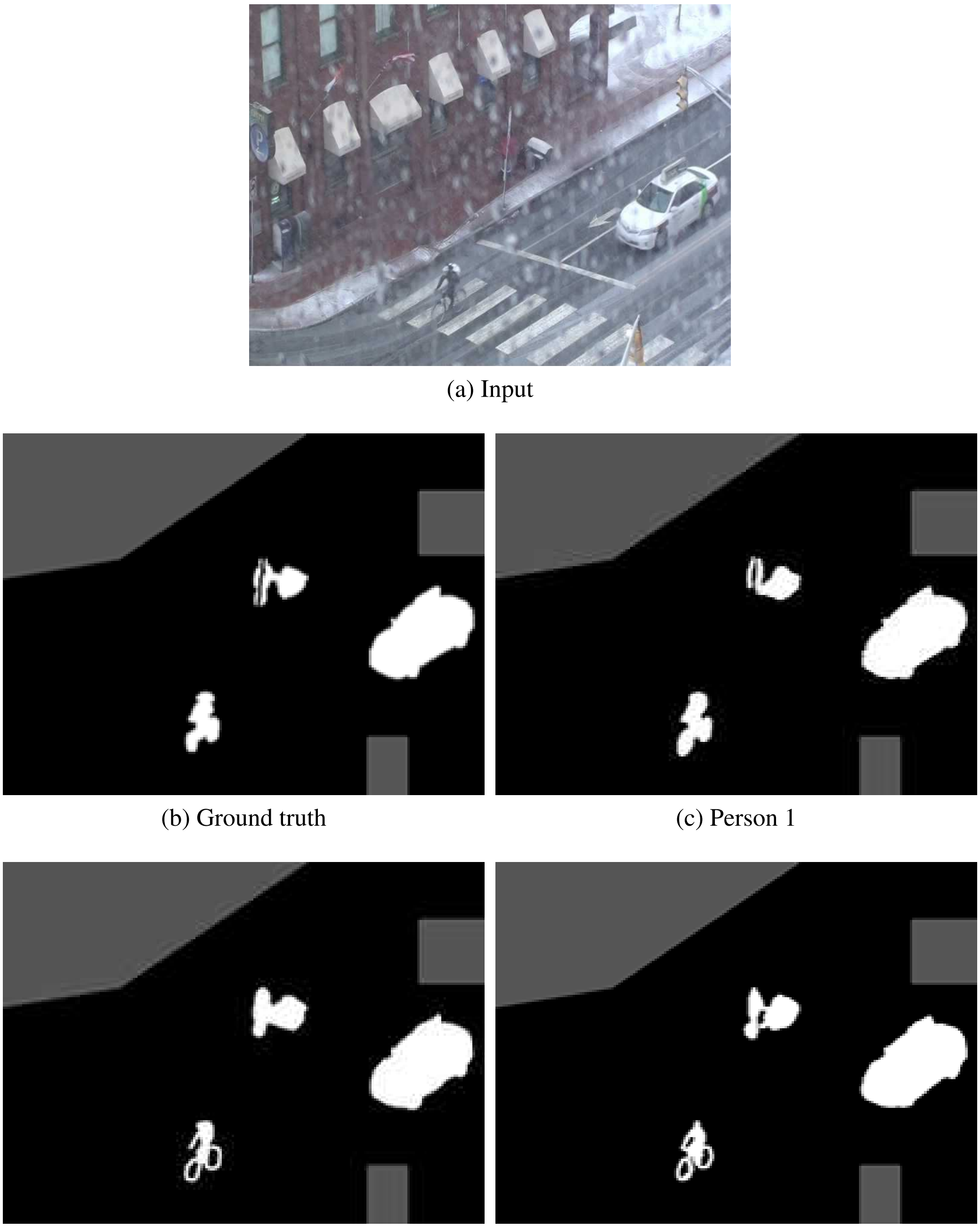
As mentioned at the beginning of the paper, our goal is to produce results sufficiently accurate to be used as ground truth. One may thus conclude that since our model does not reach a F-measure of 1 and a PWC of 0 in Table 3.2, 3.3, 3.4, and 3.5 it is not accurate enough to be used as a reference. With 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 2014 ground truth.

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 2014 dataset and invited three persons to label it. We then compared their results with the CDnet 2014 ground truth (which has also been obtained by a person). Example is given in Fig. 3.9 and quantitative results are in Table 3.7.

Table 3.7 – Results of 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

Interestingly, none of the manually labeled results 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.



(a) Input

(b) Ground truth

(c) Person 1

(d) Person 2

(e) Person 3

Figure 3.9 – Results showing the unavoidable variation between the ground truth and the manual labeling obtained by three independent persons.

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 2014 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 results into a F-measure of 0.94 and 0.93 (*cf.* Table 3.8). This shows again that a method with F-measure of 0.93 and above may be considered almost as good as the ground truth.

Table 3.8 – Evaluation results 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

Let us also mention that our method comes without any 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.

3.4.5 Experiments on the SBI2015 Dataset

We also tested our method on the Scene Background Initialization 2015 (SBI2015) dataset [155] which contains 14 videos. Since this dataset does not contain any pixel-accurate ground truth of background and foreground objects, we manually labeled each video of the dataset. Also, as the SBI2015 videos are relatively short (*e.g.* "Toscana" video contains only six frames), we randomly split each video into 20% frames for training and use the remaining 80% for testing. Due to space limitation, we only report results of our best model (MSCNN + Cascade) in Table 3.9 and also plot some representative results in Fig. 3.10.

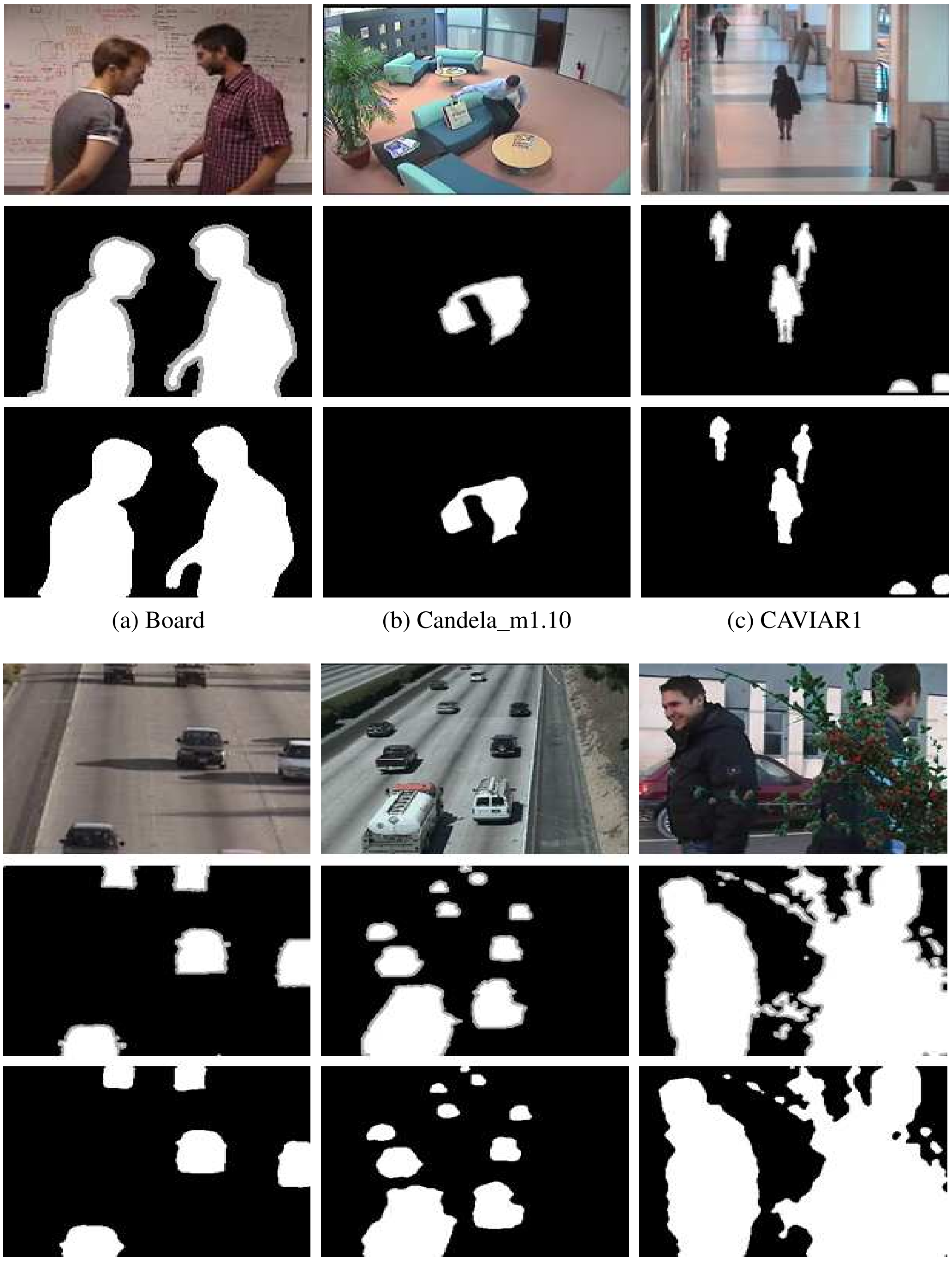
Table 3.9 – Results of *MSCNN* + *Cascade* model on the SBI2015 dataset

Video	**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

As can be seen from Table 3.9, 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.

3.4.6 Processing Time

All the experiments were conducted on a GTX970 GPU with a MATLAB implementation. For a 1,700 frames long video with frame size of 320×240, it takes roughly



(a) Board

(b) Candela_m1.10

(c) CAVIAR1

(d) HighwayI

(e) HighwayII

(f) PeopleAndFoliage

Figure 3.10 – Examples of results of *MSCNN* + *Cascade* model on the SBI2015 dataset. The first row shows input frames, the second row shows the ground truth, and the third row shows the results obtained by *MSCNN* + *Cascade* model.

14 minutes for our *MSCNN + Cascade* model to train 20 epochs with 200 frames and 2 minutes to segment the rest of the video. These 16 minutes are orders of magnitude smaller than the time required to manually label the remaining 1,500 frames. In our experience, on average it takes 30 seconds to label a 320 x 240 frame manually. Our model decreases the time cost to 1/50 comparing with the manual labeling.

3.5 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 2014 dataset and most videos of the SBI 2015 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 models we have tested.
- For a given video, only 50 to 200 frames are needed 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 2014 videos containing 8,000 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 (≈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. 3.11). Fortunately, such situations are relatively infrequent.

— Ground truthing is a subjective task and we showed that a labeling result with a F-measure ≥ 0.94 and a PWC ≤ 0.8 can be considered within the error margin of a human.

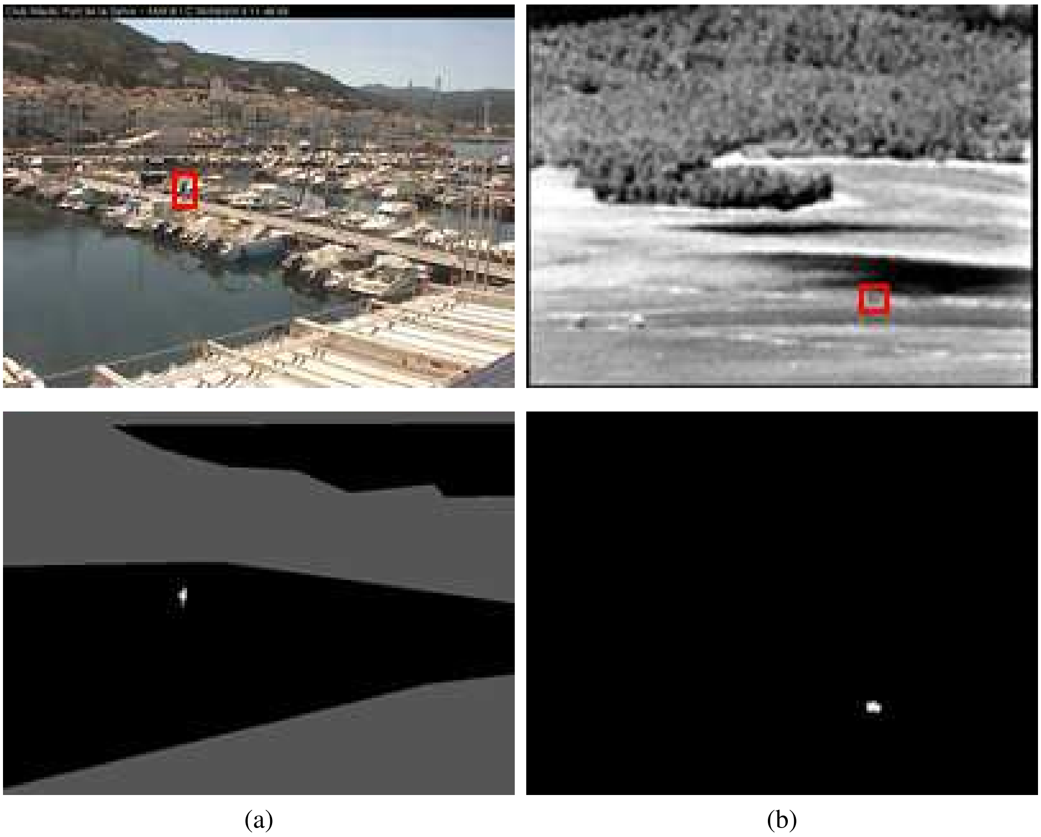


Figure 3.11 – Examples of videos for which our method does not perform well.

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 worse, especially for videos with intermittent motion object. Similar to [67], 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 [92], 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 [124], 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.

Chapter 4

Improving Pedestrian Detection Using Motion-guided Filtering

Résumé

As mentioned before, motion detection can be used to improve the performance of other computer vision tasks, *e.g.*, pedestrian detection. The goal of pedestrian detection is to detect the pedestrians in each frame of a video and highlight them out with bounding boxes. A big challenge for pedestrian detection is that false detections wildly exist because many background objects have a humanoid shape. Objects such as a chair, a fire hydrant, a street light, or just atextured area which happens to have the same features as that of a pedestrian are often wrongly associated to pedestrians. One way to solve this problem is to decrease the decision threshold of a pedestrian detector. However, while reducing the number of false detections, this will also significantly increase the miss rate, which means the real pedestrians may also be missed by the detector. In this chapter, we proposed a motion-guided filter based model. The temporal gradient of foregrounds are extracted and accumulated into motion history image (MHI). MHI is then combined with a nonlinear filter, which is used to filter the background in the video. A feedback loop is also

added to guide the filter. In the end, a merging step is applied to remove more false detections. By considering motion information, our model removes the false positive and thus reduce the false positive rate significantly. This chapter is published as a paper with title **Improving pedestrian detection using motion-guided filtering** [240] in international journal of Pattern Recognition Letter in 2016.

Commentaires

The experiments were mostly designed and implemented by the Ph.D. candidate. The experimental evaluation was also made by the Ph.D. candidate. The article was mostly written and organized by the Ph.D. candidate.

Improving pedestrian detection using motion-guided filtering

Yi Wang
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
yi.wang@usherbrooke.ca

Sébastien Piérard
Montefiore Institute, Université de Liège,
allée de la découverte 10, 4000 Liège, Belgium
sebastien.pierard@gmail.com

Songzhi Su
School of Information Science and Technology, Xiamen University,
Xiamen, Fujian, 361005 China
ssz@xmu.edu.cn

Pierre-Marc Jodoin
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
pierre-marc.jodoin@usherbrooke.ca

Keywords: Pedestrian detection, Video surveillance, Motion history image, Nonlinear filtering

Abstract

In this letter, we show how a simple motion-guided nonlinear filter can drastically improve the accuracy of several pedestrian detectors. More specifically, we address the problem of how to pre-filter an image so almost any pedestrian detector will see its false detection rate decrease. First, we roughly identify moving pixels by accumulating their temporal gradient into a motion history image (MHI). The MHI is then used in conjunction with a nonlinear filter to filter out background details while leaving untouched foreground moving objects. We also show how a feedback loop as well as a merging procedure between the filtered and the unfiltered frames can further improve results. We tested our method on 26 videos from six categories. The results show that for a given miss rate, filtering out background details reduces the false detection rate by a factor of up to 69.6. Our method is simple, computationally light, and can be implemented with any pedestrian detector. Code is made publicly available at: `https://bitbucket.org/wany1601/pedestriandetection`

4.1 Introduction

Despite the number of publications devoted to pedestrian detection, reliable human-shape detection is still a work in progress. Detecting humans is a difficult task since people may take very different poses, be pictured from different viewpoints, and be occluded by objects or other pedestrians. Also, many background objects have a humanoid shape thus leading to false detections. Objects such as a chair, a fire hydrant, or just a textured area which happens to have the same features than that of a pedestrian are often wrongly associated to pedestrians [69, 247]. At the same time, human detectors are fundamentally ambivalent. A sensitive detector (one with a low decision threshold) will detect most pedestrians but at the same time non-pedestrian background objects. On the other hand, a more conservative detector (one with a higher decision threshold) will have a low false positive rate but will suffer from a large miss rate.

In this letter, instead of proposing new features or an improved pedestrian detection classifier, we focus on the images a pedestrian detector is fed with. We propose a motion-guided nonlinear filter whose goal is to filter out background details while leaving intact everything that is likely to be a pedestrian. To achieve this, we compute a motion history image (MHI) [56] at each frame. Since the content of the MHI is highly correlated with moving objects (and thus pedestrians), we apply a Gaussian filter whose standard deviation is proportional to the content of the MHI. By doing so, fixed background objects are blurred out while areas around moving objects are left untouched. We show that the number of false positives in pre-filtered images is drastically lower than in unfiltered images. The reader shall note that although our filter has been validated with pedestrian detectors, it can also be used in conjunction with other kinds of moving object detectors.

Furthermore, a feedback loop is used to update the MHI. This is done by using the predicted pedestrians to update the background image. Our system also fuses results obtained on the original frames as well as on the filtered frames to decrease even more the false positive rate.

The main contributions of this letter are:

- We propose a simple motion-guided filter which improves by a significant amount the performance of off-shelf pedestrian detectors. The filter is independent of the detector and works on a large variety of surveillance videos.
- The motion-guided filter has two novel characteristics. First, it implements a Gaussian filter whose variance is dynamically adapted to the video (*cf.* Section 4.3.2). Second, it benefits from a feedback loop which takes into account the predicted bounding boxes (*cf.* Section 4.3.4).

4.2 Related Work

As of today, top performing pedestrian detectors mostly rely on sophisticated features or discriminative classifiers [94, 172, 173]. At test time, these classifiers output a score indicating how confident they are that a pedestrian is located in the currently-scanned window. What differentiates most pedestrian detectors are the features and the classifiers they use. Although histogram of oriented gradients (HOG) is probably the most frequently-used feature [36, 127], local binary patterns (LBP) [42] and Haar-like features [264] have also been shown effective. Since pedestrians are usually moving, several methods use spatio-temporal features such as binary motion labels [137, 229] and tracking [140, 236]. Other methods use richer features based on specialized hardware such as stereo [19, 142] and infrared features [70, 268, 68]. A trend recently emerged with deep learning where features are learned instead of being handcrafted [199].

The most common classifiers used for pedestrian detection are support vector machines (SVM) [174], AdaBoost [84], Hough forests [75], and deep learning methods such as convolutional neural networks (CNN) [199].

Motion detection is also used for pedestrian detection, [250] uses Gaussian mixture model (GMM) in luma space and temporal saliency map obtained by background subtraction to extract semantic information, which is then used to adjust the pixel-wise learning rate adaptively. In [272], a video is split into spatio-temporal texture patches, in which dynamic texture is extracted. In the end, a conventional GMM is used to separate foreground motion from background image. With an advanced conditional random field model, [132] combined multiple motion and visual saliency induced features, such as shape, foreground/background color models, and visual saliency, to extract the foreground objects in videos. However, all these methods are only focused on motion detection but never extended to pedestrian detection.

4.3 Proposed Method

As shown in Fig. 4.1, our method is a 5-step procedure made of: (i) a background subtraction and MHI computation (Section 4.3.1), (ii) a nonlinear filter (Section 4.3.2), (iii) pedestrian detection, (iv) bounding boxes fusion (Section 4.3.3), and (v) a feedback loop (Section 4.3.4).

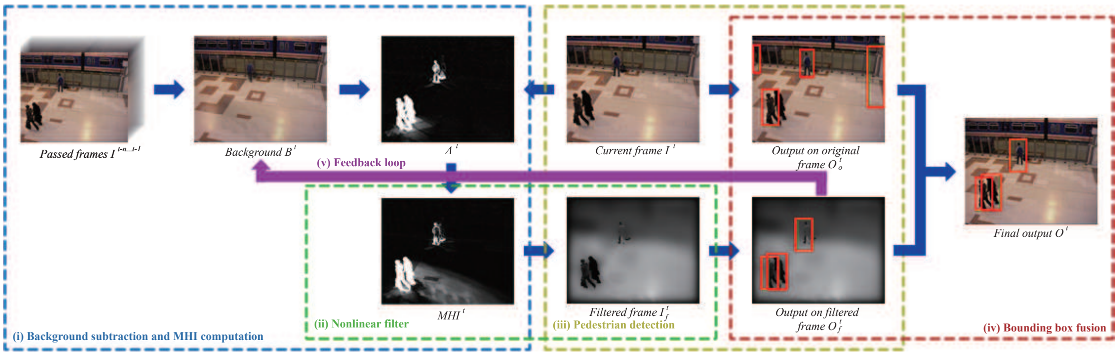

Passed frames $I^{t-n...t-1}$
Background B^t
Δ^t
Current frame I^t
Output on original frame O_o^t
(v) Feedback loop
Final output O^t
MHI ^{t}
Filtered frame I_f^t
Output on filtered frame O_f^t
(i) Background subtraction and MHI computation
(ii) Nonlinear filter
(iii) Pedestrian detection
(iv) Bounding box fusion

Figure 4.1 – Pipeline of our method. At each time t, the current frame I^t and the background image B^t are used to update the MHI^t, which is then used to filter the input image. Pedestrians are detected in the filtered image I_f^t and in I^t. The two resulting sets of bounding boxes are intersected. The ones detected in I_f^t are used to update the background image B^t.

4.3.1 Motion History Image (MHI)

The first step of our method is to identify where moving objects (and thus pedestrians) roughly are. This information will later on be used to filter out background details.

Motion is characterized with a temporal gradient:

$$\Delta^t_{(x,y)} = |B^{t-1}_{(x,y)} - I^t_{(x,y)}|, \tag{4.1}$$

where (x, y) denotes the coordinates of a pixel, $|\cdot|$ is the Euclidean norm in the RGB space, I^t is the video frame at time t, and B^{t-1} the background image at time $t - 1$. Since in this step, the goal is to roughly detect the moving objects, B^t is updated with a running average:

$$B^t_{(x,y)} = \beta^t_{(x,y)} I^t_{(x,y)} + (1 - \beta^t_{(x,y)}) B^{t-1}_{(x,y)}, \tag{4.2}$$

where $\beta_{(x,y)} \in [0, 1]$ is the updating ratio which may be fixed or, as will be shown in Section 4.3.4, adjusted according to a feedback loop. The initial background B^0 is obtained following a temporal median filter on the first 200 frames of the video.

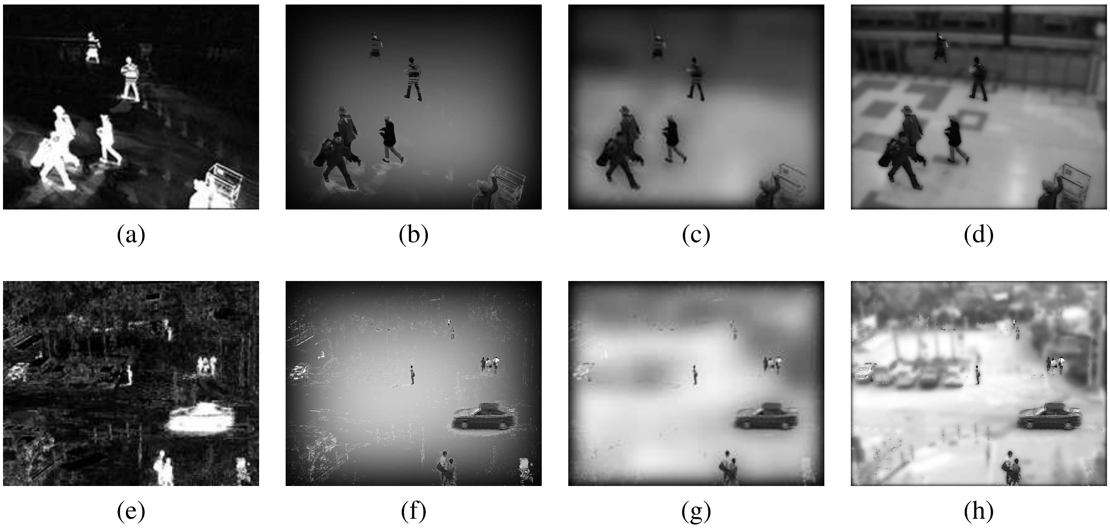
Once the temporal gradient Δ^t has been computed, we cumulate it into an MHI as follows:

$$\text{MHI}^t_{(x,y)} = \max(\Delta^t_{(x,y)}, \alpha\Delta^t_{(x,y)} + (1 - \alpha)\text{MHI}^{t-1}_{(x,y)}), \tag{4.3}$$

where $\alpha \in [0, 1]$ is the MHI updating ratio. MHI^0 is initialized with zero values. The max operator ensures the MHI always contains the latest and largest temporal gradients. In this case, Eq. 4.3 can grasp short bursts of activity caused by fast moving objects. As for the values of α and $\beta_{(x,y)}$, please refer to Section 4.3.4 and 4.4 for how we fix on it.

Note that Eq. 4.3 differs from the original MHI implementation by [56]. First, the use of an α ratio allows to adjust the speed at which the MHI is renewed in time. Second, since we directly cumulate the gradient instead of binary motion maps, there is no detection threshold and thus one less parameter to tune.

Examples of MHI are shown in Fig. 4.2a and Fig. 4.2e. As can be seen, MHI aggregates layers of motion so a pixel value is a function of the recent activity at that position. MHI values are also strongly correlated with the presence of foreground moving objects: the larger a grayscale value is at a given pixel, the more probable a moving object is at that position. As opposed to background subtraction which produces binary maps, MHI contains a much richer set of information, especially in low-contrasted areas. In fact, MHI helps compensating for camouflage problems which happens when sections of a moving object have a low temporal gradient. By cumulating gradients in time, it is likely


(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.2 – (a) MHI example on PETs 2006 video; (b) correlated filtered image with σ_{max} equals $^1/_5$ of the image height; (c) filtered image with σ_{max} equals $^1/_{20}$ of the image height; (d) σ_{max} equals $^1/_{100}$ of the image height. (e) MHI example of the CUHK square video with (f) the filtered image with σ_{max} equals $^1/_5$ of the image height; (g) σ_{max} equals $^1/_{20}$ of the image height; and (h) σ_{max} equals $^1/_{100}$ of the image height.

that a section of the moving object with a larger gradient will eventually compensate for another.

4.3.2 Nonlinear Motion-Guided Filter

As mentioned before, pedestrian detectors often wrongly detect background objects whose features happen to be similar to the ones of a pedestrian. In this section, we propose an MHI-based nonlinear motion-guided filter to decrease the false detection rate. Since the MHI-grayscale values are correlated with the presence of moving blobs, the intuition behind our method is to strongly filter areas with low MHI values and filter less (or not) for areas with higher MHI values. To achieve this goal, we first need to model the likelihood of having a moving blob at time t given the content of MHI^t, *i.e.* $foreground(\text{MHI})$.

We first came out with an empirical model for $foreground(\text{MHI})$. We did so with the CDnet 2014 dataset, the largest publicly-available video dataset with pixel-accurate ground truth. We took 54 videos and computed the MHI for each frame of them. Then $foreground(\text{MHI})$ is computed by counting foreground and background pixels for each MHI value (the MHI value is normalized into [0, 1]). The resulting curve is shown in blue in Fig. 4.3. As can be seen, the chance of having a foreground moving object is almost linearly correlated with MHI values. This is especially true for MHI values larger than $tr_{cut} = 0.2$ while pixels with MHI values below tr_{cut} hardly correlate to any motion. Note that the reason for which $foreground(\text{MHI})$ does not reach 1 is because of motion detection errors, mainly due to camouflage effects (moving objects having the same colors than fix background objects).

Now that the foreground-MHI model has been characterized, we may define our motion-guided filter. We implemented a Gaussian filter $\hat{\mathcal{G}}(0, \sigma)$ which we use to compute a filtered version I_f^t of the input image I^t at time t following a convolutional operation $\otimes$ [1]:

$$I^t_{f_{(x,y)}} = \text{gray}(I^t_{(x,y)}) \otimes \hat{\mathcal{G}}(0, \sigma^t_{(x,y)}). \tag{4.4}$$

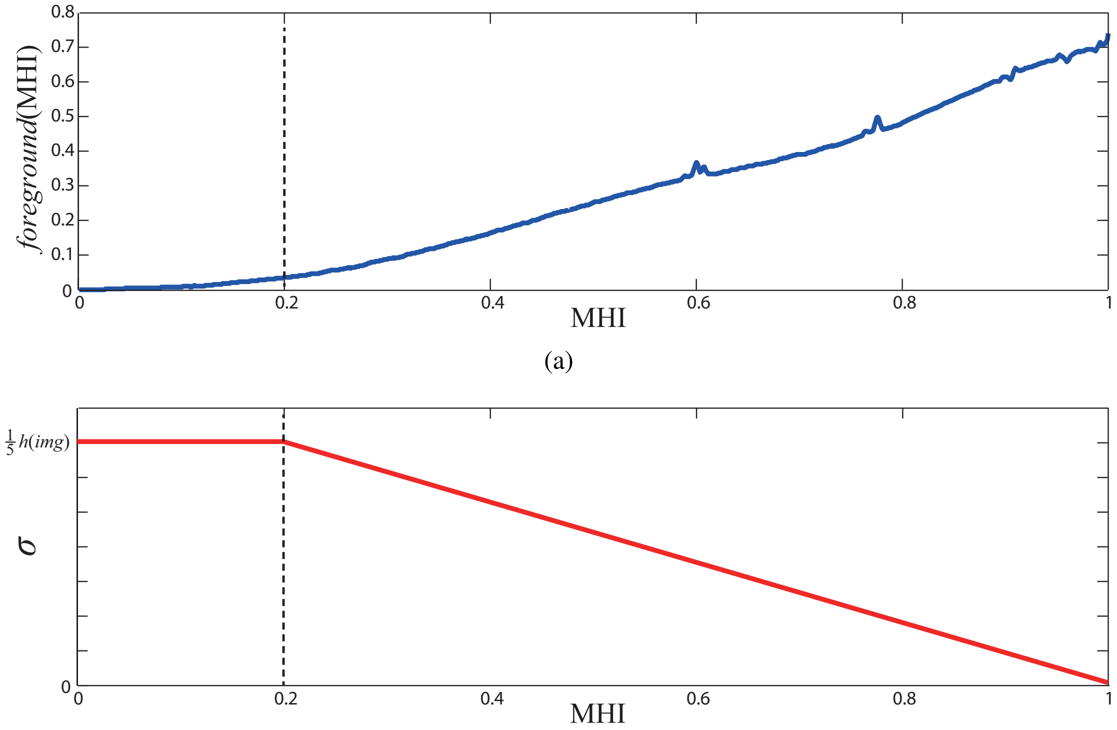
Note that $\hat{\mathcal{G}}$ is not a usual linear Gaussian filter since its standard deviation (std) $\sigma^t_{(x,y)}$ is a function of the likelihood of presence of a moving object, and therefore differs from a pixel to another. According to our model, $\sigma^t_{(x,y)}$ is calculated as:

$$\sigma^t_{(x,y)} = \min(\sigma_{max}, \sigma_{max} \times (1 + s \times (\text{MHI}^t_{(x,y)} - tr_{cut})), \tag{4.5}$$

where $s = {}^{-1}\!/\!_{1-tr_{cut}} = -1.25$ is the slope of the curve in Fig. 4.3b, and σ_{max} is the maximum standard deviation value which is set as $^1\!/\!_5$ of the image height. Having σ_{max} be a function of the image height allows our method to work both on low and high-resolution images. We empirically observed that a Gaussian filter with a std of $^1\!/\!_5$ of the image height is large enough to filter out background details, as too large σ_{max} will decrease the filtering speed, while too small σ_{max} can not remove the background details. Using a appropriate σ_{max} can also maintain the filter size small and thus reduce the filtering time. The compare is shown in Fig. 4.2. Eq. 4.5 is illustrated by the red curve in Fig. 4.3 which is the corollary of the blue curve. The std of our filter reaches the maximum for MHI values below tr_{cut} (those values for which the chances of having

[1] We transform I^t from RGB to gray because most pedestrian detectors work on grayscale images. Furthermore, working on grayscale images reduces processing time.

a moving object are low) and then decreases linearly between tr_{cut} and 1.



(a)

(b)

Figure 4.3 – (a): MHI-$foreground$(MHI) model observed on 54 videos of changedetection.net dataset. (b): The $\sigma^t_{(x,y)}$ model used in this letter, which is determined based on the MHI-$foreground$(MHI) model.

Our filtering procedure comes with one great advantage. As some videos might be suffering from camera jitter, background motion, compression artifacts, and illumination changes, things that may create strong intensities in the MHI. Although these strong intensities are inaccurate as they do not correspond to moving blobs, the only consequence of it is to filter less in these areas, which makes the performance of the pedestrian detection method closer to the one obtained by only processing the original image.

4.3.3 Merging Bounding Boxes

At this point, we have a filtered image I^t_f whose background has been filtered out. That image is then fed to a pedestrian detector (could be any detector, although results might vary from one detector to another). Compared with the results produced on the original image I^t, the result of the same detector applied on I^t_f contains a much smaller number of false detections, as will be shown in Section 4.4.

That being said, bounding boxes obtained with I^t_f are not void of false detections. However, we empirically noticed that false detections in I^t_f rarely correlate with those in I^t. While on the other hand since the nonlinear filter has little or no effect on foreground moving objects, true positives are heavily correlated in I^t_f and I^t. Thus, one way of reducing the number of false positives even more is by keeping bounding boxes that overlap in I^t_f and I^t.

By considering $BB^t = \{bb^t_1, ..., bb^t_n\}$ and $BB^t_f = \{bb^t_{f,1}, ..., bb^t_{f,m}\}$ the bounding boxes obtained with I^t and I^t_f respectively, where bb stands for the bounding box, the resulting bounding boxes returned by our system is a combination of both:

$$BB^t_{final} = \left\{ bb^t_i \in BB^t \middle| \exists j, \frac{bb^t_i \cap bb^t_{f,j}}{bb^t_i \cup bb^t_{f,j}} \geq tr_{merge} \right\}. \tag{4.6}$$

The sole objective of this equation is to keep those bounding boxes in the original frame which overlaps with those in the filtered frame. While this procedure does not affect the number of true positives and false negatives much, it significantly reduces the number of false detections.

Results on the original, filtered and merged results are shown in Fig. 4.4. As can be seen, even though false positive detections still exist in the filtered image (Fig. 4.4b), the false detections are different from those in the original frame (Fig. 4.4a). Keeping the overlapping bounding boxes in these two results into a smaller number of false positives (Fig. 4.4c).

4.3.4 Detection-Guided Model for Background Update

Since the adaptive background model updates the entire frame (*cf.* Eq. 4.2), a pedestrian which stays motionless for some time will be slowly integrated into the background. To fix this problem, we propose a detection-guided background model, for which the background updating ratio β is adapted following a feedback loop based on the detection

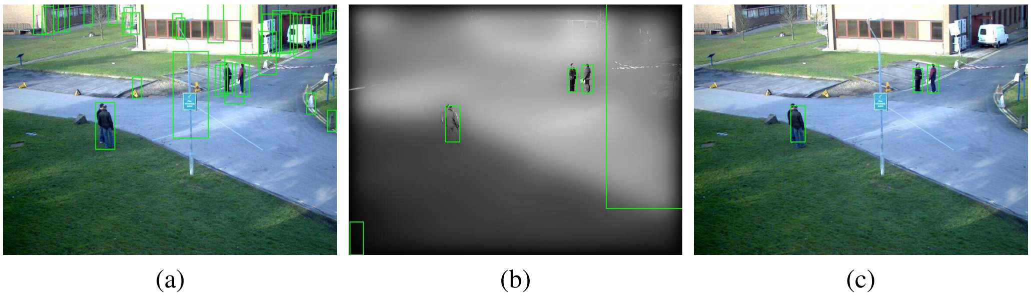


Figure 4.4 – (a) Detection result on the original frame; (b) detection result on the filtered frame; (c) detection result after merging the results from (a) and (b).

results (*i.e.* a lower ratio in areas covered by a pedestrian and a higher ratio elsewhere). This feedback loop is illustrated with the purple arrow in Fig. 4.1. To achieve this, we first use Eq. 4.7 to compute a detecting score map (DSM) for each pixel from the detection results:

$$\mathrm{DSM}^{t}_{(x,y)} = \max\left(0, \min\left(1, \frac{s^{t}_{(x,y)} - s_{min}}{s_{max} - s_{min}}\right)\right) \otimes \mathcal{G}, \tag{4.7}$$

where $\mathcal{G}$ is a 5×5 Gaussian filter with mean and standard deviation (0, 0.5) for smoothing out the DSM, s_{min} and s_{max} are the minimum and maximum confidences for the detector, and $s^{t}_{(x,y)}$ is the maximum confidence at location (x, y) at frame t estimated by the pedestrian detector:

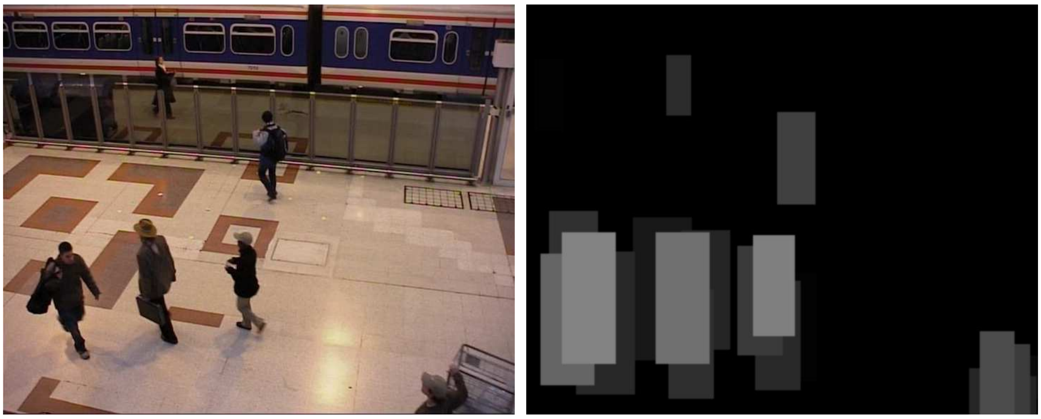
$$s^{t}_{(x,y)} = \max\left\{\text{score}\left(bb^{t}_{f}\right) \middle| bb^{t}_{f} \in BB^{t}_{f}\,, \, (x,y) \in bb^{t}_{f}\right\}, \tag{4.8}$$

where score $\left(bb^{t}_{f}\right)$ is the confidence the detector has that the estimated bounding boxes do contain a pedestrian. That confidence value is typically related to the distance between the feature vector of a bounding box and the decision hyperplane.

Then the adaptive ratio for each pixel is calculated as:

$$\beta^{t}_{(x,y)} = \beta_{max} \times (1 - W_{\mathrm{DSM}} \times \mathrm{DSM}^{t}_{(x,y)}), \tag{4.9}$$

where β_{max} is the maximum updating ratio empirically determined following some experiments (details in Section 4.4), and $W_{DSM} \in [0, 1]$ is the weight for DSM for back-



(a) Input frame

(b) Correlated DSM

Figure 4.5 – The input frame and the DSM calculated based on the frame.

ground updating. An example of DSM is shown in Fig. 4.5. The intuition behind this equation is that the more confident a pedestrian detector is over a certain pixel (x, y), the larger $\text{DSM}^t_{(x,y)}$ will be at that place and thus, the smaller the updating ratio $\beta^t_{(x,y)}$ will be.

The parameters of our method are summarized in Table 4.1.

Table 4.1 – Parameters in our pipeline.

Parameter	**Value**	**Description**
α	0.8	MHI updating ratio
β_{max}	0.016	Maximum background updating ratio
σ_{max}	$^1/_5$ image height	Maximum std of the Gaussian filter
s	-1.25	The slope of the MHI-σ model in Fig. 4.3b
tr_{cut}	0.2	MHI lower than it will not be considered
tr_{merge}	0.5	Overlapping threshold for bounding merging
W_{DSM}	0.9	DSM weight

4.4 Results and Analysis

We tested six different pedestrian detectors namely (1) the Aggregate Channel Features (ACF) by [58] which uses HOG + LUV features with boosting decision trees,

Table 4.2 – Factors by which the miss rate increases and the FPPI decreases after applying our system.

	Baseline		**Small Pedestrians**		**Standstill**		**Different Position**		**Thermal**		**Crowded Scene**	
Detector	miss rate ↑	FPPI ↓	miss rate ↑	FPPI ↓	miss rate ↑	FPPI ↓	miss rate ↑	FPPI ↓	miss rate ↑	FPPI ↓	miss rate ↑	FPPI ↓
ACF	2.1	9.7	1.6	8.1	2.5	8.3	3.2	17.5	3.8	7.4	1.9	4.6
HOG+SVM	1.6	9.8	1.3	19.9	1.8	14.8	1.6	19.6	1.3	17.6	1.1	**1.4**
C4	1.3	8.1	1.4	8.0	2.2	26.0	2.1	**69.6**	2.5	15.6	1.3	2.1
DPM	1.4	8.3	1.4	5.5	1.7	7.9	1.9	19.9	2.8	6.1	1.1	8.3
SPF	1.3	12.3	1.6	4.5	1.9	8.4	2.2	13.4	2.5	5.8	1.1	2.2
DeepPed	3.1	11.3	1.6	13.4	4.3	13.3	4.6	23.6	1.6	4.9	1.6	30.2

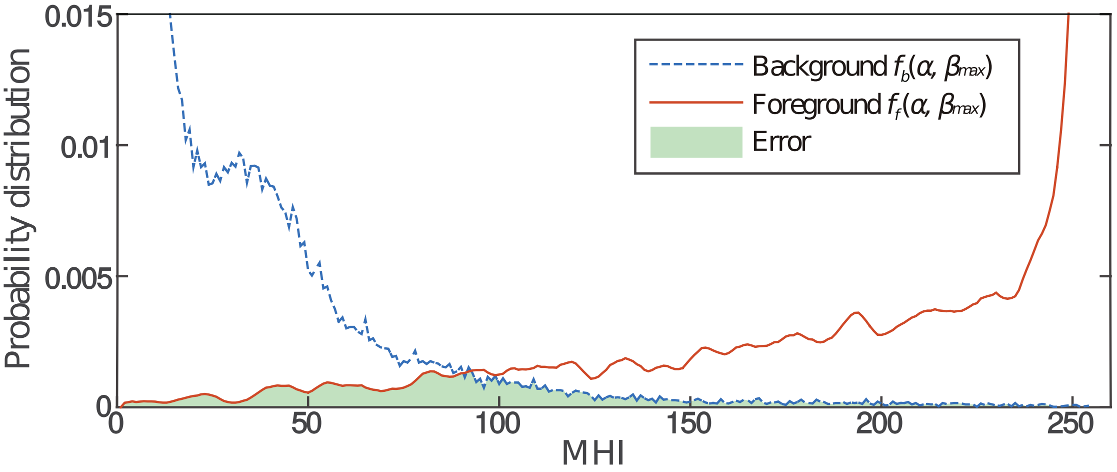
(2) HOG+SVM by [55] which uses HOG feature and SVM, (3) C4 by [247] which uses the CENTRIST visual descriptor and a linear classifier, (4) the Deformable part model (DPM) pedestrian detector by [69], a part-based detection model with modified HOG features and a latent SVM classifier, (5) DeepPed by [218], a deep learning based pedestrian detector, and (6) the Spatially Pooled Features (SPF) by [174], which extracts low-level visual features including motion-based features like us (color (LUV), gradient magnitude, orientation bins, histogram of optical flow, spatially pooled covariance and spatially pooled LBP).

We first determined parameters *i.e.* α and β_{max} for our model. For each pair of α and β_{max} values, we calculated the MHI values for videos in changedetection.net. Given pixel-level ground truth, the distributions of foreground and background over MHI values are calculated as shown in Fig. 4.6a. We then tried to minimize the error which is defined as the overlapping area between these two distributions to determine the α and β_{max} parameters as Eq. 4.10.

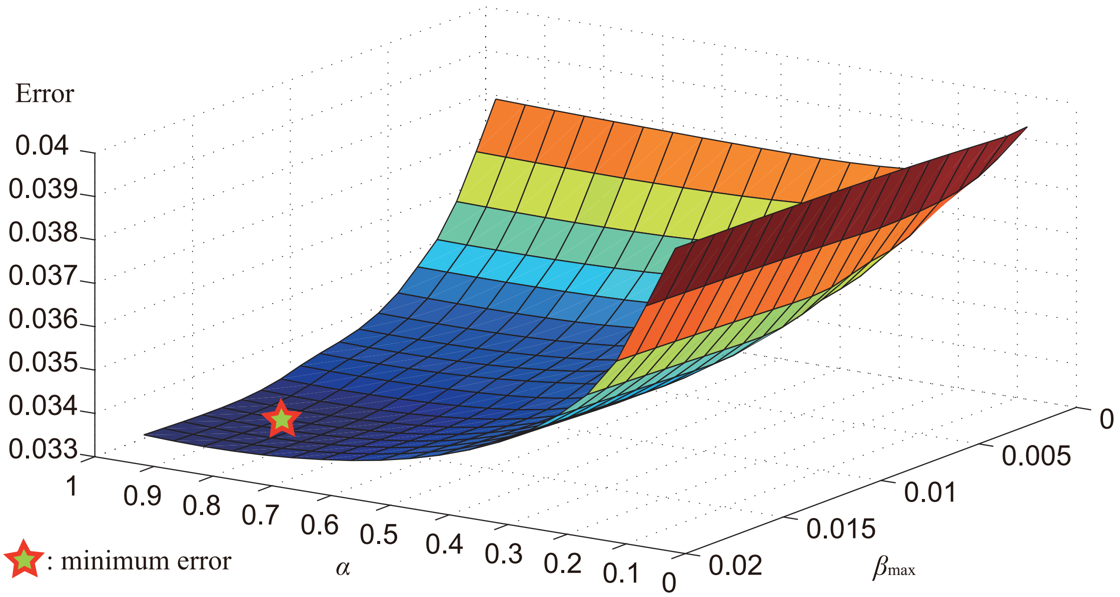
$$\alpha^*, \beta^*_{max} = \underset{\alpha, \beta_{max}}{\arg\min} \int_{x=0}^{255} \min\left(f_f^{\alpha,\beta_{max}}(x),\ f_b^{\alpha,\beta_{max}}(x) \right) dx, \tag{4.10}$$

where f_f and f_b are the distributions of foreground and background, which are calculated by normalizing their respected histograms of MHI values.

As shown in Fig. 4.6b, the error reached its minimum value with $\alpha = 0.8$ and $\beta_{max} = 0.016$, which will be used in Eq. 4.3 and Eq. 4.9. The reader shall also notice that since the error plot is relatively flat near $(0.8, 0.016)$, our model is insensitive to small changes of α and β_{max}. In other words, our method has a smooth behavior when α and β_{max} are changed.



(a) Example of foreground and background distributions ($f_f(\alpha, \beta_{max})$ and $f_b(\alpha, \beta_{max})$) calculated based on changedetection.net for a pair of values (α, β_{max}). The green area is related to the error rate one would have if the MHI value was used to determine if the pixels are in the foreground or the background. This is a quantity we want to minimize.



(b) Error map of MHI for different α and β_{max}. The optimal parameters are $\alpha = 0.8$ and $\beta_{max} = 0.016$.

Figure 4.6 – Procedure used to determine the optimal values for α and β_{max}.

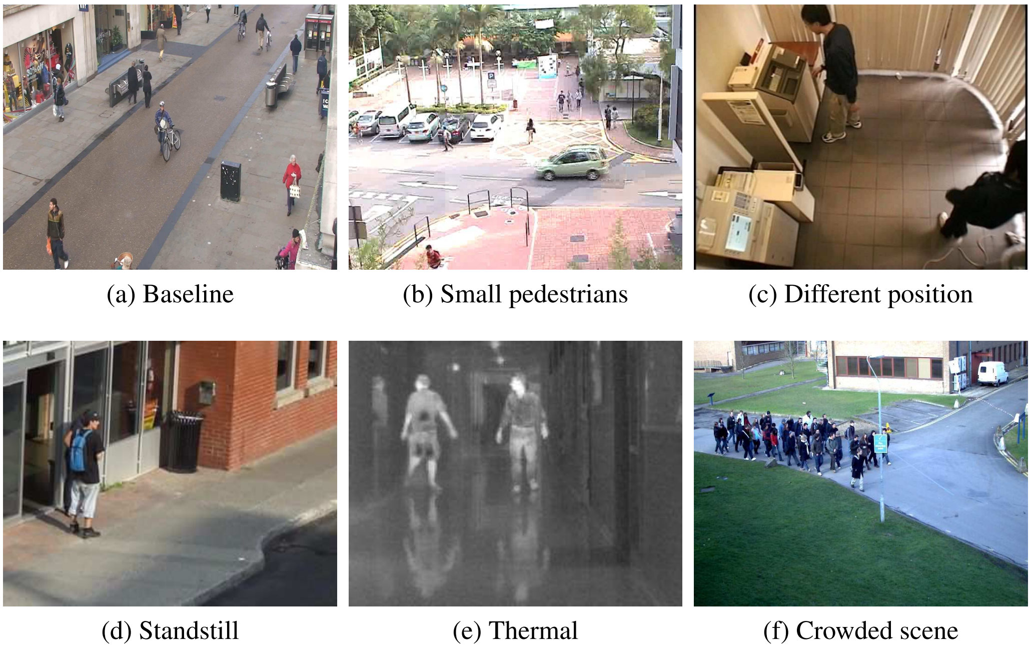

(a) Baseline
(b) Small pedestrians
(c) Different position
(d) Standstill
(e) Thermal
(f) Crowded scene

Figure 4.7 – Six categories for testing: (a) *Baseline*; (b) *Small pedestrians*; (c) *Different Position*; (d) *Standstill*; (e) *Thermal*; (f) *Crowded Scene*.

In order to gauge performances, we tested our method on 26 videos containing indoor and outdoor scenes. These videos come from the Caviar dataset [40], the changedetection.net dataset [237], the CUHK Square dataset [232], the TownCentre dataset [23] and the PETS 2009 dataset. We categorized those videos into six classes: (1) "*Baseline*" which contains videos with common circumstances; (2) "*Small Pedestrians*" which contains pedestrians with a height of roughly 50 pixels; (3) "*Different Position*" which contains pedestrians pictured from a top-down perspective; (4) "*Standstill*" which contains pedestrians that stay motionless for a certain period of time; (5) "*Thermal*" which contains videos shot by infrared cameras and (6) "*Crowded Scene*" which contains groups of pedestrians walking together. Examples of the testing dataset are shown in Fig. 4.7. Let us also mention that 18 of those videos contain either background motion, different types of reflection, strong shadows, and illumination changes.

Typical results obtained with and without our system are shown in Fig. 4.8. These snapshots illustrate how our system can strongly reduce the number of false detections while keeping the true detections almost untouched.

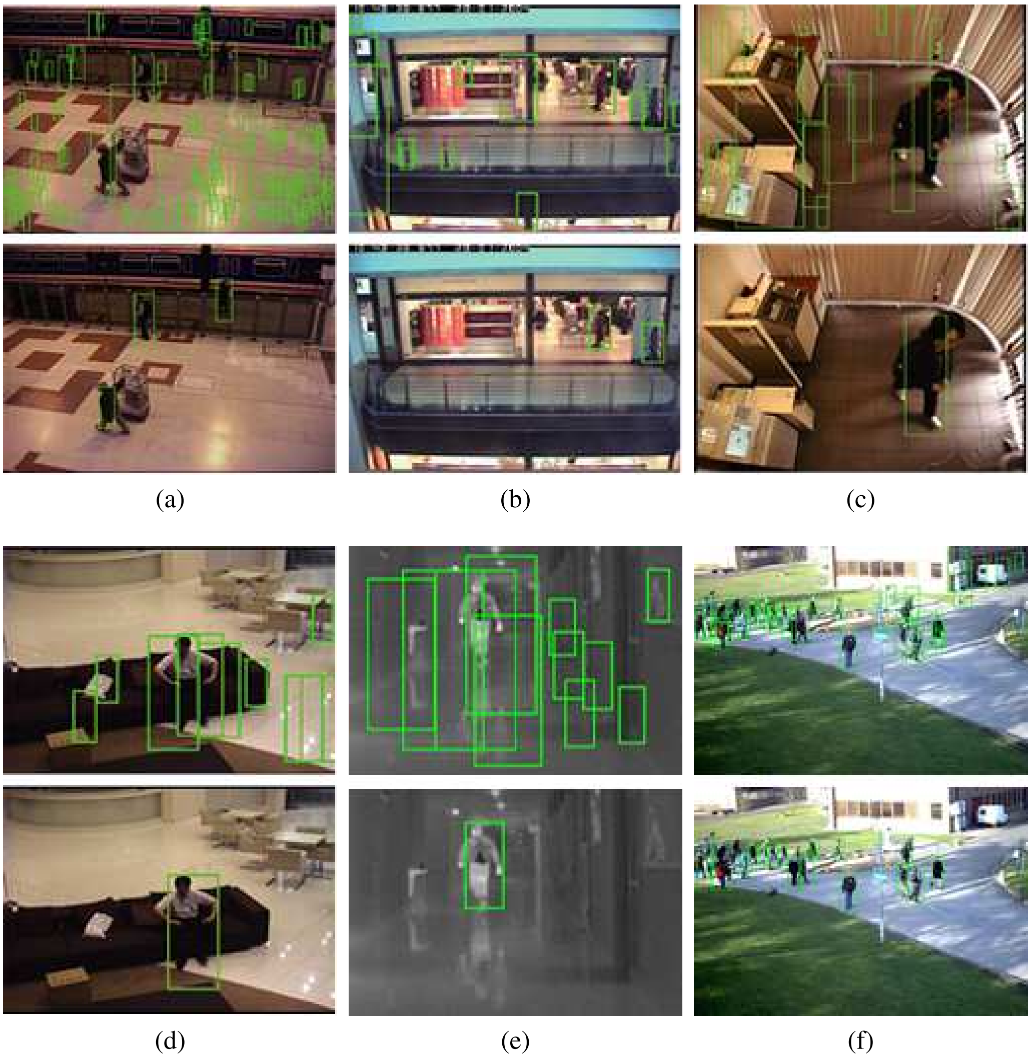


Figure 4.8 – Results obtained without [top] and with [bottom] our system. (a) C4 on a "*Baseline*" video; (b) HOG + SVM on a "*Small Pedestrians*" video; (c) DPM on a "*Different Position*" video; (d) ACF on a "*Standstill*" video; (e) SPF on a "*Thermal*" video; (f) DeepPed on a "*Crowded Scene*" video.

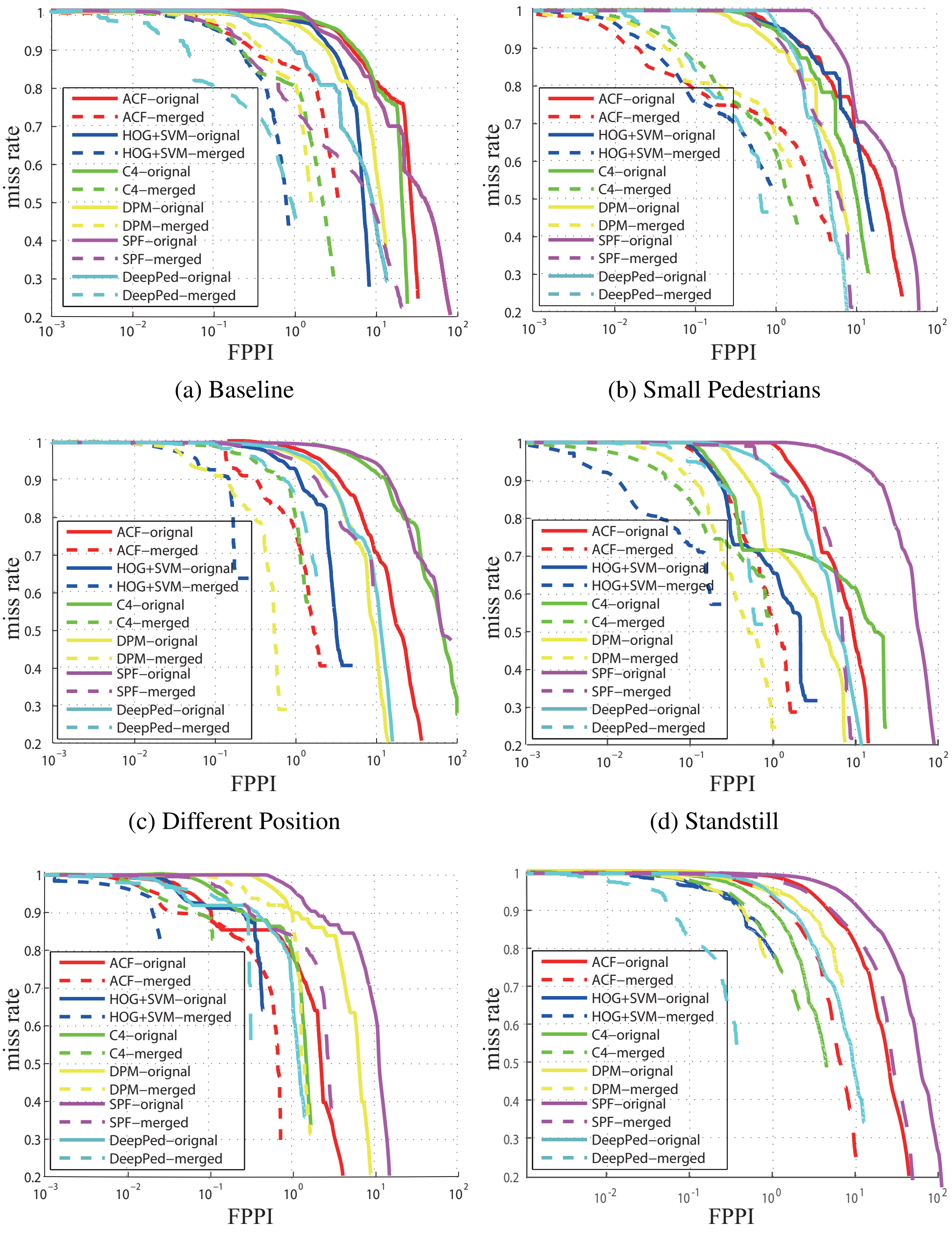
Although a variety of evaluation metrics exists [25], in this paper we report results in terms of *miss rate* and *false positives per image* (FPPI). Please note that the metrics for each category are not obtained by averaging the results from every frame of every sequence. Instead, we extract a subset of N (uniformly spaced) frames per video, where N is the number of frames in the shortest video of the category. All the frames selected in a category are considered as a unique set. This selection allows to avoid biasing the results in favor of longer videos (their lengths vary from 131 to 7,200 frames).

As shown in Fig. 4.9, our system pushes drastically to the left the curves of each pedestrian detector. In Table 4.2, we compare the miss rate and FPPI value before and after applying our system. In this case, each detector use the same set of parameters (including their threshold) when processing the filtered and the unfiltered images. The FPPI ends up decreasing by a factor between 1.4 and 69.6. On average, the FPPI decreases by a factor of 13.0 for each detector, while the miss rate increases by a factor less than 2 on average. It is quite remarkable to see that, for all six tested detectors and all six categories, the targeted FPPI decrease is always higher than the increase of miss rate that is an unavoidable side effect. This again shows how successful our method is at reducing the FPPI without increasing much the miss rate. As can be seen, even the recent DeepPed method which uses a deep learning method and as such, performs very well on its own, sees its FPPI decrease by a factor of up to 21.

Table 4.3 – FPPI values for a fixed miss rate of 0.7.

Detector		Baseline	Small Pedestrians	Standstill	Different Position	Thermal	Crowded Scene
ACF	Original	22.7	11.4	4.8	10.4	1.5	15.9
	Filtered	4.4	1.0	0.9	2.3	0.6	5.0
	Merged	2.3	1.0	0.7	1.2	0.5	4.3
HOG + SVM	Original	5.9	**10.4**	0.7	2.6	0.4	-
	Filtered	1.6	**0.4**	0.4	0.6	0.3	-
	Merged	0.6	0.4	0.2	0.2	0	-
C4	Original	18.5	6.6	2.4	**35.3**	1.2	2.6
	Filtered	3.8	2.0	1.5	5.2	0.1	1.9
	Merged	1.4	0.7	0.4	**1.1**	0.1	1.7
DPM	Original	8.3	4.0	1.3	7.1	4.3	7.1
	Filtered	2.3	1.9	0.6	1.3	1.5	2.6
	Merged	1.2	1.0	0.2	0.4	1.2	0.9
SPF	Original	15.6	8.0	5.9	25.5	6.0	28.3
	Filtered	4.6	1.0	1.5	4.7	1.7	15.2
	Merged	3.5	0.8	1.4	2.8	1.5	14.7
DeepPed	Original	3.4	2.1	1.2	1.6	0.6	4.2
	Filtered	1.4	1.0	0.3	0.7	0.3	0.4
	Merged	0.5	0.2	0.1	0.2	0.2	0.2

Table 4.3 contains FPPI values obtained after processing the original frames, the filtered



(a) Baseline

(b) Small Pedestrians

(c) Different Position

(d) Standstill

(e) Thermal

(f) Crowded Scene

Figure 4.9 – Miss Rate-FPPI curves for six video categories.

frames, and after merging the bounding boxes for a fix miss rate of 0.7. Our motion-guided filter can decrease the FPPI by a factor of up to 26 while our merging procedure can decrease it by a factor of up to 32.

Experiments were conducted on a 2.8GHz Intel computer with a MATLAB implementation. On average, for a 461×615 frame, our system takes approximately 0.03 secs to update the background, compute the MHI and filter the image. Considering the 0.06 sec to detect a pedestrian with HOG+SVM, 0.1 sec with C4, 5.6 sec with DPM, 2.7 sec with SPF, and 1.56 sec with DeepPed (GPU GTX 970), our pre-processing method does not bring a major processing overhead.

4.5 Conclusion

We proposed a system which can be combined with almost any existing pedestrian detector. The core of our method is a motion-guided filter relying on MHI to nonlinearly filter video frames by redfiltering out background details. Our results demonstrate that a motion detection algorithm can be helpful to boost the performance of a person detector, and that a person detection algorithm is valuable to predict the areas in which motion can occur. Our method leverages these two observations thanks to a feedback loop, which is an extension of [238]. The experiments show that our method decreases the FPPI rate drastically without increasing much the miss rate. Our method has been shown successful with four pedestrian detectors on six different video categories. In the future, more features can be considered to be combined in the method to improve the accuracy. We will also extend the model to apply it not only for pedestrian detection, but also for more general detecting tasks, *e.g.* vehicle detection.

Chapter 5

Extensive Benchmark and Survey of Modeling Methods for Scene Background Initialization

Résumé

The most straight forward way to extract motion from a video is the "background subtraction" strategy, which models a clean background without any foreground and compares each frame in the video with it. The significant difference between the video frame and the background model is considered to be the foreground. In this case, how to initialize the background and update it is a key technology. A large number of background initialization methods are proposed, however, the number of background initialization dataset and benchmarking is very limit. As a result, plenty negative implications are caused such as: (1) it is difficult to rationally compare the relative accuracy and robustness of different methods. (2) It is difficult to re-implement a background modeling method and reproduce its results. (3) It is hard to predict how would those methods work when the assumptions they built upon are violated. In this paper we proposed a survey of background modeling methods

and a novel benchmarking framework for background modeling. A carefully analysis of the state-of-the-art methods is also given. This article was accepted as a paper with title **Extensive Benchmark and Survey of Modeling Methods for Scene Background Initialization** to IEEE Transactions on Image Processing 2016.

Commentaires

An workshop and challenge[1] in conjunction with the International Conference on Pattern Recognition (ICPR) 2016 was organized by the other authors. The website of the dataset and the workshop was built and maintained by the Ph.D. candidate. The python evaluation code was written by the Ph.D. candidate. The experiments were mostly run by the Ph.D. candidate, and the paper was partly written by the Ph.D. candidate. The authors of the paper were alphabetically ordered.

[1] http://scenebackgroundmodeling.net/

Extensive Benchmark and Survey of Modeling Methods for Scene Background Initialization

Pierre-Marc Jodoin
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
pierre-marc.jodoin@usherbrooke.ca

Lucia Maddalena
National Research Council,
Institute for High-Performance Computing and Networking,
Via P. Castellino 111 Naples, Italy, 80131
lucia.maddalena@cnr.it

Alfredo Petrosino
Department of Applied Science, University of Naples Parthenope,
Naples, Italy, 80143
alfredo.petrosino@uniparthenope.it

Yi Wang
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
yi.wang@usherbrooke.ca

Keywords: Background initialization, Video analysis, Survey, Benchmarking

Abstract

Scene background initialization is the process by which a method tries to recover the background image of a video without foreground objects in it. Having a clear understanding about which approach is more robust and/or more suited to a given scenario is of great interest to many end users or practitioners. The aim of this paper is to provide an extensive survey of scene background initialization methods as well as a novel benchmarking framework. The proposed framework involves several evaluation metrics and state-of-the-art methods, as well as the largest video dataset ever made for this purpose. The dataset consists of several camera-captured videos that: (i) span categories focused on various background initialization challenges; (ii) are obtained with different cameras of different lengths, frame rates, spatial res-

olutions, lighting conditions, and levels of compression; (iii) contain indoor and outdoor scenes. The wide variety of our dataset prevents our analysis from favoring a certain family of background initialization methods over others. Our evaluation framework allows us to quantitatively identify solved and unsolved issues related to scene background initialization. We also identify scenarios for which state-of-the-art methods systematically fail.

5.1 Introduction

Initializing and updating a scene background model is of paramount importance for a large number of applications, such as motion detection, object counting, crowd monitoring, action recognition, and video segmentation [200, 187, 263, 265]. As such, a quick search for "background initialization" and "background modeling" leads to hundreds of scientific publications. New methods are being devised for addressing well known issues related to background initialization, such as illumination changes, intermittent object motion, background motion, low-frame rate, and highly cluttered videos [81]. Recent surveys have been published on background modeling [253, 201] as well as on motion detection [28, 54], and well settled initiatives have been carried out in order to evaluate the existing background modeling methods for the purpose of foreground detection [220, 81]. However, the initialization aspect of background modeling is often neglected, assuming *emptiness* hypotheses on initial video frames. Background initialization is still necessary, not only for the (usual) case those hypotheses are not verified, but also in further applications, such as video compression [180], video inpainting [51], privacy protection for videos [163], and computational photography [3], where the availability of an image of the background free of moving objects is fundamental.

Also known as "video bootstrapping", "background estimation", "background reconstruction", "initial background extraction", and "background generation", the goal of initializing a background model may be defined as follows: given a temporal sequence of images taken by a static camera showing a background scene with foreground objects on top of it, the aim is to recover a model that is able to provide an image of the background without the foreground objects. The background image may be valid for the entire video or updated in time in case the background configuration changes over time due to illumination changes or to the displacement of background objects.

Over the past two decades, many techniques have been proposed for background initialization, but only initial efforts have been made to propose a benchmarking framework devoted to an objective assessment [150, 155]. The lack of a comprehensive evaluation framework has a number of negative implications. First and foremost, without a common evaluation ground, it is difficult (if not impossible) to rationally compare the relative accuracy and robustness of different methods. This results into a large number of methods all claiming robustness to their own data, but with no clear understanding on which approach is best suited to a given scenario. Second, since authors rarely share code and often use their own data (which are not always publicly available), it is difficult to re-implement a method and reproduce its results. Indeed, the implementation of a given method, as well as the correct tuning of its parameters, may vary significantly from one developer to another. This results into papers comparing their methods with simple (and yet easily re-implementable) methods and implies the alienation of more complex state-of-the-art approaches. Third, while some algorithms perform well in some scenarios, it is very hard to predict how those methods work when the assumptions they are built upon are violated. As such, algorithms often tend to overfit specific scenarios and their generalizations to other videos are hardly predictable. For example, a background initialization method robust to background motion might not necessarily work well on highly cluttered videos, and vice versa.

Recognizing the importance of background initialization for the video analytics community, we believe that an extensive evaluation framework containing well-known evaluation metrics, as well as an exhaustive dataset containing different scenarios, would go a long way towards providing an objective assessment. In this perspective, we gathered a scene background modeling benchmarking framework, that the scientific community can access via the URL `http://SceneBackgroundModeling.net` (SBMnet) and organized the ICPR 2016 Scene Background Modeling (SBM) Challenge. The framework includes 79 videos distributed into eight different categories, each representing a specific challenge. Each video comes with two scene background models to be adopted as ground truth: one (or more) background image(s) and a pixel-wise non-parametric model. These ground truths allow an extensive evaluation of various methods, in terms of seven metrics. Being completely automatic, the online evaluation system allows researchers to upload their results and compare their methods with others.

The main contributions of this work are as follows:

1. Study several evaluation metrics and measure how good these metrics are at ranking models for background initialization.
2. Propose an automatic online system for ranking scene background initialization methods.
3. Propose the largest dataset ever made for gauging performance of scene background initialization methods and, from there, identify solved and unsolved issues.
4. Provide an evaluation framework for background modeling complimentary to those devoted to foreground detection.

The reminder of this paper is organized as follows. In Section 5.2 we review the state-of-the-art background initialization methods. In Section 5.3, we describe details of the SBM challenge, including the dataset, the metrics, the ranking strategy, and the probability density function model we use. Experimental results are shown in Section 5.4, while Section 5.5 draws the conclusion.

5.2 State of the Art

Background initialization methods can be classified according to different aspects [33]. Useful insight can be gained by considering the way they model the inter-pixel relationship.

Pixel-level methods: These are among the most widely implemented methods whose main characteristic is to process each pixel independently. Examples of such methods include the temporal mean [156], the temporal median [130, 143], and the temporal histogram [269], eventually corroborated by a reward/penalty mechanism in its update [48, 52]. To improve the simple temporal median filter, Wang and Suter [227] propose a RANSAC-based method that tolerates more than 50% outliers. Kim *et al.* [117]

propose a method that combines pixel-level edge difference and brightness difference to model the background. Even though straightforward, these methods have their own limits. Indeed, since they ignore spatial relationships among pixels, these methods are sensitive to "ghosting" artifacts (*i.e.*, when the estimated background contains parts of foreground objects) and other issues that require higher-level analysis, such as local or global illumination changes.

Region-level methods: These approaches take advantage of inter-pixel relationship through Markov random fields (MRF) or conditional random fields (CRF), or by subdividing the images into non-overlapping regions. MRF- and CRF-based methods involve a two-term energy function, often optimized by graph-cut [49, 44] or loopy belief propagation [252]. In [49], the data term accounts for pixel color stationarity and motion boundary consistency, while the interaction term looks for spatial consistency in the neighborhood. In [44], the data term is made of two parts: a stationary pixel color term and a predicted term for stable pixels obtained using an image inpainting technique. The method by Xu *et al.* [252] is similar to [49], although it implements a simpler data term and a different optimizer. The Photomontage [3] method also involves a MRF graph-cut optimizer. However, it first extracts regions of pixels which exclusively contain the scene background. The energy function is then constructed at the level of those regions. The work by Lin *et al.* [139] uses a classifier to determine background blocks and then updates the background image with it. Note that region-level methods usually have a higher computational complexity than pixel-level methods.

Hybrid methods: These approaches operate at both pixel- and region-level, and thus provide a compromise between efficiency and accuracy. For example, Wallflower [219] uses a pixel-wise Wiener filter [243] to estimate a background image and then refines it with a region-level and a frame-level method to avoid intermittent motion and illumination change problems. Colque and Cámara-Chávez [52] propose a hybrid variation of the method of [48], where the temporal histogram is updated region-wise. Nonaka *et al.* [168] also implement a multi-level background estimation method. Here, a pixel-wise probability density function (PDF) is estimated by using a Parzen Density Estimation [179]. The spatial similarity is considered at the region level and the brightness is normalized at each frame.

Further insight on background initialization methods can be gained by grouping these methods based on shared methodologies.

Subsequences of stable intensity: These methods come with a two-phase structure. Based on the assumption that a background pixel is one with a stable RGB color over time, a set of non-overlapping temporal subsequences with similar color values ("stable subsequences") is first selected for each pixel or image region. Then, the best subsequence is selected according to some criterion and its average color is used as background color. In [86], the authors separate a video into subsequences with maximum length of six frames for each pixel. The best subsequence is selected according to a maximum likelihood criterion. In [227], after locating the non-overlapping stable subsequences for each pixel, the best subsequence is chosen with the highest reliability as defined in [72]. In [176], the most likely stable subsequence is selected in the subinterval in which co-located regions are statistically similar, and the median is applied pixel-wise.

Iterative model completion: These methods construct the background image in an iterative manner. At first, they identify areas in the video where no activity has been detected. These areas are copy-pasted and serve as background initialization. From there, the background model is iteratively completed based on criteria that vary from one method to another. In [14, 95], video frames are first split into blocks and the stable ones are labeled as background. The missing blocks are filled by those whose frequency spectrum is coherent with the neighboring background. In [50], patches subdivided from a video are temporally clustered. The background is recursively grown by selecting the patch which provides the best continuation of the current background. In the block-level recursive technique proposed in [190], for each block location of the image sequence, a representative set is maintained which contains distinct blocks obtained along its temporal line. The background initialization is recasted as a MRF labeling problem, where the clique potentials are computed based on the combined frequency response of the candidate block and its neighborhood. It is assumed that the most appropriate block results in the smoothest response, indirectly enforcing the spatial continuity of structures within a scene. In [171], the authors temporally cluster background blocks as candidates for each location and background seeds are spatially initialized. The background model is then estimated iteratively from the seeds by considering both inter-block and intra-block smoothness constraints.

Missing data reconstruction methods: These approaches are those for which the area covered by foreground objects is considered as missing background information that shall be recovered. For example, [51] considers the background initialization process as an instance of video inpainting, aiming at eliminating from the sequence all the foreground objects (considered as holes to be filled in) using the remaining visual information to estimate a statistics of the entire background scene. In [205], the problem is formulated as a matrix completion task, later extended to tensor completion [206], where the image sequence is revealed as partially observed data. Missing entries are induced from the moving regions based on motion detection, and their reconstruction can be achieved by any matrix or tensor completion method. Also robust principal component analysis (RPCA) can be adopted for decomposing a matrix composed by the observed video frames into a low-rank matrix (the scene background) and a sparse matrix (the scene foreground) [30]. Based on this idea, a RPCA-based motion-aware regularization of graphs on the low-rank component is proposed in [103, 104] in order to better handle background variations.

Neural networks: These methods formulate the problem of background initialization as an unsupervised or supervised classification problem. A method for background estimation based on a neural network model previously adopted for change detection is proposed in [41]. The system analyzes video information and detects video scenario situations, classifying the video into four different modules to make appropriate parameter adjustment according to those situations. Learning rates for each pixel are automatically computed according to the results of the two parallel neural networks and of the video classification module. A neural background model based on self-organization is exploited in the SC-SOBS method for static [154] or PTZ [71] cameras, that well adapts to background initialization [151]. A method based on a convolutional neural network (CNN) is proposed in [88]. The CNN starts with a "contractive" stage (a series of convolutions) followed by a "refinement" stage (a series of deconvolutions), so that the output of the net has the same size than the input. Since the authors implement a L2 loss, their CNN performs a regression instead of a classification, as is usually the case for such networks.

Online and Offline methods: Several background initialization methods implement online algorithms which process a video frame by frame, without going back in time [156, 190, 57, 41]. Online approaches are well-suited for devices that cannot store more than a few frames at the same time, such as IP cameras. Ironically, one advantage of these methods is also their main limitation. While these methods always end up incorporating changes that happen in the background (*e.g.*, illumination changes that occur through the day), they may also wrongly include into the background slowly moving objects or objects which stay motionless for some time before they move away, such as cars waiting at a red light. These are the so-called *intermittent object motion* artifacts [81]. Offline algorithms compute the background image by considering the entire video as a whole [219, 53, 143]. Methods relying on a temporal median filter [130] usually fall into this category, as well as eigenbackground methods that use principal component analysis to capture the main component of the video [169, 28], or genetic algorithm methods, where survival-of-the-fittest and genetic evolution are used to search better background candidates [116]. Offline methods often have memory issues when the video is long and are generally slower than online methods.

We point out that several motion detection methods based on background modeling have been published over the years (extensive surveys can be found in [31, 28, 29], and [81]). Many of these methods involve a pixel-wise probabilistic model, such as a Gaussian mixture model (GMM) [211], a mixture of general Gaussians [8], or a kernel density estimation [63, 90]. Some methods implement a Bayesian spatio-temporal model of both the background and the foreground [202, 165, 166]. Others, like "Vibe" [15] and PBAS [93], implement a non-parametric and stochastic strategy to model each background pixel with a random subset of pixel values from the recent past. Although these methods are very good at modeling temporal variations of the background, they are nonetheless ill-suited for generating a single RGB background image, and thus in the following they are not considered for comparison.

5.3 Benchmarking Framework

5.3.1 Dataset and Video Categories

The proposed SBMnet dataset provides a realistic and diversified set of 79 videos [2] coming from our personal collection as well as from public datasets, which we referred to in Table 5.1. In order to prevent the dataset from having a bias towards a category of methods, the videos come from different cameras, including IP cameras, web cams, and DV cams, all with a different compression level, resolution, and frame rate. While some videos are computer-generated, most of them come from real surveillance cameras located indoor and outdoor, showing day-time and night-time scenes of traffic, pedestrians, and wild life.

Table 5.1 – Publicly available datasets from which the SBMnet videos come from.

Acronym	Related references	Web site
ATON	Prati *et al.* [186]	http://cvrr.ucsd.edu/aton/shadow/index.html
BMC2012	Vacavant *et al.* [220]	http://bmc.iut-auvergne.com
CDNET	Goyette *et al.* [80] Wang *et al.* [237]	http://wordpress-jodoin.dmi.usherb.ca/dataset2012/ http://wordpress-jodoin.dmi.usherb.ca/dataset2014/
CIRL	Anderson *et al.* [10]	http://www.derektanderson.com/fallrecognition/datasets.html
CMU	Sheikh *et al.* [202]	http://www.cs.cmu.edu/~yaser
EPFL	Fleuret *et al.* [73]	http://cvlab.epfl.ch/data/pom
Fish4Knowledge	Kavasidis *et al.* [113]	http://groups.inf.ed.ac.uk/f4k/
ICRA2010		http://www.cs.utexas.edu/~changhai/icra10-datasets/datasets.html, no more available.
IPPR2006		http://media.ee.ntu.edu.tw/Archer_contest/, no more available.
LASIESTA	Cuevas *et al.* [54]	http://www.gti.ssr.upm.es/data/LASIESTA
LIMU	Yoshinaga *et al.* [258]	http://limu.ait.kyushu-u.ac.jp/dataset/en/
MIT	Wang *et al.* [235]	http://www.ee.cuhk.edu.hk/~xgwang/MITtraffic.html
PETS2001		ftp://ftp.pets.reading.ac.uk/pub/PETS2001/
SABS	Brutzer *et al.* [37]	http://www.vis.uni-stuttgart.de/index.php?id=sabs
VSSN2006		http://mmc36.informatik.uni-augsburg.de/VSSN06_OSAC/, no more available.
UCF	Ali *et al.* [5]	http://crcv.ucf.edu/data/

The videos span eight categories, selected to include diverse scene background initialization challenges:

— **Basic:** the category represents a mixture of mild challenges typical of the *background motion*, *camera jitter* and *intermittent object motion* categories. Some videos have subtle background motion, others contain bad weather, some show

[2] All the videos are available through the SBMnet webpage http://SceneBackgroundModeling.net.

blurry underwater scenes. That being said, none of those challenges is very significant. These basic videos have frame rates of more than 20 fps and contain a small amount of well-contrasted foreground objects moving fluidly through the video. These videos are provided mainly as a reference on which every method should perform well.

— **Intermittent motion:** this category includes videos with scenarios known for causing "ghosting" artifacts in the estimated background. These might be caused by objects that move, then stop for a short while, and then start moving again. Some videos also include abandoned objects or background objects that suddenly start moving, *e.g.*, people that stand still for most of the video and then leave the scene. This category is intended for testing how various algorithms adapt to scenarios in which objects move in a non-fluid manner.

— **Clutter:** videos in this category contain a large number of foreground moving objects occluding each other as well as a large portion of the background. In this case, clutter is either caused by a large amount of small objects (*e.g.*, fishes, cars, and pedestrians) or by a small amount of large objects (*e.g.*, persons or foliage located right in front of the camera) occupying a large portion of the visual field. The main challenge of this category is to cope with videos whose pixels are often occupied by foreground objects more than 50% of the times, *i.e.*, above the threshold that can be tolerated by methods based on temporal median filtering.

— **Camera jitter:** this category contains indoor and outdoor videos captured by unstable cameras. The jitter magnitude varies from one video to another. Jitter is caused by wind or vibration due to an engine located close by or by a handheld camera. This can be an issue for pixel-level methods that are unable to take into account spatial information available in the neighborhood of each pixel.

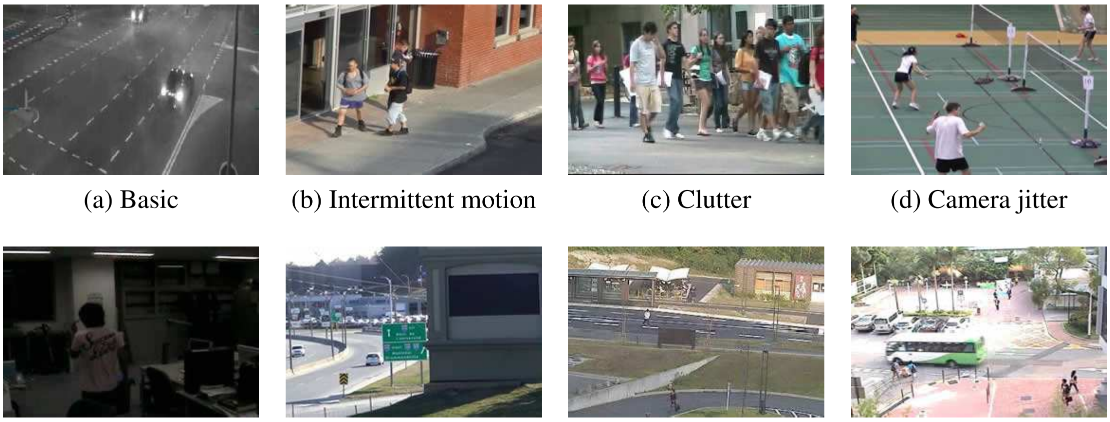
— **Illumination changes:** this category includes indoor and outdoor videos that contain mild or strong illumination changes due to light switching, curtains opening, automatic camera brightness adjustment, or varying shades when the clouds alter the sunlight. The main challenge with this category comes from the fact that the background evolves over time and thus any method relying on some kind of temporal median filter will end up smudging light-shaded and dark-shaded portions of the videos and producing a corrupted solution. This is also an issue for methods devoted to non-temporal image sequences (*e.g.*, for computational photography [82]), where the temporal order of input frames is not taken into account, and thus cannot temporally compensate illumination variations.

— **Background motion:** this category includes videos with strong and parasitic background motion. These videos contain boats on shimmering water, cars passing next to a fountain, and pedestrians passing near trees shaken by the wind. This category also includes a video with a low frame rate background motion caused by a flipping advertisement board. This category is intended for discriminating methods that have strong assumptions on the stationarity of the background [86, 190, 226, 252].

— **Very long:** videos in this category contain more than 3,500 frames. Like the *Basic* category, these videos do not contain any specific challenge as far as their content is concerned. However, their large size is meant to discriminate online and offline methods, as well as methods that are particularly expensive processing-wise and memory-wise.

— **Very short:** these videos contain a limited number of frames (less than 20) with a low frame rate (less than 1 fps). The goal of this category is to discriminate methods that require a large number of training frames or methods which assume that the video frame rate is large enough to track foreground moving objects or to compute the optical flow.

Note that every sequence in the dataset contains at least one foreground object. In this way, no video contains its own solution, *i.e.*, a background image without moving objects. At the same time, we ensure that the scene background is revealed at least once for each pixel in all videos. This condition allows all background initialization methods to use only observed values to fill the background, as opposed, for example, to video inpainting [51], where "plausible" values are used for filling-in the background image. Moreover, some of the surveillance videos have been cropped in order to eliminate the timestamps located at the top or the bottom. Example frames of the SBMnet dataset are shown in Fig. 5.1.



(a) Basic (b) Intermittent motion (c) Clutter (d) Camera jitter

(e) Illumination changes (f) Background Motion (g) Very long (h) Very short

Figure 5.1 – Samples from the SBMnet dataset.

5.3.2 Ground Truth Image and Model

The SBMnet dataset has been the bedrock of the ICPR 2016 SBM Challenge, where groups from all over the world have been invited to compare their background initialization methods. For that challenge, each video had one (or more) ground truth (GT) color background image(s)[3] that we used to compute six well known quality metrics (*cf.* Section 5.3.3).

In order to create the GT images, we applied the following Background Ground Truth (BGT) procedure:

1. For every frame of every video, we produced the binary foreground mask outlining every foreground object. This has been done either manually, or, in case of videos containing a too large number of foreground objects, by applying a background subtraction algorithm and post-processing the results. In some cases, we had to copy regions from different frames and paste them through an image manipulating system.
2. We constructed the GT color background image, by accounting at each pixel for the largest background color mode.

[3] A small proportion of the GT images has been made available at http://SceneBackgroundModeling.net.

For some sequences, more than one GT color background image was constructed, to take into account scene backgrounds which vary over time. This was typical for videos with strong illumination changes (*e.g.*, one GT image for the light-shaded portion of the video and one for the dark-shaded portion), videos with strong background motion (*e.g.*, various GT images showing different positions of background trees shaken in the wind), as well as videos with strong camera jitter. For those cases, the use of multiple GT images prevents from having temporally smudged images and allows to account for several background configurations.

The main advantage of using GT background images is that they comply with most image-based evaluation metrics, such as PSNR and MS-SSIM. However, the outcome of the ICPR 2016 SBM Challenge made us realize that the use of a single (or a small set of) background image(s) has its limits. Indeed, some videos have a nearly infinite number of valid background images, all of which being slightly different from the GT background image(s). A good example is a scene with wavy water in the background. Since no finite series of images may account for every possible configuration of waves, we also considered using a probabilistic background model. This probabilistic model is a conditional PDF $p(c_z|\mathbf{B}_z)$ which is the likelihood of observing a generic RGB color c_z at pixel z, given a collection of background colors $\mathbf{B}_z$ recorded at location z.

For a given video sequence made of k frames $I_1, \ldots, I_k$, the set $\mathbf{B}_z$ of background RGB colors observed at location z is obtained exploiting the binary foreground masks constructed for each I_j, $j \in [1, k]$. Thus, $\mathbf{B}_z$ is a series of at most k RGB values all associated to background pixels.

The likelihood of the background model for each pixel at location z may be estimated with a simple histogram

$$p(c_z|\mathbf{B}_z) = \frac{n_{b,c_z}}{||\mathbf{B}_z||}, \tag{5.1}$$

where n_{b,c_z} is the number of background pixels with color c_z recorded at location z and $||\mathbf{B}_z|| \in [1, k]$ is the number of background pixels recorded at location z.

However, computing a conditional PDF with a histogram requires a very large number of samples for it to be accurate. Short videos or highly cluttered videos would lead to sparse likelihood functions, with various colors c_z having null probabilities. In order to smooth out the PDF given the $\mathbf{B}_z$ background values observed at location z, we estimate the background conditional PDF with a Parzen-Window Density Estimation [179]. The PDF of a given color c_z at location z with respect to every background value recorded at the same location is estimated as

$$p(c_z|\mathbf{B}_z) = \frac{1}{||\mathbf{B}_z||} |\mathbf{H}|^{-\frac{1}{2}} \sum_{i=1}^{||\mathbf{B}_z||} \mathcal{K}\left(\mathbf{H}^{-\frac{1}{2}}(c_z - b_i)\right), \tag{5.2}$$

where $b_i \in \mathbf{B}_z$ is a background RGB color, $\mathcal{K}$ is a multivariate kernel, satisfying $\int \mathcal{K}(\mathbf{x})d\mathbf{x} = 1$ and $K(\mathbf{x}) \geq 0$, and $\mathbf{H}$ is a $d \times d$ symmetric positive bandwidth matrix, d is the dimension of the data c_i and b_i (*i.e.*, d=3 as the number of the color channels). In our implementation, we use a Gaussian kernel K and, as is usually the case, we assume independence of different color channels, *i.e.*, $\mathbf{H}$ is a diagonal matrix

$$\mathbf{H} = diag(\sigma_1^2, \ldots, \sigma_d^2), \tag{5.3}$$

where σ_j^2 is the bandwidth of the kernel in the j^{th} dimension. The bandwidth is estimated using the following estimator [204]:

$$\sigma = \left(\frac{4\hat{\sigma}^5}{3||\mathbf{B}_z||}\right)^{\frac{1}{5}} \approx 1.06\hat{\sigma}||\mathbf{B}_z||^{-\frac{1}{5}}, \tag{5.4}$$

where $\hat{\sigma}$ is the standard deviation (std dev) of the data. As mentioned by Silverman, Eq.(5.4) is well suited for unimodal distributions. He also mentioned that, in case of heavily skewed or bimodal distributions, one can reduce the 1.06 factor down to 0.9, and use a slightly different σ estimator. Since most of the pixels in our videos have a unimodal distribution (except for videos with strong background motion and sudden illumination changes), we decided to keep using Eq.(5.4). Note that, as mentioned by Narayana *et al.* [164], other bandwidth estimators could also be used. Therefore, the PDF of Eq. (5.2) can be written as

$$p(c_z|\mathbf{B}_z) = \frac{1}{||\mathbf{B}_z||} \sum_{i=1}^{||\mathbf{B}_z||} \prod_{j=1}^{3} \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2}\frac{(c_{z,j}-b_{i,j})^2}{\sigma_j^2}}, \tag{5.5}$$

where $c_{z,j}$ and $b_{i,j}$ indicate the j^{th} component of c_z and b_i respectively.

5.3.3 Evaluation Metrics

We adopted two types of metrics to accommodate with the two GT background models our dataset provides.

The first type of metrics includes those commonly used in the background initialization literature [155, 86, 51, 227, 14, 190, 171, 206, 104]. These are image-to-image metrics measuring the visual correctness of an estimated background image against a GT background image. Each of them exploits different aspects of image quality evaluation, thus leading to an extensive overall evaluation of results. To compute the metrics, we use the Y channel in the YCbCr color space

$$\mathrm{Y} = 0.299 \times \mathrm{R} + 0.587 \times \mathrm{G} + 0.114 \times \mathrm{B}. \tag{5.6}$$

Let B_{gt} be the Y channel of the GT background image and B_{eb} the Y channel of the background image computed by a background initialization method. The following six metrics [4] have been adopted to evaluate the compared algorithms:

1. **Average Gray-level Error (AGE)** [86, 227, 14, 190, 171, 206, 104]: It is the average of the absolute difference between the gray-level images B_{gt} and B_{eb}

$$\mathrm{AGE} = \frac{1}{N} \sum_{z=0}^{N-1} |B_{gt_z} - B_{eb z}|, \tag{5.7}$$

where N is the total number of pixels in the image. The resulting AGE value ranges between 0 and 255. According to that metric, the lower the AGE value is, the better the background estimate is.

2. **Percentage of Error Pixels (pEPs)** [86, 227, 51, 206, 104]: An *error pixel* (EP) is a pixel of the estimated background B_{eb} whose value differs from the value of the corresponding pixel in B_{gt} by more than a threshold τ (we use τ=20, as suggested in [86, 227, 190]). pEPs is the ratio between the number N of EPs and

[4] MATLAB and python scripts for computing the metrics have been made available through the SBMnet webpage.

the total number of image pixels:

$$\text{pEPs} = \frac{1}{N} \sum_{z=0}^{N-1} \left(\mathbb{1}_{|B_{gt_z} - B_{eb_z}| > \tau} \right), \tag{5.8}$$

where $\mathbb{1}$ is an indicator function. The resulting value of pEPs ranges between 0 and 1; the lower the pEPs value is, the more accurate the estimated background is.

3. **Percentage of Clustered Error Pixels (pCEPs)** [86, 227, 14, 190, 171, 206, 104]: A *clustered error pixel* (CEP) is defined as any error pixel whose 4-connected neighbors are also error pixels. pCEPs is thus the ratio between the number of CEPs and the number N of image pixels. In this case, it ignores isolated noise pixels in the estimated background, e.g., salt and pepper noise. Its value is in the [0, 1] range; the lower it is, the closer the background estimate is to the GT.

4. **Peak-Signal-to-Noise-Ratio (PSNR)** [51, 171, 206, 104, 95]: This well known and often utilized metric is defined as

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right), \tag{5.9}$$

where MAX is 255 in our case and MSE is the mean squared error between B_{gt} and B_{eb},

$$\text{MSE} = \frac{1}{N} \sum_{z=0}^{N-1} \left(B_{gt_z} - B_{eb_z} \right)^2. \tag{5.10}$$

PSNR assumes values in decibels (db); thus, the higher it is, the better the background estimate is. Although widely utilized, PSNR has two limitations that one shall keep in mind: (1) it assumes that both images are rigorously aligned, and (2) it penalizes errors at brighter pixels more than errors at darker pixels.

5. **Color image Quality Measure (CQM)** [171, 206, 104]: This metric is an improved version of PSNR [256]. It first converts RGB images to the YUV color space and then computes the PSNR value of each YUV channel separately. The

resulting PSNR values are then combined as follows:

$$\text{CQM} = \text{PSNR}_Y \times R_W + \frac{\text{PSNR}_U + \text{PSNR}_V}{2} \times C_W, \tag{5.11}$$

where R_W and C_W are biologically-inspired coefficients set to 0.9449 and 0.0551 respectively (please refer to the original paper for more details [256]). As for the PSNR, CQM values are in decibels and so the larger it is, the closer the estimated background is to the GT background image.

6. **Multi-Scale Structural Similarity Index** (**MS-SSIM**) [171, 206, 104]: This is the metric defined in [241], that uses structural distortion as an estimate of the perceived visual distortion of two images, evaluated at multiple scales. For a single scale, the SSIM of a squared image block $\mathbf{x}_{eb}$ of B_{eb} and the corresponding image block $\mathbf{x}_{gt}$ of B_{gt} is computed as:

$$\text{SSIM}(\mathbf{x}_{eb}, \mathbf{x}_{gt}) = \frac{(2\mu_{gt}\mu_{eb} + C_1)(2\sigma_{gt,eb} + C_2)}{(\mu_{gt}^2 + \mu_{eb}^2 + C_1)(\sigma_{gt}^2 + \sigma_{eb}^2 + C_2)}, \tag{5.12}$$

where μ_y and σ_y are the mean and the variance of $\mathbf{x}_y$, $y \in \{eb, gt\}$, respectively, $\sigma_{gt,eb}$ is the covariance of $\mathbf{x}_{eb}$ and $\mathbf{x}_{gt}$, $C_1 = K_1 L$, $C_2 = K_2 L$, $K_1 = 0.01, K_2 = 0.03$, and L is the range of pixel values. The SSIM of the whole images is computed as the mean of the values obtained for all corresponding image blocks. MS-SSIM aggregates SSIM values computed at different image scales, thus providing hints on the similarity of the GT and the evaluated background images at both the global and the detail level. As opposed to the previous metrics, MS-SSIM is translation invariant. It assumes values in $[-1, 1]$; the higher the value of MS-SSIM is, the better the estimated background is.

The second type of metric evaluates the quality of the estimated background image using the likelihood PDF function in Eq. (5.5). This is done with a probabilistic metric called PM$(\cdot)$ which computes the negative log-likelihood of the overall image B_{eb}, normalized against the image size N:

$$\text{PM}(B_{eb}) = -\frac{1}{N}\sum_{z=0}^{N-1} \ln p(B_{eb_z} | \mathbf{B}_z), \tag{5.13}$$

where B_{eb_z} indicates the RGB background pixel color at location z and $\mathbf{B}_z$ the list of background color values recorded at that pixel. The lower the value of PM is, the better the estimated background is.

5.3.4 Ranking Strategy

Once the metrics are calculated, we ranked the scene background initialization methods participating to the SBM Challenge with an approach similar to that of Goyette *et al.* [81]. For each video in each category, we first computed the six image-based metrics described in the previous section[5]. Each metric is then averaged category-wise. Take AGE for example. Let $|N_c|$ be the number of videos in category c, the AGE metric for that category is calculated as:

$$\text{AGE}_c = \frac{1}{|N_c|} \sum_{v=1}^{|N_c|} \text{AGE}_{v,c}. \tag{5.14}$$

This procedure is repeated for each image-based metric.

After that, we computed the metrics for the entire dataset. This has been done by averaging the eight category metrics; *e.g.*, for AGE:

$$\text{AGE}_{overall} = \frac{1}{8} \sum_{c=1}^{8} \text{AGE}_c. \tag{5.15}$$

While methods can be ranked based on any of these metrics, we also consider their ranking across all of them. The rationale for this is to give an indication of how good a method is with respect to other methods in each category and across all categories. Following the approach by Young and Ferryman [261] and Goyette *et al.* [81], we provide an average ranking *R* across all overall-average metrics, and an average ranking *RC* across all categories. Let $rank_i(m, c)$ denote the rank of a method i (*i.e.*, its position with respect to other methods) for metric m in category c. For that method, the average

[5] Note that, since the ranking was implemented before the ICPR 2016 SBM Challenge, it does not include the PM metric introduced here.

ranking over all six metrics in category c is given by:

$$rank_{i,c} = \frac{1}{6} \sum_{m=1}^{6} rank_i(m, c). \tag{5.16}$$

The average ranking RC_i across all categories for method i is then calculated as the average across all eight categories:

$$RC_i = \frac{1}{8} \sum_{c=1}^{8} rank_{i,c}. \tag{5.17}$$

For the average ranking *R*, the overall-average metric values (like that calculated by Eq. (5.15) for the AGE metric) are ranked as $rank_i(m)$. Then, R_i is the average of $rank_i(m)$ over all six metrics

$$R_i = \frac{1}{6} \sum_{m=1}^{6} rank_i(m). \tag{5.18}$$

The category-average and overall-average metrics obtained on so far 24 different methods, as well as their overall average rankings *R* and *RC*, are reported on the SBMnet website.

5.4 Experimental Results

5.4.1 Compared Methods

In the wake of the ICPR 2016 SBM Challenge, results from 14 different scene background initialization methods have been uploaded on the SBMnet website.

Four temporal statistics methods are based on temporal median filter, namely the plane *temporal median filter* (TMF) [156], *LaBGen* and *LaBGen-P* [128], and *Temporal Median Filter with Gaussian filtering* (TMFG) [141]. *TMF* is undoubtedly the simplest method reported in this paper. *LaBGen* combines a temporal median filter with a patch-wise motion detection, while *LaBGen-P* is an extension of it using a pixel-wise motion detector. *TMFG* models pixel-wise the background through a single temporal Gaussian distribution for each pixel and applies a temporal median filter only on pixels having sufficiently high probability values. *LaBGen*, *LaBGen-P*, and *TMFG* can also be seen as background initialization methods based on subsequences of stable intensity (see Section 5.2), where subsequences with similar intensity values are first selected through motion detection, and intensity values of each pixel taken along time in the stable subsequences are used to construct the estimated background.

Two other methods based on temporal statistics adopt a Gaussian mixture model, namely *Bidirectional Analysis* (B-A) and *Bidirectional Analysis and Consensus Voting* (BACV) [159]. The two offline methods implement a forward and a backward pixel-based GMM, computed by processing the video from the first to the last frame and vice versa. The background image is obtained, for each pixel, by selecting pixel values having the highest combined probability in the forward and backward models. Further consensus voting, taking into account spatial information in the neighborhood of each pixel, helps in refining the estimated result.

Two of the compared methods are based on iterative model completion. In the online method proposed in [190] (which we call *RSL2011*), for each block location a representative set is maintained which contains distinct blocks obtained along its temporal line. The background initialization is carried out in a MRF framework, where the optimal labeling solution is computed using iterated conditional modes. In *Rejection based Multipath Reconstruction* (RMR) [171], the first phase involves a temporal module that clusters the input frames and generates background candidates. Then, a module based on spatial analysis iteratively recovers the final background from background candidates, using a multipath reconstruction method guided by smoothness constraints.

Photomontage [3] is a unified framework for interactive image composition, based on a MRF graph-cut optimizer. The cost function consists of an interaction term, that penalizes perceivable seams in the composite image, and a data term, that reflects various objectives of different image editing tasks. For the specific task of background initialization, the data term adopted for achieving visual smoothness is the maximum likelihood image objective.

Motion-Aware Graph Regularized RPCA (MAGRPCA) [103] tackles background initialization as a missing data reconstruction problem. Based on RPCA, it implements a graph regularization on the low-rank component using motion estimation, in order to better handle background variations.

Finally, three of the compared methods are based on neural networks, namely *SC-SOBS-C4* [151], *AAPSA* [41], and *FC-FlowNet* [88]. *SC-SOBS-C4* is based on the self-organizing neural background model SC-SOBS [154] which was originally used for detecting moving objects. Several criteria for extracting an image of the estimated background by the multi-modal SC-SOBS model can be considered [151]. Here, the estimated background image is obtained by choosing, for each pixel, the SC-SOBS modeling weight vector that is the closest to the corresponding pixel in the background image estimated by an accurate uni-modal background initialization method (here Photomontage [3]). *AAPSA* implements two neural networks for modeling the background, each replicating a running average, which adapt their parameters at different rates. For each pixel, the method automatically decides to use or to combine the information contained into the two models to obtain the estimated background image. The third neural network method is *FC-FlowNet*, based on a CNN trained to estimate background patches. The CNN is made of a series of convolution and deconvolution.

The methods reported in this section are by nature very different. In order to measure how complementary they are, we implemented a method which combines them all. A typical approach for combining several methods is through a majority vote. Unfortunately, a pixel-wise majority vote would result into blurry and often corrupted background images. Instead, we implemented a method which, for each video, compares all background images computed by the 14 compared methods and selects one of them. Our strategy for selecting one of the computed background images is based on the assumption that, for each video, a subset of methods (which might be different from one video to another) correctly estimates the background image. As such, these correct results are all visually similar. As for the other methods which return a corrupted background image, their visual distance to any other result is unavoidably high. Our method, in the following referred to as *MS-SSIM-Selection*, returns the background image $\tilde{B}_i$ which is visually similar to as many other results as possible, *i.e.*,:

$$\tilde{B}_i = \underset{B_i}{\text{argmax}} \sum_{j=1,j\neq i}^{14} \text{MS-SSIM}_{B_i,B_j}, \tag{5.19}$$

Table 5.2 – Overall results of 15 different methods.

Method	*R*	*RC*	AGE	pEPs	pCEPs	MS-SSIM	PSNR	CQM	PM
MS-SSIM-Selection	1.17	2.75	6.16	0.057	0.021	0.94	30.0	30.8	8.56
LaBGen	3.00	5.50	6.71	0.063	0.027	0.93	28.6	29.5	8.93
LaBGen-P	3.83	6.25	7.07	0.071	0.032	0.93	28.5	29.3	8.98
Photomontage	4.33	7.25	7.20	0.069	0.026	0.92	28.0	28.9	8.64
SC-SOBS-C4	5.33	6.88	7.52	0.071	0.024	0.92	27.7	28.6	8.88
MAGRPCA	6.83	7.63	8.31	0.099	0.057	0.94	28.5	29.3	17.0
TMF	8.17	6.25	8.28	0.098	0.055	0.91	27.5	28.4	10.0
BE-AAPSA	8.17	8.75	7.91	0.087	0.045	0.91	27.1	28.0	11.4
B-A	8.50	7.63	8.34	0.076	0.018	0.91	26.2	27.2	8.97
BACV	9.67	8.63	8.58	0.072	0.026	0.91	26.1	27.1	9.25
TMFG	11.00	6.88	9.40	0.110	0.057	0.90	27.1	28.1	10.1
FC-FlowNet	11.17	10.0	9.11	0.110	0.060	0.92	27.0	27.9	10.6
RSL2011	12.17	11.25	9.04	0.100	0.050	0.89	25.8	26.8	10.0
AAPSA	13.17	11.75	9.20	0.110	0.052	0.90	25.4	26.3	12.2
RMR	13.50	10.88	9.54	0.120	0.058	0.88	26.5	27.5	10.5

where MS-SSIM$_{B_i,B_j}$ is the visual distance value between the background images estimated by methods i and j, $i, j \in [1, 14]$ in terms of the MS-SSIM metric.

Indeed, even though other metrics could have been chosen, this metric perfectly fits our selection requirement, since it well reflects the global structural relation between different images (see Section 5.3.3).

5.4.2 Overall Results

The overall results are shown in Table 5.2. The table shows the R ranking, the RC ranking, the six image-based metrics as well as the log-likelihood probabilistic metric (PM) (*cf.* Section 5.3.3). Methods have been sorted according to R. The reader shall find more detailed results for each category and each video on the SBMnet website.

As shown in Table 5.2, the top performing methods are *LaBGen* (and *LaBGen-P*), *Photomontage*, and *SC-SOBS-C4*, three very different methods. Surprisingly, the simplistic *temporal median filter* performs quite well, as it beats more than half the methods. This is a strong indication that, besides its obvious limits, the true background image of many videos is close to that obtained with TMF. Even more interesting is that *MS-SSIM-Selection* (the strategy for combining all 14 methods) has the best ranking according to every metric. This underlines the fact that, as of today, there is no such thing like a single best method for background initialization. On the contrary, methods are complementary by nature. When a method fails on some videos, others perform better and can compensate for it, and vice versa. This suggests that future work could borrow concepts from several of these methods and hopefully outperform them all.

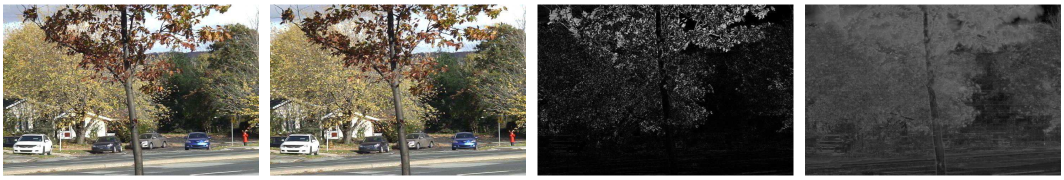

(a) (b) (c) (d)

Figure 5.2 – Sequence *fall* of "*background motion*" category: (a) result of *Photomontage*; (b) GT image; (c) absolute gray-level difference of (a) and (b); (d) negative log likelihood map of (b) as compared to the GT model. Although visually identical to the GT image, the result achieves very poor image metrics scores (AGE: 23.9, pEPs: 0.32, pCEPs: 0.11, MS-SSIM: 0.73, PSNR: 15.6, CQM: 16.7), due to tree branches not perfectly aligned, but average probabilistic metric value (PM: 9.707).

As for the metrics, pEPs and AGE are the ones whose ranking is the closest to R. This makes us conclude that if someone is to report only one metric, pEPs and AGE might be the most appropriate ones. The reader shall note that the R and RC ranking rules should be considered with care, as they might be ill-suited for some methods. For example, MAGRPCA got ironically the best MS-SSIM score but among the worst pEPs score. After careful investigation, we realized that MAGRPCA is a very accurate method, but its estimated background images often suffer from a global illumination shift. While MS-SSIM and CQM are illumination invariant, the other metrics heavily penalize global illumination errors. A global color shift of the background image could be acceptable for applications like computational photography, but would lead to inaccurate results for applications like foreground detection. Thus, the ranking rule shall always be considered with the end application in mind.

5.4.3 Category Results

In this section, we inspect the results obtained on each category of videos. We first calculated the mean and std dev of the six image metrics described in Section 5.3.3 for each category, as reported in Figs. 5.3 (a)-(f). With no surprise, the easiest category on average is the "*Basic*" category, and this is true for all six metrics. The mean metric values are always the best (*e.g.*, low for AGE, pEPs, and pCEPs, and high for PSNR, CQM, and MS-SSIM) and the std dev values are always small. This shows that, for every method, backgrounds estimated on "*Basic*" videos are always close to the ground truth. On the other hand, "*Jitter*" and "*Background motion*" categories often get the worst average metric values (*e.g.*, large for AGE, pEPs, and pCEPs, and very low for PSNR, CQM, and MS-SSIM). Also, with a low std dev, the side-by-side comparison of those metrics suggests that a majority of methods are struggling with these videos. However, careful inspection of results reveals that the main issue may come from the metrics themselves. Indeed, since a moving background may have a nearly infinite number of configurations (think of every possible shape a moving tree can take), using, as we do, a series of 10 or less background images as ground truth leads unavoidably to poor metric values. This is illustrated in Fig. 5.2 (a), showing that the background image for sequence *fall* of "*Background motion*" category estimated by the *Photomontage* method is visually identical to the GT image (Fig. 5.2 (b)). However, the six image metrics values are much worse than the corresponding mean values reported in Figs. 5.3 (a)-(f). This is because the branches of the tree in the *Photomontage* result are not aligned with those in the GT, as it can be appreciated looking at the absolute gray-level difference of the two images, reported in Fig. 5.2 (c). Thus, although the results obtained by many methods on "*Jitter*" and "*Background motion*" videos are indeed blurry, the side-by-side comparison of the metrics reported in Fig. 5.3 shall be considered with care.

In the light of the above results, we implemented the log likelihood probabilistic metric PM (*cf.* Section 5.3.3), that does not depend on an image-to-image distance function. As an example, in the case of the *Photomontage* result of Fig. 5.2, we observe that the negative log likelihood map reported in Fig. 5.2 (d) (computed through Eq. (5.5)) allows us to take into account the background motion of the leaves, stored into the GT model, in a much smoother way comparing with the absolute gray-level difference. This results in a PM value comparable to the mean value reported in Fig. 5.3(g). Overall, the "*Jitter*" and "*Background motion*" categories get the best PM values (the smaller, the better) with a very small std dev, as can be seen in Fig. 5.3 (g). This underlines the fact that, although results may be blurred on these videos, they do not suffer from strong artifacts. Interestingly, the "*Very long*" videos also got an excellent PM score. This is because long videos always have a certain level of change overtime (due to slight sun shift in the sky), that is taken care by the PM metric.

We also report in Table 5.3 the top three methods for each category according to ranking R. As one can see, the top three methods for each category are different and their respective orders are also different. The overall best methods (*e.g.*, LaBGen, LaBGen-P, Photomontage, and SC-SOBS-C4) are not always the best methods for each category. On the contrary, they can even be the worst methods, such as LaBGen-P on the "*Very long*" category.

Overall, the category-wise rankings lead us again to conclude that there is no single best method and that future work shall focus on a combination of different methods.

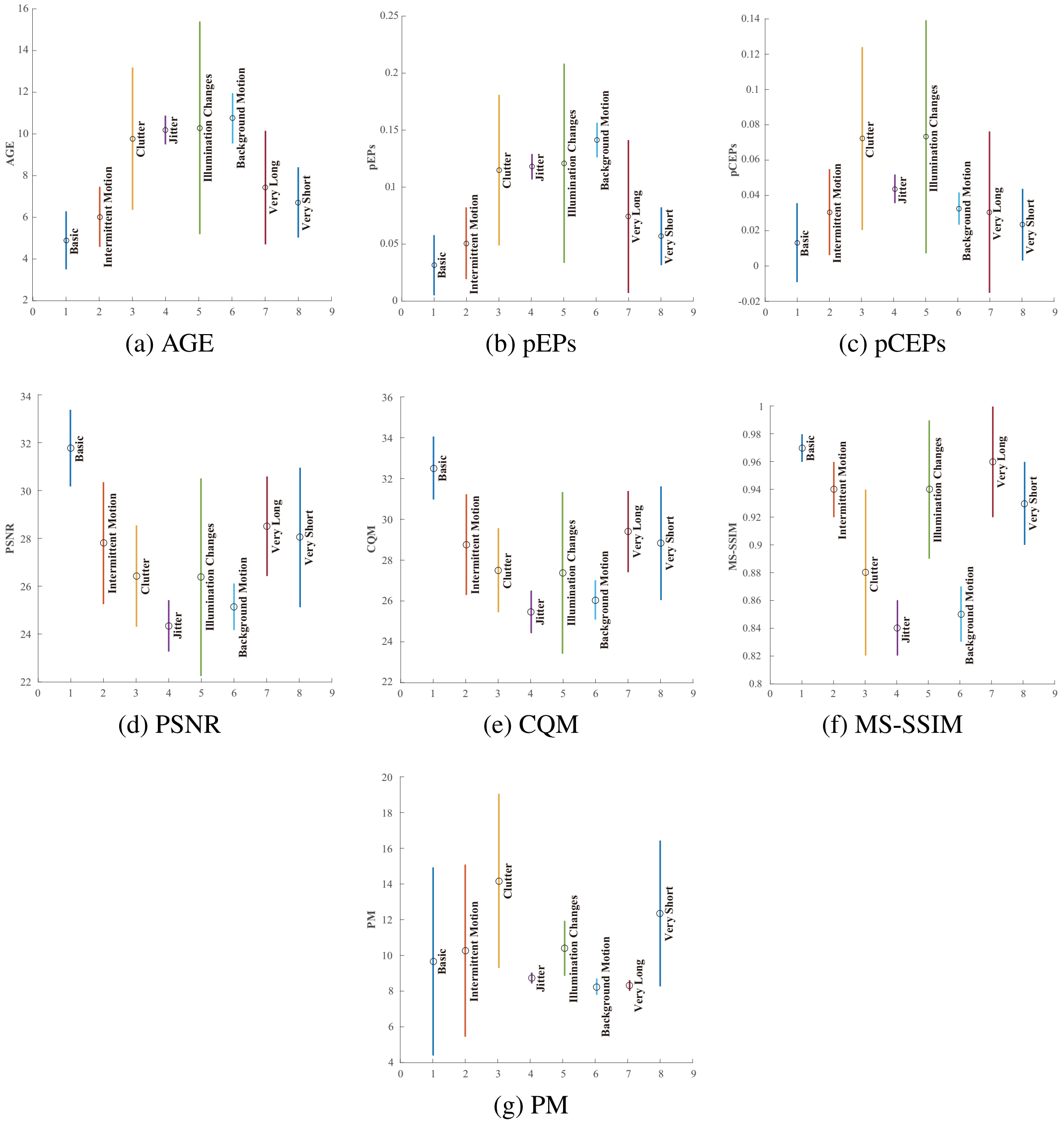
Table 5.3 – Methods with best *R* ranking for each category.

Category	1^{st}	2^{nd}	3^{rd}
Basic	TMFG	TMF	B-A
Intermittent motion	LaBGen-P	BACV	RMR
Clutter	Photomontage	SC-SOBS-C4	B-A
Jitter	TMF	TMFG	LaBGen
Illumination changes	Photomontage	LaBGen	RMR
Background motion	TMFG	BE-AAPSA	TMF
Very long	BE-AAPSA	MAGRPCA	LaBGen
Very short	Photomontage	TMF	TMFG

5.4.4 Unsolved Issues

Five different scenarios, as illustrated in Fig. 5.4, still appear challenging for background initialization:

1. Whenever a foreground object stops moving for a period of time, many methods incorporate that object in to the background or produce strong ghosting artifacts (*e.g.*, Figs. 5.4(a) and (b)).
2. When the background is not visible for a long enough period of time due to a too-short video or heavy clutter, methods get to produce artifacts (*e.g.*, Figs. 5.4 (c)-(e)).



(a) AGE

(b) pEPs

(c) pCEPs

(d) PSNR

(e) CQM

(f) MS-SSIM

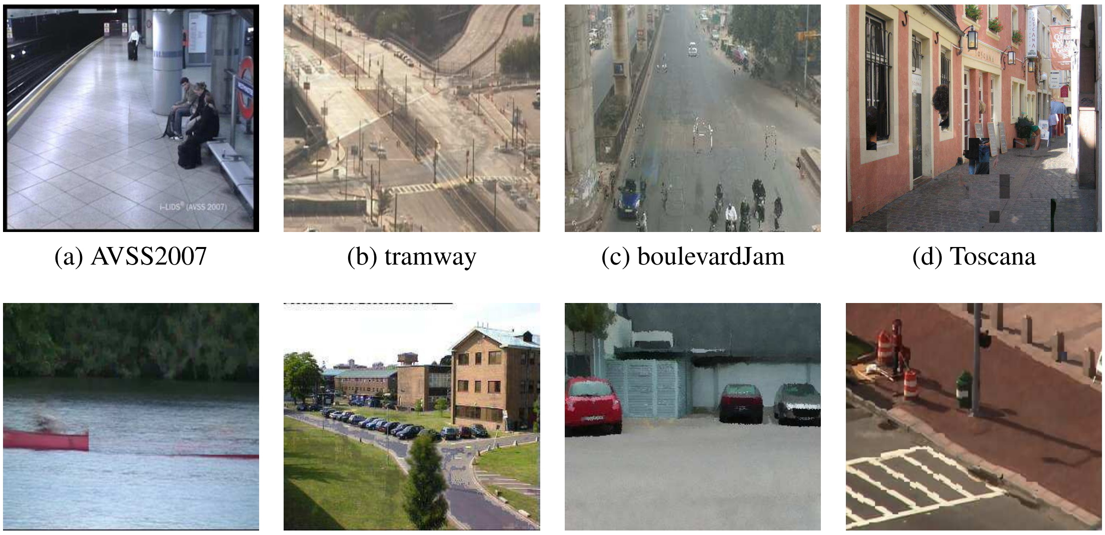
(g) PM

Figure 5.3 – Mean and standard deviation of seven metrics for different categories: (a) AGE, (b) pEPs, (c) pCEPs, (d) PSNR, (e) CQM, (f) MS-SSIM, and (g) PM. The circles are for the means, while the lines are for the std dev.

3. Strong background motion, including a wavy water surface or trees shaken by the wind, is also a concern for a lot of methods. Although this is hard to correctly assess with visual metrics, a large number of methods return blurry results, as illustrated in Fig. 5.4 (e) showing a blurry water surface.

4. Strong illumination changes is also an unsolved issue for the majority of background initialization methods, as they often produce an unrealistic mixture of different illuminations (*e.g.*, Fig. 5.4 (f)).
5. Although most methods are robust to low jitter, strong jitter has a tendency to create blurry results, as shown in Figs. 5.4 (g) and (h).

Even though most of these issues are afforded and partly solved by the best performing background modeling methods for the purpose of foreground detection, our study reveals that state-of-the-art background initialization methods fail over these scenarios. These unsolved issues in background initialization will be surely the new trend towards which all researchers should move when designing new and widely applicable methods.



(a) AVSS2007 (b) tramway (c) boulevardJam (d) Toscana

(e) canoe (f) Dataset3Camera2 (g) O_MC_02 (h) sidewalk

Figure 5.4 – Typical failures: (a) and (b) ghosting artifacts caused by intermittent motion; (c), (d) and (e) artifacts caused by heavy clutter where background pixels are almost never seen; (e) blurry water surface caused by a wavy water; (f) result containing a mixture of two different illumination settings; (g) and (h) blurry images caused by strong camera jitter.

5.5 Concluding Remarks

In this paper, we proposed an extensive survey of scene background initialization methods as well as a benchmarking framework. The framework includes a large dataset, seven evaluation metrics as well as 14 different state-of-the-art methods. The SBMnet dataset we proposed includes 79 videos in eight categories, corresponding to eight different classes of challenges for video background initialization. Based on the dataset, a complete analysis of representative state-of-the-art methods is provided. Their complementarity is further analyzed by proposing a possible ideal optimum method, named *MS-SSIM-Selection*, that selects, among them, the best one for each video. We analyzed advantages and limitations of the image-based metrics useful for single background images and proposed a probability-based metric which accounts for a complete distribution of background over the whole video. Adopting this pool of metrics allowed us to underline the strength and weaknesses of each method, giving a good understanding of the solved and still unsolved issues, that deserve future in-depth research. Moreover, deeper analysis is required by the identification of an overall ranking strategy that is suited for all the applications of background initialization. Methods other than *MS-SSIM-Selection* could also be devised to better highlight and exploit the advantages of existing background initialization methods.

Acknowledgments

This research was supported by Project PON01_01430 PT2LOG under the Research and Competitiveness PON, funded by the European Union via structural funds, with the responsibility of the Italian Ministry of Education, University, and Research (MIUR), and by INTEROMICS Flagship Project, funded by MIUR.

Conclusion

As a basic computer vision task, motion detection has attracted researchers' attention for decades. So many works have been done in all aspects of it. Nowadays, motion detection has achieved significant improvements, and a lot of problems in it are claimed to be solved. However, this does not mean in general motion detection is a solved issue. Unsolved problems still remains.

In this work, we focused on four aspects of motion detection: (1) How to evaluate motion detection methods objectively; (2) How to use an interactive deep learning method to detect motion in a video; (3) How to use motion information to improve the performance of state-of-the-art pedestrian detectors; and (4) how to evaluate background initialization methods.

Motion detection benchmark To evaluate motion detection methods objectively, we built the CDnet 2014 dataset. With 75 videos separated in 11 categories that cover most of the challenges in motion detection, the CDnet 2014 dataset is by far the biggest and most objective motion detection dataset in the world. Seven metrics are used for the evaluation. We also provide an online evaluation system that researchers can use to compare their motion detection methods with others. As of today, about 40 methods have been uploaded to the CDnet 2014 website. The CDnet 2014 paper has also been cited 133 times.

Based on the benchmarking and evaluation made in this thesis, we discovered that:

1. For most challenging categories, state-of-the-art methods are remarkably accurate. However, several challenges are still quite difficult for most of the methods.

Overall, a majority of methods fail under the following circumstances:

(a) **Videos with strong global movements**: Local background movement (*e.g.* a tree shaken by the wind and shimmering water), and low level global movements (*e.g.* camera jitter) in a video are not a problem for most state-of-the-art methods. However, videos with strong global background movement, such as videos shot by a PTZ camera, are still a big issue for almost every method.

(b) **Videos with blurry foreground edges**: It is not easy to accurately label the foreground if its edge is unclear. This usually happens when the resolution of the video is low, or when the video is shot at night.

(c) **Videos with low framerate**: With a low framerate, the location of a moving object can change drastically from one frame to another. Tracking objects in such a video can be very challenging. At the same time, optical flow is more difficult to be calculated (if not impossible). In that case, any method based on tracking or optical flow may fail when the video framerate is low.

2. Combining motion detection results of different methods can help to improve the performance. Nevertheless, generating motion detection masks with different methods and then combining them is computationally expensive. For applications which require fast detection, combing results of several methods is not recommended.

Deep foreground segmentation To detect and label foreground objects in videos, we implemented a semi-automatic motion detection method. Our model is composed of a multi-scale CNN [123] with a cascaded structure. Our deep learning method is as accurate as a human while being 40 times faster than manual labeling.

Results reveal that our method can be applied on all kinds of challenging videos. As deep learning methods have redefined the limits of various applications, it can be predicted that more deep learning methods will be proposed for motion detection.

Pedestrian detection We also proposed a model to combine motion features with state-of-the-art pedestrian detectors. To achieve this goal, we extracted motion information and accumulated it into an MHI. The MHI was then used to filter the video to remove the false positive detection. At the end, a feedback loop as well as a merging procedure between the filtered and the unfiltered frames are used to further improve results. Our model has been tested with six state-of-the-art pedestrian detectors, all their performances have been significantly improved with our proposed strategy.

Our work clearly proved that motion detection technology can be used to improve other computer vision tasks. The only concern is that if the extracted motion features are not accurate, the overall performance may decrease when motion features are used. However, as the performance of state-of-the-art motion detection methods keep improving, motion features will be more widely used in conjunction of higher-level computer vision applications.

Scene background modeling benchmark To evaluate the background modeling methods, we proposed the largest background modeling dataset SBMnet and a novel benchmarking framework. The SBMnet dataset contains 79 videos in eight categories. The benchmarking framework includes seven evaluation metrics, which allows us to quantitatively identify solved and unsolved issues related to scene background modeling. We also concretely identify scenarios for which state-of-the-art methods systematically fail and propose concrete ideas for future works. So far, an ICPR workshop has been organized based on SBMnet and 24 different background modeling methods have been submitted to SBMnet.

Although background initialization has several applications such as video compression, video inpainting, *etc*., it attracted less attention than motion detection. One big reason for that is the difficulty to evaluate the performance of background initialization. As most of the evaluation metrics are image based, while for some cases, a limited number of image cannot represent all the status of the background in a video. Even though we proposed a probabilistic model for this issue, other evaluation metrics could be proposed in the future.

Future Work

After I finish my Ph.D., I would like to explore more the area of motion detection. There are plenty of things to try based on the works that I have done during my Ph.D.

From the previous projects, the following projects could be explored:

1. The convolutional neural networks based motion detection method that we proposed (described in Chapter 3) achieved excellent performance. However, some improvements can still be made. First, our model has the ability to learn the background of a video, however, it cannot be trained on one video and generalize to another video, especially when the color distribution of the new video varies from one of the training video. In this case, transfer learning technology may help to solve this problem. Second, our model does not use any temporal information which is an important feature for videos. Combining temporal features with the current model can hopefully improve its performance. 3D convolutional layers may work for this case. Last but not least, the current model is semi-automatic, which means that it still needs a certain level of human interaction. How to reduce the amount of user interaction or even remove the user interaction with minimum drop in performance can be an interesting question that shall be answered in the future.
2. The "pedestrian detection using motion-guided filtering project" (described in Chapter 4) can also be improved in the future. The current model uses only grayscale information of the video. Though increases the speed, it discards the color information of the video. If the speed of the non-linear filtering can be increased (*e.g.*) super pixel or other region based filtering), color and other features can be used to improve the performance of the model.

Beyond that, state-of-the-art deep learning methods can be utilized to explore new directions.

1. Most motion detection methods have to either keep a stable background model and update it slowly; or make the background model flexible, thus the model can react to any foreground movement fast. With a low updating ratio, the stable background model is robust to noise, but takes longer time to adjust to background changes (*i.e.* global illumination changes, or a foreground object stops moving and turns to be background). On the other hand, a flexible background model may adapt to these changes fast, however, it may be affected by the noise or foreground movement easily (*i.e.* a slow moving object, which should not be considered as background).

   As a deep learning model, Long Short-term Memory (LSTM) network can be used to solve this problem. LSTM network is a recurrent neural network (RNN), which can learn the background updating ratio dynamically from the previous frames of the video and adapt it when a new frame arrives. For example, if a motion is caused by noise, the LSTM will not or rarely update the background model; while if it is the movement of a real foreground, LSTM can hopefully detect it and merge it into the background fast once it stops moving.

Appendix A

Publications

1. Jodoin P. M., Maddalena, L., Petrosino, A. and **Wang Y.** "*Extensive Benchmark and Survey of Background Modeling Methods*", Submitted to Transactions on Image Processing, 2016.
2. **Wang, Y.**, Piérard, S., Su, S. Z., and Jodoin, P. M. "*Improving pedestrian detection using motion-guided filtering*", in press at Pattern Recognition Letters, 2016.
3. **Wang, Y.**, Luo, Z. M., and Jodoin, P. M. "*Interactive Deep Learning Method for Segmenting Moving Objects*", in press at Pattern Recognition Letters, 2016.
4. **Wang, Y.**, Piérard, S., Su, S. Z., and Jodoin, P. M. "*Nonlinear Background Filter to Improve Pedestrian Detection*", in New Trends in Image Analysis and Processing–ICIAP 2015 Workshops, pp. 535-543.
5. **Wang, Y.**, Jodoin, P.M., Porikli, F., Konrad, J., Benezeth, Y., and Ishwar, P., "*CDnet 2014: An Expanded Change Detection Benchmark Dataset*", in Proc. IEEE Workshop on Change Detection (CDW-2014) at CVPR-2014, pp. 387-394.
6. Jodoin, P.M., Pierard, S., **Wang, Y.**, Droogenbroeck, V., "*Overview and benchmarking of motion detection methods*", chapter 24. Chapman and Hall/CRC, July 2014.
7. Jodoin, P.M., Benezeth, Y., **Wang, Y.**, "*Meta-tracking for video scene understanding*", in proc. of Advanced Video and Signal Based Surveillance, 2013, pp. 1-6.

Bibliography

[1] 2nd IEEE Change Detection Workshop. in conjunction with CVPR. www.changedetection.net, 2014.

[2] T. Aach and A. Kaup. Bayesian algorithms for adaptive change detection in image sequences using markov random fields. *Signal Process. Image Commun.*, 7(2):147–160, 1995.

[3] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. In *ACM Trans. Graph.*, volume 23-3, pages 294–302, 2004.

[4] D. S. Alex and A. WAHI. Bsfd: Background subtraction frame difference algorithm for moving object detection and extraction. *J. Theoretical and Applied Information Tech.*, 60(3):623–628, 2014.

[5] S. Ali and M. Shah. A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1–6, 2007.

[6] G. Allebosch, F. Deboeverie, P. Veelaert, and W. Philips. Efic: edge based foreground background segmentation and interior classification for dynamic camera viewpoints. In *Intern. Conf. Advanced Concepts Intelligent Vis. Syst.*, pages 130–141, 2015.

[7] G. Allebosch, D. Van Hamme, F. Deboeverie, P. Veelaert, and W. Philips. C-efic: Color and edge based foreground background segmentation with interior classification. In *Intern. Joint Conf. Comput. Vis., Imaging Comput. Graph.*, pages 433–454, 2015.

[8] M. S. Allili, N. Bouguila, and D. Ziou. A robust video foreground segmentation by using generalized gaussian mixture modeling. In *Proc. IEEE Conf. Comput. Robot Vis.*, pages 503–509, 2007.

[9] M. S. Allili, N. Bouguila, and D. Ziou. Finite general gaussian mixture modeling and application to image and video foreground segmentation. *J. Electron. Imaging*, 17(1):1–13, 2008.

[10] D. Anderson, R. H. Luke, J. M. Keller, M. Skubic, M. Rantz, and M. Aud. Linguistic summarization of video for fall detection using voxel person and fuzzy logic. *Comput. Vis. Image Und.*, 113(1):80–89, 2009.

[11] D. Avola, L. Cinque, G. L. Foresti, C. Massaroni, and D. Pannone. A keypoint-based method for background modeling and foreground detection using a ptz camera. *Pattern Recognit. Lett.*, pages 1–10, 2016.

[12] M. Babaee, D. T. Dinh, and G. Rigoll. A deep convolutional neural network for background subtraction. *arXiv preprint arXiv:1702.01731*, 2017.

[13] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *Proc. IEEE Intern. Conf. Comput. Vis.*, pages 1–8, 2007.

[14] D. Baltieri, R. Vezzani, and R. Cucchiara. Fast background initialization with recursive Hadamard transform. In *Proc. IEEE Intern. Conf. Advanced Video and Signal-based Surveillance*, pages 165–171, 2010.

[15] O. Barnich and M. Van Droogenbroeck. Vibe: a powerful random technique to estimate the background in video sequences. In *Proc. IEEE Intern. Conf. Acoustics Speech Signal Process.*, pages 945–948, 2009.

[16] O. Barnich and M. Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. In *IEEE Trans. Image Process.*, volume 20-6, pages 1709–1724, 2011.

[17] F. Bashir and F. Porikli. Performance evaluation of object detection and tracking systems. In *Proc. IEEE Workshop on PETS*, pages 7–14, 2006.

[18] B. E. Bayer. Color imaging array, July 20 1976. US Patent 3,971,065.

[19] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. Pedestrian detection at 100 frames per second. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2903–2910, 2012.

[20] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Comparative study of background subtraction algorithms. *J. Electron. Imaging*, 19(3): 1–12, 2010.

[21] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Review and evaluation of commonly-implemented background subtraction algorithms. In *Proc. IEEE Intern. Conf Intern. Conf. Pattern Recognit.*, pages 1–4, 2008.

[22] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Comparative study of background subtraction algorithms. *J. Electron. Imaging*, 19(3): 1–12, 2010.

[23] B. Benfold and I. Reid. Stable multi-target tracking in real-time surveillance video. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3457–3464, 2011.

[24] Y. Bengio. Learning deep architectures for ai. *Foundations and trends® in Mach. Learning*, 2(1):1–127, 2009.

[25] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: the CLEAR MOT metrics. *EURASIP J. Image Video Process.*, 2008: 1–10, 2008.

[26] S. Bianco, G. Ciocca, and R. Schettini. How far can you get by combining change detection algorithms? *arXiv preprint arXiv:1505.02921*, 2015.

[27] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Trans. Pattern Anal. Machine Intell.*, 23(3):257–267, 2001.

[28] T. Bouwmans. Traditional and recent approaches in background modeling for foreground detection: An overview. *Comput. Scie. Review*, 11:31–66, 2014.

[29] T. Bouwmans. Recent advanced statistical background modeling for foreground detection-a systematic survey. *Recent Patents Comput. Sci.*, 4(3):147–176, 2011.

[30] T. Bouwmans and E. H. Zahzah. Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance. *Comput. Vis. Image Und.*, 122:22–34, 2014.

[31] T. Bouwmans, F. El Baf, and B. Vachon. Background modeling using mixture of Gaussians for foreground detection-a survey. *Recent Patents Comput. Sci.*, 1(3): 219–237, 2008.

[32] T. Bouwmans, C. Silva, C. Marghes, M. S. Zitouni, H. Bhaskar, and C. Frelicot. On the role and the importance of features for background modeling and foreground detection. *arXiv preprint arXiv:1611.09099*, 2016.

[33] T. Bouwmans, L. Maddalena, and A. Petrosino. Scene background initialization: a taxonomy. *Pattern Recognit. Lett.*, pages 1–9, 2017.

[34] T. Bouwmans, A. Sobral, S. Javed, S. K. Jung, and E.-H. Zahzah. Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset. *Comput. Scie. Review*, 23:1–71, 2017.

[35] M. Braham and M. Van Droogenbroeck. Deep background subtraction with scene-specific convolutional neural networks. In *Proc. IEEE Conf. Syst., Signals and Image Process.*, pages 1–4, 2016.

[36] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE Trans. Pattern Anal. Machine Intell.*, 33(9):1820–1833, 2011.

[37] S. Brutzer, B. Höferlin, and G. Heidemann. Evaluation of background subtraction techniques for video surveillance. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1937–1944, 2011.

[38] D. Butler, S. Sridharan, and V. J. Bove. Real-time adaptive background segmentation. In *Proc. IEEE Intern. Conf. Acoustics Speech Signal Process.*, volume 3, pages 349–352, 2003.

[39] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *J. ACM*, 58(3):1–37, 2011.

[40] CaviarDataset. http://homepages.inf.ed.ac.uk/rbf/CAVIAR/, 2003.

[41] M. Chacon-Murguia, G. Ramirez-Alonso, and J. Ramirez-Quintana. Evaluation of the background modeling method auto-adaptive parallel neural network architecture in the sbmnet dataset. In *Proc. IEEE Workshop on Intern. Conf. Pattern Recognit.*, pages 137–142, 2016.

[42] G. Chen, Y. Ding, J. Xiao, and T. Han. Detection evolution with multi-order contextual co-occurrence. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1798–1805, 2013.

[43] S. Chen, J. Zhang, Y. Li, and J. Zhang. A hierarchical model incorporating segmented regions and pixel descriptors for video background subtraction. *IEEE Trans. Syst., Industrial Informatics*, 8(1):118–127, 2012.

[44] X. Chen, Y. Shen, and Y. H. Yang. Background estimation using graph cuts and inpainting. In *Proc. Graphics Interface*, pages 97–103, 2010.

[45] Y. Chen, J. Wang, and H. Lu. Learning sharable models for robust background subtraction. In *Proc. IEEE Conf. Multimedia Expo*, pages 1–6, 2015.

[46] Y.-T. Chen, C.-S. Chen, C.-R. Huang, and Y.-P. Hung. Efficient hierarchical method for background subtraction. *Pattern Recognit.*, 40(10):2706–2715, 2007.

[47] L. Cheng, M. Gong, D. Schuurmans, and T. Caelli. Real-time discriminative background subtraction. *IEEE Trans. Image Process.*, 20(5):1401–1414, 2011.

[48] Y.-C. Chung, J.-M. Wang, and S.-W. Chen. Progressive background images generation. In *Comput. Vis. Graph. Image Process.*, pages 858–865, 2002.

[49] S. Cohen. Background estimation as a labeling problem. In *Proc. IEEE Intern. Conf. Comput. Vis.*, volume 2, pages 1034–1041, 2005.

[50] A. Colombari and A. Fusiello. Patch-based background initialization in heavily cluttered video. *IEEE Trans. Image Process.*, 19(4):926–933, 2010.

[51] A. Colombari, M. Cristani, V. Murino, and A. Fusiello. Exemplar-based background model initialization. In *Proc. ACM Intern. Workshop on Video Surveillance Sensor Networks*, pages 29–36, 2005.

[52] R. M. Colque et al. Progressive background image generation of surveillance traffic videos based on a temporal histogram ruled by a reward/penalty function. In *SIBGRAPI Conf. Graph., Patterns Images*, pages 297–304, 2011.

[53] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Trans. Pattern Anal. Machine Intell.*, 25(10):1337–1342, 2003.

[54] C. Cuevas, R. Martínez, and N. García. Detection of stationary foreground objects: A survey. *Comput. Vis. Image Und.*, 152:41–57, 2016.

[55] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, volume 1, pages 886–893, 2005.

[56] J. W. Davis. Hierarchical motion history images for recognizing human motion. In *Proc. IEEE Conf. Workshop on Detection Recognit. Events Video*, pages 39–46, 2001.

[57] M. De Gregorio and M. Giordano. Background modeling by weightless neural networks. In *Intern. Conf. Image Anal. Process.*, pages 493–501, 2015.

[58] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Trans. Pattern Anal. Machine Intell.*, 36(8):1532–1545, 2014.

[59] Y. Dong, T. Han, and G. N. DeSouza. Illumination invariant foreground detection using multi-subspace learning. *Intern. J. Knowledge-based Intell. Eng. Syst.*, 14 (1):31–41, 2010.

[60] R. P. Duin. The combining classifier: to train or not to train? In *Proc. IEEE Intern. Conf Intern. Conf. Pattern Recognit.*, volume 2, pages 765–770, 2002.

[61] R. P. Duin and D. M. Tax. Experiments with classifier combining rules. In *Intern. Workshop on Multiple Classifier Syst.*, pages 16–29, 2000.

[62] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. Background and foreground modeling using nonparametric kernel density for visual surveillance. *Proc. IEEE*, 90:1151–1163, 2002.

[63] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *Proc. European Conf. Comput. Vis.*, pages 751–767. Springer, 2000.

[64] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proc. IEEE*, 90(7):1151–1163, 2002.

[65] R. H. Evangelio and T. Sikora. Complementary background models for the detection of static and moving objects in crowded environments. In *Proc. IEEE Intern. Conf. Advanced Video and Signal-based Surveillance*, pages 71–76, 2011.

[66] R. H. Evangelio, M. Pätzold, and T. Sikora. Splitting gaussians in mixture models. In *Proc. IEEE Intern. Conf. Advanced Video and Signal-based Surveillance*, pages 300–305, 2012.

[67] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Trans. Pattern Anal. Machine Intell.*, 35(8):1915–1929, 2013.

[68] B. Fardi, U. Schuenert, and G. Wanielik. Shape and motion-based pedestrian detection in infrared images: a multi sensor approach. In *Proc. IEEE Intell. Vehicles Symp.*, pages 18–23, 2005.

[69] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. Pattern Anal. Machine Intell.*, 32(9):1627–1645, 2010.

[70] A. Fernández-Caballero, J. C. Castillo, J. Serrano-Cuerda, and S. Maldonado-Bascón. Real-time human segmentation in infrared videos. *Expert Syst. Appl.*, 38(3):2577–2584, 2011.

[71] A. Ferone and L. Maddalena. Neural background subtraction for pan-tilt-zoom cameras. *Conf. IEEE Intern. Conf. Syst., Man and Cyb.*, 44(5):571–579, 2014.

[72] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[73] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Trans. Pattern Anal. Machine Intell.*, 30(2): 267–282, 2008.

[74] N. Friedman and S. Russell. Image segmentation in video sequences: A probabilistic approach. In *Proc. Conf. Uncertainty artificial intell.*, pages 175–181, 1997.

[75] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *Decis. Forests Comput. Vis. Med. Image Anal.*, pages 143–157. Springer, 2013.

[76] X. Gao, T. E. Boult, F. Coetzee, and V. Ramesh. Error analysis of background adaption. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, volume 1, pages 503–510, 2000.

[77] M. Gong, Y. Qian, and L. Cheng. Integrated foreground segmentation and boundary matting for live videos. *IEEE Trans. Image Process.*, 24(4):1356–1370, 2015.

[78] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[79] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection.net: A new change detection benchmark dataset. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 1–8, 2012.

[80] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection. net: A new change detection benchmark dataset. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 1–8, 2012.

[81] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. A novel video dataset for change detection benchmarking. *IEEE Trans. Image Process.*, 23(11): 4663–4679, 2014.

[82] M. Granados, H.-P. Seidel, and H. Lensch. Background estimation from non-time sequence images. In *Proceed. Graphics Interface*, pages 33–40, 2008.

[83] M. D. Gregorio and M. Giordano. Change detection with weightless neural networks. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 403–407, 2014.

[84] W. Guo, Y. Xiao, and G. Zhang. Multi-scale pedestrian detection by use of adaboost learning algorithm. In *Intern. Conf. Virtual Reality Visualization*, pages 266–271, 2014.

[85] Z. Guo, L. Zhang, and D. Zhang. A completed modeling of local binary pattern operator for texture classification. *IEEE Trans. Image Process.*, 19(6):1657–1663, 2010.

[86] D. Gutchess, M. Trajkovics, E. Cohen-Solal, D. Lyons, and A. K. Jain. A background model initialization algorithm for video surveillance. In *Proc. IEEE Intern. Conf. Comput. Vis.*, volume 1, pages 733–740, 2001.

[87] T. S. Haines and T. Xiang. Background subtraction with dirichlet processes. In *Proc. European Conf. Comput. Vis.*, pages 99–113, 2012.

[88] I. Halfaoui, F. Bouzaraa, and O. Urfalioglu. CNN-based initial background estimation. In *Proc. IEEE Workshop on Intern. Conf. Pattern Recognit.*, pages 101–106, 2016.

[89] B. Han and L. S. Davis. Density-based multifeature background subtraction with support vector machine. *IEEE Trans. Pattern Anal. Machine Intell.*, 34(5):1017–1023, 2012.

[90] B. Han, D. Comaniciu, and L. Davis. Sequential kernel density approximation through mode propagation: applications to background modeling. In *Asian Conf. Comput. Vis.*, volume 39, pages 1–6, 2004.

[91] Z. Hao, W. Wen, Z. Liu, and X. Yang. Real-time foreground-background segmentation using adaptive support vector machine algorithm. In *Intern. Conf. Artificial Neural Networks*, pages 603–610, 2007.

[92] H. Hattori, V. Naresh Boddeti, K. M. Kitani, and T. Kanade. Learning scene-specific pedestrian detectors without real data. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3819–3827, 2015.

[93] M. Hofmann, P. Tiefenbacher, and G. Rigoll. Background segmentation with feedback: The pixel-based adaptive segmenter. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 38–43, 2012.

[94] Y.-L. Hou and G. K. Pang. People counting and human detection in a challenging situation. *IEEE Trans. Syst., Man, Cybernetics Part A: Syst. Humans*, 41(1):24–33, 2011.

[95] H.-H. Hsiao and J.-J. Leou. Background initialization and foreground segmentation for bootstrapping video sequences. *EURASIP J. Image Video Process.*, 12: 1–19, 2013.

[96] W.-C. Hu, C.-H. Chen, T.-Y. Chen, D.-Y. Huang, and Z.-C. Wu. Moving object detection and tracking from video captured by moving camera. *J. Vis. Commun. Image Represent.*, 30:164–180, 2015.

[97] Q. Huang, M. Han, B. Wu, and S. Ioffe. A hierarchical conditional random field model for labeling and segmenting images of street scenes. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1953–1960, 2011.

[98] S.-S. Huang, L.-C. Fu, and P.-Y. Hsiao. Region-level motion-based background modeling and subtraction using mrfs. *IEEE Trans. Image Process.*, 16(5):1446–1456, 2007.

[99] T. Huynh-The, O. Banos, S. Lee, B. H. Kang, E.-S. Kim, and T. Le-Tien. Nic: a robust background extraction algorithm for foreground detection in dynamic scenes. *IEEE Trans. Circuits and Syst. for Video Tech.*, pages 1–13, 2016.

[100] ImageNet. `http://www.image-net.org/`, 2014.

[101] P. Jaikumar, A. Singh, and S. K. Mitra. Background subtraction in videos using bayesian learning with motion information. In *British Machine Vis. Conf.*, volume 2008, pages 615–624, 2008.

[102] O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *Proc. IEEE Workshop on Motion Video Comput.*, pages 22–27, 2002.

[103] S. Javed, S. K. Jung, A. Mahmood, and T. Bouwmans. Motion-aware graph regularized rpca for background modeling of complex scenes. In *Proc. IEEE Workshop on Intern. Conf. Pattern Recognit.*, pages 120–125, 2016.

[104] S. Javed, A. Mahmood, T. Bouwmans, and S. K. Jung. Spatiotemporal low-rank modeling for complex scene background initialization. *IEEE Trans. Circuits Syst. Video Tech.*, pages 1–14, 2016.

[105] J.-P. Jodoin, G.-A. Bilodeau, and N. Saunier. Background subtraction based on local shape. *arXiv preprint arXiv:1204.6326*, 2012.

[106] P.-M. Jodoin, M. Mignotte, and J. Konrad. Statistical background subtraction using spatial cues. *IEEE Trans. Circuits and Syst. for Video Tech.*, 17(12):1758–1763, 2007.

[107] P.-M. Jodoin, Y. Benezeth, and Y. Wang. Meta-tracking for video scene understanding. In *Proc. IEEE Intern. Conf. Advanced Video and Signal-based Surveillance*, pages 1–6, 2013.

[108] P.-M. Jodoin, L. Maddalena, A. Petrosino, and Y. Wang. Extensive benchmark and survey of modeling methods for scene background initialization. *Accepted by Transactions on Image Processing*, 2017.

[109] L. Juan, J. Dengbiao, L. Bo, R. Yaduan, and C. Qimei. A nonparametric approach to foreground detection in dynamic backgrounds. *IEEE J. China Commun.*, 12 (2):32–39, 2015.

[110] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based Surveillance Syst.*, pages 135–144. Springer, 2002.

[111] M. Karaman, L. Goldmann, D. Yu, and T. Sikora. Comparison of static background segmentation methods. In *Proc. SPIE Visual Commun. and Image Process.*, pages 1–12, 2005.

[112] K. Karman and A. von Brandt. Moving object recognition using an adaptive background memory. *Time-varying Image Processing and Moving Object Recognition*, 2:297–307, 1990.

[113] I. Kavasidis, S. Palazzo, R. Di Salvo, D. Giordano, and C. Spampinato. An innovative web-based collaborative platform for video annotation. *Multimedia Tools Appl.*, 70(1):413–432, 2014.

[114] H. Kim, R. Sakamoto, I. Kitahara, T. Toriyama, and K. Kogure. Robust foreground extraction technique using gaussian family model and multiple thresholds. In *Asian Conf. Comput. Vis.*, pages 758–768, 2007.

[115] 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.

[116] T. Kim, S. Lee, and J. Paik. *Evolutionary Algorithm-Based Background Generation for Robust Object Detection*, pages 542–552. Springer Berlin Heidelberg, 2006.

[117] T. Kim, S. Lee, and J. Paik. Evolutionary algorithm-based background generation for robust object detection. In *Intern. Conf. Intelligent Computing*, pages 542–552. Springer, 2006.

[118] D. Kit, B. Sullivan, and D. Ballard. Novelty detection using growing neural gas for visuo-spatial memory. In *Proc. IEEE/RSJ Conf. Intell. Robots Syst.*, pages 1194–1200, 2011.

[119] J. Kittler, M. Hatef, R. P. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Machine Intell.*, 20(3):226–239, 1998.

[120] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *arXiv preprint arXiv:1210.5644*, 2012.

[121] F. Kristensen, P. Nilsson, and V. Öwall. Background segmentation beyond rgb. In *Asian Conf. Comput. Vis.*, pages 602–612, 2006.

[122] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Rapport technique, University of Toronto, 2009.

[123] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information process. systems*, pages 1097–1105, 2012.

[124] P. Kumar, S. Ranganath, and W. Huang. Queue based fast background modelling and fast hysteresis thresholding for better foreground segmentation. In *Proc. IEEE Conf. Inf., Commun. Signal Process.*, volume 2, pages 743–747, 2003.

[125] L. I. Kuncheva, C. J. Whitaker, C. A. Shipp, and R. P. Duin. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis & Applications*, 6 (1):22–31, 2003.

[126] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM Trans. Graph.*, volume 22-3, pages 277–286, 2003.

[127] C. Q. Lai and S. S. Teoh. A review on pedestrian detection techniques based on histogram of oriented gradient feature. In *Proc. IEEE Student Conf. Research Development*, pages 1–6, 2014.

[128] B. Laugraud, S. Piérard, and M. Van Droogenbroeck. Labgen-p: A pixel-level stationary background generation method based on labgen. In *Proc. IEEE Workshop on Intern. Conf. Pattern Recognit.*, pages 107–113, 2016.

[129] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[130] B. Lee and M. Hedley. Background estimation for video surveillance. In *Image vis. comput. New Zealand*, pages 315–320, 2002.

[131] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Trans. Image Process.*, 13 (11):1459–1472, 2004.

[132] W.-T. Li, H.-S. Chang, K.-C. Lien, H.-T. Chang, and Y.-C. F. Wang. Exploring visual and motion saliency for automatic video object extraction. *IEEE Trans. Image Process.*, 22(7):2600–2610, 2013.

[133] W. Li, V. Mahadevan, and N. Vasconcelos. Anomaly detection and localization in crowded scenes. *IEEE Trans. Pattern Anal. Machine Intell.*, 36(1):18–32, 2014.

[134] Y. Li. On incremental and robust subspace learning. *Pattern Recognit.*, 37(7): 1509–1518, 2004.

[135] D. Liang and S. Kaneko. Improvements and experiments of a compact statistical background model. *arXiv preprint arXiv:1405.6275*, 2014.

[136] C.-C. Lien, Y.-M. Jiang, and L.-G. Jang. Large area video surveillance system with handoff scheme among multiple cameras. In *Mach. Vis. Appl.*, pages 463–466, 2009.

[137] C.-F. Lin, C.-S. Chen, W.-J. Hwang, C.-Y. Chen, C.-H. Hwang, and C.-L. Chang. Novel outline features for pedestrian detection system with thermal images. *Pattern Recognit.*, pages 3440–3450, 2015.

[138] H.-H. Lin, T.-L. Liu, and J.-H. Chuang. A probabilistic svm approach for background scene initialization. In *Proc. IEEE Intern. Conf. Image Process.*, volume 3, pages 893–896, 2002.

[139] H.-H. Lin, T.-L. Liu, and J.-H. Chuang. Learning a scene background model via classification. *IEEE Trans. Signal Process.*, 57(5):1641–1654, 2009.

[140] D. Liu, X. Wang, and J. Song. A robust pedestrian detection based on corner tracking. In *Intern. Conf. Inf. Sci. Tech.*, pages 207–211, 2015.

[141] W. Liu, Y. Cai, M. Zhang, H. Li, and H. Gu. Scene background estimation based on temporal median filter with gaussian filtering. In *Proc. IEEE Workshop on Intern. Conf. Pattern Recognit.*, pages 132–136, 2016.

[142] D. F. Llorca, I. Parra, R. Quintero, C. Fernández, R. Izquierdo, and M. Sotelo. Stereo-based pedestrian detection in crosswalks for pedestrian behavioural modelling assessment. In *Intern. Conf. Inf. Control, Automa. Robot.*, pages 102–109, 2014.

[143] B. Lo and S. Velastin. Automatic congestion detection system for underground platforms. In *Proc. IEEE Symp. Intell. Multimedia Video Speech Process.*, pages 158–161, 2001.

[144] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3431–3440, 2015.

[145] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, volume 2, pages 1150–1157, 1999.

[146] X. Lu. A multiscale spatio-temporal background model for motion detection. In *Proc. IEEE Intern. Conf. Image Process.*, pages 3268–3271, 2014.

[147] X. Lu and R. Manduchi. Fast image motion segmentation for surveillance applications. *Image Vis. Comput.*, 29(2):104–116, 2011.

[148] Z. Luo, P.-M. Jodoin, S.-Z. Li, and S.-Z. Su. Traffic analysis without motion features. In *Proc. IEEE Intern. Conf. Image Process.*, pages 3290–3294, 2015.

[149] L. Maddalena and A. Petrosino. The SOBS algorithm: what are the limits? In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, 2012.

[150] L. Maddalena and A. Petrosino. Background model initialization for static cameras. In T. Bouwmans, F. Porikli, B. Hoferlin, and A. Vacavant, editors, *Background Modeling Foreground Detection Video Surveillance*, pages 1–16. Chapman and Hall/CRC, 2014.

[151] L. Maddalena and A. Petrosino. Extracting a background image by a multimodal scene background model. In *Proc. IEEE Workshop on Intern. Conf. Pattern Recognit.*, pages 143–148, 2016.

[152] L. Maddalena and A. Petrosino. A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Trans. Image Process.*, 17(7): 1168–1177, 2008.

[153] L. Maddalena and A. Petrosino. A fuzzy spatial coherence-based approach to background/foreground separation for moving object detection. *Neural Computing and Appl.*, 19(2):179–186, 2010.

[154] L. Maddalena and A. Petrosino. The sobs algorithm: what are the limits? In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 21–26, 2012.

[155] L. Maddalena and A. Petrosino. Towards benchmarking scene background initialization. In *Intern. Conf. Image Anal. Process.*, pages 469–476, 2015.

[156] N. J. McFarlane and C. P. Schofield. Segmentation and tracking of piglets in images. *Mach. Vis. Appl.*, 8(3):187–193, 1995.

[157] J. M. McHugh, J. Konrad, V. Saligrama, and P.-M. Jodoin. Foreground-adaptive background subtraction. *IEEE Signal Process. Lett.*, 16(5):390–393, 2009.

[158] E. Mémin and P. Pérez. Dense estimation and object-based segmentation of the optical flow with robust techniques. *IEEE Trans. Image Process.*, 7(5):703–719, 1998.

[159] T. Minematsu, A. Shimada, and R.-I. Taniguchi. Background initialization based on bidirectional analysis and consensus voting. In *Proc. IEEE Workshop on Intern. Conf. Pattern Recognit.*, pages 126–131, 2016.

[160] A. Miron and A. Badii. Change detection based on graph cuts. In *Proc. IEEE Conf. Syst., Signals and Image Process.*, pages 273–276, 2015.

[161] A. Mittal and N. Paragios. Motion-based background subtraction using adaptive kernel density estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, volume 2, pages 1–8, 2004.

[162] A. Morde, X. Ma, and S. Guler. Learning a background model for change detection. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 15–20, 2012.

[163] Y. Nakashima, N. Babaguchi, and J. Fan. Automatic generation of privacy-protected videos using background estimation. In *Proc. IEEE Conf. Multimedia Expo*, pages 1–6, 2011.

[164] M. Narayana, A. Hanson, and E. Learned-Miller. Background modeling using adaptive pixelwise kernel variances in a hybrid feature space. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2104–2111, 2012.

[165] M. Narayana, A. R. Hanson, and E. G. Learned-Miller. Improvements in joint domain-range modeling for background subtraction. In *British Machine Vis. Conf.*, pages 1–11, 2012.

[166] M. Narayana, A. R. Hanson, and E. G. Learned-Miller. Background subtraction: separating the modeling and the inference. *Mach. Vis. Appl.*, 25(5):1163–1174, 2014.

[167] J. C. Nascimento and J. S. Marques. Performance evaluation of object detection algorithms for video surveillance. *IEEE Trans. Multimedia*, 8(4):761–774, 2006.

[168] Y. Nonaka, A. Shimada, H. Nagahara, and R.-i. Taniguchi. Evaluation report of integrated background modeling based on spatio-temporal features. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 9–14, 2012.

[169] N. M. Oliver, B. Rosario, and A. P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Trans. Pattern Anal. Machine Intell.*, 22 (8):831–843, 2000.

[170] D. Ortego and J. C. SanMiguel. Stationary foreground detection for video-surveillance based on foreground and motion history images. In *Proc. IEEE Intern. Conf. Advanced Video and Signal-based Surveillance*, pages 75–80, 2013.

[171] D. Ortego, J. C. SanMiguel, and J. M. Martínez. Rejection based multipath reconstruction for background estimation in video sequences with stationary objects. *Comput. Vis. Image Und.*, 147:23–37, 2016.

[172] W. Ouyang and X. Wang. A discriminative deep model for pedestrian detection with occlusion handling. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3258–3265, 2012.

[173] W. Ouyang and X. Wang. Single-pedestrian detection aided by multi-pedestrian detection. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3198–3205, 2013.

[174] S. Paisitkriangkrai, C. Shen, and A. van den Hengel. Pedestrian detection with spatially pooled features and structured ensemble learning. *IEEE Trans. Pattern Anal. Machine Intell.*, 38(6):1243–1257, 2016.

[175] T. Parag, A. Elgammal, and A. Mittal. A framework for feature selection for background subtraction. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, volume 2, pages 1916–1923, 2006.

[176] G. Park. Background initialization by spatiotemporal similarity. *J. Korean Society Broadcast Eng.*, 12(3):289–292, 2007.

[177] J. Park, A. Tabb, and A. C. Kak. Hierarchical data structure for real-time background subtraction. In *Proc. IEEE Intern. Conf. Image Process.*, pages 1849–1852, 2006.

[178] D. H. Parks and S. S. Fels. Evaluation of background subtraction algorithms with post-processing. In *Proc. IEEE Intern. Conf. Advanced Video and Signal-based Surveillance*, pages 192–199, 2008.

[179] E. Parzen. On estimation of a probability density function and mode. *Annals Mathematical Stat.*, 33(3):1065–1076, 1962.

[180] M. Paul. Efficient video coding using optimal compression plane and background modelling. *Image Processing, IET*, 6(9):1311–1318, 2012.

[181] M. Piccardi. Background subtraction techniques: a review. In *Conf. IEEE Intern. Conf. Syst., Man and Cyb.*, volume 4, pages 3099–3104, 2004.

[182] F. Porikli. Multiplicative background-foreground estimation under uncontrolled illumination using intrinsic images. In *Proc. IEEE Conf. Appl. of Comput. Vis.*, volume 2, pages 20–27, 2005.

[183] F. Porikli and O. Tuzel. Bayesian background modeling for foreground detection. In *Proc. ACM Intern. Workshop on Video Surveillance Sensor Networks*, pages 55–58, 2005.

[184] F. Porikli and C. Wren. Change detection by frequency decomposition: Waveback. In *Proc. Workshop on Image Anal. for Multimedia Interactive Services*, pages 1–6, 2005.

[185] A. Prati, R. Cucchiara, I. Mikic, and M. M. Trivedi. Analysis and detection of shadows in video streams: A comparative evaluation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, volume 2, pages 571–576, 2001.

[186] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara. Detecting moving shadows: Algorithms and evaluation. *IEEE Trans. Pattern Anal. Machine Intell.*, 25(7): 918–923, 2003.

[187] J. Quesada and P. Rodriguez. Automatic vehicle counting method based on principal component pursuit background modeling. In *Proc. IEEE Intern. Conf. Image Process.*, pages 3822–3826, 2016.

[188] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: a systematic survey. *IEEE Trans. Image Process.*, 14(3):294–307, 2005.

[189] G. Ramírez-Alonso and M. I. Chacón-Murguía. Auto-adaptive parallel som architecture with a modular analysis for dynamic object segmentation in videos. *Neurocomputing*, 175:990–1000, 2016.

[190] V. Reddy, C. Sanderson, and B. C. Lovell. A low-complexity algorithm for static background estimation from cluttered image sequences in surveillance contexts. *EURASIP J. Image Video Process.*, 2011(1):1–14, 2011.

[191] D. Riahi, P. St-Onge, and G. Bilodeau. Rectgauss-tex: Blockbased background subtraction. *Technical Report EPM-RT-2012-03, Ecole Polytechnique de Montreal*, pages 1–9, 2012.

[192] P. L. Rosin and E. Ioannidis. Evaluation of global image thresholding for change detection. *Pattern Recognit. Lett.*, 24(14):2345–2356, 2003.

[193] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Trans. Graph.*, volume 23-3, pages 309–314, 2004.

[194] J. Rymel, J. Renno, D. Greenhill, J. Orwell, and G. A. Jones. Adaptive eigenbackgrounds for object detection. In *Proc. IEEE Intern. Conf. Image Process.*, volume 3, pages 1847–1850, 2004.

[195] H. Sajid and S.-C. S. Cheung. Background subtraction for static & moving camera. In *Proc. IEEE Intern. Conf. Image Process.*, pages 4530–4534, 2015.

[196] A. Schick, M. Bäuml, and R. Stiefelhagen. Improving foreground segmentations with probabilistic superpixel markov random fields. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 27–31, 2012.

[197] M. Sedky, M. Moniri, and C. Chibelushi. Object segmentation using full-spectrum matching of albedo derived from colour images. *US patent no. 2374109 12.10*, 2011.

[198] M. Sedky, M. Moniri, and C. C. Chibelushi. Spectral-360: A physics-based technique for change detection. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 399–402, 2014.

[199] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3626–3633, 2013.

[200] M. J. Shafiee, P. Siva, P. Fieguth, and A. Wong. Embedded motion detection via neural response mixture background modeling. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 19–26, 2016.

[201] M. Shah, J. D. Deng, and B. J. Woodford. Video background modeling: recent approaches, issues and our proposed techniques. *Mach. Vis. Appl.*, 25(5):1105–1119, 2014.

[202] Y. Sheikh and M. Shah. Bayesian modeling of dynamic scenes for object detection. *IEEE Trans. Pattern Anal. Machine Intell.*, 27(11):1778–1792, 2005.

[203] V. Sikri. Proposition and comprehensive efficiency evaluation of a foreground detection algorithm based on optical flow and canny edge detection for video surveillance systems. In *Proc. IEEE Conf. Wireless Commun., Signal Process. and Networking*, pages 1466–1472, 2016.

[204] B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[205] A. Sobral, T. Bouwmans, and E.-h. Zahzah. Comparison of matrix completion algorithms for background initialization in videos. In V. Murino et al., editor, *Intern. Conf. Image Anal. Process.*, pages 510–518. Springer, 2015.

[206] A. C. Sobral and E.-h. Zahzah. Matrix and tensor completion algorithms for background model initialization: A comparative evaluation. *Pattern Recognit. Lett.*, pages 1–12, 2017.

[207] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin. Flexible background subtraction with self-balanced local sensitivity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 408–413, 2014.

[208] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin. A self-adjusting approach to change detection based on background word consensus. In *Proc. IEEE Conf. Appl. of Comput. Vis.*, pages 990–997, 2015.

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

[210] C. Stauffer and E. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(8):747–757, 2000.

[211] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, volume 2, pages 246–252, 1999.

[212] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J. M. Buhmann. Topology free hidden markov models: Application to background modeling. In *Proc. IEEE Intern. Conf. Comput. Vis.*, volume 1, pages 294–301, 2001.

[213] J. K. Suhr, H. G. Jung, G. Li, and J. Kim. Mixture of gaussians-based background subtraction for bayer-pattern image sequences. *IEEE Trans. Circuits and Syst. for Video Tech.*, 21(3):365–370, 2011.

[214] D. M. Tax, M. Van Breukelen, R. P. Duin, and J. Kittler. Combining multiple classifiers by averaging or by multiplying? *Pattern Recognit.*, 33(9):1475–1485, 2000.

[215] M. Tepper, A. Newson, P. Sprechmann, and G. Sapiro. Multi-temporal foreground detection in videos. In *Proc. IEEE Intern. Conf. Image Process.*, pages 4599–4603, 2015.

[216] Y.-L. Tian and A. Hampapur. Robust salient motion detection with complex background for real-time video surveillance. In *Proc. IEEE Conf. Appl. of Comput. Vis.*, volume 2, pages 30–35, 2005.

[217] F. Tiburzi, M. Escudero, J. Bescós, and J. M. Martínez. A ground truth for motion-based video-object segmentation. In *Proc. IEEE Intern. Conf. Image Process.*, pages 17–20, 2008.

[218] D. Tomè, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi, and S. Tubaro. Deep convolutional neural networks for pedestrian detection. *Signal Process. Image Commun.*, 47:482–489, 2016.

[219] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Proc. IEEE Intern. Conf. Comput. Vis.*, volume 1, pages 255–261, 1999.

[220] A. Vacavant, T. Chateau, A. Wilhelm, and L. Lequièvre. A benchmark dataset for outdoor foreground/background extraction. In *Asian Conf. Comput. Vis.*, pages 291–300, 2012.

[221] M. Van Droogenbroeck and O. Paquot. Background subtraction: Experiments and improvements for vibe. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 32–37, 2012.

[222] S. Varadarajan, P. Miller, and H. Zhou. Spatial mixture of gaussians for dynamic background modelling. In *Proc. IEEE Intern. Conf. Advanced Video and Signal-based Surveillance*, pages 63–68, 2013.

[223] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proc. ACM Intern. Conf. Multimedia*, pages 689–692, 2015.

[224] R. Vezzani and R. Cucchiara. Video surveillance online repository (visor): an integrated framework. *Multimedia Tools Appl.*, 50(2):359–380, 2010.

[225] B. Wang and P. Dudek. A fast self-tuning background subtraction algorithm. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 395–398, 2014.

[226] H. Wang and D. Suter. A novel robust statistical method for background initialization and visual surveillance. In *Asian Conf. Comput. Vis.*, pages 328–337, 2006.

[227] H. Wang and D. Suter. A novel robust statistical method for background initialization and visual surveillance. In *Asian Conf. Comput. Vis.*, pages 328–337, 2006.

[228] H. Wang and D. Suter. A consensus-based method for tracking: Modelling background scenario and foreground appearance. *Pattern Recognit.*, 40(3):1091–1105, 2007.

[229] J. Wang, W. Fu, J. Liu, and H. Lu. Spatiotemporal group context for pedestrian counting. *IEEE Trans. Circuits Syst. Video Tech.*, 24(9):1620–1630, 2014.

[230] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Trans. Graph.*, 24(3):585–594, 2005.

[231] K. Wang, Y. Liu, C. Gou, and F.-Y. Wang. A multi-view learning approach to foreground detection for traffic surveillance applications. *IEEE Trans. Video Tech.*, 65(6):4144–4158, 2016.

[232] M. Wang, W. Li, and X. Wang. Transferring a generic pedestrian detector towards specific scenes. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3274–3281, 2012.

[233] R. Wang, F. Bunyak, G. Seetharaman, and K. Palaniappan. Static and moving object detection using flux tensor with split gaussian models. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 414–418, 2014.

[234] W.-H. Wang and R.-C. Wu. Fusion of luma and chroma gmms for hmm-based object detection. In *Pacific-Rim Symp. Image Video Tech.*, pages 573–581, 2006.

[235] X. Wang, X. Ma, and W. E. L. Grimson. Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models. *IEEE Trans. Pattern Anal. Machine Intell.*, 31(3):539–555, 2009.

[236] X. Wang, M. Wang, and W. Li. Scene-specific pedestrian detection for static video surveillance. *IEEE Trans. Pattern Anal. Machine Intell.*, 36(2):361–374, 2014.

[237] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar. Cdnet 2014: an expanded change detection benchmark dataset. In *Proc. IEEE Workshop on Comput. Vis. Pattern Recognit.*, pages 387–394, 2014.

[238] Y. Wang, S. Piérard, S.-Z. Su, and P.-M. Jodoin. Nonlinear background filter to improve pedestrian detection. In *Intern. Conf. Image Anal. Process.*, pages 535–543, 2015.

[239] Y. Wang, Z. Luo, and P.-M. Jodoin. Interactive deep learning method for segmenting moving objects. *Pattern Recognit. Lett.*, pages 1–10, 2016.

[240] Y. Wang, S. Piérard, S.-Z. Su, and P.-M. Jodoin. Improving pedestrian detection using motion-guided filtering. *Pattern Recognit. Lett.*, pages 1–7, 2016.

[241] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *Signals, Syst. Comput.*, volume 2, pages 1398–1402, 2003.

[242] S. K. Warfield, K. H. Zou, and W. M. Wells. Simultaneous truth and performance level estimation (staple): an algorithm for the validation of image segmentation. *IEEE Trans. Medical Imaging*, 23(7):903–921, 2004.

[243] N. Wiener. *Extrapolation, interpolation, and smoothing of stationary time series*, volume 2. MIT press Cambridge, 1949.

[244] L. Wixson. Detecting salient motion by accumulating directionally-consistent flow. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(8):774–780, 2000.

[245] C. Wojek and B. Schiele. A dynamic conditional random field model for joint labeling of object and scene classes. In *Proc. European Conf. Comput. Vis.*, pages 733–747, 2008.

[246] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Anal. Machine Intell.*, 19(7): 780–785, 1997.

[247] J. Wu and J. M. Rehg. CENTRIST: A visual descriptor for scene categorization. *IEEE Trans. Pattern Anal. Machine Intell.*, 33(8):1489–1501, 2011.

[248] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Trans. Pattern Anal. Machine Intell.*, 37(9):1834–1848, 2015.

[249] H. Xia, S. Song, and L. He. A modified gaussian mixture background model via spatiotemporal distribution with shadow detection. *Signal, Image and Video Process.*, 10(2):343–350, 2016.

[250] Y. Xia, R. Hu, Z. Wang, and T. Lu. Moving foreground detection based on spatio-temporal saliency. *Intern. J. Comput. Sci. Issues*, 10(1):79–84, 2013.

[251] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Trans. Syst. Man Cybern.*, 22(3):418–435, 1992.

[252] X. Xu and T. Huang. A loopy belief propagation approach for robust background estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1–7, 2008.

[253] Y. Xu, J. Dong, B. Zhang, and D. Xu. Background modeling methods in video analysis: A review and comparative evaluation. *CAAI Trans. Intell. Tech.*, pages 43–60, 2016.

[254] Z. Xu, P. Shi, and I. Y.-H. Gu. An eigenbackground subtraction method using recursive error compensation. In *Pacific-Rim Conf. Multimedia*, pages 779–787, 2006.

[255] D. K. Yadav and K. Singh. A combined approach of kullback–leibler divergence and background subtraction for moving object detection in thermal video. *Infrared Physics and Tech.*, 76:21–31, 2016.

[256] Y. Yalman and İ. ERTÜRK. A new color image quality measure based on yuv transformation and psnr for human vision system. *Turkish J. Electron. Eng. Comput. Sci.*, 21(2):603–612, 2013.

[257] B. Yang and L. Zou. Robust foreground detection using block-based rpca. *Optik-International Journal for Light and Electron Optics*, 126(23):4586–4590, 2015.

[258] S. Yoshinaga, A. Shimada, H. Nagahara, and R.-i. Taniguchi. Statistical local difference pattern for background modeling. *IPSJ Trans. Comput. Vis. Appl.*, 3: 198–210, 2011.

[259] S. Yoshinaga, A. Shimada, H. Nagahara, and R.-i. Taniguchi. Background model based on intensity change similarity among pixels. In *Korea-Japan Joint Workshop on Frontiers of Computer Vision*, pages 276–280, 2013.

[260] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural Inf. Process. Syst.*, pages 3320–3328, 2014.

[261] D. P. Young and J. M. Ferryman. Pets metrics: On-line performance evaluation service. In *Proc. IEEE Workshop on Visual Surveillance Performance Evaluation Tracking Surveillance*, pages 317–324, 2005.

[262] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[263] D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 628–635, 2013.

[264] S. Zhang, C. Bauckhage, and A. Cremers. Informed haar-like features improve pedestrian detection. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 947–954, 2014.

[265] X. Zhang, T. Huang, Y. Tian, and W. Gao. Background-modeling-based adaptive prediction for surveillance video coding. *IEEE Trans. Image Process.*, 23(2): 769–784, 2014.

[266] Y. Zhang, M. X. Li, and J. Hou. Application of non-parametric kernel density background modeling method in intelligent video surveillance system. In *Applied Mechanics and Materials*, volume 799, pages 1117–1120, 2015.

[267] C. Zhao, X. Wang, and W.-K. Cham. Background subtraction via robust dictionary learning. *EURASIP J. Image Video Process.*, 2011(1):1–12, 2011.

[268] X. Zhao, Z. He, S. Zhang, and D. Liang. Robust pedestrian detection in thermal infrared imagery using a shape distribution histogram feature and modified sparse representation classification. *Pattern Recognit.*, 48(6):1947–1960, 2015.

[269] J. Zheng, Y. Wang, N. Nihan, and M. Hallenbeck. Extracting roadway background image: Mode-based approach. *J. Transportation Research Board*, 1944: 82–88, 2006.

[270] J. Zhong and S. Sclaroff. Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In *Proc. IEEE Intern. Conf. Comput. Vis.*, pages 44–50, 2003.

[271] D. Zhou and H. Zhang. Modified gmm background modeling and optical flow for detection of moving objects. In *Conf. IEEE Intern. Conf. Syst., Man and Cyb.*, volume 3, pages 2224–2229, 2005.

[272] M. S. Zitouni, H. Bhaskar, and M. Al-Mualla. Robust background modeling and foreground detection using dynamic textures. In *Intern. Joint Conf. Comp. Vis., Imaging Comp. Graphics Theory Appl.*, pages 1–8, 2016.

[273] Z. Zivkovic and F. V. D. Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognit. Lett.*, 27:773–780, 2006. ISSN 0167-8655.

[274] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognit. Lett.*, 27 (7):773–780, 2006.