SuBSENSE: A Universal Change Detection Method With Local Adaptive Sensitivity

Pierre-Luc St-Charles, *Student Member, IEEE*, Guillaume-Alexandre Bilodeau, *Member, IEEE*, and Robert Bergevin, *Member, IEEE*

***Abstract*— Foreground/background segmentation via change detection in video sequences is often used as a stepping stone in high-level analytics and applications. Despite the wide variety of methods that have been proposed for this problem, none has been able to fully address the complex nature of dynamic scenes in real surveillance tasks. In this paper, we present a universal pixel-level segmentation method that relies on spatiotemporal binary features as well as color information to detect changes. This allows camouflaged foreground objects to be detected more easily while most illumination variations are ignored. Besides, instead of using manually set, frame-wide constants to dictate model sensitivity and adaptation speed, we use pixel-level feedback loops to dynamically adjust our method's internal parameters without user intervention. These adjustments are based on the continuous monitoring of model fidelity and local segmentation noise levels. This new approach enables us to outperform all 32 previously tested state-of-the-art methods on the 2012 and 2014 versions of the ChangeDetection.net dataset in terms of overall F-Measure. The use of local binary image descriptors for pixel-level modeling also facilitates high-speed parallel implementations: our own version, which used no low-level or architecture-specific instruction, reached real-time processing speed on a midlevel desktop CPU. A complete C++ implementation based on OpenCV is available online.**

***Index Terms*— Background subtraction, change detection, foreground segmentation, surveillance, spatiotemporal features, video signal processing.**

I. Introduction

THE use of change detection algorithms to identify regions of interest in video sequences has long been a stepping stone in high level surveillance applications. In their simplest form, they allow the subtraction of static background from scenes where relevant objects are always in motion. In most cases however, "foreground" objects may move intermittently (e.g. cars at a traffic light), they may not be focal points in a camera's field of view, and uninteresting background regions may also exhibit dynamic behavior (e.g. swaying tree branches, water fountains). Simplistic background subtraction methods and traditional image segmentation approaches are thus ill-suited for active region labeling in real video surveillance tasks. Modern change detection algorithms are generally split into three parts: first, a background model of the scene is created and periodically updated by analyzing frames from the video sequence. Then, preliminary foreground/background segmentation labels (or probabilities) are assigned to all pixels of every new frame based on their similarity to the model. Finally, regularization is used to combine information from neighboring pixels and to make sure uniform regions are assigned homogeneous labels.

Background modeling can be approached in many different ways: to allow high-speed implementations, most methods rely on independent pixel-level models which are assembled into a larger background model. Color intensities are typically used to characterize local pixel representations in non-parametric models or for local distribution estimation in parametric models. Due to its simplicity, this kind of approach cannot directly account for spatiotemporal coherence between pixels and instead delegates this responsibility to the regularization step. This often culminates in less-than-ideal segmentation results due to the absence of texture analysis, especially when camouflaged foreground objects are involved.

Because this is a two-class segmentation problem and because of the wide range of possible scenarios, parameters that control model sensitivity and adaptation rate are usually left to the user to define. These can be very difficult to adjust for particular problems, especially when illumination variations, dynamic background elements and camouflaged objects are all present in a scene at the same time. Additionally, good knowledge of the data and of the change detection algorithm itself is required to achieve optimal performance in any given situation. Previously, most methods have used global thresholds under the assumption that all observations would show similar behavior throughout the analyzed sequences, which is rarely the case. While it is possible to dynamically adjust such parameters based on comparisons between observations and values predicted by the model, this approach cannot be used continuously due to disparities caused by foreground objects. In all cases, a typical outcome of bad parameterization is segmentation noise: usually observed under the form of "blinking pixels" (i.e. pixels that often switch between foreground and

Manuscript received July 17, 2014; revised September 18, 2014; accepted November 18, 2014. Date of publication December 4, 2014; date of current version December 22, 2014. This work was supported in part by the Fonds de Recherche du Québec—Nature et Technologies (FRQNT) under Grant 2014-PR-172083 and in part by the Regroupement pour l'étude des environnements partagés intelligents répartis FRQ-NT strategic cluster. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Amit K. Roy Chowdhury.

P.-L. St-Charles and G.-A. Bilodeau are with the Laboratoire d'Interprétation et de Traitement d'Images et Vidéo, École Polytechnique de Montréal, Montréal, QC H3T 1J4, Canada (e-mail: pierre-luc.st-charles@polymtl.ca; gabilodeau@polymtl.ca).

R. Bergevin is with the Laboratoire de Vision et Systèmes Numériques, Université Laval, Québec, QC G1V 0A6, Canada (e-mail: robert.bergevin@gel.ulaval.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2014.2378053

background classification over time), such noise indicates that the model is too sensitive to change in a certain region.

In this paper, we present a *universal* method for change detection, meaning it can be directly applied on most video surveillance scenarios and still result in near-optimal performance. Our work has already been partially described in [1] and [2]; here, we give new and more detailed explanations on the different parts of our method while providing in-depth analyses of new results. In short, we first use an improved spatiotemporal binary similarity descriptors along with color intensities to characterize pixel representations in a nonparametric paradigm. This novel approach allows the detection of subtle local changes caused by camouflaged foreground objects, which are not typically visible at the pixel level. It also ignores disturbances caused by soft shadows and other types of illumination variations or noise. Pixel representations are captured and stored as "samples" in pixel-level models, which as a whole form our background model. Due to the conservative update strategy and neighbor-spread rules used to update these samples, our model is resistant to shaking cameras and intermittent object motion.

Then, in order to improve out-of-the-box flexibility in complex scenarios, we propose a new feedback scheme to dynamically control our algorithm's sensitivity and adaptation speed. This component is based on the continuous analysis of background dynamics at the pixel level: it treats frame regions differently based on recent observations, their similarity to pixel models as well as local segmentation noise levels. While segmentation noise is often seen as detrimental in foreground/background labeling tasks, we show that it can be used to guide feedback when it is managed correctly, thus solving the dynamic adjustment problem in the presence of foreground objects. In our case, the highly sensitive nature of our pixel representations allows sparse noise to be generated very easily, also improving the efficiency of our approach. Ultimately, our feedback scheme can identify complex motion patterns and adjust our model's parameters locally without compromising segmentation performance elsewhere.

To keep our method's complexity minimal and its implementation simple, we avoid using preprocessing and color conversion/normalization on analyzed frames. Furthermore, our segmentation decisions do not rely on pixel-wise probabilities or on an energy function, and require no region-level or object-level processing. Our regularization step is also simplified to morphological operations and median blurring, which are able to eliminate all salt-and-pepper segmentation noise. A complete evaluation on the 2012 and 2014 versions of the ChangeDetection.net (CDnet) dataset [3], [4] shows that we outperform all 32 previously ranked methods in terms of overall *F-Measure*, as well as in nine out of eleven categories (including baseline). These methods include ViBe [5] and ViBe+ [6], which use a similar sample-based background model that only relies on color, and PBAS [7], which uses a feedback scheme that does not monitor segmentation noise for its adjustments. In the end, our results demonstrate that our approach is suitable for most complex change detection challenges, but also that good generalization is not earned at the expense of performance in simple scenarios. The full C++ implementations of our method's two stages presented in Section III (i.e. with and without feedback) are available online.[1]

II. Related Work

Most methods used for change detection in video sequences are based on the idea that, when using stationary cameras, disparities between an analyzed frame and a background reference are usually indicative of foreground objects. The advantage behind this concept is that no prior knowledge is required to detect the objects, as long as their appearance differs enough from the background (i.e. they are not camouflaged). As opposed to solutions based on object detection, this approach can accurately describe the contour of moving objects instead of simply returning their bounding box. However, finding a good reference image in order to do actual "background subtraction" is almost always impossible due to the dynamic nature of real-world scenes.

Instead of relying on an existing reference image for change detection, the earliest adaptive methods used pixel-wise intensity averages and Kalman filtering to create parametric background models from which comparisons are made. This kind of approach is robust to noise and can slowly adapt to global illumination variations, but is generally inadequate against shadows and multimodal background regions. Gaussian Mixture Models (GMM) [8], [9] were introduced to solve the latter problem and remain to this day a very popular solution. They allow dynamic background elements to be modeled through color intensities at individual pixel locations using a mixture of Gaussian probability density functions. New adaptive and more flexible variations of GMM were also proposed over the years [10]–[13] to allow dynamic numbers of components for modeling as well as to improve their convergence rate.

Nonparametric models based on Kernel Density Estimation (KDE) [14] were also introduced early on and improved in more recent state-of-the-art methods [11], [15], [16]. Unlike parametric models, these rely directly on local intensity observations to estimate background probability density functions at individual pixel locations. Most of them however only incorporate observations on a first-in, first-out basis, and are thus unable to model both long-term and short-term periodic events without holding on to large amounts of data. The stochastic sampling approach presented in [5] and [17] and improved in [6] and [7] solves this problem by using a random observation replacement policy in its model. Note that our own approach is derived from it. The codebook methods of [18] and [19] present another alternative to solve this problem: they cluster observations into codewords and store them in local dictionaries, allowing for a wider range of representations to be kept in the background model. Kim et al. also proposed in [18] a normalized color distance measure that quickly gained popularity in change detection due to its relatively low cost and robustness to illumination variations. Non-parametric models are also often considered for hardware [5] or high-speed parallel implementations due to their

[1] Available at https://bitbucket.org/pierre_luc_st_charles.

data-driven nature. For example, the method of [20], which relies on pixel-level template matching, reported on the CDnet website exceeding 800 frames per second while processing 320×240 videos using a single mid-level GPU (GTX 460).

Unprecedented methods based on artificial neural networks [21], [22] have been proposed and achieve good results on various change detection scenarios without prior knowledge of the involved motion patterns. However, they require a training period of variable length depending on the presence of foreground objects in the early frames of the video sequences. To address this problem, weightless neural networks based on binary nodes were instead used for online learning in [23]. Other works have also focused on improving foreground/background label coherence using advanced regularization techniques based on connected components [6], [24], superpixels and Markov Random Fields [25], or even static/abandoned object detection [24], [26]. Some methods instead rely on region-level [27], frame-level [28], [29] or hybrid frame/region-level [30], [31] comparisons to explicitly model the spatial dependencies of neighboring pixels. The recently proposed method of Wang et al. [32] also relies on a hybrid, multi-level system: it combines flux tensor-based motion detection and classification results from a Split Gaussian Mixture Model (SGMM), and uses object-level processing to differentiate immobilized foreground objects from "ghost" artifacts.

The use of local binary descriptors to improve the spatial awareness of change detection methods has also been studied: Heikkilä and Pietikäinen were the first to propose a solution based on Local Binary Patterns (LBP) [33]. Their method was demonstrated to be tolerant to illumination variations and robust against multimodal background regions. This robustness is achieved using LBP histograms at the pixel level, but at the expense of sensitivity to subtle local texture changes. An improved version of this method was proposed by Zhang et al. in [34]: by computing LBP feature histograms on consecutive frames and then merging them, they benefit from using underexploited temporal motion information. Another approach was suggested in [35] that does not require histograms: Scale-Invariant Local Ternary Patterns (SILTP) can be used directly at the pixel level to detect local changes when incorporated in a modified KDE framework. Finally, Local Binary Similarity Patterns (LBSP) were demonstrated in [36] to surpass traditional color comparisons when used to detect subtle changes in baseline scenarios via Hamming distance thresholding. We chose to improve upon LBSP features for our complete method due to their simplicity and effectiveness in change detection (detailed in Section III-A). We also chose not to use histograms in our model, and instead we directly incorporate LBSP descriptors in a simplified density estimation framework based on a step kernel (like the one used in [5]).

As for the use of feedback mechanisms to adjust parameters on-the-fly in this context, many parametric methods (such as [9]–[11]) already used local variance analysis or comparison results to guide segmentation behavior. Moreover, post-processing components are sometimes used to trigger model update mechanisms (like in the case of static object detectors [24], [26] or frame-level components [30]). However, feedback is rarely used solely at the pixel level to control both change detection sensitivity and model update rate. A notable exception is the Pixel-Based Adaptive Segmenter (PBAS) [7], which exploits "background dynamics" (i.e. the study of background motion patterns and model fidelity) to control local decision thresholds and update rates. In Section III-C, we improve upon their design by allowing continuous background monitoring and by restraining parameter adjustments to regions with unstable segmentation behavior. These regions are identified through the detection of blinking pixels, which were also used heuristically in [6] to guide model updates.

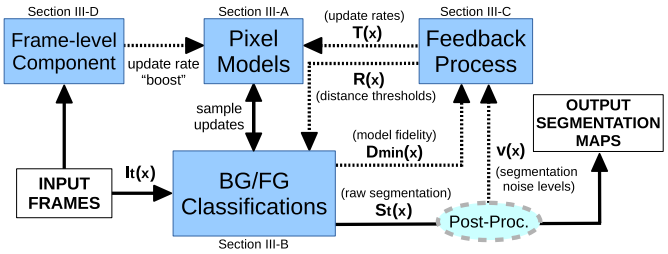

Fig. 1. Block diagram of SuBSENSE; dotted lines indicate feedback relations. The role of each block and variable is detailed in sections III-A through III-D. In our case, post-processing used to generate the output segmentation maps from raw labeling data is based on typical median filtering and blob smoothing operations. This component is an integral part of our method since it provides segmentation noise levels to our feedback process.

For more information on foreground/background segmentation via change detection, note that many review papers and surveys [37]–[41] have been published over the years on the subject.

III. Methodology

As stated in the previous section, our proposed approach is based on the adaptation and integration of Local Binary Similarity Pattern (LBSP) features in a nonparametric background model that is then automatically tuned using pixel-level feedback loops. We coined our complete method SuBSENSE, short for "Self-Balanced SENsitivity SEgmenter". We detail how it works in three steps: first, we show in Section III-A how individual pixels are characterized using spatiotemporal information based on RGB values and LBSP features; then, we present in Section III-B how these representations can be gathered, updated and used in a stochastic, sample-based model, resulting in a fairly simple and effective change detection method; finally, we show in Section III-C how model fidelity and segmentation noise monitoring drives the feedback scheme we used to control our algorithm's sensitivity and adaptation speed locally. Extra implementation details about our method and its frame-level analysis component are given in Section III-D.

An overview of SuBSENSE's architecture is presented in Fig. 1 based on a block diagram.

A. Pixel-Level Modeling

Pixel-level modeling, as opposed to region-level or object-level modeling, usually allows high-speed parallel implementations to be developed with relative ease due

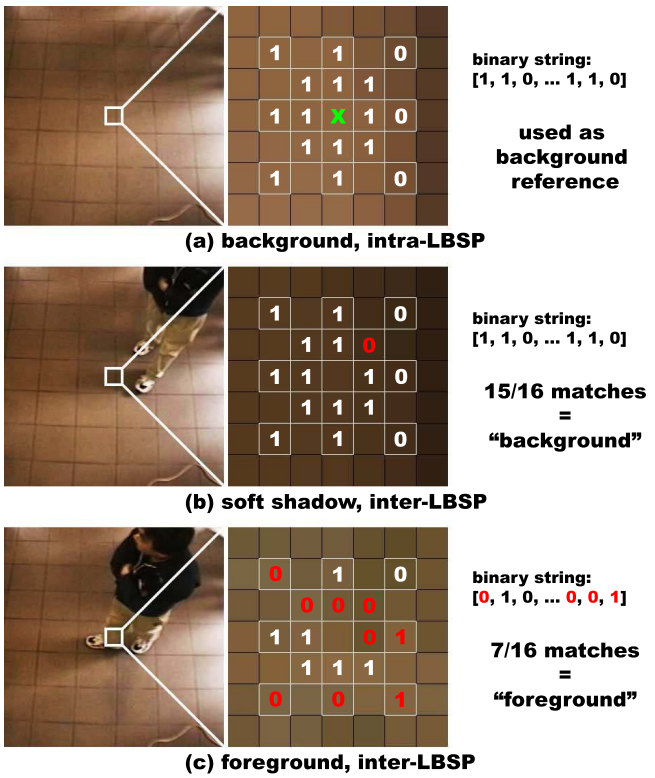

Fig. 2. Simplified description and comparison of LBSP features using frames picked from the copyMachine sequence of CDnet. In row (a), an intra-LBSP feature is computed to serve as the basis for other comparisons; in rows (b) and (c), inter-LBSP features are computed using (a)'s reference intensity (the green x), while soft shadow and foreground are respectively visible. Row (b)'s feature can be matched with (a)'s since textures are similar and the relative intensity threshold is correctly scaled in bright-to-dark transitions; this is not the case in row (c). Note that in fact, LBSP features are computed and compared on each color channel.

to how the workload is already split and kept isolated at a low level. However, the absence of information sharing between such local models puts the entire burden of spatial (or spatiotemporal) labeling coherence on the method's regularization scheme. To counter this, we characterize pixel-level representations using not only their RGB values, but Local Binary Similarity Pattern (LBSP) features, which operate in the spatiotemporal domain. This approach improves the odds of detecting camouflaged objects when their texture differs from the background's, and can even tolerate illumination changes when all local color intensities vary equally over time. Moreover, these features have a very low computational cost, and are discriminative enough to be used directly in pixel models without relying on local histograms.

As described in [36] and shown in Fig. 2, LBSP features are computed on a predefined 5×5 grid. They can be considered a counterpart to Local Binary Pattern (LBP) and Local Ternary Pattern (LTP) features: instead of assigning binary codes based on whether a given adjoining intensity is lesser or greater than the central reference, they assign them based on similarity (via absolute difference thresholding). More specifically, the following equation is used to compute an LBSP binary string centered at a given location x:

$$LBSP(x) = \sum_{p=0}^{P-1} d(i_p\,,\, i_x) \cdot 2^p \tag{1}$$

with

$$d(i_p\,,\, i_x) = \begin{cases} 1 & \text{if } |i_p - i_x| \leq T_d \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where i_x is the "central reference" and corresponds to the intensity of the pixel at x, i_p corresponds to the intensity of the pth neighbor of x on the predefined pattern, and T_d is the internal similarity threshold. In short, LBSP features require the same number of internal operations per comparison as LBP and fewer than LTP, but are more discriminative in the context of change detection (see the experimental results of Section IV-A). Furthermore, the pattern seen in Fig. 2 covers more pixels than the basic version of LBP or LTP (typically the 8-connected neighbors) without having to interpolate intensity values. LBSP features can also be made sensitive to spatiotemporal variations by picking a central reference intensity (i_x) from a previous frame. This is called inter-LBSP by the original authors, in opposition to intra-LBSP when computations are kept within a single frame. Examples of inter-LBSP computations are also presented in Fig. 2.

The first major improvement we propose to traditional LBSP features is related to their internal threshold, T_d. As discussed in [35], the replacement of a similar threshold used in LTP by a term that is relative to the reference intensity (i_x) makes the binary descriptors much more resistant to illumination variations. In our case, due to the nature of inter-LBSP features, this modification becomes even more effective against shadows, as presented in the middle row of Fig. 2. Simply put, since we always use a reference intensity from a previous frame (Fig. 2.a, the green X mark) to compute inter-LBSP descriptors (Fig. 2.b and c), and since shadows are always characterized by bright-to-dark color transitions, the similarity threshold will remain high and local pixels are likely to stay classified as background. However, in the case of dark-to-bright transitions (which are less likely to be shadows, and more likely to be relevant changes), the similarity threshold will remain low, thus avoiding possible false background classifications. In short, to apply this modification to LBSP features, we only need to replace (2) by the following equation, where T_r is the new relative internal threshold (bound to [0,1]):

$$d(i_p\,,\, i_x) = \begin{cases} 1 & \text{if } |i_p - i_x| \leq T_r \cdot i_x \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

We determined experimentally that using $T_r \approx 0.3$ resulted in noise-free, discriminative descriptors in nearly all tested video sequences. However, for optimal flexibility in our final method, this value was automatically scaled over time based on the texture content of the analyzed scenes so that all videos would bear approximately the same overall gradient magnitude sums. That way, scenes with very little background texture or few object edges would rely on much lower T_r values than scenes with cluttered backgrounds, making the former much more sensitive to local texture variations than the latter. We measure the gradient magnitude of individual frames by summing the Hamming weights (rescaled to [0, 1]) of all its intra-LBSP descriptors. We then slowly adjust T_r

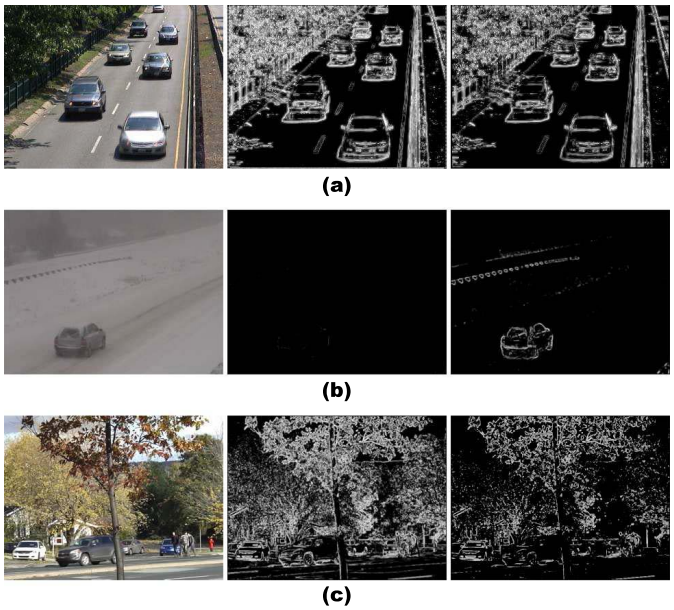

Fig. 3. Examples of gradient magnitude maps obtained in different CDnet sequences by computing the Hamming weight of dense intra-LBSP descriptors with $T_r = 0.3$ (central column) and an automatically set T_r (right-hand column); bright areas indicate highly textured regions and sharp edges, while dark areas indicate flat regions. (a) Highway #0826. (b) Blizzard #0182. (c) Fall #0289.

if this measure is too low or too high (i.e. outside a predetermined interval). In Fig. 3, we can see that the highway sequence of CDnet (a) is not affected by these automatic adjustments, as it presents a good balance between cluttered and flat regions. On the other hand, the mostly texture-less blizzard sequence (b) automatically settles for $T_r \approx 0.1$, which accentuates foreground object contours, and the fall sequence causes T_r to increase above 0.3, reducing potential segmentation noise induced by noisy texture patterns in trees.

In summary, for our nonparametric pixel models, we define a single background pixel representation (or "sample", as referred to in the following sections) in RGB space as a combination of color intensities (8-bit values) and intra-LBSP binary strings (16-bit vectors). We do not convert RGB values to a format with normalized brightness as we determined that the extra computational cost was not worth the improved resistance to illumination variations, to which LBSP features already provide ample flexibility. When trying to match a background sample to an actual observation on a given frame, we first compare color values using L1 distance. If a match is still possible after thresholding, we generate inter-LBSP binary strings using the color values of the sample as reference and using the 5×5 comparison grid on the current input frame. Then, inter-LBSP and intra-LBSP strings (see Fig. 2) are compared via Hamming distance thresholding. Therefore, to consider a background sample similar to a local observation, both color values and binary descriptors must be successfully matched. In contrast to a purely color-based approach, this two-step verification is inclined to reject more matches, thus to classify more pixels as foreground (and therefore generate more segmentation noise, as desired).

In the next section, we show how this kind of pixel-level modeling can be used in a basic sample consensus framework to achieve good change detection performance. Note that we tested our pixel models using different local binary descriptors and we present our results in Section IV-A.

B. Change Detection via Sample Consensus

To be properly used for change detection, our pixel-level modeling approach requires maintenance and classification rules that do not rely on the clustering or averaging of samples, as LBSP features cannot easily be combined. Thus, we opted for a basic sample consensus approach similar to ViBe's [5]: derived from [42], it determines if a given observation should be considered foreground or background based on its similarity to recently observed samples. This sample-based framework allows our *color-LBSP* pixel representations to be used effectively to detect even the most subtle local changes while staying quite robust to irrelevant motion in complex scenarios.

The way it works is rather simple: first, the background model, noted B, is formed through the combination of pixel models, which each contain a set of N recent background samples:

$$B(x) = \{B_1(x), B_2(x), \ldots, B_N(x)\} \tag{4}$$

These samples, as described in the previous section, are matched against their respective observation on the input frame at time t, noted $I_t(x)$, to classify the pixel at coordinate x as foreground (1) or background (0). A simplified version of this classification test where we ignore our two-step *color-LBSP* verification is presented in (5); pseudocode for the two-step approach is presented in [1].

$$S_t(x) = \begin{cases} 1 & \text{if } \#\left\{dist\left(I_t(x), B_n(x)\right) < R, \forall n\right\} < \#_{min} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where S_t is the output segmentation map, $dist\left(I_t(x), B_n(x)\right)$ returns the distance between the current observation and a given background sample, R is the maximum distance threshold and $\#_{min}$ is the minimum number of matches required for a background classification. In this context, a small R value means that the model has to be very accurate in order to successfully classify pixels as background. Using a larger R leads to better resistance against irrelevant change, but also makes it harder to detect foreground objects that are very similar to the background. We further discuss how this parameter is used to obtain individual color and LBSP distance thresholds in Section III-C.

We fixed $\#_{min} = 2$ for our method as it was demonstrated in [5]–[7], [17], and [42] to be a reasonable trade-off between noise resistance and computational complexity. However, the number of samples per pixel model (N) has to be raised above the usual $N = 20$ proposed in [5]: this is due to the larger representation space induced by LBSP features, which are described using 16 bits instead of 8. Typically, N is used to balance the precision and sensitivity of sample-based methods: using fewer samples leads to more sensitive but less precise models, and vice-versa. As it can be seen in Fig. 4, when using only color information, the overall *F-Measure* score (which indicates the "balanced" performance

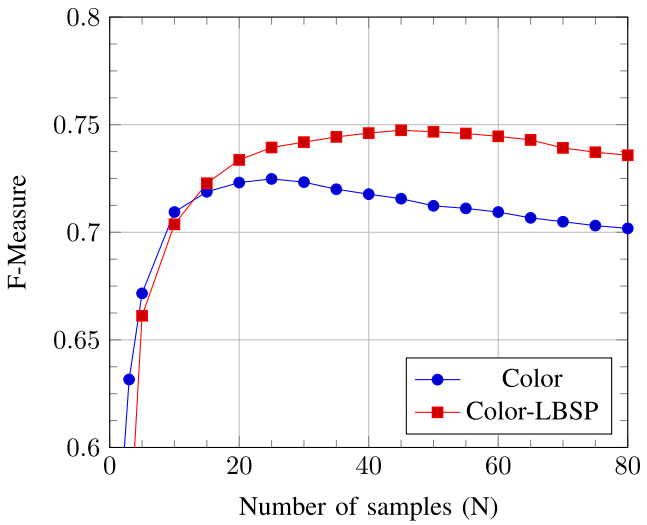

Fig. 4. Average F-Measure scores obtained on the 2012 CDnet dataset for different numbers of background samples, using *color-only* and *color-LBSP* configurations of our method (without feedback).

of an algorithm) tends to saturate when N reaches 20. Yet, with our own pixel-level modeling approach, it stabilizes at a higher value.

Although these values depend on the complexity of the studied scenes, we determined that based on the 2012 CDnet dataset, a minimum of $N = 35$ samples was required for our method to be used universally, with $N = 50$ being preferred for better precision. Increasing N does not directly affect the average pixel classification time in stable regions, as the first $\#_{min}$ are usually enough to break off the matching process. It can however lengthen it in dynamic background regions (where matches are sparser) and when foreground is present. As a side note, the relative processing time difference between these two configurations was about 15% on the entire 2012 CDnet dataset.

We update pixel models using a conservative, stochastic, two-step approach similar to [5]: first, every time a pixel at x is classified as background using (5), a randomly picked sample of $B(x)$ has a $1/T$ probability to be replaced by the observation at $I_t(x)$, where T is a "time subsampling factor", as defined in [5]. Then, one of the neighbors of $B(x)$ also has a $1/T$ probability of seeing one of its samples replaced by this same observation. This new parameter controls the adaptation speed of our background model: small values lead to high update probabilities (and thus to the rapid evolution of the model) and vice-versa. It is adjusted automatically, as described in the next section.

The fact that samples are replaced randomly instead of based on when they were last modified insures that a solid history of long-term and short-term background representations can be kept in our pixel models. Likewise, since new samples can only be inserted when a local pixel is recognized as background, this approach prevents static foreground objects from being assimilated too fast (as is often the case for methods using "blind update" strategies). In theory, this conservative approach implies that, given enough contrast, some foreground objects will never be incorporated into the background model. In practice, noise and camouflage always cause gradual foreground erosion, meaning that all static objects will eventually be classified as background.

The second update step described earlier (i.e. the "spatial diffusion" step, as named by the authors of [5]) allows regions that are homogeneous with the background to be absorbed much faster. In other words, ghost artifacts, which are commonly defined as falsely classified background regions due to the removal of an object from the observed scene, can be eliminated rapidly since they share many similarities with other parts of the background. Moreover, this same "diffusion" step improves the spatial coherency of the background model to the point where limited camera motion can be tolerated. Besides, relying on texture information prevents the spread of samples across object boundaries. Simply put, even if a sample is wrongfully transported from one region to another, the odds that it might be matched in the new region are much lower due to the use of LBSP features, which would detect a textural change near the border. In fact, a static foreground object with a color similar to the background may be correctly classified for a long time, given that its border is noticeable.

Overall, as presented in Section IV-A, this basic method is very robust against global illumination changes and soft shadows, and it can easily detect camouflaged foreground objects that are left unexposed when using a traditional color-based approach. However, using LBP-like features at the pixel level still results in false foreground classifications in most dynamic background regions, as textures are much harder to match than color values. The last part of our method, which is presented in the following section, benefits from this predicament.

C. Background Monitoring and Feedback Scheme

So far, we have seen how R, the maximum sample distance threshold, and T, the model update rate, are the two most important parameters in our method. They essentially control its precision and sensitivity to local changes and can determine how easily moving elements are integrated in the model. In [5] and [6], global values were determined empirically for both and used frame-wide. This kind of approach is flawed, as using a global strategy to control model maintenance and labeling decisions implies that all pixels will always present identical behavior throughout the analyzed video sequence. In reality, this assumption almost never holds since an observed scene can present background regions with different behaviors simultaneously, and these can vary over time. Moreover, even if it were possible to fix parameters frame-wide and obtain good overall performance, finding an optimal set of values for a specific application requires time as well as good knowledge of the method and dataset.

So, in our case, we consider R and T pixel-level state variables and adjust them dynamically to avoid these parameterization problems. Ideally, to increase overall robustness and flexibility, we would need to increase R in dynamic background regions to reduce the chance of generating false positives, and T wherever foreground objects are most likely to be immobilized to make sure they do not corrupt the model.

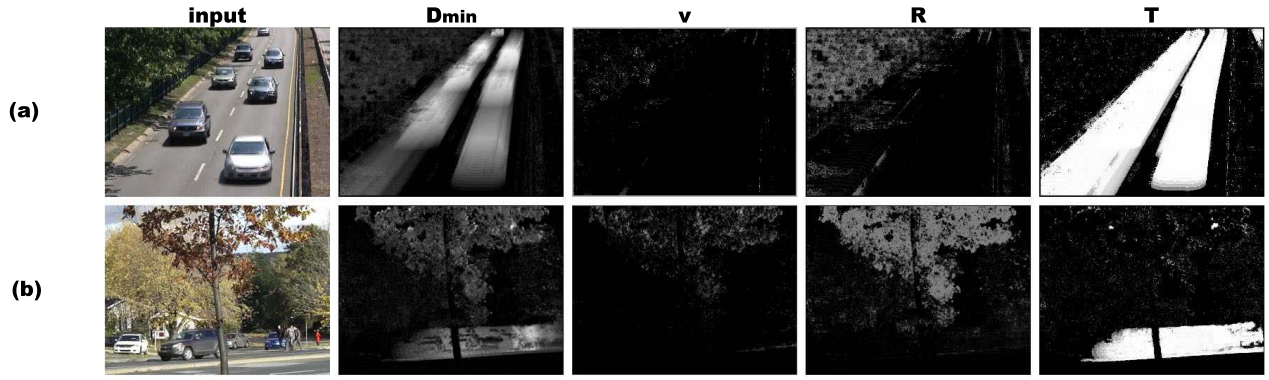

Fig. 5. Typical 2D distributions for our monitoring variables (D_{min} and v), local distance thresholds (R) and local update rates (T) on a baseline sequence of CDnet ("highway", row a), and on a sequence with important dynamic background elements ("fall", row b). In R's 2D map, bright areas indicate high distance thresholds (thus easier sample-observation matches), while in T, they indicate low update probabilities (meaning the pixel models stay mostly unchanged over time). We can notice in both cases that trees carry high R and low T values, and vice-versa for road; this means that foreground objects will more easily be detected on the road, and are less likely to corrupt the model over time. (a) Highway #0826. (b) Fall #0289.

However, wrongfully increasing T in dynamic regions can cause the model to stop adapting to useful background representations, and increasing R in static regions can worsen camouflage problems, leading in both cases to even more false classifications. Therefore, to properly adjust these variables, we first need to determine the nature of the region which overlies a given pixel x while avoiding region-level or object-level analyses because they are time-consuming.

A feedback approach based on the analysis of pixel-level background motion patterns ("background dynamics") was proposed in [7] for this purpose: it uses the results of recent comparisons between pixel models and local observations to control local distance thresholds and update rates. Their technique, albeit successful, has an important drawback: since local comparison results (i) cannot be used when x is classified as foreground, and (ii) are only used for feedback when $B(x)$ is updated, the response time of variable adjustments is rather long (especially in noisy regions). This means that intermittent dynamic background motion (e.g. swaying tree branches due to wind bursts) will still cause many false classifications.

What we propose instead is a feedback scheme based on a two-pronged background monitoring approach. Similar to [7], we first measure background dynamics based on comparison results between our pixel models and local observations, but we do this continuously, without any regards to classification results or model updates. This ensures an optimal response time to events in all observed frame regions. Additionally, we measure local segmentation noise levels based on the detection of blinking pixels. This allows dynamic background regions to be distinguished from static regions where foreground objects might be present, essentially guiding which local adjustments should be constrained to avoid over-adaptation and camouflage problems. Two new dynamic controllers for R and T are then introduced, which both rely on local "indicators" emanating from the monitoring of background dynamics and segmentation noise.

So, first of all, the idea behind analyzing background dynamics is to measure the motion entropy of a single pixel location over a small temporal window based on model fidelity. To obtain an indicator of such behavior, we use a recursive moving average, defined as

$$D_{min}(x) = D_{min}(x)\cdot(1-\alpha) + d_t(x)\cdot\alpha \tag{6}$$

where α is the learning rate, and $d_t(x)$ the minimal normalized *color-LBSP* distance between all samples in $B(x)$ and $I_t(x)$. Since $D_{min}(x)$ is bound to the [0,1] interval, an entirely static background region would have $D_{min}(x) \approx 0$, and a dynamic region to which the model cannot adapt to would have $D_{min}(x) \approx 1$. Following the same logic, areas with foreground objects would also present high D_{min} values since foreground detection is defined through disparities between pixel models and local observations. This is why [7] avoided using a similar continuous monitoring approach; in our case, it simply means that we cannot use this indicator by itself to control R and T, as both risk deteriorating when foreground objects stay in the same area for too long. This kind of behavior can be observed in Fig. 5: large foreground objects (in this case, cars), just like dynamic background elements, can steadily increase local $D_{min}(x)$ values.

The monitoring of blinking pixels can help solve the complication behind our continuous approach for D_{min}. In this case, it is similar to measuring the segmentation entropy of individual pixel locations, and it allows our method to distinguish noisy regions from purely static regions. This kind of differentiation can guide adjustments so that dynamic background motion triggers feedback mechanisms that regular background or immobile foreground regions cannot. To obtain such an indicator, we first define a 2D map of pixel-level accumulators, noted v. Then, for every new segmented frame S_t, we compute the binary map of all blinking pixels at time t, noted X_t, by using an XOR operation with the previous segmentation results, S_{t-1}. Finally, we update v using

$$v(x) = \begin{cases} v(x) + v_{incr} & \text{if } X_t(x) = 1 \\ v(x) - v_{decr} & \text{otherwise} \end{cases} \tag{7}$$

where v_{incr} and v_{decr} are respectively 1 and 0.1, and $v(x)$ is always ≥ 0. This formulation means that regions with little labeling noise would typically have $v(x) \approx 0$, while regions with unstable labeling would have large positive $v(x)$ values.

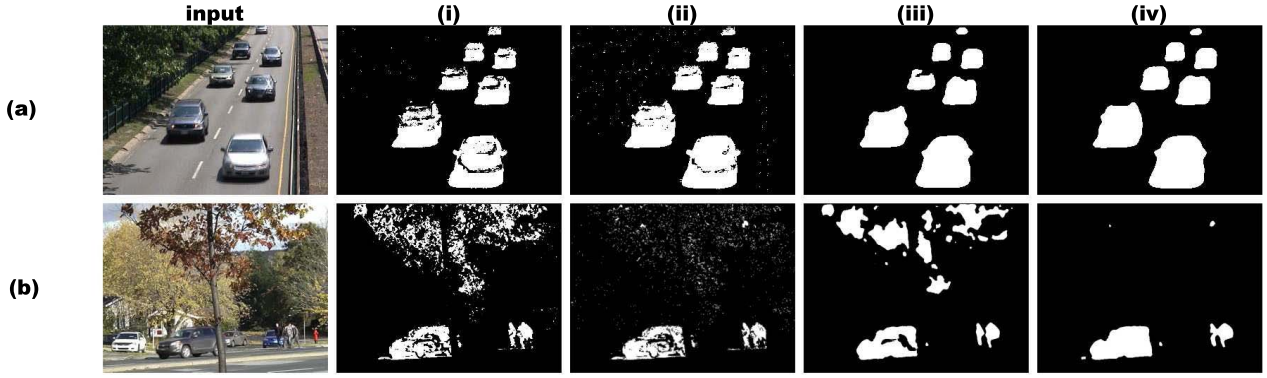

Fig. 6. Segmentation results obtained with our proposed model on a baseline sequence of CDnet ("highway", row a), and on a sequence with important dynamic background elements ("fall", row b), where (i) used no feedback or post-processing, (ii) used feedback but without post-processing, (iii) used post-processing but without feedback, and (iv) used both. While false classifications are present in all variations, we can note that the results of (iv) are the most consistent with object boundaries. (a) Highway #0826. (b) Fall #0289.

Directly using an XOR operation to detect blinking pixels can be inconvenient since the borders of moving foreground objects would also be included in the result; this can however be dealt with by nullifying all areas in X_t which intersect with the post-processed and dilated version of S_t. In the end, as shown in Fig. 5, the distribution of values in v can help highlight all frame regions where dynamic background elements are truly present, as opposed to regions where change has been recently seen.

With D_{min} and v defined, we can now introduce dynamic controllers for the main parameters of our method. First, local distance thresholds can be recursively adjusted for each new frame using

$$R(x) = \begin{cases} R(x) + v(x) & \text{if } R(x) < (1 + D_{min}(x) \cdot 2)^2 \\ R(x) - \frac{1}{v(x)} & \text{otherwise,} \end{cases} \tag{8}$$

where $R(x)$ is a continuous value always ≥ 1. The exponential relation between $R(x)$ and $D_{min}(x)$ is chosen over a linear relation since it favors sensitive behavior in static regions (and thus helps generate sparse segmentation noise), but also provides robust and rapidly scaling thresholds elsewhere. Here, the segmentation noise indicator $v(x)$ is used as a factor which, in dynamic regions, allows faster threshold increments and can even freeze $R(x)$ in place when $D_{min}(x)$ recedes to lower values. This is particularly helpful against intermittent dynamic background phenomena, as described earlier. However, in static regions, this same factor allows $R(x)$ to stay low even when $D_{min}(x) \approx 1$, which is especially useful when foreground objects are immobilized over x.

Besides, note that in (5), R is only an abstract value used to simplify notation; the actual distance thresholds for color and LBSP comparisons are obtained from $R(x)$ using

$$R_{color}(x) = R(x) \cdot R^0_{color} \tag{9}$$

and

$$R_{lbsp}(x) = 2^{R(x)} + R^0_{lbsp}, \tag{10}$$

where R^0_{color} and R^0_{lbsp} respectively carry the default color and LBSP distance thresholds (30 and 3 in our case). These minima are reached when $R(x) = 1$ and they represent the smallest amount of relevant local change our model can perceive. We define the relation behind $R_{lbsp}(x)$ as nonlinear due to the binary nature of LBSP descriptors and their comparison operator (Hamming distance).

On the other hand, local update rates are recursively adjusted using

$$T(x) = \begin{cases} T(x) + \frac{1}{v(x) \cdot D_{min}(x)} & \text{if } S_t(x) = 1 \\ T(x) - \frac{v(x)}{D_{min}(x)} & \text{if } S_t(x) = 0, \end{cases} \tag{11}$$

where $T(x)$ is limited to the $[T_{lower}, T_{upper}]$ interval (by default, $[2, 256]$), and $D_{min}(x)$ is used concurrently with $v(x)$ to determine the variation step size.[2] In short, this relation dictates that regions with very little entropy (both in terms of segmentation and motion) will see rapid update rate increases (i.e. sudden drops in model update probabilities) whenever pixels are classified as foreground (noted $S_t(x) = 1$). In other words, in regions that are static and motionless ($v(x) \approx D_{min}(x) \approx 0$), once foreground is detected, the model adaptation process will be instantly halted (due to very high $T(x)$). This process can be resumed slowly based on the value of $v(x)$, and only once the foreground is gone. As for dynamic background regions (or otherwise unstable regions), this equation simply dictates that variations will be much smoother, allowing the model to keep adapting to new observations even through continuous foreground classifications.

Overall, our feedback process depends not only on the results of internal comparisons between pixel models and local observations, but also on past labeling results. Due to how $v(x)$ works, our approach is more effective when sparse segmentation noise is present under the form of blinking pixels in most analyzed frame regions. Fortunately, due to the sensitive nature of LBSP descriptors, such noise can easily be generated when local distance thresholds are properly adjusted. Furthermore, the use of a median filter as a post-processing operation eliminates all of it, leaving the actual output of our method intact at a very low cost. We present in Fig. 5 the typical 2D distributions of R, T, D_{min} and v on frames showing static and dynamic background regions.

[2]Note that in practice, the indeterminate $\frac{0}{0}$ form cannot be achieved in the second right-hand statement of (11) as $D_{min}(x)$ never actually reaches 0 due to its infinite impulse response property.

In Fig. 6 we present segmentation results obtained with and without our feedback process (as well as with and without post-processing). In the latter, we can observe that before post-processing, the feedback-less configuration (Fig. 6.i) displays many bad foreground blobs, but less random salt-and-pepper (blinking pixel) noise than the configuration that used feedback (Fig. 6.ii). After post-processing however, the version with feedback (Fig. 6.iv) is much more reliable. This demonstrates how important sparse segmentation noise is in our feedback process, and how easily we can eliminate it.

D. Further Details

We stated earlier that our pixel models can handle short-term and long-term motion patterns due to the stochastic nature of their update rules. This is however not the case for our feedback process, as D_{min} is updated using a recursive moving average formula. To counter this, we keep two sets of D_{min} variables simultaneously up-to-date using different learning rates (i.e. a short-term one, $\alpha^{ST} = 25$, and a long-term one, $\alpha^{LT} = 100$). Then, when updating $R(x)$ via (8) and $T(x)$ via (11), we respectively use the current minimum and maximum $D_{min}(x)$ value between the two moving averages. This modification allows smoother distance thresholds and update rates adjustments, increasing the stability and responsiveness of our feedback scheme against most types of periodic background disturbances.

Also, to improve performance in scenarios with drastic background changes (e.g. "light switch" events) or moving cameras, we added a lightweight frame-level analysis component to our method. Its goal is to automatically scale T_{upper} and T_{lower} and trigger partial model resets in extreme cases (like the frame-level component of [30]). It works by analyzing discrepancies between short-term and long-term temporal averages of downscaled input frames (using the same learning rates as D_{min}). When disparities are omnipresent and persistent, it indicates that the analysis region suffered from an important change, and that the model should allow faster adaptations frame-wide. Downscaling the input frames allows faster overall processing and helps ignore shaking cameras while temporal averages are used to blur out and reduce the effects of foreground objects on the discrepancies analysis. This solution allows our method to compose with sudden and continuous camera movements, as long as the observed scene presents enough high-level detail. It is much less computationally expensive than analyzing camera movements using optical flow or feature detection/matching techniques. By-products of this component, namely the downscaled input frames and the temporal averages for different CDnet sequences, are shown in Fig. 7.

For implementation details on how the default parameters are scaled in this frame-level component, in the feedback process or for grayscale images, the reader is invited to refer to the source code.[3]

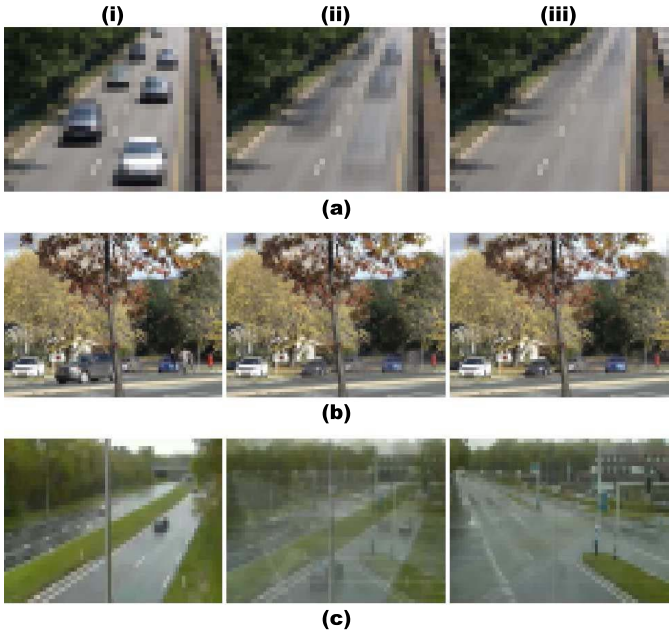

Fig. 7. By-products of the proposed frame-level analysis component at various times of CDnet sequences; column (i) shows the downscaled input frame used to update the moving averages, and columns (ii) and (iii) respectively show the sequence's short-term and long-term moving averages. While the difference between these two is negligible for the entire highway (row a) and fall (row b) sequences, it quickly becomes significant in the twoPositionPTZCam sequence (row c) right after the camera rotates. The pixel-wise analysis of discrepancies between these two moving averages allows the detection of such drastic events. (a) Highway #0826. (b) Fall #0289. (c) TwoPositionPTZCam #0356.

IV. EXPERIMENTS

To properly evaluate our method, we need to rely on more than a few hand-picked frames from typical surveillance videos. It is very difficult to compare state-of-the-art change detection methods, as many of them were tested on small datasets with few scenarios, and using different ground truth sources. Fortunately, a real benchmark was introduced for the 2012 CVPR Workshop on Change Detection [3]. Unlike its predecessors, the ChangeDetection.net (CDnet) dataset offers a wide variety of segmentation scenarios set in realistic conditions along with accurate ground truth data. More specifically, $\sim 88,000$ frames obtained from 31 video sequences were manually labeled and split into six categories: baseline, camera jitter, dynamic background, intermittent object motion, shadow and thermal. It was originally tested on 19 state-of-the-art methods, but has since been used to rank dozens more on their website, thus becoming a solid reference for method comparisons. This dataset was also updated for the 2014 version of the same CVPR Workshop [4], adding 22 videos and $\sim 70,000$ annotated frames in five new, much harder categories: bad weather, low framerate, night videos, pan-tilt-zoom and turbulence. In both versions, the official metrics used to rank methods are *Recall* (Re), *Specificity* (Sp), *False Positive Rate* (FPR), *False Negative Rate* (FNR), *Percentage of Wrong Classifications* (PWC), *Precision* (Pr) and *F-Measure* (FM). For their specific descriptions, refer to [3] and [4].

We primarily use *F-Measure* to compare the performance of different methods as it was found in [3] to be closely correlated with the ranks used on the CDnet website, and is generally accepted as a good indicator of overall performance. We chose not to compare methods using their overall ranks on

[3]Available at https://bitbucket.org/pierre_luc_st_charles/subsense.

TABLE I
AVERAGE PERFORMANCE COMPARISON OF DIFFERENT MODEL CONFIGURATIONS ON THE 2012 CDNET DATASET

Configuration	Pr	Re	FM
Color	0.841	0.672	0.710
LBP	0.350	0.686	0.394
SILTP	0.402	0.628	0.410
LBSP	0.772	0.758	0.721
Color-LBP	0.727	0.782	0.713
Color-SILTP	0.712	0.801	0.715
Color-LBSP	0.743	0.821	0.745
Color-LBSP-Feedback	0.856	0.831	0.826
Color-LBSP-Feedback-LessNoise	0.790	0.877	0.816

the CDnet website based on three reasons: 1) due to the non-linearity of overall ranks, adding or removing a method from the "comparison pool" (even when it is clearly outperformed by others) can substantially affect how top methods are ranked; 2) since the ranking system relies on both FPR and Sp, which are reciprocal ($FPR = 1-Sp$), "precise" methods are unduly favored over "sensitive" ones; and 3) because change detection is typically an unbalanced binary classification problem (there are far more background pixels than foreground pixels in analyzed sequences), using PWC as defined in [3] once again favors "precise" methods. To provide a better assessment of the overall performance of our method compared to others, we also evaluated top-ranked methods in separate tables using Matthew's Correlation Coefficient (MCC). This metric is designed to appraise the performance of binary classifiers in unbalanced problems, and is defined by

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}}, \tag{12}$$

where TP, TN, FP and FN are defined like in [3].

As required, we used a unique parameter set for all videos to determine the true flexibility of our method. Like we said earlier, the post-processing operations we use are only based on median blur and morphological operations and serve as our regularization step, eliminating irrelevant blobs/holes from segmentation maps. Note that all our segmentation results can be downloaded online via the CDnet website.[4]

A. CDnet 2012

To demonstrate our first key contribution (i.e. *color-LBSP* is preferable to other pixel-level characterization approaches), we present in Table I how our pixel models fare in comparison to similar alternatives when all are used on the 2012 CDnet dataset under the sample-based approach detailed in Section III-B. Note that these scores are the averages obtained over six scenario categories based on the evaluation technique of CDnet. Here, we can immediately see that 3×3 LBP and SILTP features are ill-suited to this kind of pixel-level modeling. SILTP features can detect texture variations more easily, but just like LBP, they are far too sensitive in most scenarios and cannot properly detect color changes between two frames when textures are unaffected. Nonetheless, both LBP and SILTP configurations obtain decent results when combined with color. According to *F-Measure* scores, only LBSP features are preferable to color information in this framework; this is due to their ability to detect color and texture changes simultaneously. Furthermore, the *color-LBSP* configuration offers performance on par with the best methods presented in [3]. In general, we can note that configurations that used local binary descriptors were inherently more sensitive to change than the baseline color-based approach (as visible through *Recall* scores), albeit at the expense of segmentation precision. However, among all tested binary descriptor configurations, *color-LBSP* is the best choice because it combines good sensitivity and precision.

We also inserted into Table I the results obtained using our *color-LBSP* configuration with the proposed feedback process (noted *color-LBSP-Feedback*) to demonstrate our second key contribution, i.e. continuous dynamic parameter adjustments can drastically improve performance. We can observe that *color-LBSP* obtains worse overall *Precision* and *Recall* scores (and thus a worse *F-Measure*) than our complete method. This is due to the sensitive nature of our pixel models, which cause many false classifications when they are not properly adjusted to their overlying regions. In the case of *color-LBSP*, a compromise had to be made to reach good overall flexibility (i.e. sensitivity had to be globally lowered to accommodate for complex scenarios). This comparison demonstrates that adjusting local parameters dynamically is extremely beneficial for this kind of approach, especially in terms of *Precision*. Note that for *color-LBSP* and *color-LBSP-Feedback*, the results were obtained by independently tuning their threshold values (i.e. R^0_{color} and R^0_{lbsp}) for optimal overall *F-Measures*. Interestingly, and as we expected, the *color-LBSP-Feedback* configuration performed much better with lower default thresholds (i.e. with a higher theoretical sensitivity) than *color-LBSP*, even in simple scenarios with completely static backgrounds; this can be explained by the affinity of our feedback scheme for noisier segmentation results. Furthermore, we present how this same feedback scheme performs when some sparse noise is removed using a 3×3 median filter before detecting blinking pixels (under the *color-LBSP-Feedback-LessNoise* configuration). From these results, we can again see that guiding local change sensitivity and update rates using segmentation noise is beneficial, as the *Precision* and *F-Measure* scores obtained for this new configuration are lower than those of *color-LBSP-Feedback*.

The complete results of our best configuration (*color-LBSP-Feedback*, noted SuBSENSE below) on this same dataset are then displayed in Table II. While these numbers may not mean much by themselves, we can see that overall performance in the baseline and shadow categories is very good, and our method's *Recall* is generally high. These two categories consist of simple sequences where pedestrians and cars are the main focus with various types of camouflage and illumination variation problems involved. We can also notice

[4] Available at http://wordpress-jodoin.dmi.usherb.ca/method/139/.

TABLE II
COMPLETE RESULTS FOR SUBSENSE ON THE 2012 CDNET DATASET

Category	*Re*	*Sp*	*FPR*	*FNR*	*PWC*	*Pr*	*FM*
baseline	0.9520	0.9982	0.0018	0.0480	0.3574	0.9495	0.9503
camera jitter	0.8243	0.9908	0.0092	0.1757	1.6469	0.8115	0.8152
dynamic background	0.7768	0.9994	0.0006	0.2232	0.4042	0.8915	0.8177
interm. object motion	0.6578	0.9915	0.0085	0.3422	3.8349	0.7957	0.6569
shadow	0.9419	0.9920	0.0080	0.0581	1.0120	0.8646	0.8986
thermal	0.8161	0.9908	0.0092	0.1839	2.0125	0.8328	0.8171
overall	0.8282	0.9938	0.0062	0.1718	1.5446	0.8576	0.8260

TABLE III
OVERALL AND PER-CATEGORY F-MEASURE COMPARISONS, CDNET 2012 DATASET[a]

Method	$FM_{overall(2012)}$	$FM_{baseline}$	$FM_{cam.jitter}$	$FM_{dyn.bg}$	$FM_{interm.obj.mot.}$	FM_{shadow}	$FM_{thermal}$
SuBSENSE	**0.826**	**0.950**	**0.815**	**0.818**	*0.657*	**0.899**	**0.817**
CwisarD [43]	*0.778*	0.908	*0.781*	0.809	0.567	0.841	0.762
Spectral-360 [44]	0.777	0.933	0.716	0.787	0.566	*0.884*	0.776
DPGMM [45]	0.776	0.929	0.748	*0.814*	0.542	0.813	*0.813*
SGMM-SOD [26]	0.766	0.921	0.672	0.688	**0.715**	0.865	0.735
PBAS [7]	0.753	0.924	0.722	0.683	0.575	0.860	0.756
PSP-MRF [25]	0.737	0.929	0.750	0.696	0.565	0.791	0.693
SC-SOBS [22]	0.728	*0.933*	0.705	0.669	0.592	0.779	0.692
ViBe+ [6]	0.722	0.871	0.754	0.720	0.509	0.815	0.665
KDE [14]	0.672	0.909	0.572	0.596	0.409	0.803	0.742
ViBe [5]	0.668	0.870	0.600	0.565	0.507	0.803	0.665
GMM [9]	0.662	0.825	0.597	0.633	0.520	0.737	0.662

[a] Note that red-bold entries indicate the best result in a given column, and blue-italics the second best.

that the intermittent object motion category poses the biggest challenge; this is true for any change detection solution that does not focus explicitly on static object detection and segmentation. This category mostly contains videos where objects are abandoned and parked cars suddenly start moving. The camera jitter and dynamic background scenarios are well handled by our approach, as in both cases, overall *F-Measure* and *Precision* are above 80%. The same can be said for thermal-infrared videos; in this case, our automatic distance threshold adjustments allow very good *Recall* despite numerous important camouflage problems in all sequences.

We show in Tables III and VI how our method compares to recent and classic state-of-the-art solutions, based on their overall 2012 CDnet results. Due to a lack of space, we only listed the classic and top-ranked methods out of the 32 that were published as of July 2014. In Table III, SuBSENSE is better in five out of six categories, with a 6.2% relative overall *F-Measure* improvement over the previous best method (CwisarD). Our performance in the intermittent object motion category is well above the average set by the methods tested in [3], and is only surpassed by SGMM-SOD, which used static object detection to specifically target this kind of scenario. While it cannot be deduced from the results shown here, our per-category *Recall* scores are the main reason why our *F-Measures* are so high compared to other methods; this again demonstrates that our approach can handle most camouflage problems due to its ability to detect very subtle spatiotemporal changes and locally adjust its sensitivity to change. The smaller overview shown in Table VI also support this conclusion: both our overall Recall and Precision scores are much higher than most others. We can also note that overall MCC scores are somewhat correlated with F-Measure scores, and a similar gap between our method and the second best is visible. Typical segmentation results for our proposed method as well as for the next best method (Spectral-360) and a classic one (GMM) are shown in Fig. 8.

B. CDnet 2014

For the 2014 update of the CDnet dataset, we first show in Table IV the complete results of SuBSENSE on all new scenario categories. It is easy to see how these new scenarios are much harder to deal with than the original ones: only bad weather and turbulence scenarios seem to result in acceptable performance (with *F-Measures* greater than 80%). The former consists of outdoor surveillance footage taken under snowy conditions and the latter shows long distance thermal-infrared video surveillance with important air turbulence due to a high temperature environment. Results in the pan-tilt-zoom category, on the other hand, are our worse so far; in this kind of scenario, the basic assumption behind all low-level change

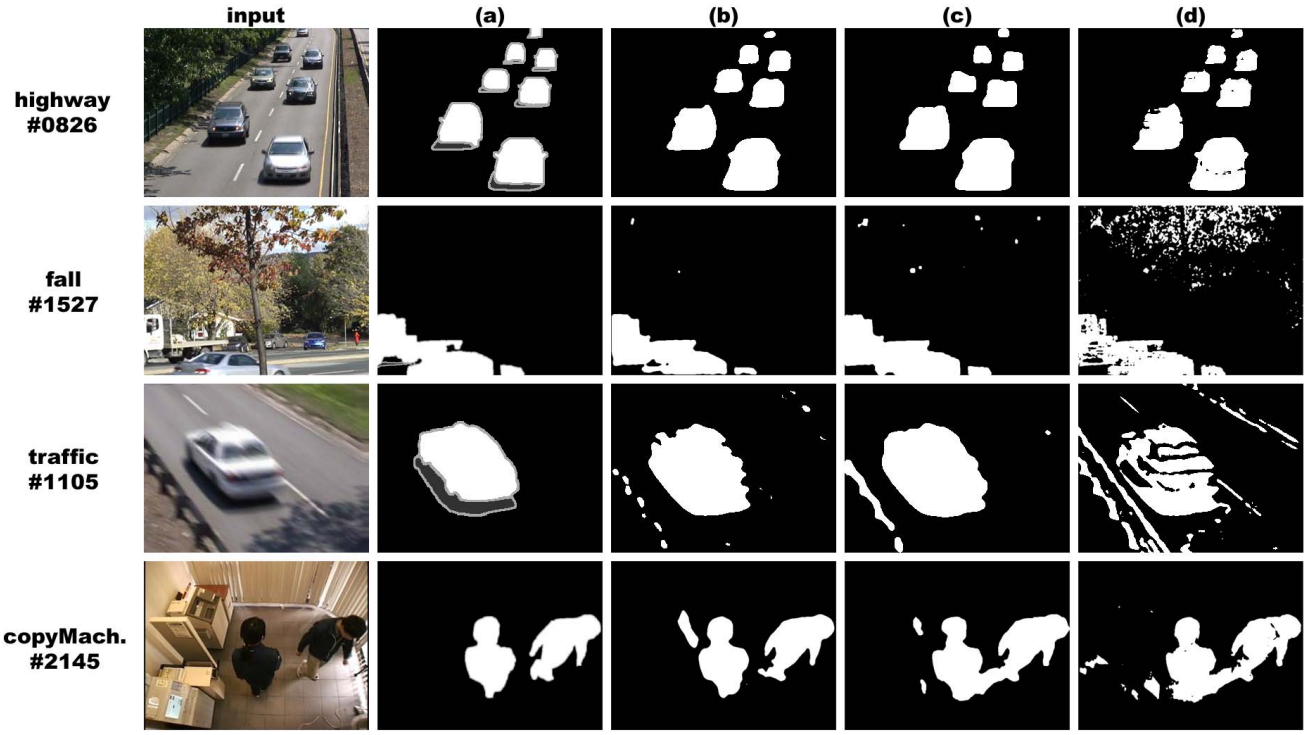

Fig. 8. Typical segmentation results for various sequences of the 2012 version of the CDnet dataset; column a) shows groundtruth maps, b) shows our segmentation results, c) Spectral-360's results and d) GMM's results. From top to bottom, the sequences are highway (from the baseline category), fall (dynamic background), traffic (camera jitter), and copyMachine (shadow). Note that gray areas are not evaluated.

TABLE IV
COMPLETE RESULTS FOR SUBSENSE ON THE 2014 CDNET DATASET

Category	*Re*	*Sp*	*FPR*	*FNR*	*PWC*	*Pr*	*FM*
bad weather	0.8213	0.9989	0.0011	0.1787	0.4527	0.9091	0.8619
low framerate	0.8537	0.9938	0.0062	0.1463	0.9968	0.6035	0.6445
night videos	0.6570	0.9766	0.0234	0.3430	3.7718	0.5359	0.5599
pan-tilt-zoom	0.8306	0.9629	0.0371	0.1694	3.8159	0.2840	0.3476
turbulence	0.8050	0.9994	0.0006	0.1950	0.1527	0.7814	0.7792
overall (2014)	0.7935	0.9863	0.0136	0.2064	1.8378	0.6228	0.6386
overall (2012+2014)	0.8124	0.9904	0.0096	0.1876	1.6780	0.7509	0.7408

detection methods, i.e. the camera remains static, is violated. Our frame-level analysis component allows our method to have a minimum level of functionality, but despite the apparent simplicity of most sequences, half of the segmentation results would hardly be of any use to other applications. Using a more sophisticated, high-level approach would result in better performance than a pixel-level change detection solution in this kind of scenario. Besides, even though scores in the low framerate category are generally low, in reality, our method performed well on all but one video. In this particular sequence, a marina is filmed at 0.17 frames per second under wavering global lighting conditions while many background elements (boats, water) show intense dynamic behavior. Finally, night videos also pose a significant challenge: the footage in this category is only taken from urban traffic monitoring cameras at night, meaning that photon shot noise, compression artifacts, camouflaged objects and glare effects from car headlights must all be handled simultaneously.

We then compare SuBSENSE to all other top-ranked methods tested on this same dataset in Tables V and VII. First, in Table V, our *F-Measure* scores once again stand out as well above average. In fact, even in the PTZ category, all evaluated methods obtained similar or worse scores, despite some of them using more sophisticated frame-level motion analysis approaches. Overall, we still obtain the best *F-Measure* scores in four out of five categories, and surpass the second best method with a 6.5% relative improvement in overall *F-Measure* for the 2014 dataset only. This indicates that our method is extremely flexible and can adapt even to the most difficult change detection scenarios. In Table VII, we can see that MCC is not as correlated with F-Measure as it was for the 2012 dataset; our method still achieves the

TABLE V
OVERALL AND PER-CATEGORY F-MEASURE COMPARISONS, CDNET 2014 DATASET[a]

Method	$FM_{overall(2012+2014)}$	$FM_{overall(2014)}$	$FM_{bad\ weather}$	$FM_{low\ framerate}$	$FM_{night\ videos}$	FM_{ptz}	$FM_{turbulence}$
SuBSENSE	**0.741**	**0.639**	**0.862**	**0.645**	**0.560**	*0.348*	**0.779**
FTSG [32]	*0.728*	*0.600*	*0.823*	0.626	*0.513*	0.324	0.713
CwisarDH [23]	0.681	0.549	0.684	0.641	0.374	0.322	0.723
Spectral-360 [44]	0.673	0.558	0.757	*0.644*	0.483	**0.365**	0.543
Bin Wang's [20]	0.658	0.501	0.767	0.469	0.380	0.135	*0.756*
SC-SOBS [22]	0.596	0.437	0.662	0.546	0.450	0.041	0.488
KNN [11]	0.594	0.492	0.759	0.549	0.420	0.213	0.520
KDE [14]	0.571	0.445	0.757	0.548	0.436	0.037	0.448
GMM [9]	0.569	0.461	0.738	0.537	0.410	0.152	0.466

[a] Note that red-bold entries indicate the best result in a given column, and blue-italics the second best.

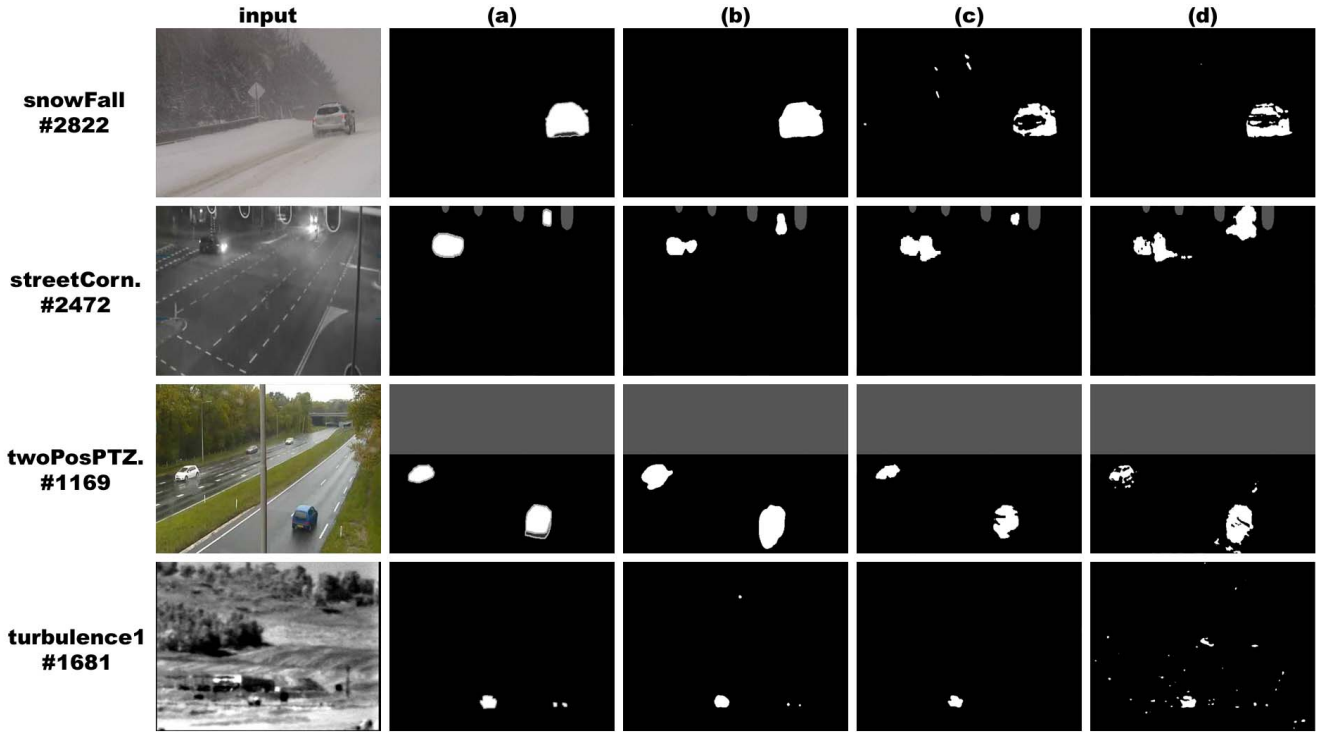

Fig. 9. Typical segmentation results for various sequences of the 2014 version of the CDnet dataset; column a) shows groundtruth maps, b) shows our segmentation results, c) FTSG's results and d) GMM's results. From top to bottom, the sequences are snowFall (from the bad weather category), streetCornerAtNight (night videos), twoPositionPTZCam (PTZ), and turbulence1 (turbulence). Note that gray areas are not unevaluated.

TABLE VI
AVERAGE PERFORMANCE COMPARISON OF DIFFERENT METHODS ON THE 2012 CDNET DATASET

Method	Pr	Re	FM	MCC
SuBSENSE	**0.858**	**0.828**	**0.826**	**0.827**
Spectral-360 [44]	*0.846*	0.777	*0.777*	*0.784*
DPGMM [45]	0.793	*0.827*	0.776	0.782
SGMM-SOD [26]	0.834	0.770	0.766	0.776
CwisarD [43]	0.774	0.818	0.778	0.773
PBAS [7]	0.816	0.784	0.753	0.766

TABLE VII
AVERAGE PERFORMANCE COMPARISON OF DIFFERENT METHODS ON THE 2014 CDNET DATASET

Method	Pr	Re	FM	MCC
SuBSENSE	0.751	**0.812**	**0.741**	**0.754**
FTSG [32]	*0.770*	*0.766*	*0.728*	*0.753*
CwisarDH [23]	**0.773**	0.661	0.681	0.709
Spectral-360 [44]	0.705	0.735	0.673	0.703
Bin Wang's [20]	0.716	0.704	0.658	0.671
SC-SOBS [22]	0.609	0.762	0.596	0.615

best overall score, but only marginally. Note that since part of the 2014 dataset is withheld for online testing, all MCC scores were computed based on the publicly available dataset, which may explain this difference. Besides, the second and third best methods, FTSG and CwisarDH, receive noticeably better overall Precision scores. Again, we show in Fig. 9

segmentation results for our proposed method as well as for the second best one (FTSG) and a classic one (GMM). Note that for all 2014 results, we used the same parameters as for the 2012 dataset.

C. Processing Speed

We did not optimize any component of our final method; it ran on a third generation Intel i5 CPU at 3.3 GHz with no architecture-specific instruction, using OpenCV's background subtraction interface (C++) and saving output frames on a local hard drive. Nonetheless, the feedback-less version (*color-LBSP* in Section IV-A) clocked an average of over 90 frames per second on the entire 2012 CDnet dataset, and the complete method (SuBSENSE) processed the same dataset at 45 (both were running one sequence per CPU core). Comparing these results with those of recent state-of-the-art methods on the same dataset is impossible due to the wide variety of platforms involved for testing, the lack of a common reference and the rarity of openly available source code.

Besides, our method can be fully initialized with a single image, requires no training and performs all dynamic adjustments on-line, for every new frame. The low-level nature of our approach also favors high-speed parallel implementations, as no region-level or object-level processing is required. Processing speed could be further improved by reducing the total number of samples in our pixel models (N), at the expense of some flexibility in complex scenarios. This same parameter could also be controlled dynamically to reduce overall computational cost since fewer samples are usually needed for good classifications in static background regions.

V. Conclusion

We presented a novel, highly efficient and *universal* foreground/background segmentation algorithm based on change detection in video sequences. Our method uses spatiotemporal information based on color and texture to characterize local representations in pixel-level models while staying robust to most types of illumination variations, including shadows. It also relies on a pixel-level feedback scheme that automatically adjusts internal sensitivity to change and update rates. Our approach continuously monitors both local model fidelity and segmentation noise to guide these adjustments, allowing for fast responses to intermittent dynamic background motion. As such, it can be effectively used in complex surveillance scenarios presenting many different challenges simultaneously.

Experiments on the largest change detection dataset available yet have shown that, in terms of average *F-Measure*, we surpass all previously tested methods in nine out of eleven scenario categories as well as overall. Categories where our segmentation results were still inaccurate can be considered the next major challenge in change detection: in those cases, the assumptions that have been commonly followed since the late 1990s no longer hold (e.g. the camera no more static). These experiments have also confirmed the benefit of using LBSP features in our pixel models as well as the benefit of using our continuous parameter adjustment scheme based on model fidelity and segmentation noise.

A number of improvements can still be considered for our method; for example, region-level or object-level analyses could be used as extra regularization steps to improve the shape consistency of blobs over time. Also, more sophisticated post-processing operations based on connected components or Markov Random Fields could also help eliminate larger noise patches from our final segmentation results. Besides, since our method is relatively simple and operates at the pixel level, it has a lot of potential for hardware and high-speed parallel implementations.

References

[1] P.-L. St-Charles and G.-A. Bilodeau, "Improving background subtraction using local binary similarity patterns," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, Mar. 2014, pp. 509–515.

[2] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin, "Flexible background subtraction with self-balanced local sensitivity," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 414–419.

[3] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.net: A new change detection benchmark dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 1–8.

[4] 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 Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 387–394.

[5] O. Barnich and M. Van Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," *IEEE Trans. Image Process.*, vol. 20, no. 6, pp. 1709–1724, Jun. 2011.

[6] M. Van Droogenbroeck and O. Paquot, "Background subtraction: Experiments and improvements for ViBe," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 32–37.

[7] M. Hofmann, P. Tiefenbacher, and G. Rigoll, "Background segmentation with feedback: The pixel-based adaptive segmenter," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 38–43.

[8] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," in *Proc. 13th Conf. Uncertainty Artif. Intell.*, 1997, pp. 175–181.

[9] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. Jun. 1999, pp. 246–252.

[10] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proc. 17th Int. Conf. Pattern Recognit.*, vol. 2. Aug. 2004, pp. 28–31.

[11] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 773–780, 2006.

[12] D.-S. Lee, "Effective Gaussian mixture learning for video background subtraction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 827–832, May 2005.

[13] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-Based Surveillance Systems*. New York, NY, USA: Springer-Verlag, 2002, pp. 135–144.

[14] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Proc. 6th Eur. Conf. Comput. Vis.*, 2000, pp. 751–767.

[15] A. Mittal and N. Paragios, "Motion-based background subtraction using adaptive kernel density estimation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. Jul. 2004, pp. II-302–II-309.

[16] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1778–1792, Nov. 2005.

[17] M. Van Droogenbroeck and O. Barnich, "ViBe: A disruptive method for background subtraction," in *Background Modeling and Foreground Detection for Video Surveillance*, T. Bouwmans, F. Porikli, B. Hoferlin, and A. Vacavant, Eds. Boca Raton, FL, USA: CRC Press, Jun. 2014, ch. 7.

[18] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Background modeling and subtraction by codebook construction," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2004, pp. 3061–3064.
[19] M. Wu and X. Peng, "Spatio-temporal context for codebook-based dynamic background subtraction," *AEU-Int. J. Electron. Commun.*, vol. 64, no. 8, pp. 739–747, 2010.
[20] B. Wang and P. Dudek, "A fast self-tuning background subtraction algorithm," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 401–404.
[21] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Trans. Image Process.*, vol. 17, no. 7, pp. 1168–1177, Jul. 2008.
[22] L. Maddalena and A. Petrosino, "The SOBS algorithm: What are the limits?" in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 21–26.
[23] M. De Gregorio and M. Giordano, "Change detection with weightless neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 409–413.
[24] A. Morde, X. Ma, and S. Guler, "Learning a background model for change detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 15–20.
[25] A. Schick, M. Bauml, and R. Stiefelhagen, "Improving foreground segmentations with probabilistic superpixel Markov random fields," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 27–31.
[26] R. H. Evangelio and T. Sikora, "Complementary background models for the detection of static and moving objects in crowded environments," in *Proc. 8th IEEE Int. Conf. Adv. Video Signal Based-Surveill.*, Aug. 2011, pp. 71–76.
[27] J.-P. Jodoin, G.-A. Bilodeau, and N. Saunier. (2012). "Background subtraction based on local shape." [Online]. Available: http://arxiv.org/abs/1204.6326
[28] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian computer vision system for modeling human interactions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 831–843, Aug. 2000.
[29] D.-M. Tsai and S.-C. Lai, "Independent component analysis-based background subtraction for indoor surveillance," *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 158–167, Jan. 2009.
[30] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, vol. 1. Sep. 1999, pp. 255–261.
[31] Y. Nonaka, A. Shimada, H. Nagahara, and R. Taniguchi, "Evaluation report of integrated background modeling based on spatio-temporal features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 9–14.
[32] R. Wang, F. Bunyak, G. Seetharaman, and K. Palaniappan, "Static and moving object detection using flux tensor with split Gaussian models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 420–424.
[33] M. Heikkila and M. Pietikainen, "A texture-based method for modeling the background and detecting moving objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 657–662, Apr. 2006.
[34] S. Zhang, H. Yao, and S. Liu, "Dynamic background modeling and subtraction using spatio-temporal local binary patterns," in *Proc. 15th IEEE Int. Conf. Image Process.*, Oct. 2008, pp. 1556–1559.
[35] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikainen, and S. Z. Li, "Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1301–1306.
[36] G.-A. Bilodeau, J.-P. Jodoin, and N. Saunier, "Change detection in feature space using local binary similarity patterns," in *Proc. Int. Conf. Comput. Robot Vis.*, May 2013, pp. 106–112.
[37] D. H. Parks and S. S. Fels, "Evaluation of background subtraction algorithms with post-processing," in *Proc. 5th IEEE Int. Conf. Adv. Video Signal Based Surveill.*, Sep. 2008, pp. 192–199.
[38] S. Herrero and J. Bescós, "Background subtraction techniques: Systematic evaluation and comparative analysis," in *Advanced Concepts for Intelligent Vision Systems*, vol. 5807, J. Blanc-Talon, W. Philips, D. Popescu, and P. Scheunders, Eds. Berlin, Germany: Springer-Verlag, vol. 5807, 2009, pp. 33–42.
[39] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Comparative study of background subtraction algorithms," *J. Electron. Imag.*, vol. 19, no. 3, p. 033003, 2010.
[40] S. Brutzer, B. Hoferlin, and G. Heidemann, "Evaluation of background subtraction techniques for video surveillance," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 1937–1944.
[41] P.-M. Jodoin, S. Piérard, Y. Wang, and M. Van Droogenbroeck, "Overview and benchmarking of motion detection methods," in *Background Modeling and Foreground Detection for Video Surveillance*, T. Bouwmans, F. Porikli, B. Hoferlin, and A. Vacavant, Eds. Boca Raton, FL, USA: CRC Press, Jun. 2014, ch. 1.
[42] H. Wang and D. Suter, "A consensus-based method for tracking: Modelling background scenario and foreground appearance," *Pattern Recognit.*, vol. 40, no. 3, pp. 1091–1105, 2007.
[43] M. De Gregorio and M. Giordano, "A WiSARD-based approach to CDnet," in *Proc. 11th BRICS Countries Congr.*, Sep. 2013, pp. 172–177. [Online]. Available: http://www.ieeexplore.us/xpl/articleDetails.jsp?tp=&arnumber=6855846
[44] M. Sedky, M. Moniri, and C. C. Chibelushi, "Spectral-360: A physics-based technique for change detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 405–408.
[45] T. S. F. Haines and T. Xiang, "Background subtraction with Dirichlet process mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 670–683, Apr. 2014.

Pierre-Luc St-Charles (S'13) received the B.Eng. degree in computer engineering from the École Polytechnique de Montréal, Montréal, QC, Canada, in 2013, where he is currently pursuing the Ph.D. degree. He has been working part-time with the Computer Research Institute of Montréal, Montréal, since 2011. His research interests include image and video segmentation, multimodal registration, and video surveillance applications.

Guillaume-Alexandre Bilodeau (M'10) received the B.Sc.A. degree in computer engineering and the Ph.D. degree in electrical engineering from Université Laval, Québec, QC, Canada, in 1997 and 2004, respectively.

He was appointed as an Assistant Professor with the École Polytechnique de Montréal, Montréal, QC, Canada, in 2004, and an Associate Professor in 2011, where he has been a Full Professor since 2014. His research interests encompass in image and video processing, video surveillance, object recognition, content-based image retrieval, and medical applications of computer vision.

Dr. Bilodeau is a member of the Province of Québec's Association of Professional Engineers and the REPARTI Research Network.

Robert Bergevin (M'84) received the B.Eng. degree in electrical engineering and the M.A.Sc. degree in biomedical engineering from the École Polytechnique de Montréal, Montréal, QC, Canada, and the Ph.D. degree in electrical engineering from McGill University, Montréal. His research interests are in image analysis and cognitive vision. His main works address in generic modeling and recognition of objects in static images and tracking and modeling of people and animals in image sequences.

He is a member of the Computer Vision and Systems Laboratory with Université Laval, Québec, QC, Canada, where he is a Professor with the Department of Electrical and Computer Engineering. He is also a member of the Province of Québec's Association of Professional Engineers and serves as an Area Editor of the *Computer Vision and Image Understanding* journal.