A Self-Adjusting Approach to Change Detection Based on Background Word Consensus

Pierre-Luc St-Charles, Guillaume-Alexandre Bilodeau
LITIV lab., Dept. of Computer & Software Eng.
École Polytechnique de Montréal
Montréal, QC, Canada
{pierre-luc.st-charles,gabilodeau}@polymtl.ca

Robert Bergevin
LVSN - REPARTI
Université Laval
Québec, QC, Canada
robert.bergevin@gel.ulaval.ca

Abstract

Although there has long been interest in foreground-background segmentation based on change detection for video surveillance applications, the issue of inconsistent performance across different scenarios remains a serious concern. To address this, we propose a new type of word-based approach that regulates its own internal parameters using feedback mechanisms to withstand difficult conditions while keeping sensitivity intact in regular situations. Coined "PAWCS", this method's key advantages lie in its highly persistent and robust dictionary model based on color and local binary features as well as its ability to automatically adjust pixel-level segmentation behavior. Experiments using the 2012 ChangeDetection.net dataset show that it outranks numerous recently proposed solutions in terms of overall performance as well as in each category. A complete C++ implementation based on OpenCV is available online.

1. Introduction

The segmentation of foreground and background regions in video sequences based on change detection is a fundamental, yet challenging task in computer vision. Often simply called background subtraction, it has been well studied over the years, but so far no approach has been able to efficiently manage complex scenarios. In fact, most methods must be finely tuned to achieve optimal results on different video sequences: for example, dynamic outdoor scenes can rarely be modeled as easily as static indoor scenes, forcing the use of a less sensitive approach to reduce false classifications. Ideally, an adequate algorithm should be able to handle challenges such as rapid illumination variations, dynamic background regions (rippling water, swaying trees, etc.) and intermittent motion without supervision or readjustments. However, very few can actually accomplish this, and even fewer can do it without requiring extensive training or sequence preprocessing.

Here, we propose a background subtraction method suitable for a large variety of scenarios without manual parameter readjustment. Specifically, we use a self-adjusting approach based on a new kind of pixel-level word model. PAWCS (Pixel-based Adaptive Word Consensus Segmenter), our novel method, improves upon other codebook approaches by using persistence-based words to adequately model the background over longer periods of time. Furthermore, our method introduces real-time learning and adaptation capabilities as well as a complementary frame-level dictionary. Its reliance on Local Binary Similarity Patterns [2] at the pixel level to better detect spatio-temporal changes as well as its dynamically constrained, feedback-driven parameter control scheme also contribute in large part to its novelty. More precisely, our contributions are 1) the introduction of a new word-based model that has the ability to capture and retain background representations over long periods of time. It uses color values, LBSP features and persistence indicators all grouped into background words to assimilate spatio-temporal information at the pixel level. This model combines both local and global background representations in a two-step classification process; 2) the proposal of a novel feedback control scheme able to isolate and process frame regions differently based on background dynamics and segmentation results. Since it allows internal parameters to be adjusted after every new pixel classification, it is mostly unaffected by noisy regions and intermittent dynamic background motion.

Evaluations based on the 2012 ChangeDetection.net (CDnet) dataset [5] indicate that our new approach outperforms 10 recently proposed methods not only in terms of overall performance, but also in all scenario categories taken individually. Furthermore, F-Measure comparisons reveal that the relative improvement between our method and the previous best is over 10% overall.

978-1-4799-6683-7/15 $31.00 © 2015 IEEE
DOI 10.1109/WACV.2015.137

1.1. Related Work

The CDnet workshop [5] recently offered a remarkable proving ground for all background subtraction methods: different families of approaches could finally be directly compared in various categories. Among them, pixel-based approaches solely rely on individual pixel intensities to track background representations. Despite the simplicity of this concept, two of the three best overall methods of [5], namely PBAS [8] and ViBe+ [19], follow it as they are both based on local intensity sampling. Most classic methods based on Gaussian Mixture Models (GMM) [16] or Kernel Density Estimation (KDE) [3] can also be considered part of this family. On the other hand, texture-based methods provide richer background representations by also analyzing the relationships between neighboring pixels: those presented in [7, 10, 2, 15] all use Local Binary Patterns (LBP) or LBP-like features for this purpose and achieve good tolerance against illumination variations. Although pixel-based and texture-based approaches have their benefits, they also have their drawbacks, and "hybrid" techniques such as [18, 17] have emerged to capitalize on their full combined potential. Furthermore, recent work using neural networks ([11, 12]) and Markov Random Fields ([14, 21]) has also focused on improving spatio-temporal segmentation coherence in a similar mindset.

There are very different ways to build and update background models: for example, amid non-parametric approaches, both PBAS and ViBe+ rely on the sample consensus ideas of [20, 1], while methods based on KDE use such samples to estimate the probability density functions of the background. Likewise, the codebook method proposed in [9] and improved in [21] relies on the clustering of local recent pixel representations into codewords to build its model. Although this last approach can be considered unique and effective, only a handful of authors actually followed the idea of using such codewords to keep track of background representations. Nonetheless, Kim et al.'s model [9] offers many advantages over traditional ones, such as better adaptability against multimodal regions while having a smaller memory footprint. In the original pixel-level codebook initialization procedure, the measure of a codeword's maximum negative run-length is used to filter out bad local representations while keeping the periodically reoccurring ones inside the model. In other words, local codebooks are able to retain most forms of dynamic background while rejecting any kind of foreground that might have been present during training based on periodicity and number of occurrences. However, this filtering step requires a predetermined threshold that could vary based on the sequence type and the size of the initialization window. Our proposed background model is inspired by this previous work.

As stated earlier, a problem common to most existing methods is that they lack flexibility: even though they can provide good results on individual sequences when adjusted properly, few of them can actually perform equally across large datasets. In this regard, PBAS [8] opened up a new horizon on the automatic adjustment of a method's internal parameters, but also suffered from delayed and fluctuating sensitivity variations – especially when faced with intermittent dynamic background motion. Our own self-adjustment strategy is inspired by this work and solves this sensitivity problem by using blinking pixel information.

2. Description of the PAWCS method

Our method is based on the characterization and monitoring of background representations at the pixel level using a word-based approach without clustering. The general idea behind this new concept is to register the appearances of pixels over time as "background words" in local dictionaries using color and texture information. These words are then considered good representations of the background when they are persistent, *i.e.* when they reoccur often. Unlike other approaches where such representations are simply considered data samples, our solution favors persistent background words over infrequent ones, which can be discarded and replaced by better alternatives. The way we characterize background representations is described in section 2.1, and the general guidelines of our learning and update strategies are described in section 2.2.

To improve how background representations are compared and maintained in difficult conditions, we dynamically adjust sensitivity thresholds and learning rates used in our segmentation decisions and model update rules. These adjustments are made based on the analysis of background dynamics (*i.e.* local variation patterns and model fidelity) and past segmentation results. This feedback process is presented in section 2.3.

Note that a C++ implementation of PAWCS which includes all specific details about the operations described below is available online[1], and a general overview is presented in Fig. 1 using a block diagram.

2.1. Background Words and Dictionaries

In order to properly characterize and match pixel representations with adequate sensitivity, we integrate LBSP features with colors in pixel-level models as in [15]. These LBP-like features were demonstrated in [2] to be more sensitive to local changes than simple color comparisons. However, when used by themselves, they are often too sensitive to change in dynamic background regions and highly contrasted/noisy areas. This is why we combine them with color values in our own local models. Therefore, we take direct pixel color resemblance (based on L1 distance) into consideration along with LBSP intersections (via Ham-

[1] https://bitbucket.org/pierre_luc_st_charles/pawcs

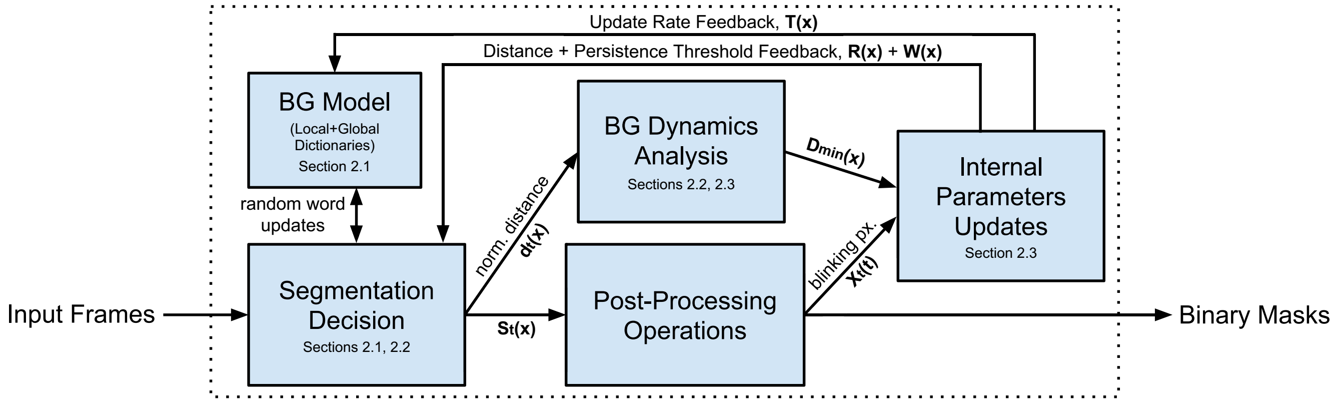

Figure 1. Block diagram of the Pixel-based Adaptive Word Consensus Segmenter. The role of each variable is presented in the following sections. In our implementation, the post-processing operations are only based on median blur and blob smoothing operations.

ming distance) to test the similarity of pixel representations. The combination of color and LBSP comes at a negligible cost since the required color information must already be kept locally to compute inter-frame LBSP descriptors (as described in [2]). The novelty that we propose is to tie color/LSBP pairs to a persistence indicator which reflects their periodicity and number of occurrences amongst recent local background observations (detailed in section 2.2). This modification allows some pixel representations to be considered more important than others and also dictates which should be replaced based on their relevance in the model. The color/LBSP/persistence triplets we obtain are defined as background words (BWs), and the pixel-level models in which they are collected as local dictionaries. Implementation-wise, a BW uses one byte per RGB channel for color information, two bytes for LBSP binary strings (based on the 5x5 pattern of [2]) and integers to store its total occurrence count as well as time indexes at which it was first/last seen. Finally, note that in addition to the use of a local dictionary for each pixel x (noted $C_L(x)$ below), we also propose to use a global dictionary for the whole frame, which is detailed in section 2.2.

Similarly to [15], to improve robustness of our dictionaries against illumination variations, we use the following encoding formula for LBSP descriptors:

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

with

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

where i_x is the reference intensity value, i_p is the pixel intensity of the p-th neighbor of i_x on the predefined P-length LBSP pattern, and T_r is the relative LBSP threshold (by default, we use $\frac{1}{3}$).

2.2. Learning and Update Strategy

Learning. Contrarily to previous work, we keep all our BWs separated inside their local dictionaries instead of clustering them into codewords. This way, rather than having a few BWs with very different general representations and high persistence, we obtain many overlapping BWs with lower persistence, each covering a smaller portion of the representation space. This kind of approach where overlap between local representations is allowed is very similar to the ideology behind sample-consensus methods: to obtain a "consensus" between a pixel model and the current local observation (noted $I_t(x)$), a minimal subset of intersecting background representations needs to be found. Therefore, instead of looking for one BW match with an acceptable persistence to consider a pixel background, we would first need to calculate the persistence sum of all matched BWs in a local dictionary. To do this, we use

$$q_{tot}(x) = \sum_{\omega \in C_L(x)} q(\omega, t) \;\; \big|\, \big(dist(I_t(x), \omega) < R(x)\big), \tag{3}$$

where $q(\omega, t)$ returns the persistence value of a BW at time t (described further down in Eq. 5), $dist(I_t(x), \omega)$ returns the color/LBSP distances between a given BW and the local observation at x, and $R(x)$ expresses the local color/LBSP similarity thresholds (further discussed in section 2.3). Finally, the pixel at x would be classified as foreground (1) or background (0) based on $q_{tot}(x)$:

$$S_t(x) = \begin{cases} 1 & \text{if } q_{tot}(x) < W(x) \\ 0 & \text{otherwise} \end{cases}\,, \tag{4}$$

where $S_t(x)$ is the output segmentation map, and $W(x)$ is a dynamic persistence threshold that will also be detailed in section 2.3.

By its design, our "word-consensus" solution is much more resilient to erroneous model drifting due to bad representation updates. It also allows us to share individual

BWs between local dictionaries without the risk of corrupting them entirely. Moreover, any form of clustering would have been incompatible with the pixel-level characterization approach we use since LBSP descriptors cannot be gathered or averaged easily. The increased cost of this new approach due to the rising number of BWs to check for intersections is, in practice, counter-balanced by the absence of clustering and can be further reduced when all BWs inside a dictionary are already sorted by persistence. In our case, we incorporated a simplified bubble sort algorithm inside the local dictionary loop of Eq. 3 so that words could be individually swapped once while processing every new frame.

As discussed earlier, we improve the state of the art by proposing a persistence-based approach to determine which BWs hold the most importance in a local dictionary. The actual persistence of a given BW is obtained at time t using

$$q(\omega, t) = \frac{n_{occ}}{(t_{last} - t_{first}) + 2\cdot(t - t_{last}) + t_o}, \qquad (5)$$

where n_{occ} is the BW's total number of occurrences, t_{first} and t_{last} are, respectively, the time at which it was first and last seen, and t_o is a predetermined offset value. The goal of this equation is analogous to the maximum negative run-length measure of the codebook method[9]: it helps quickly eliminate BWs that do not reoccur very often. Note that, in our case, persistence values are generally more affected by a distant last occurrence than by a prolonged lifetime, as the second term of the denominator, $(t - t_{last})$, is multiplied by 2. This means that we confer more importance to BWs that are seen often over long periods of time while still taking their periodicity into account. The time offset t_o is only used to prevent newly created BWs from having important persistence values; since persistence thresholds are dynamically controlled (we discuss this in section 2.3), its actual size has little effect on the rest of the algorithm, as long as it is high enough (by default, we used $t_o = 1000$). With this formula, a mandatory training and analysis phase to filter out unwanted BWs is unneeded, as the importance of all BWs in a given local dictionary can be obtained at any time. Hence, the actual segmentation process can be completely initialized within the first two frames of any sequence, with results becoming more stable over time. Furthermore, previously unseen BWs can be immediately inserted into local dictionaries where they will either grow in importance or be quickly replaced by others.

Update. Because we rely on the L1 distance to compare the color component of BWs, to improve robustness against illumination variations, we randomly update color representations inside BWs with actual color observations from the analyzed frame. This is only done when their corresponding local textures (as expressed through LBSP features) are almost identical and when color distortion (as described by Eq. 2 of [9]) is negligible. The idea behind these random updates is that only a small proportion of all BWs should be immediately modified when an observed scene's illumination changes; those that are left untouched can then be used to bring the model back to a previous state if said change was only temporary. This first update mechanism also helps prevent the saturation of local dictionaries due to the accumulation of BWs that have similar LBSP descriptors and normalized colors but different brightness.

Besides, an important trait shared between recent sample-consensus methods (*e.g.* [8, 19, 1, 15]) can be easily adapted to our new approach: improved spatial coherency via pixel-level representations diffusion. Simply put, once a given pixel x has been classified as background, adjacent pixel models sometimes see one of their local representations be randomly replaced by $I_t(x)$, the current observation at x. First described in [1], this technique leads not only to better spatial coherence in areas with intermittent and periodic motion, but it also helps reabsorb falsely labeled background regions into the model. To achieve a very similar result, we simply have to randomly pick an adjoining local dictionary of $C_L(x)$ after Eq. 4 classifies x as background and update it again, increasing the number of occurrences for BWs that match $I_t(x)$. This modification further increases the persistence of common BWs in static areas, which in turn helps against camouflaged and immobile foreground objects. Furthermore, unlike the spatial context improvement proposed in the codebook approach of [21], this change could be regarded as "proactive" since it allows the model to passively build resistance against disturbances such as vibrating cameras.

The model described so far is very flexible and adapts to different challenges. For example, it behaves almost exactly like the one in [9] for static regions as, once stable, it only requires one or two BWs per local dictionary for proper classifications. However, in dynamic background regions, it behaves a lot more like a sample consensus model, where up to 20 BWs could be active at once, all sharing similar lower persistence. The randomized BW updates presented earlier also help in scenes where illumination varies periodically; nonetheless, as shown in Fig. 2.c, some false classifications are still present in sequences with strong dynamic backgrounds such as moving tree branches or water fountains. This is due in large part to the highly specific and sensitive nature of the local representations we opted for: BWs tied to rare periodic changes near stable regions often carry very low persistence values due to their uniqueness, and are thus often discarded and replaced before being seen again.

Global dictionary. The need to retain each of these exceptional representations drives us to another important part of our model, the frame-level dictionary. This new "global" dictionary helps decrease the number of bad foreground classifications by confirming that a given observation $I_t(x)$

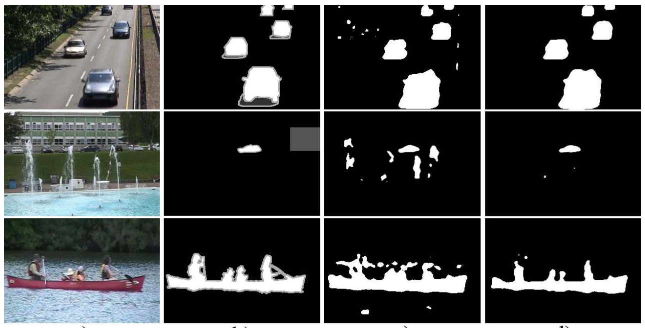

Figure 2. Segmentation results obtained on frame 1064 of the 'highway' sequence (top row), frame 717 of 'fountain' (middle row) and frame 957 of 'canoe' (bottom row), where (a) is the input image, (b) is the ground truth, (c) is the result obtained without the global dictionary and feedback improvements and (d) is our final result. Note that both (c) and (d) use a 9x9 median filter as post-processing.

cannot be matched with BWs regularly seen around x. Its structure is relatively simple: it contains a predetermined number of BWs acting as global representations, but instead of generating them directly from observations within the sequence, they are randomly copied from those contained in local dictionaries. To improve the chances of finding a match for a given representation, the color distortion measure of [9] is again used along with the L1 distance for color comparisons, and the Hamming distance used to compare LBSP binary strings is replaced by a Hamming weight function. Since these global BWs are shared between local models, their persistence can no longer be updated using Eq. 5; we instead assign them 2D persistence maps. This way, we can keep track of where they were recently seen by locally accumulating $q(\omega, t)$ for all matching $\omega \in C_L(x)$. Moreover, we can spread global BWs region boundaries by using blur operations on this 2D map, and then decimate it every given number of frames to keep the spread and accumulation under control. In the end, foreground classification for a given pixel x can be overruled if a matching global word with good local persistence is found. Specifically, Eq. 4 is replaced by

$$S_t(x) = \begin{cases} 1 & \text{if } q_{tot}(x) + q_G(x) < W(x) \\ 0 & \text{otherwise} \end{cases}, \quad (6)$$

where $q_G(x)$ returns the value contained at x in the 2D persistence map of a matched global BW (if one is found; 0 otherwise).

2.3. Feedback-Driven Self-Adjustments

So far, we have presented how our model is built and managed, but an additional part of its flexibility resides in how all of its important internal parameters are automatically adjusted. This is our second contribution. Internal parameters can be redefined as pixel-level state variables, and summarized in three classes: update rates (noted $T(x)$) which control the frequency of random actions such as BW color updates, local neighboring dictionary updates and global dictionary updates; distance thresholds (noted $R(x)$) which control how different two BWs can be while still being considered similar; and finally, persistence thresholds (noted $W(x)$) which control the minimal matching BW persistence required to classify a pixel as background.

To adjust these variables, we first rely on pixel-level background dynamics indicators, noted $D_{min}(x)$, which continuously monitor how analogous our pixel models are to real observations. Our D_{min} is essentially a recursive rolling average map of the normalized minimal differences between observations and BWs kept in local dictionaries. It is not restricted to updates when $S_t(x) = 0$, meaning it can smoothly adapt to most intermittent dynamic background motion. More specifically, we use

$$D_{min}(x) = D_{min}(x) \cdot (1 - \alpha) + d_t(x) \cdot \alpha \quad (7)$$

with

$$d_t(x) = \begin{cases} d_{color/lbsp}(x) + \frac{W(x) - q_{tot}}{W(x)} & \text{if } q_{tot} < W(x) \\ d_{color/lbsp}(x) & \text{otherwise} \end{cases}, \quad (8)$$

where $d_{color/lbsp}(x)$ is the minimal normalized distance between color values and LBSP descriptors in $C_L(x)$, and α is a constant learning rate. The persistence sum (q_{tot}) is used in this context to increase the value of D_{min} when the pixel at x cannot be classified as background based on the persistence threshold, $W(x)$. Note that all $D_{min}(x)$ values are bound to the $[0, 1]$ interval, where 0 would indicate a perfectly static background and 1 a highly dynamic background or a region that the model cannot properly adapt to. This formulation also implies that foreground objects temporarily immobilized over x will cause $D_{min}(x)$ to keep rising until said object is gone or has been integrated to the model. As such, we cannot use $D_{min}(x)$ alone to control the entire feedback process since it might lead to improper variable adjustments in some scenarios.

As a complement to $D_{min}(x)$, we propose another pixel-level indicator, noted $v(x)$, which helps reveal the nature of the region over a given pixel x. This indicator works based on the assumption that dynamic background regions will show more blinking foreground pixels in raw segmentation results than uniform regions (whether purely foreground or background). As such, it can guide feedback by constraining parameter adjustments when they are not truly needed (as is the case for intermittent foreground object motion). The way segmentation noise is captured and turned into pixel-level indicators is simple: first, for every new segmentation map S_t, an XOR operation with S_{t-1} exposes a map of all new blinking pixels, noted X_t. Moving object borders are removed from this result by nullifying all pixels of X_t that intersect with the post-processed version of S_t. Then, for every pixel x, we treat $v(x)$ as an accumulator:

$$v(x) = \begin{cases} v(x)+1 & \text{if } X_t(x)=1 \\ v(x)-0.1 & \text{otherwise} \end{cases} \tag{9}$$

Note that we also clamp the result of $v(x)$ to non-negative values. This approach dictates that static foreground or background regions would have $v(x) \approx 0$, while regions with noisy labels would have $v(x) \gg 0$. As a side note, we kept post-processing operations to a minimum and rely only on simple morphological operations and median filtering to get rid of salt and pepper noise in S_t; this is enough to eliminate all blinking pixels in our final results.

Based on the combination of D_{min} and v, we introduce new dynamic controllers for our internal parameters. First, we define the complete relation behind local update rates:

$$T(x) = \begin{cases} T(x) + \frac{1}{v(x)\cdot D_{min}(x)} & \text{if } S_t(x)=1 \\ T(x) - \frac{v(x)}{D_{min}(x)} & \text{if } S_t(x)=0 \end{cases} \tag{10}$$

where $T(x)$ is bound to the [1,256] interval. Like other sample-consensus methods, we use a $1/T(x)$ approach to calculate local update probabilities. Thus, considering the fact that high $T(x)$ values lead to fewer updates, this relation means that when foreground pixels are detected in static background regions with low segmentation noise, model updates will almost immediately stop. In other words, $T(x)$ will max out quickly due to $v(x) \approx D_{min}(x) \approx 0$. However, dynamic and noisy background regions will keep allowing model updates for much longer, as in those cases, $v(x) \gg 0$ and $D_{min}(x) \approx 1$, which results in smoother variations. This is beneficial, as in such situations, foreground pixels are more likely to be segmentation noise.

In most cases of dynamic background motion, having a model that updates very frequently is usually not enough to reduce false foreground labeling. Adjusting distance thresholds for local representation matching is often required; in our case, both color and LBSP distance thresholds are inherited from $R(x)$. The basic feedback relation between this variable and our pixel-level indicators can be described as

$$R(x) = \begin{cases} R(x) + v(x) & \text{if } R(x) < (1+D_{min}(x)\cdot 2)^2 \\ R(x) - \frac{1}{v(x)} & \text{otherwise} \end{cases} \tag{11}$$

Note that $R(x)$ can only be strictly greater or equal to 1; this lower limit reflects the lowest possible distance thresholds, *i.e.* the noise threshold used in perfectly static regions. We opted to control $R(x)$ via $D_{min}(x)$ based on an exponential relation, which we determined was better suited to handle highly uncertain local representation matching when $D_{min}(x) \gg 0$. Here, $v(x)$ directly controls the variation step size of $R(x)$; in static regions, it prevents $R(x)$ from being increased too fast (which helps against camouflage problems), and in dynamic regions, it prevents it from fluctuating based on the current value of $D_{min}(x)$. Following that, we obtain color and LBSP distance thresholds, respectively $R_{color}(x)$ and $R_{desc}(x)$, using

$$R_{color}(x) = R^o_{color} \cdot R(x) \tag{12}$$

and

$$R_{desc}(x) = R^o_{desc} + 2^{R(x)}, \tag{13}$$

where R^o_{color} and R^o_{desc} are their minimal values (we use 20 and 2). In this case, while color thresholds are directly proportional to $R(x)$, LBSP thresholds rely on an exponential relation which reflects their nature: when $R(x)$ is high enough, it allows them to be completely ignored in most dynamic regions, dramatically reducing spatial sensitivity.

Since distance and persistence are already closely related through their influence on the feedback process (via D_{min}), we keep the relation behind persistence thresholds simple, defining it as

$$W(x) = \frac{q(\omega_0, t)}{R(x)\cdot 2}, \tag{14}$$

where ω_0 is the first BW of $C_L(x)$; it is used as a reference here since it generally carries the highest persistence within its local dictionary (since it is constantly being sorted). The idea behind $W(x)$ is to always have at least one local BW with enough persistence to assign a pixel as background by itself. Furthermore, as stated earlier, this takes care of the need to rely on a specific persistence offset value since all thresholds stored in W are kept relative to the actual persistence values of local dictionaries. This means that, unlike most change detection methods, we do not predefine how long old background representations may be kept in the model once they are no longer seen. This duration depends on the nature of the region, on the periodicity of newer representations and on how persistent the old representations were during their lifetime. This is a strong advantage in practice, especially against intermittently moving foreground objects. For example, our model could theoretically keep the true background of a parking lot active forever, even if cars are parked in every available spot, given enough "empty parking" footage.

3. Experiments

In order to properly evaluate how well our new approach handles a wide spectrum of complex scenarios, we decided to test it using the CDnet dataset and compare our results to the methods presented in [5] and those reported on the official website. This dataset is one of the largest to date in this regard: it contains a total of 31 video sequences obtained

Category	*Re*	*Sp*	*FPR*	*FNR*	*PWC*	*Pr*	*FM*
BSL	0.9408	0.9980	0.0020	0.0592	0.4491	0.9394	0.9397
CJT	0.7840	0.9935	0.0065	0.2160	1.4220	0.8660	0.8137
DBG	0.8868	0.9989	0.0011	0.1132	0.1917	0.9038	0.8938
IOM	0.7487	0.9945	0.0055	0.2513	2.3536	0.8392	0.7764
SHD	0.9172	0.9932	0.0068	0.0828	1.0230	0.8710	0.8913
THM	0.8504	0.9910	0.0090	0.1496	1.4018	0.8280	0.8324
overall	0.8547	0.9949	0.0051	0.1453	1.1402	0.8746	0.8579

Table 1. Complete results for our proposed method, PAWCS, using the 2012 CDnet dataset and evaluation tools.

Method	$FM_{Overall}$	FM_{BSL}	FM_{CJT}	FM_{DBG}	FM_{IOM}	FM_{SHD}	FM_{THM}
PAWCS	**0.858**	**0.940**	**0.814**	**0.894**	**0.776**	**0.891**	**0.832**
DPGMM[6]	*0.776*	0.929	0.748	*0.814*	0.542	0.813	0.813
SGMM-SOD[4]	0.766	0.921	0.672	0.688	*0.715*	0.865	0.735
PBAS[8]	0.753	0.924	0.722	0.683	0.574	0.860	0.756
LOBSTER[15]	0.751	0.924	0.742	0.568	0.577	*0.873*	*0.825*
PSPMRF[14]	0.737	0.929	0.750	0.696	0.564	0.791	0.693
SCSOBS[12]	0.728	*0.933*	0.705	0.669	0.592	0.778	0.692
ViBe+[19]	0.722	0.871	*0.754*	0.720	0.509	0.815	0.665
SOBS[11]	0.716	0.925	0.709	0.644	0.563	0.772	0.683
Cheby[13]	0.700	0.865	0.642	0.752	0.386	0.833	0.723
KNN[22]	0.679	0.841	0.689	0.686	0.503	0.747	0.604

Table 2. Overall and per-category F-Measure scores of the 8 best methods originally tested on the CDnet dataset, as well as more recent ones. Red/bold entries indicate the best result and blue/italics the second best.

from various sources. These are split into six different categories: baseline (BSL), camera jitter (CJT), dynamic background (DBG), intermittent object motion (IOM), shadow (SHD) and thermal (THM). The ground truth, which consists of $\sim 90,000$ manually labeled frames, is provided along with evaluation tools to ensure that all methods are always tested uniformly. The performance metrics we use are as described in [5] and include recall (Re), specificity (Sp), false positive rate (FPR), false negative rate (FNR)[2], percentage of wrong classifications (PWC), precision (Pr) and F-Measure (FM). Note that due to space constraints, we only present quantitative results below; qualitative results are provided as supplemental material to this paper, and our complete results can be downloaded from the CDnet website.

First, we show in Table 1 the averaged metrics for our complete method in individual categories as well as overall. We can notice how the increased sensitivity of our method is demonstrated through good recall scores (and low false negative rates). This is, as stated earlier, due to the highly specific nature and long lifetime of the color/LBSP/persistence triplets we use to characterize local background representations. Besides, each test group shows balanced results in both recall and precision, which lead to good F-Measure scores. This also indicates that our method does not suffer from sensitivity problems as much as PBAS did due to its feedback components, thanks to our own two-variable dynamic controllers. The most challenging category in our case is intermittent object motion, followed by camera jitter and thermal; these three were also considered the most challenging for all methods in [5].

Then, we show in Table 2 per-category and overall F-Measure scores for the best methods evaluated during the 2012 Change Detection Workshop, as well as more recent ones. As noted in [5], the general performance of a method is usually closely related to its F-Measure score, hence its selection as the common ground for all our comparisons. As such, we can see that we outperform all other solutions in all six scenario categories, with baseline and thermal showing the smallest improvement. The intermittent object motion category gives outstanding results in our case; this may be explained by the increased lifetime of background representations due to our persistence analysis strategy, which leads to decreased foreground absorption speed and instantaneous model restitutions in previously occluded

[2]Note that the FNR formula used in [5] is incorrect and has since been modified on www.changedetection.net; we used the up-to-date version for Table 1.

areas. Besides, we can also notice a relative performance improvement of 10.6% for the overall results over the second best approach, DPGMM. An interesting comparison can also be made here: LOBSTER, which was proposed in [15], is clearly outperformed by our method, and obtains somewhat close scores in only two out of six categories. Although both use color and LBSP features to model pixel appearances, this ultimately shows that a word-based approach using persistence indicators and self-adjusting parameters is preferable for most change detection scenarios.

4. Conclusion

We proposed a new background subtraction algorithm that analyses the periodicity of local representations based on color and texture information in order to build a new type of persistence-based model. We showed how this algorithm can improve its segmentation results by continuously adjusting its internal parameters via feedback loops. Experiments show that our new method outperforms the best methods tested for the 2012 CDnet workshop as well as more recent ones in terms of F-Measure scores. Our proposed model benefits from the properties of both classic word-based models and sample consensus models by using non-overlapping background words that rely on local representation persistence information. Our results could be further improved by using more sophisticated post-processing operations.

References

[1] O. Barnich and M. Van Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Trans. Image Process.*, 20(6):1709–1724, 2011. 2, 4

[2] G.-A. Bilodeau, J.-P. Jodoin, and N. Saunier. Change detection in feature space using local binary similarity patterns. In *Int. Conf. Comput. Robot Vis.*, pages 106–112, 2013. 1, 2, 3

[3] A. M. Elgammal, D. Harwood, and L. S. Davis. Non-parametric model for background subtraction. In *Proc. 6th European Conf. on Comput. Vis.*, pages 751–767, 2000. 2

[4] R. Evangelio and T. Sikora. Complementary background models for the detection of static and moving objects in crowded environments. In *Proc. IEEE Int. Conf. Advanced Video and Signal Based Surveillance*, pages 71–76, Aug 2011. 7

[5] 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*, pages 1–8, 2012. 1, 2, 6, 7

[6] T. Haines and T. Xiang. Background subtraction with dirichlet process mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(4):670–683, April 2014. 7

[7] M. Heikkila and M. Pietikainen. A texture-based method for modeling the background and detecting moving objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):657–662, 2006. 2

[8] M. Hofmann, P. Tiefenbacher, and G. Rigoll. Background segmentation with feedback: The pixel-based adaptive segmenter. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, pages 38–43, 2012. 2, 4, 7

[9] 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, June 2005. 2, 4, 5

[10] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikainen, and S. Li. Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1301–1306, 2010. 2

[11] L. Maddalena and A. Petrosino. A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Trans. Image Process.*, 17(7):1168–1177, July 2008. 2, 7

[12] L. Maddalena and A. Petrosino. The SOBS algorithm: What are the limits? In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, pages 21–26, 2012. 2, 7

[13] A. Morde, X. Ma, and S. Guler. Learning a background model for change detection. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, pages 15–20, 2012. 7

[14] A. Schick, M. Bauml, and R. Stiefelhagen. Improving foreground segmentations with probabilistic superpixel markov random fields. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, pages 27–31, 2012. 2, 7

[15] P.-L. St-Charles and G.-A. Bilodeau. Improving background subtraction using local binary similarity patterns. In *Proc. IEEE Winter Conf. Applicat. Comput. Vis.*, pages 509–515, March 2014. 2, 3, 4, 7, 8

[16] 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 –252 Vol. 2, 1999. 2

[17] T. Tanaka, A. Shimada, R.-i. Taniguchi, T. Yamashita, and D. Arita. Towards robust object detection: integrated background modeling based on spatio-temporal features. In *Proc. 9th Asian Conf. Comput. Vis. - Vol. I*, pages 201–212, 2010. 2

[18] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *Proc. IEEE Int. Conf. Comput. Vis.*, volume 1, pages 255–261 vol.1, 1999. 2

[19] M. Van Droogenbroeck and O. Paquot. Background subtraction: Experiments and improvements for ViBe. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, pages 32–37, 2012. 2, 4, 7

[20] H. Wang and D. Suter. A consensus-based method for tracking: Modelling background scenario and foreground appearance. *Pattern Recognit.*, 40(3):1091–1105, 2007. 2

[21] M. Wu and X. Peng. Spatio-temporal context for codebook-based dynamic background subtraction. *Int. J. Electron. Commun.*, 64(8):739 – 747, 2010. 2, 4

[22] 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. 7