Simultaneous foreground detection and classification with hybrid features

Jaemyun Kim[†], Adín Ramírez Rivera[‡], Byungyong Ryu[†], Oksam Chae[†*]
[†]Dept. of Computer Engineering, Kyung Hee University, Gyeonggi-do, South Korea
[‡]Escuela Informática y Telecomunicaciones, Universidad Diego Portales, Santiago, Chile
sense21c@khu.ac.kr, adin.ramirez@mail.udp.cl, read100nm@khu.ac.kr, oschae@khu.ac.kr

Abstract

In this paper, we propose a hybrid background model that relies on edge and non-edge features of the image to produce the model. We encode these features into a coding scheme, that we called Local Hybrid Pattern (LHP), that selectively models edges and non-edges features of each pixel. Furthermore, we model each pixel with an adaptive code dictionary to represent the background dynamism, and update it by adding stable codes and discarding unstable ones. We weight each code in the dictionary to enhance its description of the pixel it models. The foreground is detected as the incoming codes that deviate from the dictionary. We can detect (as foreground or background) and classify (as edge or inner region) each pixel simultaneously. We tested our proposed method in existing databases with promising results.

1. Introduction

In these days, the number of video-based surveillance systems is increasing due to its use in several applications, such as vision-based traffic system [4], video segmentation [13, 15], or human behavior analysis [1]. Most methods for video-based surveillance rely on moving object detection. One popular approach to implement the latter is background subtraction. Although background modeling has been widely studied, there remain many unsolved challenges. Several methods [4, 9, 13, 14, 17, 18, 21, 27, 28] have tried to solve problems from global and local illumination changes, dynamic background (such as trees waving and rippling water), background changes (new background appearance, *birth* and *death*), and shadow effects as shown in Fig. 1. We classify traditional background modeling methods as pixel- and edge-based methods. Nonetheless, recent surveys [3, 26] present a complete view of the field.

Pixel-based methods use the information of pixels, such as intensity or color, to represent consecutive frames [17, 18, 27, 28]. These methods are robust to changes in location and orientation of background objects. However, they are sensitive to pixel-value changes, such as global and local illumination changes. Single modal representation was used [18, 28] (e.g., median, average, or single Gaussian), but background variation from illumination is not well represented by a single model. To overcome this limitation, mixture of Gaussians (MoG) are used to generate a stable background [27]. In this type of methods, each pixel has k Gaussian models with weights representing their priority. Nevertheless, these methods depend on the correct number of background variation k, as well as an accurate update ratio for foreground with slow motion. Heikkilä and Pietikäinen [7] proposed a mixture of histograms that are based on local binary patterns (LBP) to model the background. They borrowed ideas from the MoG methods, and changed the way of modeling the distributions. Other methods [25] that model the background based on LBP has been proposed too. Also, codebook-based methods are used for background modeling [6, 17]. These approaches use codewords with variable size to represent pixel variation instead of using Gaussians for each pixel distribution. The automatic generation of codewords, however, in changing environments may produce a large number of codewords. On the other hand, there are non-parametric methods, such as the method proposed by Hoffman *et al.* [8] or previously proposed ViBe [2], that do not assume a shape on the distributions but rather use a history of samples to make decisions about the foreground.

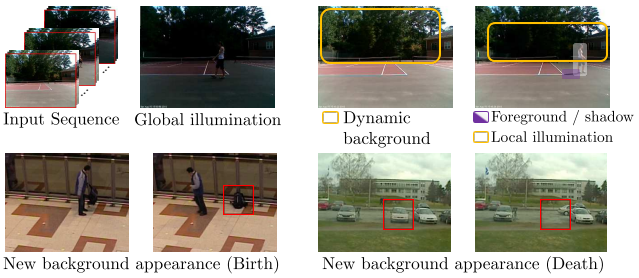
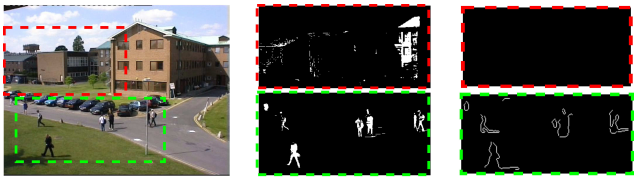


Figure 1. Examples for the background variations, such as global and local illumination, dynamic background, shadows effect from moving objects, and background changing (*birth* and *death*).

*Corresponding author



Frame | Pixel based method | Edge-segment based method

Figure 2. Problems of pixel and edge based methods. Red-dashed boxes show the region of illumination change, and green-dashed boxes show foreground region. Pixel-based methods are sensitive to illumination changes and lose some boundary of the foregrounds, as shown in the second column. Edge-segment based (or edge based) methods are robust against illumination variations, as the red-dashed box on third column shows. However, these methods cannot define the inner information of foregrounds (see green-dashed box on third column).

On the other hand, edge-based methods use only location of edges (not all pixel locations) to represent the background model. Edges are more stable to illumination changes than pixel values. Sequential frames, however, may not have exactly the same edge position, and have shape and length changes due to noisy frames. Existing edge-pixel-based methods use binary information [4, 13] or models [11] as existence of edge on each pixel. This strategy may generate many false alarms for foreground detection due to edge distortion from adjacent frames. To solve the edge-distortion problem, edge-segment-based methods emerged to take advantage of the edge existence and its shape information [9]. An edge-segment is the concatenation of adjacent edges, and they inherit the problems of edges: shape and position changes. Thus, naive comparison of edge-segments produce similar results as edge-pixel-based approaches. Statistical edge-segment-based methods extract movement of edge-segments including edge distortion [14, 16, 21]. These methods solve the edge-variation problem by accumulating edge existence from a training sequence. Each accumulated region represents an edge-segment distribution and each region refines their statistical properties after each frame to generate a stable background model. Since edge-based and edge-segment-based methods detect foreground as edges, these methods depend on a post-processing to extract the regions defined by the detected edges. Moreover, these methods have problems updating their background model to adjust for background *birth* and *death*.

In this paper, we propose a local information encoding algorithm and a hybrid background modeling method that uses a mixture of edge and pixel representations. Thus, it mixes the advantages of pixel-based and edge-based methods for background modeling (and simultaneously tries to solve their common problems, see Fig. 2).

In this method we encode the information of each pixel and its neighborhood (both the structural and texture information), and compute the frequency of the edge and inner non-edge characteristics by generating a Local Hybrid Pattern (LHP) code that represents either edge or inner information. We put a flag in the LHP code to separate the two types of information.

To generate the background model, Adaptive Dictionary Model (ADM), we mix the ideas of previous methods [7,8,27], as we use a dynamic history of compressed information (the LHP codes) that is weighted and maintained through an online updating mechanism. Previous methods that use hybrid information and similar encoding patterns exist [12, 19, 30]; however, they encode information of the generated codebook, which may change later thus creating inconsistencies with the coded information. Moreover, both features are used simultaneously regardless of the type of region. Thus, the fact that gradient is not significant on flat regions is overlooked. On the contrary, our proposed method uses the most relevant feature in each region (*i.e.*, flat or edge regions).

In detail, first we create an adaptive dictionary of LHP codes for each pixel. Then, we update it similarly to previous methods [27], we weight each code and dynamically add or remove them in the modeling process. The flag in the LHP codes induces two different distributions, between edge and inner, in the model's frequency. Thus, we can jointly model the structure and texture information of the pixels in the background. For foreground detection, we generate a probability map for each pixel, based on the LHP code dictionary probabilities. Consequently, detected candidate foregrounds are divided into edge and inner regions, and the final decision of foregrounds uses the two candidate regions. By using these edge and inner characteristics, we produce results that are less sensitive to global and local illumination changes and noise. Moreover, our method can detect both foreground object and its boundary stably and simultaneously.

Our main contributions are as follows:

- A hybrid background modeling technique that classifies each detected pixel as one of two groups: inner or edge. Finding edges and inner regions simultaneously provides an advantage of the proposed method over existing ones. This information can be fed to other algorithms for further processing, such as providing inner edges of pedestrians for determining their pose, or separating merged objects through the edge information in the detection, among others.
- A Local Hybrid Pattern (LHP) coding scheme to jointly represent texture and edges.
- A dictionary technique, Adaptive Dictionary Model (ADM), to create structures to model the underlying distributions of the characteristics that appear in each pixel through the dynamic maintenance of the history of observed codes.

- Foreground classification using heterogeneous information (characteristics of edge and inner regions).

2. Local Hybrid Pattern

For training and testing sequences, we convert each input image to a coded image in which every pixel is classified as edge or inner region. Each region type maintains different properties. The edge region has boundary properties, such as a primary edge direction and an edge magnitude. These properties help to solve the edge overlapping problem—*i.e.*, when an edge of another object (foreground) may be confused with background edges. On the other hand, the inner information helps to enrich the discrimination of the model, as texture and flat regions do not have a prominent direction and the inclusion of edge information in that areas will introduce noise into the model.

To decide which type of code we will generate for a particular pixel, we inspect the maximum edge response of the local neighborhood of a given pixel. If the pixel's maximum edge response is high enough, we will consider it part of an edge region; otherwise it will be an inner region, and we will code it accordingly. We calculate the gradient response by using a compass mask—note that any set of masks that generates M discrete directions can be used, in our case we used four ($M = 4$) directional Sobel masks.

Let $g_i(x)$ denote the gradient response of the ith mask at the pixel position x in a given image. Then, we can find the maximum response of a given pixel position x by

$$g^*(x) = \max\{g_i(x) \mid \forall\, i\}. \tag{1}$$

We create a binary image f that flags each pixel, at position x, as edge (0) or inner (1) through

$$f(x) = \begin{cases} 0 & \text{if } g^*(x) > T_{\text{edge}}, \\ 1 & \text{otherwise,} \end{cases} \tag{2}$$

where x is the pixel location, and T_{edge} is a threshold to decide whether the response is significant enough.

In the following we will incorporate this flag into the code (in the most significant bit, for representation sake) to decide whether the information corresponds to an edge or an inner region. Then the rest of the bits in the code can be used to represent the edge and inner information accordingly. We generate the code as

$$C(x) = \begin{cases} 2^{n-1} f(x) + C_{\text{edge}}(x) & \text{if } f(x) = 1, \\ 2^{n-1} f(x) + C_{\text{inner}}(x) & \text{otherwise,} \end{cases} \tag{3}$$

where n is the length (number of bits) of the code C to be generated, and C_{edge} and C_{inner} are code functions for edge and inner pixels, respectively, that generate codes of size $n-1$ bits as shown in Fig. 3.

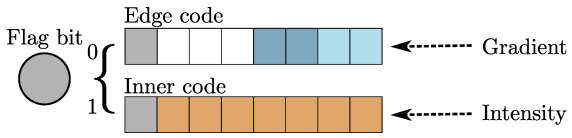


Figure 3. Code generation for LHP. The code is separated for either edge or inner region. In this paper, the edge code uses the principal direction and quantized gradient magnitude (two bits each), while inner code uses the quantized intensity information (the size depends on the quantization).

In the case of edge-based code generation, we can use any type of edge-based coding, such as LDP [10, 29], LDN [22–24], LBP [7], or even dynamic-features codes [20], among others. In this paper, we propose to use a code that uses the principal directional number, P, and its orthogonal gradient magnitude, $g_{\perp P}$. For the principal direction we use 2 bits, and for the gradient we use 2 bits to represent such information. The coding function is

$$C_{\text{edge}}(x) = 4P(x) + g_{\perp P(x)}(x), \tag{4}$$

where the principal directional number for the pixel at position x is computed by

$$P(x) = \arg\max_i \{g_i(x) \mid 0 \le i \le M-1\}, \tag{5}$$

which ranges from 0 to $M-1$, where M is the number of directional masks used, and the orthogonal magnitude is encoded in a ternary pattern by

$$g_{\perp P(x)}(x) = \begin{cases} 0 & \text{if } g_j < -T_{\text{tern}}, \\ 1 & \text{if } -T_{\text{tern}} \le g_j \le T_{\text{tern}}, \\ 2 & \text{if } g_j > T_{\text{tern}}, \end{cases} \tag{6}$$

where j is the orthogonal direction of the principal direction at pixel x, and T_{tern} is a threshold to divide the magnitude into a ternary pattern.

To compute the code of the inner features, we simply use the first significant bits of the gray intensity of the given pixel x. In our case, we set the general code C to have $n = 8$ bits; thus we have seven remaining bits to use for the C_{inner} code. Therefore, we compute the code by doing a right shift operation on the intensity level, I, of the pixel x by

$$C_{\text{inner}}(x) = I(x) \gg b, \tag{7}$$

where $\gg$ shifts b bits of the left-operand (in general, $b \ge 1$ due to the code length n).

3. Adaptive Dictionary Model

The proposed background model is comprised by an adaptive code dictionary at each pixel. We define a dictionary as a set of codes with an associated weight value, Adaptive Dictionary Model (ADM). To construct the codes

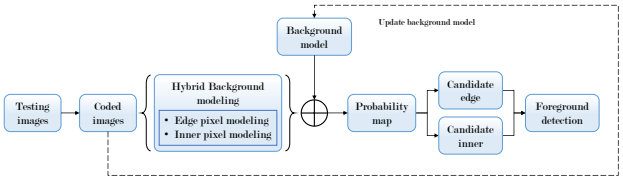


Figure 4. Overall flow of the proposed background modeling algorithm.

we use the LHP introduced in Section 2—although any coding method can be used. The dictionaries are updated online during the sequence processing. This approach allows us to recover stable codes from the incoming sequence, and discard unstable or noisy codes. The dictionaries act as an approximation of the underlying distributions of features at each pixel (as we maintain the occurrence of stable structural and textural features on it). With the difference that each dictionary holds the frequency of quantized features (the codes) instead of a continuous feature. Moreover, the use of LHP helps to adaptively discriminate the features that are dominant in each neighborhood. For example, on neighborhoods with high structural information, the non-relevant color information is discarded; while in neighborhoods with low edginess, the texture information is preserved. For detection, we classify the incoming pixels as background if they are within the learned code sets, or as foreground otherwise. An overall flowchart of the proposed method is shown in Fig. 4.

3.1. Code-based Background Modeling

We model the background as a set of dictionaries $B = \{D_x\}$, for every position x in the image. Our dictionary is a set of codes that is actively changing during the sequence processing, unlike common codebook approaches that learn a static codebook and use it afterwards. Thus, let

$$D_x = \{(c_0, w_0), (c_1, w_1), \ldots, (c_{N-1}, w_{N-1})\}, \quad (8)$$

be the dictionary for the pixel at position x, that is comprised of tuples (c_i, w_i), where c_i is the ith code with its own associated weight w_i. The dictionaries have a limit of N codes, to keep them manageable.

3.2. Background Model Updating

For adjusting background changes, such as background *birth* and *death*, the background model needs to be updated frame by frame. Foreground objects in a sequence can change to background when they stay stationary for a long time, what we called *birth*. In contrast, hidden background appears when an object moves or leaves the scene, what we called *death*.

Let $B^t = \{D_x^t \mid \forall x\}$ be the current background model at frame t. Then we compute the next state of the background model B^{t+1} by updating the dictionary D_x^t at each pixel position x.

Let $c = C^t(x)$ be the code for position x at frame t computed by (3), and $(d_i, w_i) \in D_x$ be the ith tuple of the corresponding dictionary. We update each associated weight of all the codes in the dictionary through $w_i^{t+1} = u(c, d_i, w_i^t)$, such that

$$u(c, d_i, w_i) = \begin{cases} (1-\alpha)w_i + \alpha & \text{if } m(c, d_i) = 1 \\ (1-\alpha)w_i & \text{otherwise,} \end{cases} \quad (9)$$

where u is the update function, c is the incoming code, d_i is the corresponding ith code in the dictionary, w_i is the associated weight, α is the learning rate at which we learn and forget the codes—similar to previous research [27] on online learning—, and $m(\cdot, \cdot)$ is a function that matches the code c with the dictionary code d_i. We define m as

$$m(c, d) = \begin{cases} m_{\text{edge}}(c, d) & \text{if } c \text{ is edge,} \\ m_{\text{inner}}(c, d) & \text{otherwise,} \end{cases} \quad (10)$$

as c is a code created through (3), we can know whether it is an edge code by checking its most significant bit, then we need to check the codes according to their types through

$$m_{\text{edge}}(c, d) = \begin{cases} 1 & \text{if } c = d, \\ 0 & \text{otherwise,} \end{cases} \quad (11)$$

and

$$m_{\text{inner}}(c, d) = \begin{cases} 1 & \text{if } |c - d| \leq T_{\text{inner}}, \\ 0 & \text{otherwise,} \end{cases} \quad (12)$$

where T_{inner} is a threshold to relax the matching of the quantized intensities.

In case no match was found during this process, a new code will be added to the dictionary D_x. To do so, the least significant code on it will be dropped (if it exists), and a new tuple will be created with the not-found code c and starting weight α. That is,

$$D_x^{t+1} = \begin{cases} \{D_x^t \setminus \{(d_j, w_j)\}\} \cup \{(c, \alpha)\} & \text{if } |D_x| \geq N, \\ D_x^t \cup \{(c, \alpha)\} & \text{otherwise,} \end{cases} \quad (13)$$

where $\setminus$ is the set difference operator, (d_j, w_j) is the tuple with the minimum weight w_j, and $|D_x|$ is the number of elements in D_x.

4. Foreground Detection

Our proposed background model incorporates both edge and inner region distributions, so we can mix their advantages for the foreground detection. For example, when the illumination of the scene varies, we can improve accuracy of foreground detection by using the robustness of the edge-based detection, and the use of inner information helps refining the results and filling the holes in the detection. To achieve these results, our method divides foreground detection into two steps: candidate foreground construction, and foreground classification.

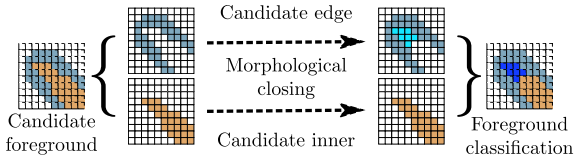


Figure 5. We apply morphological closing to each candidate foreground (edge and inner), and check if the regions overlap. If they do, we keep them as foreground, otherwise they are considered noise.

4.1. Candidate Foreground

As every incoming frame is transformed into its coded version through (3), and its model gets updated through (9), we can use the code information and the background model to infer the probability of a code of belonging to the dictionary. Thus, for every pixel in the coded image C^t we compute its probability of being background, by

$$p(x) = \mathcal{P}\left(C^t(x), D_x^t\right) \tag{14}$$

where p is the probability map of the incoming frame, $\mathcal{P}$ is a function that returns the probability of belonging to a given dictionary, and $C^t(x)$ is the code of the pixel x at frame t. We define the probability $\mathcal{P}$ as

$$\mathcal{P}(c, D) = \begin{cases} norm(w_i) & \text{if } \exists\, d \in D : m(c,d) = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

where c is a code, D is a dictionary, $m(\cdot,\cdot)$ is (10), and

$$norm(w_i) = \frac{w_i}{\sum_j w_j}, \tag{16}$$

is a function that normalizes the weights in the given dictionary to approximate them as probabilities.

Finally, the candidate region is

$$F_{\text{cand}}(x) = \begin{cases} 1 & \text{if } p(x) < T_{\text{cand}}, \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

where T_{cand} is a threshold of the probability of what is consider foreground, for all x.

4.2. Foreground Classification

We need to refine the candidate foreground, as spurious edge and non-edge regions may appear. To reduce the false positives, we find regions of edge and non-edge features that are contiguous to each other. Thus, we perform a closing operation on the candidate regions to fill holes to some extent, and then find the regions that intersect with each other—see Fig. 5. Then, we keep the regions that intersected with each other as foreground, and remove the rest.

As we know the origin of each code, due to our code generation, we can detect and classify each detected pixel as boundary or inner region. This distinction helps us to provide more information on the detected objects.

5. Experimental Results

We examined the proposed method performance against several datasets from the ChangeDetection database [5]. The ChangeDetection database includes a total of six different categories that show typical visual data captured today in surveillance, smart environment, and video analytics applications. We chose five datasets that contain moving objects with small background motion, pedestrians movement, abandoned object, moving objects with their shadows, and new background appearance by an un-parking car, to evaluate the hybrid features used in our proposed algorithm. In detail, the tested datasets were Highway (HIGH), Pedestrians (PED), PETS 2006 (P2006), Bungalows (BUN), and WinterDriveway (WD) of ChangeDetection database. They were extracted from three categories: Baseline (HIGH, PED, and P2006), Shadows (BUN), and Intermittent Object Motion (WD). Regarding the HIGH and PED sequences, both have fewer background movements but have illumination changes, and the P2006 have objects that stay a long time and become new background (however, this behavior is not reflected in the ground truth), and WD shows real background after a parked object (leaving car) moves away.

We compared our method against five state of the art methods. Namely, Pixel-Based Adaptive Segmenter (PBAS) [8], visual background extractor (ViBe) [2], a multiple cue system based on SALBP (MultiCue) [19], texture description with local binary patterns (LBP) [7], and the classical Mixture of Gaussians (MoG) [27]. To compare the results with the previous methods we used precision (Pr), recall (Re), and reported F-measure against other methods defined as

$$Pr = \frac{TP}{TP + FP}, \tag{18}$$

$$Re = \frac{TP}{TP + FN}, \tag{19}$$

$$F = \frac{2(Pr)(Re)}{Pr + Re}, \tag{20}$$

where TP are true positives, FP are false positives, and FN are false negatives, as described by the evaluation protocol of the ChangeDetection database [5].

5.1. Parameter Setting

We have two sets of parameters in our algorithm, one for the coding algorithm LHP, and one for the background modeling algorithm ADM. The parameters for the LHP edge codes are $T_{\text{edge}} = 100$ which determines what pixels will be considered edges or inner regions. Higher values of this parameter will make thinner boundaries, while small values will produce thicker ones; and $T_{\text{tern}} = 100$ characterizes the type of the edges' gradient response, as we are considering anything with less than T_{edge} to be a non-edge region,

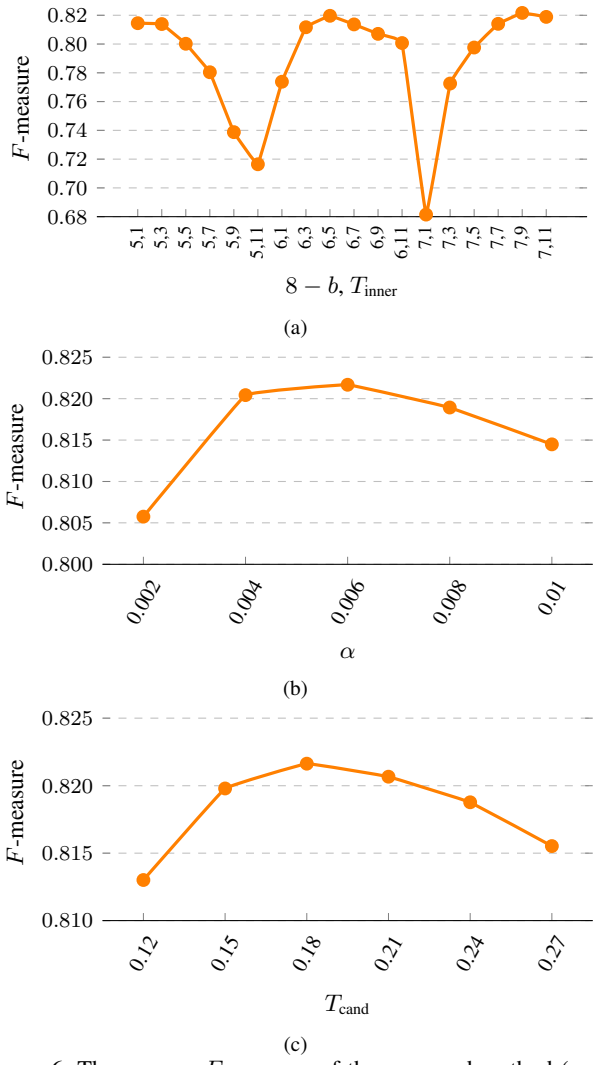


Figure 6. The average F-measure of the proposed method (over all the datasets) while varying (a) the bit length of inner code [$8-b$, where b is as in (7)] and its matching distance (T_{inner}), (b) the weight value for initial code and increment of the weight that is matched (α), and (c) the percentage of the weight to be considered background (T_{cand}).

we are using the same value to divide the gradient into two groups. Regarding the texture part of the LHP code, we performed several tests [as shown in Fig. 6(a)] to evaluate the behavior of the detection using different values of b and T_{inner}. We found that higher code lengths ($b = 1$) gave better performance in comparison to smaller code lengths. On the other hand, the threshold for matching the inner codes T_{inner} shows a behavior dependent on the code length; for higher code lengths more flexibility works best ($T_{\text{inner}} = 9$), on the contrary for small code lengths a less flexible threshold gives better results ($T_{\text{inner}} = 3$). That behavior can be explained as follows. Low code lengths have a higher quantization; thus, a tighter comparison will be better as

Table 1. Quantitative results of the proposed method: ADM.

Dataset	Precision	Recall	F	F (PP)
HIGH	0.914	0.912	0.913	0.942
PED	0.927	0.877	0.901	0.914
P2006	0.720	0.697	0.708	0.713
BUN	0.980	0.919	0.949	0.957
WD	0.640	0.634	0.637	0.668
Avg.	**0.836**	**0.808**	**0.822**	**0.839**

the codes are already grouped together. On the other hand, with higher code lengths, we need more flexible thresholds as there is more diversity in the codes due to less quantization. Finally, for the rest of the experiments we picked $b = 1$ (yielding a code length of seven) and $T_{\text{inner}} = 9$.

For the background modeling algorithm, we use a dictionary size of $N = 10$ which means that we can hold up to ten different modes in our model per pixel. We evaluated several learning rates for the algorithm [shown in Fig. 6(b)], and settle with a learning rate $\alpha = 0.006$. Moreover, the threshold to evaluate what is considered background or foreground in the candidate foreground stage was tested and reported in Fig. 6(c), and found that the best value was $T_{\text{cand}} = 0.18$. From now onward, when we refer to experiments these values were used to perform them (unless otherwise stated explicitly).

5.2. Quantitative Evaluation

We did a quantitative evaluation following the protocol proposed by Goyette *et al*. [5], and show results without a post processing median filter (ADM) and with it (ADM+PP). We show on Table 1 several metrics of the proposed algorithm in all the datasets. Moreover, we compared the performance against other methods that have similar cornerstone ideas on the background's processing (for a fair comparison), shown in Fig. 7. In general, PBAS [8] get comparable results with our method, and both obtain better result than the rest. In specific, we get comparable results against PBAS on three datasets (HIGH, PED, and BUN). For the P2006 sequence, our proposed method absorbed the objects left behind into the background (and the ground truth does not take into consideration that the objects should be considered background after some time)—see Section 5.3 for a deeper discussion. Regardless, we are the second best in that dataset. Moreover, if we use a parameter set learned only from the P2006 sequence ($T_{\text{edge}} = T_{\text{tern}} = 61$, $b = 1$, $T_{\text{inner}} = 11$, $\alpha = 0.0005$, and $T_{\text{cand}} = 0.12$ obtained as in Section 5.1), we get an F-measure of 85% (without PP) and 87% (with PP)—which is higher than PBAS. For the WD dataset, we obtained more than 10% F-measure than the second best. In the latter, the sequence reveals new background and it should be incorporated quickly into the background model to stabilize it. In comparison to previous methods, we can do it rather

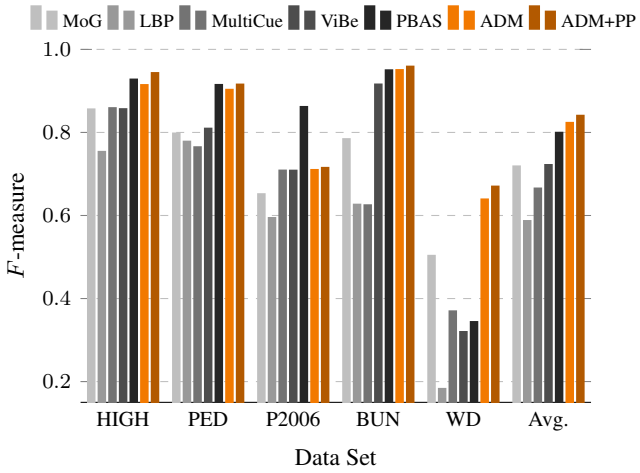


Figure 7. F-measure comparison of different methods.

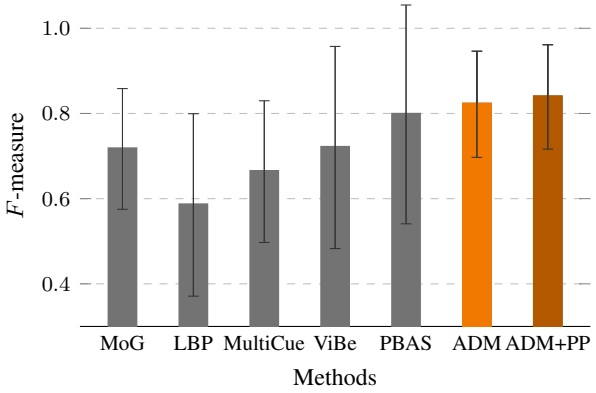


Figure 8. Average and standard deviation of F-measure comparison of different methods.

quickly without compromising the detection of the car that is considered foreground.

In general, our average result on all the datasets is better than previous methods. And we have a smaller standard deviation—as shown in Fig. 8—in comparison to the second best method. That means that our method obtains more consistent results in comparison to previous methods.

5.3. Qualitative Results

Fig. 9 shows sample frames from all the different methods in all the datasets. We observe that the proposed method recovers contiguous regions inside the objects while obtaining well-defined boundaries. Moreover, the proposed method can recover better-blurred parts, like the front wheel of the bicycle on the PED dataset, while other methods miss it or recover noisy regions (like MoG).

In the case of the P2006, we can see that our algorithm absorbs a backpack that was left on the floor. That explains part of our low F-measure in that sequence, as we used the same parameters for all the sequences. One way to improve the results is to tune the learning rate to adjust it to incorporate stationary objects into the dictionaries more slowly.

Figure 10. Usefulness of hybrid features against stationary objects. One person (from frame 490) loiters around (until frame 946), and he has very slow and small movement to drop a backpack on the floor. At the moment, the background learned his appearance. However, the edges are different from the learned structure, which allows its detection (*c.f.* frame 977 and 1145).

For the WD dataset, new background is revealed as part of the motion of a parked car leaving. Our proposed algorithm adjusts to the background changes quickly and detects the new foreground object. However, this sequence presents a challenge to the previous methods as they recover ghosts on the foreground and noisy regions.

In general, our dictionary method is more flexible and simple than previous non-parametric approaches, while still obtaining comparable results. Moreover, the ability to change the dictionary with the sequence makes our method more adaptable in comparison to a learned codebook from a given set of data. Additionally, the distinction between edge and inner regions grants us more flexibility to further process each set of pixels independently.

Our proposed method shows robustness of foreground detection when the foreground moves very slow and is absorbed by the background model. Fig. 10 shows a person that comes to the scene and loiters for several frames until frame 946, and he drops a backpack to the floor. In this scenario, the inner region of the foreground is learned to the background easily but edge region is detected as foreground. This example shows the advantages of using hybrid features, as the drawback of one can be circumvented by the other.

6. Conclusions

In this paper, we proposed a hybrid method to create a background model of scenes that jointly models object's boundaries as well as its inner regions. The proposed method uses edge and inner features to create a dictionary of codes that model the underlying feature distributions at each pixel. These distributions are used to classify the foreground as the codes of incoming frames that deviate from the learned distributions. We performed several experiments in which the proposed method obtained promising results. Additionally, our proposed coding scheme (LHP) provides joint information of the interest regions (as we detect and classify the detected region simultaneously) that can be used in further steps, such as pose estimation, behavior recognition, among others, for a target application.

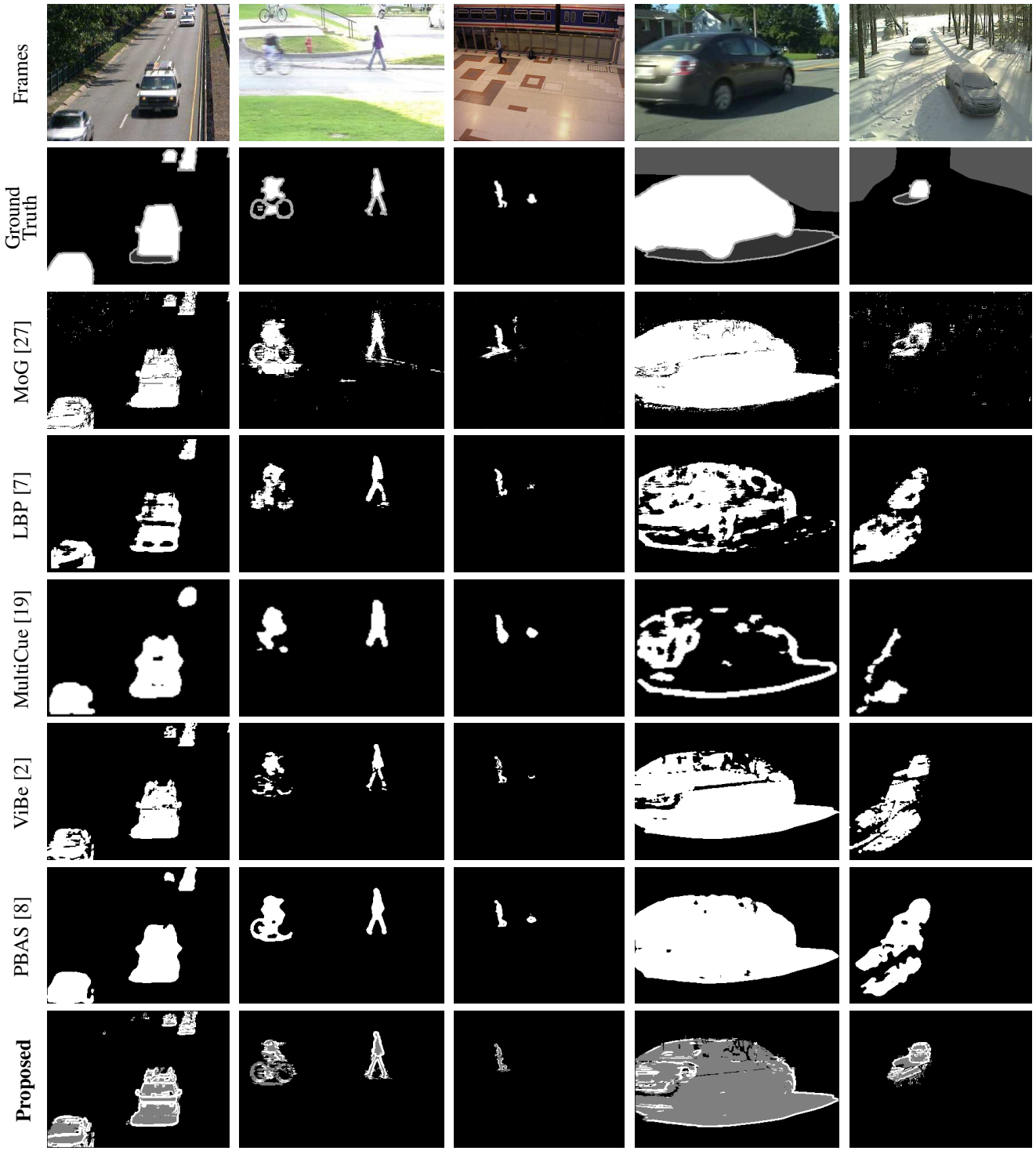


Figure 9. Examples of detection on several datasets on each of the different sequences. (a) HIGH, (b) PED, (c) P2006, (d) BUN, (e) WD from ChangeDetection database. (In the proposed method the white areas are edge features, and the gray areas are non-edge features.)

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2015R1A2A2A01006412), and by CONICYT grant FONDECYT Iniciación No. 11130098.

References

[1] P. Banerjee and S. Sengupta. Human motion detectionandtracking for video surveillance. In *In National Conference on Communication*, IITBombay, India, Feb. 2008.

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

[3] T. Bouwmans. Traditional and recent approaches in background modeling for foreground detection: An overview. *Computer Science Review*, 1112(0):31–66, 2014.

[4] D. Dailey, F. Cathey, and S. Pumrin. An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE Trans. Intell. Transp. Syst.*, 2000.

[5] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. changedetection.net: A new change detection benchmark dataset. In *CVPRW*, 2012.

[6] J.-M. Guo, C.-H. Hsia, Y.-F. Liu, M.-H. Shih, C.-H. Chang, and J.-Y. Wu. Fast background subtraction based on a multilayer codebook model for moving object detection. *IEEE Trans. Circuits Syst. Video Technol.*, 23(10):1809–1821, Oct. 2013.

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

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

[9] M. Hossain, M. Dewan, and O. Chae. Moving object detection for real time video surveillance: An edge based approach. *IEICE Transactions*, 2007.

[10] T. Jabid, M. H. Kabir, and O. Chae. Robust facial expression recognition based on local directional pattern. *ETRI Journal*, vol.32, no.5(5):784–794, Oct. 2010.

[11] V. Jain, B. Kimia, and J. Mundy. Background modeling based on subpixel edges. In *ICIP 2007*. t, Sept. 2007.

[12] O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *Proceedings of the Workshop on Motion and Video Computing*, MOTION '02, pages 22–, Washington, DC, USA, 2002. IEEE Computer Society.

[13] C. Kim and J. Hwang. Fast and automatic video object segmentation and tracking for content-based applications. *IEEE Trans. Circuits Syst. Video Technol.*, 2002.

[14] J. Kim, M. Murshed, A. Ramirez Rivera, and O. Chae. Background modelling using edge-segment distributions. *International Journal of Advanced Robotic Systems*, 10, 2013.

[15] J. Kim, A. Ramirez Rivera, B. Ryu, K. Ahn, and O. Chae. Unattended object detection based on edge-segment distributions. In *Proc. IEEE AVSS*, pages 283–288, 2014.

[16] J. Kim, A. Ramirez Rivera, G. Song, B. Ryu, and O. Chae. Edge-segment-based background modeling: Non-parametric online background update. In *Proc. IEEE AVSS*, 2013.

[17] 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.

[18] N. J. B. McFarlane and C. P. Schofield. Segmentation and tracking of piglets in images. *Mach. Vis. Appl.*, pages 187–193, 1995.

[19] S. Noh and M. Jeon. A new framework for background subtraction using multiple cues. In K. Lee, Y. Matsushita, J. Rehg, and Z. Hu, editors, *Computer Vision – ACCV 2012*, volume 7726 of *Lecture Notes in Computer Science*, pages 493–506. Springer Berlin Heidelberg, 2013.

[20] A. Ramirez Rivera and O. Chae. Spatiotemporal directional number transitional graph for dynamic texture recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, PP(99):1–1, 2015.

[21] A. Ramirez Rivera, M. Murshed, J. Kim, and O. Chae. Background modeling through statistical edge-segment distributions. *IEEE Trans. Circuits Syst. Video Technol.*, 2013.

[22] A. Ramirez Rivera, J. Rojas Castillo, and O. Chae. Local directional number pattern for face analysis: Face and expression recognition. *IEEE Trans. Image Process.*, 2012.

[23] A. Ramirez Rivera, J. Rojas Castillo, and O. Chae. Local gaussian directional pattern for face recognition. In *International Conference on Pattern Recognition (ICPR)*, pages 1000–1003, Nov. 2012.

[24] A. Ramirez Rivera, J. Rojas Castillo, and O. Chae. Recognition of face expressions using local principal texture pattern. In *Image Processing, 2012. ICIP 2012. IEEE International Conference on*, Sept. 2012.

[25] C. Silva, T. Bouwmans, and C. Frélicot. An extended center-symmetric local binary pattern for background modeling and subtraction in videos. In *10th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, Berlin, Germany, 2015.

[26] A. Sobral and A. Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122(0):4–21, 2014.

[27] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 246–252, 1999.

[28] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):780–785, 1997.

[29] B. Zhang, Y. Gao, S. Zhao, and J. Liu. Local derivative pattern versus local binary pattern: Face recognition with high-order local pattern descriptor. *IEEE Trans. Image Process.*, 19(2):533–544, Feb. 2010.

[30] Z. Zhang, C. Wang, B. Xiao, S. Liu, and W. Zhou. Multi-scale fusion of texture and color for background modeling. In *Proc. IEEE AVSS*, pages 154–159, 2012.