A Texture-Based Method for Modeling the Background and Detecting Moving Objects

Marko Heikkilä and Matti Pietikäinen, *Senior Member, IEEE*

Abstract

This paper presents a novel and efficient texture-based method for modeling the background and detecting moving objects from a video sequence. Each pixel is modeled as a group of adaptive local binary pattern histograms that are calculated over a circular region around the pixel. The approach provides us with many advantages compared to the state-of-the-art. Experimental results clearly justify our model.

Index Terms

Motion, texture, background subtraction, local binary pattern.

I. Introduction

Background subtraction is often one of the first tasks in machine vision applications, making it a critical part of the system. The output of background subtraction is an input to a higher-level process that can be, for example, the tracking of an identified object. The performance of background subtraction depends mainly on the background modeling technique used. Especially natural scenes put many challenging demands on background modeling since they are usually dynamic in nature including illumination changes, swaying vegetation, rippling water, flickering monitors etc. A robust background modeling algorithm should also handle situations where new objects are introduced to or old ones removed from the background. Furthermore, the shadows of the moving and scene objects can cause problems. Even in a static scene frame-to-frame changes can occur due to noise and camera jitter. Moreover, the background modeling algorithm should operate in real-time.

A large number of different methods for detecting moving objects have been proposed and many different features are utilized for modeling the background. Most of the methods use only the pixel color or intensity information to make the decision. To the authors' knowledge, none of the earlier studies have utilized discriminative texture features in dealing with the problem. Only some simple statistics of neighborhoods may have been considered. This is maybe due to the high computational complexity and limited performance of texture methods.

In this article, we propose an approach that uses discriminative texture features to capture background statistics. An early version of the method based on block-wise processing was presented in [1]. For our method we chose the local binary pattern (LBP) texture operator [2], [3], which has recently shown excellent performance in many applications and has several properties that favor its usage in background modeling. Perhaps the most important properties of the LBP operator are its tolerance against illumination changes and its computational simplicity. In order to make the LBP even more suitable to real-world scenes, we propose a modification to the operator and use this modified LBP throughout this paper. Unlike in most other approaches, the features in background modeling are computed over a larger area than a single pixel. This approach provides us with many advantages and improvements compared to the state-of-the-art. Our method tries to address all the issues mentioned earlier except the handling of shadows which turned out to be an extremely difficult problem to solve with background modeling. In [4], a comprehensive survey of moving shadow detection approaches is presented.

II. Related Work

A very popular technique is to model each pixel in a video frame with a Gaussian distribution. This is the underlying model for many background subtraction algorithms. A simple technique is to calculate an average image of the scene, to subtract each new video frame from it and to threshold the result. The adaptive version of this algorithm updates the model parameters recursively by using a simple adaptive filter. This single Gaussian model was used in [5].

The previous model does not work well in the case of dynamic natural environments since they include repetitive motions like swaying vegetation, rippling water, flickering monitors, camera jitter etc. This means that the scene background is not completely static. By using more than one Gaussian distribution per pixel it is possible to handle such backgrounds. In [6], the mixture of Gaussians approach was used in a traffic monitoring application. The model for pixel intensity consisted of three Gaussian distributions corresponding to the road, vehicle and shadow distributions.

One of the most commonly used approaches for updating the Gaussian mixture model was presented in [7]. Instead of using the exact EM algorithm, an online K-means approximation was used. Many authors have proposed improvements and extensions to this algorithm. In [8], new update algorithms for learning mixture models were presented. They also proposed a method for detecting moving shadows using an existing mixture model. In [9], not only the parameters but also the number of components of the mixture is constantly adapted for each pixel. In [10], the mixture of Gaussians model was combined with concepts defined by region level and frame level considerations.

The Gaussian assumption for the pixel intensity distribution does not always hold. To deal with the limitations of parametric methods, a non-parametric approach to background modeling was proposed in [11]. The proposed method utilizes a general non-parametric kernel density estimation technique for building a statistical representation of the scene background. The probability density function for pixel intensity is estimated directly from the data without any assumptions about the underlying distributions. In [12], a quantization/clustering technique to construct a non-parametric background model was presented. The background is encoded on a pixel by pixel basis and samples at each pixel are clustered into the set of codewords.

Some presented background models consider the time aspect of a video sequence. The decision depends also on the previous pixel values from the sequence. In [13], [14], an autoregressive process was used to model the pixel value distribution over time. In [15], a Hidden Markov Model (HMM) approach was adopted.

Some authors have modeled the background using edge features. In [16], the background model was constructed from the first video frame of the sequence by dividing it into equally sized blocks and calculating an edge histogram for each block. The histograms were constructed using pixel-specific edge directions as bin indices and incrementing the bins with the corresponding edge magnitudes. In [17], a fusion of edge and intensity information was used.

Motion-based approaches have also been proposed for background subtraction. The algorithm presented in [18] detects salient motion by integrating frame-to-frame optical flow over time. Salient motion is assumed to be motion that tends to move in a consistent direction over time. The saliency measure used is directly related to the distance over which a point has traveled with a consistent direction.

Region-based algorithms usually divide an image into blocks and calculate block-specific features. Change detection is achieved via block matching. In [19], the block correlation is measured using the NVD (Normalized Vector Distance) measure. In [16], an edge histogram calculated over the block area is used as a feature vector describing the block.

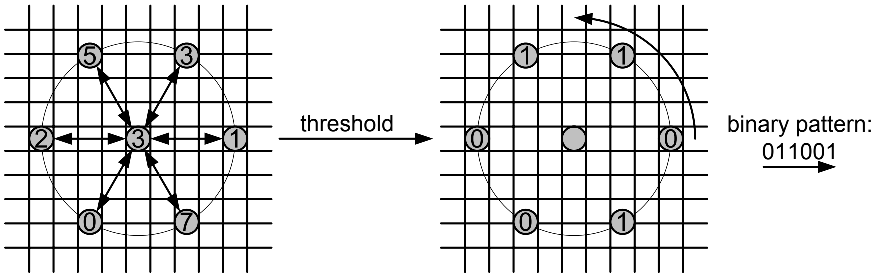


Fig. 1. Calculating the binary pattern.

III. TEXTURE DESCRIPTION WITH LOCAL BINARY PATTERNS

LBP is a gray-scale invariant texture primitive statistic. It is a powerful means of texture description. The operator labels the pixels of an image region by thresholding the neighborhood of each pixel with the center value and considering the result as a binary number (binary pattern). The basic version of the LBP operator considers only the eight neighbors of a pixel [2], but the definition can be easily extended to include all circular neighborhoods with any number of pixels [3]:

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c)2^p, \qquad s(x) = \begin{cases} 1 & : \quad x \geq 0 \\ 0 & : \quad x < 0 \end{cases}, \tag{1}$$

where g_c corresponds to the gray value of the center pixel (x_c, y_c) of a local neighborhood and g_p to the gray values of P equally spaced pixels on a circle of radius R. By extending the neighborhood one can collect larger-scale texture primitives. The values of neighbors that do not fall exactly on pixels are estimated by bilinear interpolation. In practice, Eq. (1) means that the signs of the differences in a neighborhood are interpreted as a P-bit binary number, resulting in 2^P distinct values for the binary pattern. The 2^P-bin histogram of the binary patterns computed over a region is used for texture description. See Fig. 1 for an illustration of the LBP operator.

LBP has several properties that favor its usage in background modeling. Due to the invariance of the LBP features with respect to monotonic gray-scale changes, our method can tolerate the considerable gray-scale variations common in natural images and no normalization of input images is needed. Unlike many other approaches, the proposed features are very fast to compute, which is an important property from the practical implementation point of view. LBP is a non-parametric method, which means that no assumptions about the underlying distributions are needed. Furthermore, the operator does not require many parameters to be set and has high discriminative power.

A limitation of the LBP is that it does not work very robustly on flat image areas such as sky, where the gray values of the neighboring pixels are very close to the value of the center pixel. This due to the thresholding scheme of the operator. Think of a case where g_p and g_c have the values 29 and 30, respectively. From Eg. (1) we see that in this case $s(x)$ outputs a value of 0. If the values were 30 and 30, $s(x)$ would return with 1. In order to make the LBP more robust against these negligible changes in pixel values, we propose to modify the thresholding scheme of the operator by replacing the term $s(g_p - g_c)$ in Eq. (1) with the term $s(g_p - g_c + a)$. The bigger the value of $|a|$ is, the bigger changes in pixel values are allowed without affecting the thresholding results. In order to retain the discriminative power of the LBP operator, a relatively small value for a should be used. In our experiments a was given a value of 3. The results presented in [1] show that good results can also be achieved by using the original LBP. With the modified version, our background subtraction method consistently behaves more robustly and thus should be preferred over the original one.

IV. A Texture-Based Approach

In this section, we introduce our approach to background subtraction. The algorithm can be divided into two phases, *background modeling* and *foreground detection*, described in Sections IV-A and IV-B. In Sec. IV-C, some guidelines for how to select the parameter values are given.

A. Background Modeling

Background modeling is the most important part of any background subtraction algorithm. The goal is to construct and maintain a statistical representation of the scene that the camera sees. As in most earlier studies, the camera is assumed to be non-moving. We model each pixel of the background identically, which allows a high speed parallel implementation if needed. We chose to utilize texture information when modeling the background and the LBP was selected as the measure of texture because of its good properties. In the following, we explain the background model update procedure for one pixel, but the procedure is identical for each pixel.

We consider the feature vectors of a particular pixel over time as a pixel process. The LBP histogram computed over a circular region of radius R_{region} around the pixel is used as the feature vector. The radius R_{region} is a user-settable parameter. The background model for the pixel consists of a group of adaptive LBP histograms, $\{\vec{m}_0, ..., \vec{m}_{K-1}\}$, where K is selected by the user. Each model histogram has a weight between 0 and 1 so that the weights of the K model histograms sum up to one. The weight of the k^{th} model histogram is denoted by ω_k.

Let us denote the LBP histogram of the given pixel computed from the new video frame by $\vec{h}$. At the first stage of processing, $\vec{h}$ is compared to the current K model histograms using a proximity measure. We chose to use the histogram intersection as the measure in our experiments:

$$\cap(\vec{a}, \vec{b}) = \sum_{n=0}^{N-1} \min(a_n, b_n), \tag{2}$$

where $\vec{a}$ and $\vec{b}$ are the histograms and N is the number of histogram bins. This measure has an intuitive motivation in that it calculates the common part of two histograms. Its advantage is that it explicitly neglects features which only occur in one of the histograms. The complexity is very low as it requires very simple operations only. The complexity is linear for the number of histogram bins: $\mathcal{O}(N)$. Notice that it is also possible to use other measures such as $\mathcal{X}^2$. The threshold for the proximity measure, T_P, is a user-settable parameter.

If the proximity is below the threshold T_P for all model histograms, the model histogram with the lowest weight is replaced with $\vec{h}$. The new histogram is given a low initial weight. In our experiments, a value of 0.01 was used. No further processing is required in this case.

More processing is required if matches were found. We select the best match as the model histogram with the highest proximity value. The best matching model histogram is adapted with the new data by updating its bins as follows:

$$\vec{m}_k = \alpha_b \vec{h} + (1 - \alpha_b)\vec{m}_k, \qquad \alpha_b \in [0, 1], \tag{3}$$

where α_b is a user-settable learning rate. Furthermore, the weights of the model histograms are updated:

$$\omega_k = \alpha_w M_k + (1 - \alpha_w)\omega_k, \qquad \alpha_w \in [0, 1], \tag{4}$$

where α_w is another user-settable learning rate and M_k is 1 for the best matching histogram and 0 for the others. The adaptation speed of the background model is controlled by the learning rate parameters α_b and α_w. The bigger the learning rate the faster the adaptation.

All of the model histograms are not necessarily produced by the background processes. We can use the persistence of the histogram in the model to decide whether the histogram models the background or not. It can be seen from Eq. (4) that the persistence is directly related to the histogram's weight: the bigger the weight, the higher the probability of being a background histogram. As a last stage of the updating procedure, we sort the model histograms in decreasing order according to their weights, and select the first B histograms as the background histograms:

$$\omega_0 + ... + \omega_{B-1} > T_B, \qquad T_B \in [0, 1], \tag{5}$$

where T_B is a user-settable threshold.

B. Foreground Detection

Foreground detection is done before updating the background model. The histogram $\vec{h}$ is compared against the current B background histograms using the same proximity measure as in the update algorithm. If the proximity is higher than the threshold T_P for at least one background histogram, the pixel is classified as background. Otherwise, the pixel is marked as foreground.

C. Selection of Parameter Values

Because of the huge amount of different combinations, finding a good set of parameter values must be done more or less empirically. In this section, some guidelines for how to select values for the parameters of the method are given.

The radius R_{region} defines the region for histogram calculation. A small value makes the information encoded in the histogram more local. If the shape information of moving objects is of interest, smaller region size should be used. Choosing an LBP operator with a large value of P makes the histogram long and thus computing the proximity becomes slow. Using a small number of neighbors makes the histogram shorter but also means losing more information. Furthermore, the memory consumption of the method depends on the choice of P. Since correlation between pixels decreases with distance, much of the textural information can be obtained from local neighborhoods. Thus the radius R of the LBP operator is usually kept small. The proximity threshold T_P for histogram comparison is easy to find by experimenting with different values. Values between 0.6 and 0.7 gave good results for all of our test sequences.

A proper value for the parameter K depends on the scene being modeled. In case of a unimodal scene, a small value for K is sufficient. For multimodal scenes, more histograms are needed to learn the background. According to the experiments, in most cases, a good value for K lies between 2 and 5. It should be noticed that the bigger the value of K, the slower the processing, and the greater the memory requirements. The parameter T_B is closely related to K. It is used to select the background histograms as described in Sec. IV-A. A small value is sufficient for unimodal scenes, whereas a larger value is required in the multimodal case.

The proposed method adapts itself to the changes of the scene by continuously updating the background model. The adaptation speed is controlled by two learning rate parameters α_b and α_w, shown in Equations (3) and (4), respectively. The bigger the parameter values, the more weight is given to recent observations, and the faster the adaptation. According to the experiments, in most of the cases, the best results are achieved with small values for these parameters.

V. Experiments

To the authors' knowledge, there is still a lack of globally accepted test sets for the extensive evaluation of background subtraction algorithms. Due to this fact, most authors have used their own test sequences. This makes the comparison of the different approaches rather difficult. In [13], an attempt to address this problem was made. In most cases, the comparison of different approaches is done based on visual interpretation, i.e., by looking at processed images provided by the algorithms. Numerical evaluation is usually done in terms of the number of *false negatives* (the number of foreground pixels that were missed) and *false positives* (the number of background pixels that were marked as foreground). The ground truth is achieved by manually labeling some frames from the video sequence.

In [1], we used both visual and numerical methods to evaluate our approach. We compared our method to four other methods presented in the literature. The test sequences included both indoor and outdoor scenes. According to the test results, the overall performance of our method was better than the performance of the comparison methods for the test sequences used. The method used block-wise processing. In certain applications, when accurate shape information is needed, this kind of approach cannot be used. Making the decision pixel-wise usually offers more accurate results for shape extraction. In Sec. IV, we extended our algorithm for pixel-wise processing and all the tests presented in this article are carried out using this method.

Fig. 2 shows some results for our method using some of the test sequences from [1]. The first two frames on the upper left are from an indoor sequence where a person is walking in a laboratory room. Background subtraction methods that rely only on color information will most

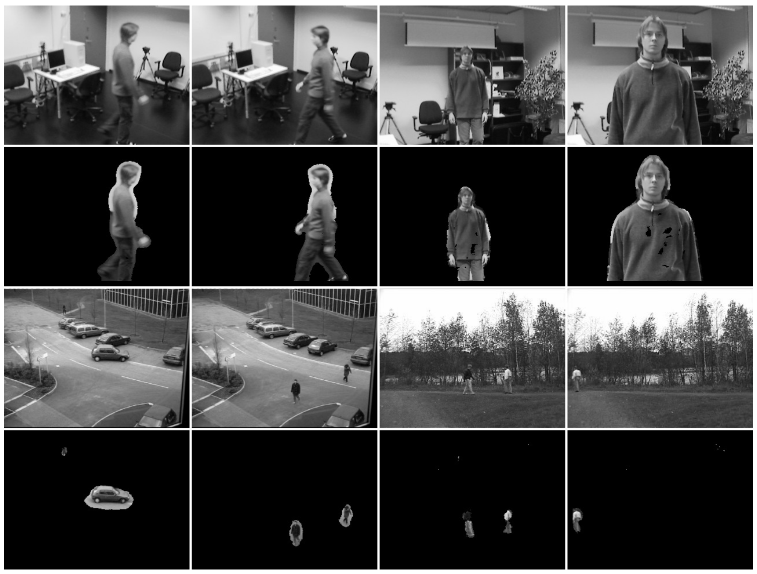


Fig. 2. Some detection results of our method. The first row contains the original video frames. The second row contains the corresponding processed frames. The image resolution is 320×240 pixels.

probably fail to detect the moving object correctly because of the similar color of the foreground and the background. The next two frames are from an indoor sequence where a person walks towards the camera. Many adaptive pixel-based methods output a huge amount of false negatives on the inner areas of the moving object, because the pixel values stay almost the same over time. The proposed method gives good results, because it exploits information gathered over a larger area than a single pixel. The first two frames on the lower left are from an outdoor sequence which contains relatively small moving objects. The original sequence has been taken from the PETS database (*ftp://pets.rdg.ac.uk/*). The proposed method successfully handles this situation and all the moving objects are detected correctly. The last two frames are from an outdoor sequence that contains heavily swaying trees and rippling water. This is a very difficult scene from the background modeling point of view. Since the method was designed to handle also multimodal backgrounds, it manages the situation relatively well. The values for the method parameters are given in Table I. For the first three sequences, the values were kept untouched. For the last sequence, values for the parameters K and T_B were changed to adjust the method for increased multimodality of the background.

TABLE I

THE PARAMETER VALUES OF THE METHOD FOR THE RESULTS IN FIGURES 2 AND 3.

Fig.	Sequence(s)	$LBP_{P,R}$	R_{region}	K	T_B	T_P	α_b	α_w
2	1-3	$LBP_{6,2}$	9	3	0.4	0.65	0.01	0.01
2	4	$LBP_{6,2}$	9	5	0.8	0.65	0.01	0.01
3	1-7	$LBP_{6,2}$	9	4	0.8	0.70	0.01	0.01

In [13], a test set for evaluating background subtraction methods was presented. It consists of seven video sequences, each addressing a specific canonical background subtraction problem. In the same paper, ten different methods were compared using the test set. We tested our method against this test set and achieved the results shown in Fig. 3. When compared to the results in [13], the overall performance of our method seems to be better than that of the other methods. We did not change the parameter values of our method between the test sequences, although better results could be obtained by customizing the values for each sequence. See Table I for the parameter values used. Like most of the other methods, our method was not capable of handling the *Light Switch* problem. This is because we do not utilize any higher level processing that could be used to detect sudden changes in background.

We also compared the performance of our method to the widely used method of Stauffer and Grimson [7] by using the five test sequences presented in [1]. The sequences include both indoor and outdoor scenes and five frames from the each sequence are labeled as the ground truth. The results are shown in Fig. 4. The numbers of error classifications were achieved by summing the errors from the five processed frames corresponding to the ground truth frames. For all five sequences, our method gave less false negatives than the comparison method. In the case of false positives, our method was better in two cases. For the rest of the three sequences, the difference is very small. It should be noticed that for the proposed method, most of the false positives occur on the contour areas of the moving objects (see Fig. 2). This is because the features are extracted from the pixel neighborhood. According to the overall results, the proposed method outperforms the comparison method for the used test sequences.

Since our method has relatively many parameters, there naturally arises a question: how easy

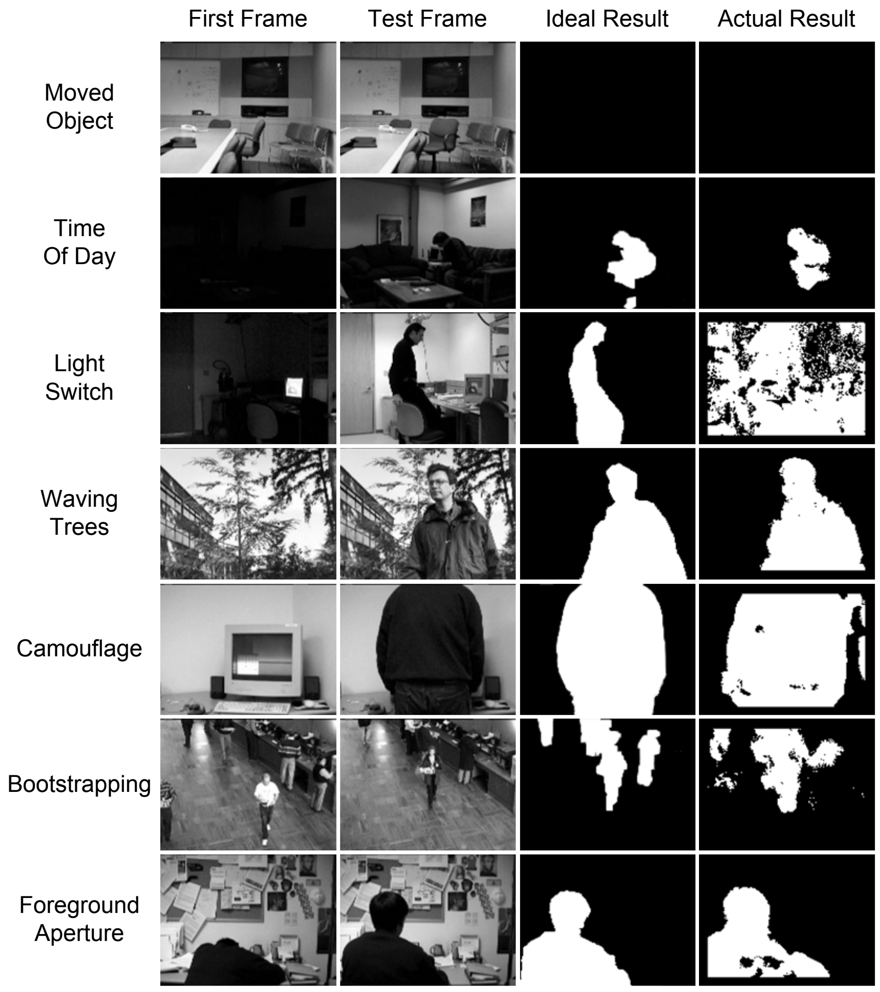
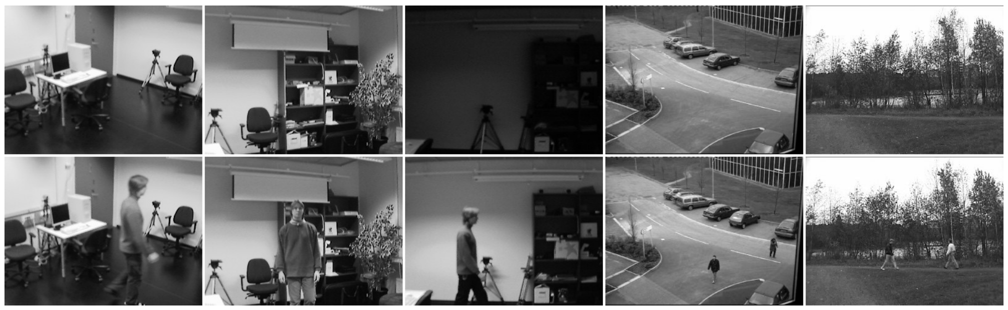


Fig. 3. Detection results of our method for the test sequences presented in [13]. The image resolution is 160×120 pixels.

or difficult is to obtain a good set of parameter values? To see how sensitive the proposed method is to small changes of its parameter values, we calculated the error classifications for different parameter settings. Because of a huge amount of different combinations, only one parameter was varied at a time. The measurements were made for several video sequences, including indoor and outdoor scenes. The results for the first sequence of Fig. 2 are plotted in Fig. 5. It can be clearly seen that, for all parameters, a good value can be chosen across a wide range of values. The same observation was made for all the measured sequences. This property significantly eases the selection of parameter values. Furthermore, the experiments have shown that a good set of



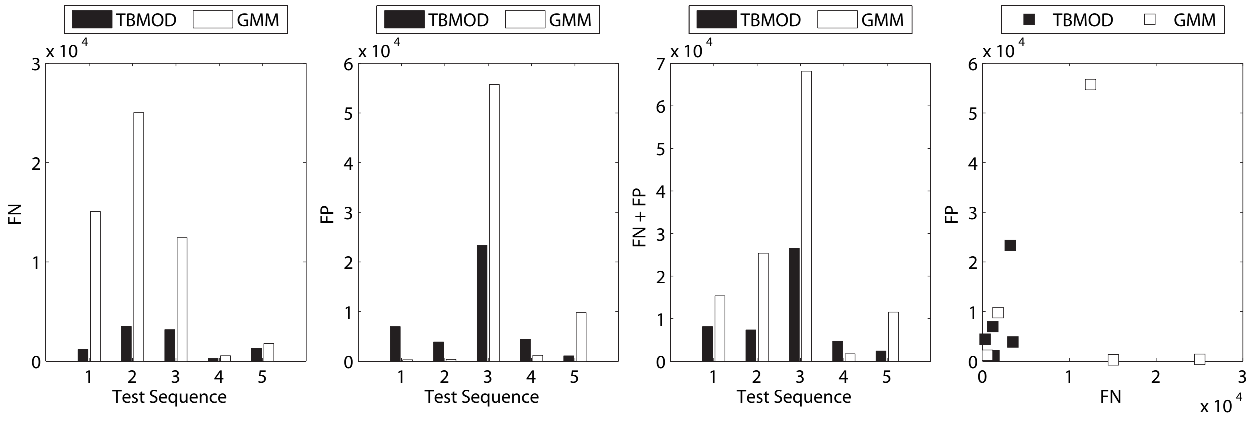


Fig. 4. Comparison results of the method presented in this article (TBMOD) and the method of Stauffer and Grimson [7] (GMM). Five different test sequences were used. The first image row corresponds to the first frames of the sequences. The second row is for frames that contain moving object(s). FN and FP stand for false negatives and false positives, respectively.

parameters for a sequence usually performs well also for other sequences (see Table I).

We also measured the speed of the proposed method. For the parameter values used in the tests, a frame rate of 15 fps was achieved. We used a standard PC with a 1.8 GHz processor and 512 MB of memory in our experiments. The image resolution was 160×120 pixels. This makes the method well suited to systems that require real-time processing.

VI. Conclusions

A novel approach to background subtraction was presented, in which the background is modeled using texture features. The features are extracted by using the modified local binary pattern (LBP) operator. Our approach provides us with several advantages compared to other methods. Due to the invariance of the LBP features with respect to monotonic gray-scale changes, our method can tolerate considerable illumination variations common in natural scenes. Unlike

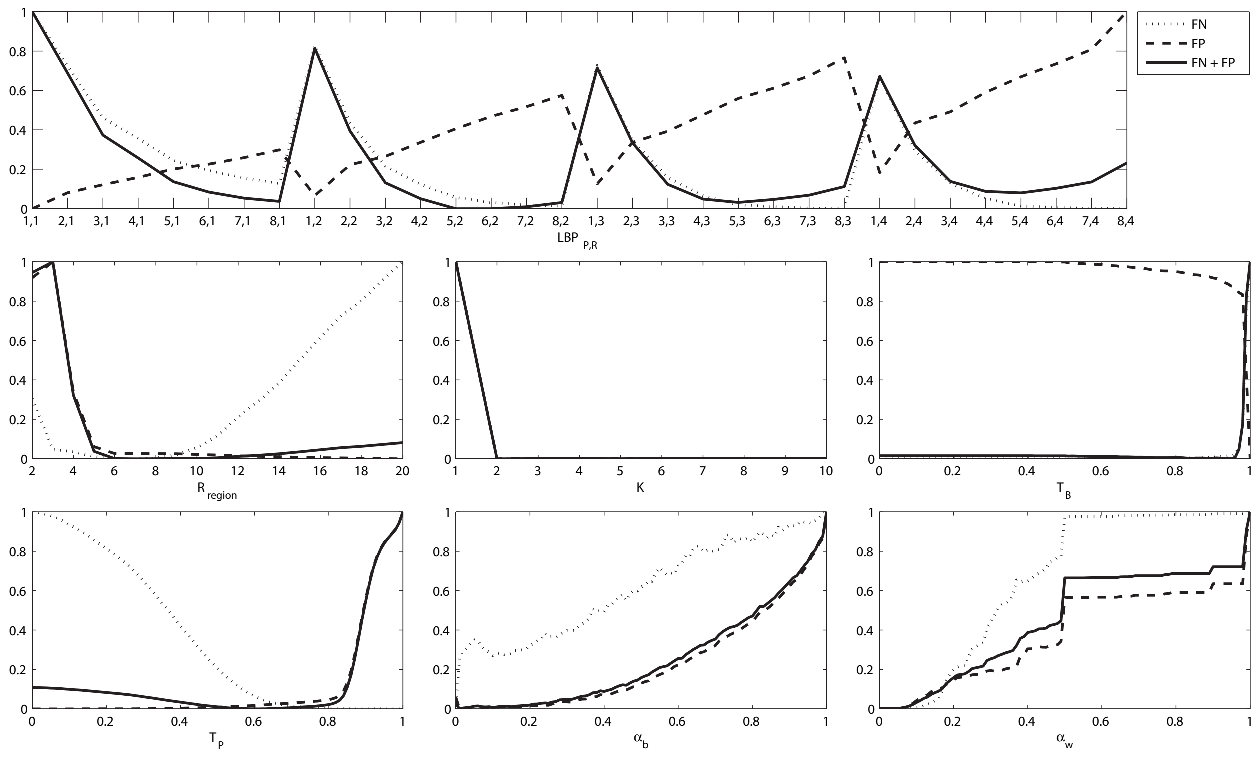


Fig. 5. Number of false negatives (FN) and false positives (FP) for different parameter values for the first sequence of Fig. 2. While one parameter was varied, other parameters were kept fixed at the values given in Table I. The results are normalized between zero and one: $\vec{x} = (\vec{x} - min(\vec{x}))/max(\vec{x} - min(\vec{x}))$.

many other approaches, the proposed features are very fast to compute, which is an important property from the practical implementation point of view. The proposed method belongs to non-parametric methods, which means that no assumptions about the underlying distributions are needed.

Our method has been evaluated against several video sequences including both indoor and outdoor scenes. It has proven to be tolerant to illumination variations, the multimodality of the background, and the introduction/removal of background objects. Furthermore the method is capable of real-time processing. Comparisons to other approaches presented in the literature have shown that our approach is very powerful when compared to the state-of-the-art.

Currently, the method requires a non-moving camera, which restricts its usage in certain applications. We plan to extend the method to support also moving cameras. The preliminary results with a pan-tilt-zoom camera are promising. The proposed method also has relatively many parameters. This could be a weakness, but at the same time, it allows the user extensive control over method behavior. A proper set of parameters can be easily found for a given application scenario.

Acknowledgment

This work was supported by the Academy of Finland. The authors also want to thank Prof. Janne Heikkilä for his contribution.

References

[1] M. Heikkilä, M. Pietikäinen, and J. Heikkilä, “A Texture-Based Method for Detecting Moving Objects,” *Proc. British Machine Vision Conf.*, vol. 1, pp. 187–196, 2004.

[2] T. Ojala, M. Pietikäinen, and D. Harwood, “A Comparative Study of Texture Measures with Classification Based on Feature Distributions,” *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[3] T. Ojala, M. Pietikäinen, and T. Mäenpää, “Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[4] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, “Detecting Moving Shadows: Algorithms and Evaluation,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 7, pp. 918–923, 2003.

[5] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, “Pfinder: Real-Time Tracking of the Human Body,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[6] N. Friedman and S. Russell, “Image Segmentation in Video Sequences: A Probabilistic Approach,” *Proc. Conf. Uncertainty in Artificial Intelligence*, pp. 175–181, 1997.

[7] C. Stauffer and W. E. L. Grimson, “Adaptive Background Mixture Models for Real-Time Tracking,” *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 246–252, 1999.

[8] P. KaewTraKulPong and R. Bowden, “An Improved Adaptive Background Mixture Model for Real-Time Tracking with Shadow Detection,” *Proc. European Workshop on Advanced Video Based Surveillance Systems*, 2001.

[9] Z. Zivkovic, “Improved Adaptive Gaussian Mixture Model for Background Subtraction,” *Proc. Int’l Conf. Pattern Recognition*, vol. 2, pp. 28–31, 2004.

[10] Q. Zang and R. Klette, “Robust Background Subtraction and Maintenance,” *Proc. Int’l Conf. Pattern Recognition*, vol. 2, pp. 90–93, 2004.

[11] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, “Background and Foreground Modeling Using Nonparametric Kernel Density Estimation for Visual Surveillance,” *Proc. IEEE*, vol. 90, no. 7, pp. 1151–1163, 2002.

[12] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, “Background Modeling and Subtraction by Codebook Construction,” *Proc. IEEE Int’l Conf. Image Processing*, vol. 5, pp. 3061–3064, 2004.

[13] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, “Wallflower: Principles and Practice of Background Maintenance,” *Proc. IEEE Int’l Conf. Computer Vision*, vol. 1, pp. 255–261, 1999.

[14] A. Monnet, A. Mittal, N. Paragios, and R. Visvanathan, “Background Modeling and Subtraction of Dynamic Scenes,” *Proc. IEEE Int’l Conf. Computer Vision*, vol. 2, pp. 1305–1312, 2003.

[15] J. Kato, T. Watanabe, S. Joga, J. Rittscher, and A. Blake, “An HMM-Based Segmentation Method for Traffic Monitoring Movies,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1291–1296, 2002.

[16] M. Mason and Z. Duric, “Using Histograms to Detect and Track Objects in Color Video,” *Proc. Applied Imagery Pattern Recognition Workshop*, pp. 154–159, 2001.

[17] S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld, “Detection and Location of People in Video Images Using Adaptive Fusion of Color and Edge Information,” *Proc. Int’l Conf. Pattern Recognition*, vol. 4, pp. 627–630, 2000.

[18] L. Wixson, “Detecting Salient Motion by Accumulating Directionally-Consistent Flow,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 774–780, 2000.

[19] T. Matsuyama, T. Ohya, and H. Habe, “Background Subtraction for Non-Stationary Scenes,” *Proc. Asian Conf. Computer Vision*, pp. 622–667, 2000.