Change Detection in Feature Space using Local Binary Similarity Patterns

Guillaume-Alexandre Bilodeau, Jean-Philippe Jodoin
LITIV lab., Dept. of computer and software engineering
École Polytechnique de Montréal
Montréal, Canada
gabilodeau@polymtl.ca, jean-philippe.jodoin@polymtl.ca

Nicolas Saunier
Dept. of civil, geological and mining engineering
École Polytechnique de Montréal
Montréal, Canada
nicolas.saunier@polymtl.ca

***Abstract*—In general, the problem of change detection is studied in color space. Most proposed methods aim at dynamically finding the best color thresholds to detect moving objects against a background model. Background models are often complex to handle noise affecting pixels. Because the pixels are considered individually, some changes cannot be detected because it involves groups of pixels and some individual pixels may have the same appearance as the background. To solve this problem, we propose to formulate the problem of background subtraction in feature space. Instead of comparing the color of pixels in the current image with colors in a background model, features in the current image are compared with features in the background model. The use of a feature at each pixel position allows accounting for change affecting groups of pixels, and at the same time adds robustness to local perturbations. With the advent of binary feature descriptors such as BRISK or FREAK, it is now possible to use features in various applications at low computational cost. We thus propose to perform background subtraction with a small binary descriptor that we named Local Binary Similarity Patterns (LBSP). We show that this descriptor outperforms color, and that a simple background subtractor using LBSP outperforms many sophisticated state of the art methods in baseline scenarios.**

***Keywords*-Change detection; Local binary patterns; Local binary descriptor; Background subtraction**

I. INTRODUCTION

Over the years, a large body of works has been published on the topic of background subtraction. Research is still ongoing since background subtraction offers complementary capabilities over object detection methods. Contrarily to object detection for which a detector (e.g. HOG-based [1], PCA-HOG-based [2]) has to be trained for known objects, background subtraction can detect unknown or unexpected objects as long as their appearance contrast with a background model. The difficulty in background subtraction is to determine if an object contrasts sufficiently with the background. Most proposed methods aim at dynamically finding the best color thresholds at individual pixel positions. To do so, complex methods are used to model the background to account for change in illumination, camera noise, and periodic motion. Nevertheless, because the pixels are considered individually, some changes cannot be detected because it involves groups of pixels and some individual pixels in these groups may keep the same appearance as the background. The body area in Fig.1(a), which has intensities similar to the background, illustrates this case. Euclidean distance between intensities cannot detect a large amount of the body (Fig.1(c)).

To solve this problem, we propose to formulate the problem of background subtraction in feature space. Instead of comparing the color of pixels in the current image with colors in a background model, features in the current image are compared with features in the background model. The use of a feature at each pixel position allows accounting for change affecting groups of pixels and at the same time adds robustness to local perturbations (see Fig.1(d)). With the advent of binary feature descriptors such as BRISK [3], D-Brief [4], or FREAK [5], it is now possible to use features in various applications at low computational cost. Unfortunately, because these binary feature descriptors are designed for matching tasks, they are computed over relatively large image regions. They cannot be used without modification for tasks such as background subtraction because they are not local enough.

We propose a new binary feature descriptor suitable for background subtraction, named LBSP (Local Binary Similarity Patterns), that describes a 5×5 region in way that is related to Local Self-Similarity [6]. We tested our binary feature descriptor on a subset of the dataset of CDnet [7]. We show that LBSP gives better performance than pixel color, and with just simple learning of stable binary codes, it outperforms several state of the art methods.

In summary, the main contributions of this paper are:

1) we reinterpreted state of the art feature descriptors to design a new binary feature descriptor that is suited for tasks such as background subtraction. By its design based on pixel difference, LBSP has the property that it can be calculated both inside a region in an image, or across regions between two images or two regions in the same image to capture differences both in intensity and in texture;
2) we characterized thoroughly our new feature descriptor in contrast to pixel color for background subtraction. We show that LBSP outperforms color in background subtraction tasks;
3) we evaluated a simple background subtractor using

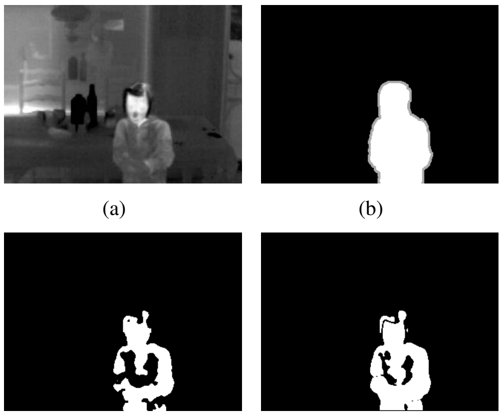

Figure 1. Effect of detecting change based on individual pixels versus on features calculated on a small neighborhood. (a) Frame 837 from the diningRoom video [7], (b) Ground-truth foreground, (c) Change detected based on individual pixels with Euclidean distance [8], and (d) Change detected in feature space using LBSP.

LBSP on a subset of the standard dataset CDnet. We show that thanks to LBSP, this simple background subtractor outperforms many complex background subtractors from the state of the art.

II. Related work

In its most basic form, background subtraction is performed by comparing a new frame with a background frame. To account for noise, the background frame is updated by making a mean of the pixel values over time [8]. Quickly, it was noticed that to account for dynamic backgrounds, more than one color value should be associated to a given background pixel location. Therefore, parametric background models like Gaussian mixture model (GMM) [9] and non-parametric background models like kernel density estimation (KDE) [10] were introduced. Since then, most works focused on improving over GMM or KDE, by for example, improving the update speed and adding robustness to shadows [11]. In other works like Vibe+ [12], [13] and PBAS [14], instead of building the probability distribution of the background for a pixel using a Parzen window, the researchers select background samples at random, and if the pixel in the new frame matches a certain proportion of background samples, it is assumed to be background and new samples may be selected for update. While Vibe+ uses many heuristics to improve the detected foreground (morphological operation, blinking pixels detection, etc.), PBAS learns parameters and thresholds from background dynamics. To integrate stopping objects the background Vibe+ and PBAS propagate background updates to neighboring pixels, even the ones marked as foreground. Other methods like SC-SOBS [15], [16] learns the background in a neural network framework, and the background update at each pixel is influenced by the labeling decision at neighboring pixels. As it can be noticed, most recent methods tend to account for neighboring pixels to add robustness to noise affecting individual pixels. Therefore, methods like RECTGAUSS-Tex [17], [18] perform background subtraction on image blocks before applying GMM. Foreground is detected only if there is enough motion in each block to modify its color histogram. For a more detailed discussion of these techniques, readers can refer to recent surveys [8], [19], [20].

Some methods attempted to perform background subtraction in feature space. Notably, local binary patterns (LBP) were used by Heikkila *et al* [21], and census-like patterns were used by Nonaka *et al* [22]. These methods performed reasonably well, but the features that are used can only capture change in texture, not change in intensity. Furthermore, because these features are computed based on comparisons with a center pixel (a bit s(x) is the result of a comparison of neighboring intensities with the intensity of the center pixel, see eq.1), change cannot be detected if the intensity of the center pixel I_c remains larger (or smaller) than each neighboring pixel I_x after a change in a scene. In eq.1, a is used to add robustness for noise affecting the center pixel.

$$s(x) = \begin{cases} 1 & I_x \geq I_c + a \\ 0 & I_x < I_c + a \end{cases} \tag{1}$$

Recently, Jodoin *et al* [23] proposed to use the local self-similarity descriptor for background subtraction. This work has some similarity with ours, but because it uses the LSS descriptor which is calculated on a large region, there are borders of falsely detected pixels around the detected foreground. Some morphological operations were performed to improve the detected foreground by removing extra pixels, but holes or spaces between legs cannot be recovered that way. Furthermore, LSS is slow to compute on complete image. To solve the shortcomings of both LBP and LSS, we propose a new feature descriptor that is binary and fast to compute, works on small regions, and captures both change in texture and change in intensity.

III. LBSP

A. Motivation

Binary feature descriptors are fast and very discriminative. Most of them are based on evaluating many comparisons between the intensity of pairs of pixels in different configurations. They are robust to noise because they include a very large number of binary comparisons. They can also discriminate texture and intensity difference because they are typically used on distinctive points that are not localized on regions with uniform color. This cannot be guaranteed in background subtraction tasks where the features are calculated densely over the entire image but still have to be discriminative at every location.

To solve this problem and to capture change both in texture and in intensity, the feature should be calculated

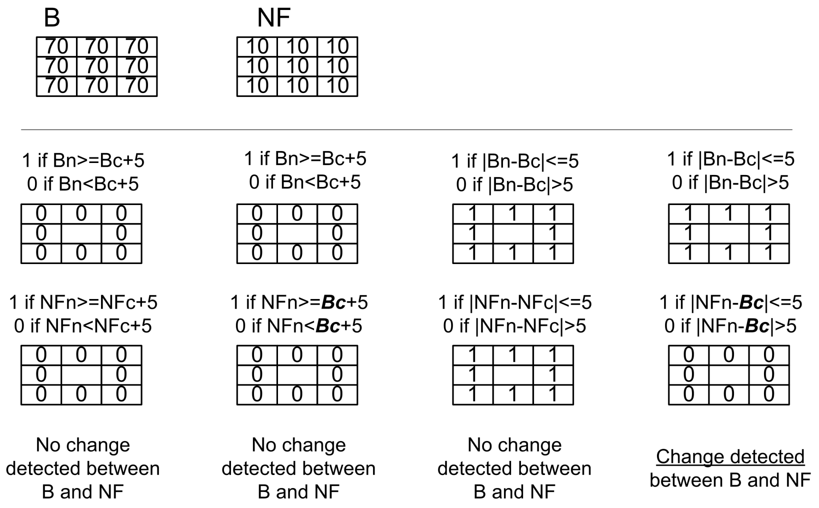

Figure 2. Results of four different binary operation on image region B and image region NF. Bc and NFc stand for the center pixel of the regions B and NF respectively. Bn and NFn stand for the pixels around the center of the regions B and NF respectively. In the bottom part of the figure, the first and second columns use comparisons to generate binary representations of the regions using in one case, intra-region operations, and in the other case both intra-region and inter-region operations. The third and fourth columns use thresholding on the absolute difference to generate binary representations using in one case intra-region operations, and in the other case, both intra-region and inter-region operations. In all case, a threshold of 5 is used.

inside the background image for creating the background model, and between the background and a new frame for foreground detection. The region to calculate the feature should be small, and most importantly, the feature should not be based strictly on comparisons ($<, >, \geq, \leq$). This is illustrated in Fig.2 on the problematic case of a large change of intensity. The case of different textures is not illustrated in the figure because it is not an issue with any binary feature descriptors. The problem with using comparisons arises from the mandatory use of a threshold a to account for noise affecting the center pixel, particularly for uniform regions (see eq.1). As illustrated in the figure (first two columns), considering only comparisons ($<, >, \geq, \leq$) does not allow to distinguish two uniform regions. In the example provided, using inter-region comparisons would solve this problem only for changes of intensities toward smaller values, not toward larger values because they are equivalent as adding a larger value of a for a $<$ operation. Note that using inter-region FREAK-like patterns of binary comparisons does not alleviate this problem. It is not specific to the use of a central pixel. Using absolute difference on the other hand allows detecting both changes in intensity, that is changes toward smaller or larger values (Fig.2, last two columns). Therefore, the feature that we propose is based on binary thresholding on absolute difference operations applied in a way similar to LSS [6], but over a small area and calculated inside one image (intra-LBSP) and between two images (inter-LBSP) to capture both texture and intensity changes.

B. Intra-image and Inter-image LBSP

The principle of LBSP is to compare a center pixel with neighboring pixels and check for similarity. In that sense, LBSP is based on the same principle as LSS [6], but it is different in two ways: (1) instead of comparing a center patch with neighboring patches, the comparisons are done for individual pixels, (2) instead of calculating a sum of squared differences between patches which results in a vector of integer values, the differences between pixels are thresholded to give a binary code.

LBSP is computed on a $n \times n$ region R, and the neighboring pixels to compare with the center pixel may be all the pixels or a subset of P pixels in R. LBSP is defined as

$$LBSP_R(x_c, y_c) = \sum_{p=0}^{P-1} d(i_p - i_c) 2^p \tag{2}$$

with

$$d(x) = \begin{cases} 1 & |x| \leq T_d \\ 0 & |x| > T_d \end{cases} \tag{3}$$

where i_c corresponds to the intensity of the center pixel (x_c, y_c) of R and i_p to the intensity of the pth pixel in a set of P pixels neighboring (x_c, y_c) in R. T_d is a similarity threshold. The size of R and the number and the pattern for the P pixels to code may be selected based on the application. LBSP can be computed intra-region and inter-region by selecting the center pixel to be in the same region as the neighboring pixels, or by selecting the center pixel to be in another region (elsewhere in the image, or in another image).

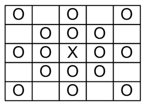

Figure 3. Pattern used to calculate LBSP. X: Center pixel, O: Neighbors used for computing binary code.

IV. Simple background subtraction with LBSP

Recall that the goal of our work is to show that a spatially compact binary feature descriptor outperforms intensity in change detection tasks, like background subtraction. To study the benefit of LBSP in background subtraction, we have designed a very simple background subtractor, such that we can focus on the impact of using LBSP instead of pixel color. Our background subtractor has no update process. It just compares the current frame with a background model constructed in the F first frames. Initially, the first frame is modeled using intra-region LBSP at all pixel locations (i_p and i_c are selected in the same frame). LBSP is computed on a 5×5 region using the pattern of Fig.3. This pattern is inspired from other descriptors like FREAK [5], or DAISY [24], where the sampling is denser closer to the pattern center. This pattern gives 16-bit codes. LBSP is computed on the R, G, and B channels which gives a final 48-bit code. Within the first F frame, a binary code is labeled as background if it is repeated a minimum of B times consecutively. A background image M is also updated with the intensities corresponding to the background binary codes. After F, the background model does not change. It is an array of binary feature descriptors $LBSP_M$ computed intra-frame. Then, for foreground detection, the current image is modeled using inter-frame LBSP (i_p is selected in frame f and i_c is selected in the background image M). This gives an array of binary feature descriptors $LBSP_f$. If there is no change for a given pixel position (x, y), the inter-frame LBSP, $LBSP_f(x, y)$, will be the same as the intra-frame LBSP, $LBSP_M(x, y)$. If not, there is either a change in intensity or a change in local texture. Labeling decision (foreground: fg, background: bg) are taken using

$$L_f(x,y) = \begin{cases} bg & H(LBSP_f(x,y), LBSP_M(x,y) \leq T_h \\ fg & H(LBSP_f(x,y), LBSP_M(x,y) > T_h \end{cases} \tag{4}$$

where $H()$ is the Hamming distance between two descriptors and T_h is a similarity threshold. The selection of the thresholds is discussed in detail the experiments.

V. Performance evalution

To evaluate the use of LBSP in background subtraction, we used the CDnet dataset [7]. Because our simple method does not update the background model, we have tested our background subtraction only on the baseline and thermal subset of videos. We have used exactly the same metrics as used in [7]. Let TP = number of true positives, TN = number of true negatives, FN = number of false negatives, and FP = number of false positives. The 7 metrics used are:

1) Recall (Re): $TP/(TP + FN)$
2) Specificity (Sp): $TN/(TN + FP)$
3) False Positive Rate (FPR): $FP/(FP + TN)$
4) False Negative Rate (FNR): $FN/(TN + FP)$
5) Percentage of Wrong Classifications (PWC): $100(FN + FP)/(TP + FN + FP + TN)$
6) Precision (Pr): $TP/(TP + FP)$
7) $F - measure$: $2 * Pr * Re/(Pr + Re)$

To compute the metrics, we used the tools provided by the authors of [7] available online (http://www.changedetection.net). Unless stated otherwise, the parameters used for our background subtractor are $T_d = 30$, $T_h = 28$, $B = 30$, and $F = 200$. We first compare the performance of LBSP with color. Then, we investigate the roles of T_d and T_h, and finally, we compare the performance of the LBSP background subtractor with the state of the art.

A. Comparison with simple color background subtraction

To compare LBSP to color, we programmed a color background subtractor that works exactly as in section IV, but using instead the intensity levels and Euclidean distance to compare the intensities. We plotted the receiver operating characteristic (ROC) curves for the baseline scenario dataset (Fig.4). As expected, LBSP gives better performance than color because it is more robust to noise as it considers neighboring pixels in labeling decisions. Therefore, the FPR is smaller for a given recall. In fact, the change in intensity threshold (T_d) can be set to a larger value when calculating LBSP because texture is an alternative for detecting change. LBSP has two ways to detect change, and therefore it is possible to be more restrictive on intensity change, because often there is also a texture change when there is a real foreground object moving in a scene. Although not perfect, the properties of LBSP allow detecting more change in regions with similar intensity because of change in local textures as shown in Fig.1

B. Parameters analysis

We investigated the effect of the parameters on the ROC of LBSP. We plotted the ROC curves with $T_h = 1(low), 28, 48(high)$ and with $T_d = 3(low), 30, 90(high)$ (Fig. 5). The ROC curves show that T_d has more effect on LBSP performance than T_h. If T_d is low, LBSP will model textures composed of small intensity changes and as such, will be very sensitive to noise. The binary codes will not be stable. On the other hand, if T_d is high, textures composed of small intensity changes will be considered as uniform patches. The binary codes will be robust to noise, but at the expense of detailed texture models. To some extent, with a high value of T_d, comparing intra-LBSP with inter-LBSP

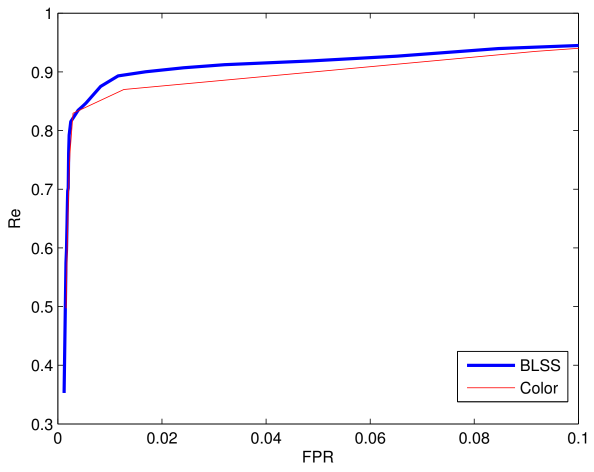

Figure 4. ROC curves of LBSP and color background subtraction on the baseline scenario of CDnet [7].

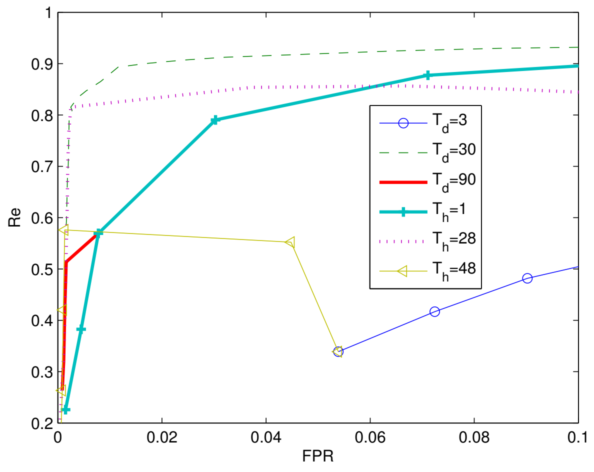

Figure 5. ROC curves of LBSP for various values of fix parameters on the baseline scenario of CDnet [7].

would allow detecting only large change of intensity and very contrasted textures. For T_h, all the following comments apply to the comparison of intra-LBSP with inter-LBSP. If T_h is low, any change in texture or intensity will be detected provided that they are larger than T_d. If T_h is high, only complementary textures and intensity change will be detected. Therefore, the value of T_d is more critical than T_h as any value of T_h gives us background subtraction results. T_d should be set to be higher than noise. Fig. 6 summarizes the effects of T_d and T_h in change detection.

C. Comparison with the state of the art

To complete our evaluation of LBSP in change detection, we compared our method with selected methods tested on

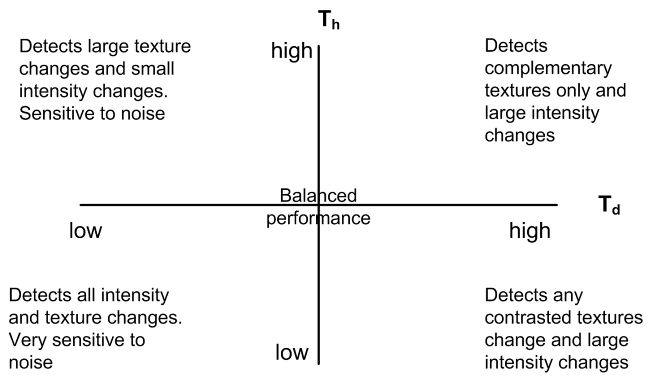
T_h
high
low
T_d
low
high
Balanced performance
Detects large texture changes and small intensity changes. Sensitive to noise
Detects complementary textures only and large intensive changes
Detects all intensity and texture changes. Very sensitive to noise
Detects any contrasted textures change and large intensity changes

Figure 6. Summary of the effect of LBSP parameters in change detection

CDnet and reported in [7]. We selected some of the best methods and some classical methods (see Tables I and II). We ranked the methods in a similar fashion as in [7], the methods are ranked based on their average rank for each metric. We tested on the baseline and thermal subsets (9 videos, 27149 frames) as our simple test method does not include background adaptation. Among the selected methods, our method ranks second on both subsets. It is characterized by a very low FNR and high Pr, better than average PWC, average Re, and average FPR. LBSP outperforms Euclidean distance on color as shown previously, although this time Euclidean distance benefits from a temporal average. Compared to the method using LSS, LBSP performs better because it detects much fewer false positives. The problem with LSS is that the descriptor is too large spatially to measure change at the pixel precision. On the other hand, LBSP is small enough. As discussed by the authors of [7], the pixels near the boundary of moving objects (2 or 3 pixels) are often corrupted by motion blur, and therefore it is very difficult to assign a reliable label. As such, the fact that we use a 5×5 descriptors does not affect significantly the quality of detection near the object boundaries. The KDE method [22] that combines texture detection and color detection separately is not very successful on these two datasets because texture does not help on uniform regions. On those regions, it only relies on individual pixel colors. In our case, we always rely on a group of 16 pixels to improve the quality of detection both on uniform and textured regions.

VI. Conclusion

In this paper, we demonstrated the benefit of performing background subtraction in feature space. We proposed a novel binary feature descriptor, named LBSP, adapted to change detection as it can be computed intra- and inter-regions. We showed that background subtraction with LBSP outperforms background subtraction with pixel colors, and we characterized thoroughly the parameters of LBSP. Furthermore, we compared a simple background subtractor with

Method	Re	Sp	FPR	FNR	PWC	Pr	F-Measure	Mean rank
SC-SOBS [15]	0.9327	0.9980	0.0020	0.0673	**0.3747**	**0.9333**	0.9341	1.88
LBSP	0.8067	0.9977	0.0023	**0.0074**	0.9168	0.9275	0.8623	3.00
PBAS [14]	0.9594	0.9970	0.0030	0.0406	0.4858	0.9242	0.8941	3.13
Vibe+ [12]	0.8283	0.9974	0.0026	0.1717	0.9631	0.8715	0.9262	3.88
Euclidean [8]	0.8385	0.9955	0.0045	0.1615	1.026	0.872	0.9114	4.25
GMM [11]	0.5863	**0.9987**	**0.0013**	0.4137	1.9381	0.7119	**0.9532**	4.38
LSS [23]	**0.9732**	0.9865	0.0135	0.0268	1.3352	0.8494	0.7564	4,88
KDE [22]	0.7472	0.9954	0.0046	0.2528	1.8058	0.7392	0.7998	6.13

Table I
RESULTS ON BASELINE DATASET [7]. BOLDFACE INDICATES THAT THE METHOD RANKS FIRST FOR THIS METRIC. RECALL: Re, SPECIFICITY :Sp, FALSE POSITIVE RATE: FPR, FALSE NEGATIVE RATE: FNR, PERCENTAGE OF WRONG CLASSIFICATIONS: PWC, PRECISION: Pr.

Method	Re	Sp	FPR	FNR	PWC	Pr	F-Measure	Mean rank
PBAS [14]	0.7283	0.9934	0.0066	0.2717	**1.5398**	0.7556	0.8922	2.75
LBSP	0.6535	0.9916	0.0083	**0.0142**	2.0774	**0.7794**	0.6924	3.38
Vibe+ [12]	0.5411	0.9974	0.0026	0.4589	2.8201	0.6646	0.9477	3.38
SC-SOBS [15]	0.6003	0.9957	0.0043	0.3997	1.9841	0.6923	0.8857	3.50
GMM [11]	0.3395	**0.9993**	**0.0007**	0.6605	4.8419	0.4767	**0.9709**	4.25
LSS [23]	**0.9036**	0.9692	0.0308	0.0964	3.2612	0.7297	0.6433	4.38
KDE [22]	0.4147	0.9981	0.0019	0.5853	5.4152	0.4989	0.9164	4.50
Euclidean [8]	0.5111	0.9907	0.0093	0.4889	3.8516	0.6313	0.8877	5.35

Table II
RESULTS ON THERMAL DATASET [7]. BOLDFACE INDICATES THAT THE METHOD RANKS FIRST FOR THIS METRIC. RECALL: Re, SPECIFICITY :Sp, FALSE POSITIVE RATE: FPR, FALSE NEGATIVE RATE: FNR, PERCENTAGE OF WRONG CLASSIFICATIONS: PWC, PRECISION: Pr.

LBSP against more complex state of the art background subtractors and shown that LBSP performs favorably. Future work will be to integrate LBSP inside a more complex background subtractor, and benefit even more from LBSP by using it in matching to compensate for camera jitter.

ACKNOWLEDGMENT

This work is supported by grants and a scholarship from the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005, pp. 886–893.

[2] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1627 –1645, 2010.

[3] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *ICCV*, 2011.

[4] T. Trzcinski and V. Lepetit, "Efficient discriminative projections for compact binary descriptors," in *ECCV*, 2012.

[5] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint." in *CVPR*, 2012, pp. 510–517.

[6] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *CVPR*, June 2007.

[7] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.net: A new change detection benchmark dataset," in *CVPRW*, June 2012, pp. 1 –8.

[8] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Comparative study of background subtraction algorithms," *Journal of Electronic Imaging*, vol. 19, no. 3, pp. 1 – 12, 2010.

[9] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *CVPR*, 1999, pp. 2246–2252.

[10] A. M. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric model for background subtraction," in *ECCV*, 2000, pp. 751–767.

[11] P. Kaewtrakulpong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Workshop on Advanced Video Based Surveillance Systems*, vol. 5308, 2001.

[12] M. Van Droogenbroeck and O. Paquot, "Background subtraction: Experiments and improvements for ViBe," in *CVPRW*, June 2012, pp. 32 –37.

[13] O. Barnich and M. Van Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," *Image Processing, IEEE Transactions on*, vol. 20, no. 6, pp. 1709 –1724, June 2011.

[14] M. Hofmann, P. Tiefenbacher, and G. Rigoll, "Background segmentation with feedback: The pixel-based adaptive segmenter," in *CVPRW*, June 2012, pp. 38 –43.

[15] L. Maddalena and A. Petrosino, “The sobs algorithm: What are the limits?” in *CVPRW*, June 2012, pp. 21 –26.

[16] ——, “A self-organizing approach to background subtraction for visual surveillance applications,” *Image Processing, IEEE Transactions on*, vol. 17, no. 7, pp. 1168–1177, Jul. 2008.

[17] P. Darvish Zadeh Varcheie, M. Sills-Lavoie, and G.-A. Bilodeau, “A multiscale region-based motion detection and background subtraction algorithm,” *Sensors*, vol. 10, no. 2, pp. 1041–1061, 2010.

[18] P. Varcheie, M. Sills-Lavoie, and G. A. Bilodeau, “An efficient region-based background subtraction technique,” in *Computer and Robot Vision, 2008. CRV ’08. Canadian Conference on*, May, pp. 71–78.

[19] S. Brutzer, B. Hoferlin, and G. Heidemann, “Evaluation of background subtraction techniques for video surveillance,” in *CVPR*, June 2011, pp. 1937 –1944.

[20] A. Bugeau and P. Perez, “Detection and segmentation of moving objects in complex scenes,” *Computer Vision and Image Understanding*, vol. 113, no. 4, pp. 459 – 476, 2009.

[21] M. Heikkilä and M. Pietikäinen, “A texture-based method for modeling the background and detecting moving objects,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 4, pp. 657–662, 2006.

[22] Y. Nonaka, A. Shimada, H. Nagahara, and R. Taniguchi, “Evaluation report of integrated background modeling based on spatio-temporal features,” in *CVPRW*, June 2012, pp. 9 –14.

[23] J.-P. Jodoin, G.-A. Bilodeau, and N. Saunier, “Background subtraction based on local shape,” *CoRR*, vol. abs/1204.6326, 2012.

[24] E. Tola, V. Lepetit, and P. Fua, “A fast local descriptor for dense matching,” in *CVPR*, 2008.