Background Segmentation with Feedback: The Pixel-Based Adaptive Segmenter

Martin Hofmann, Philipp Tiefenbacher, Gerhard Rigoll
Institute for Human-Machine Communication
Technische Universität München
martin.hofmann@tum.de, philtief@gmail.com, rigoll@tum.de

Abstract

In this paper we present a novel method for foreground segmentation. Our proposed approach follows a non-parametric background modeling paradigm, thus the background is modeled by a history of recently observed pixel values. The foreground decision depends on a decision threshold. The background update is based on a learning parameter. We extend both of these parameters to dynamic per-pixel state variables and introduce dynamic controllers for each of them. Furthermore, both controllers are steered by an estimate of the background dynamics. In our experiments, the proposed Pixel-Based Adaptive Segmenter (PBAS) outperforms most state-of-the-art methods.

1. Introduction

In many image processing and computer vision scenarios, an important preprocessing step is to segment moving foreground objects from a mostly static background. Major application scenarios are in the field of mobile devices, video games and visual surveillance, e.g. for detection of unattended luggage, person counting, face recognition and gait recognition.

The general idea of background segmentation is to automatically generate a binary mask which divides the set of pixels into the set of foreground and the set of background pixels. In the simplest case, a static background frame can be compared to the current frame. Pixels with high deviation are determined as foreground. This simplistic method might work in certain specialized scenarios. However, often an empty background frame is not available at all, background has subtle motion, or light is gradually changing. Thus, to model the background, a multitude of more sophisticated methods have been developed in the recent past.

In this paper, we present a new method which builds on several previous methods and adds ideas from control system theory. We call the resulting method Pixel-Based Adaptive Segmenter (PBAS), because several parameters are adaptively adjusted at runtime for each pixel separately. We evaluate our proposed method on the *Change Detection Challenge* [3]. This collection of databases features a wide range of scenarios from potentially relevant application scenarios making this database an ideal testbed for competitive performance evaluation. On this challenge we present significant performance gain on most current state-of-the-art approaches.

First we present related background segmentation methods in Section 2. We then explain our Pixel-Based Adaptive Segmenter in detail in Section 3. Our method uses a variety of parameter, which are evaluated and tuned in Section 4.1. The best set of parameters is fixed and declared as the standard settings. In Section 4.4 we compare our results to other background segmentation methods.

2. Related Work

Over the recent past, a multitude of algorithms and methods for background modeling have been developed. Mentioning all of them would go beyond the scope of this paper. Excellent survey papers can be found in [6] and [7]. However, we want to mention relevant approaches, which our method builds on:

One of the most prominent and most widely used methods are those based on Gaussian Mixture Models (GMM) [8]. Here, each pixels is modeled as a mixture of weighed Gaussian distributions. Pixels, which are detected as background are used to improve the Gaussian mixtures by an iterative update rule. This parametric method has very low memory complexity. In SACON (SAmple COnsensus) [9], the background model is defined by a non-parametric method. Each pixel in the background model is defined by a history of the N most recent image values at each pixel. The history of background images is filled using a first-in first-out strategy. By contrast to this in-order filling, in ViBe [1], which is also a non-parametric method, the N background values are updated by a random scheme. More over, updated pixels can "diffuse" their current pixel value into neighboring pixel using another random selection method.

Our PBAS method can also be categorized as a non-parametric method, since we also use a history of N image

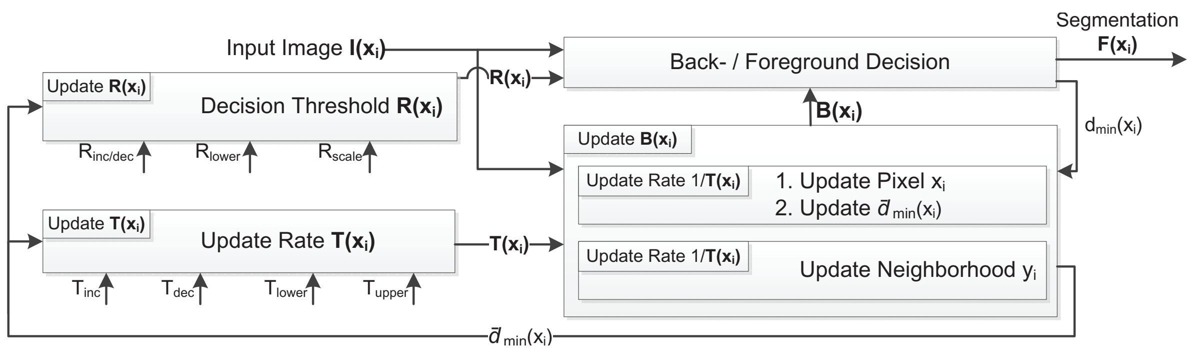

Figure 1: Overview of the Pixel-Based Adaptive Segmenter.

values as the background model as in SACON. We use a similar random update rule as in ViBe. However, in Vibe, the randomness parameters as well as the decision threshold are fixed for all pixels. In contrast, we do not treat these values as parameters, but instead as adaptive state variables, which can dynamically change over time for each pixel separately.

3. The Pixel-Based Adaptive Segmenter

This section describes the Pixel-Based Adaptive Segmenter, which follows a non-parametric paradigm. Thus, every pixel x_i is modeled by an array of recently observed background values.

Our method consists of several components which are depicted as a state machine in Figure 1. As a central component, the decision block decides for or against foreground based on the current image and a background model $\boldsymbol{B}(x_i)$. This decision is based on the per-pixel threshold $R(x_i)$. Moreover, the background model has to be updated over time in order to allow for gradual background changes. In our model, this update depends on a per-pixel learning parameter $T(x_i)$.

Now, the essential and novel idea of our PBAS approach is that both of these two per-pixel thresholds dynamically change based on an estimate of the background dynamics.

In the following, we first describe the decision process and the update of the background. Then we detail our dynamic update method of both the decision threshold $R(x_i)$ and the learning parameter $T(x_i)$.

3.1. Segmentation Decision

The goal of every background segmentation method is to come to a binary decision, whether a pixel belongs to the foreground or to the background. This decision process takes the input image and compares it in some way to a model of the background. In our case, the background model $\boldsymbol{B}(x_i)$ is defined by an array of N recently observed pixel values:

$$\boldsymbol{B}(x_i) = \{B_1(x_i), \dots, B_k(x_i), \dots, B_N(x_i)\} \quad (1)$$

A pixel x_i is decided to belong to the background, if its pixel value $I(x_i)$ is closer than a certain decision threshold $R(x_i)$ to at least $\#_{min}$ of the N background values. Thus, the foreground segmentation mask is calculated as

$$F(x_i) = \begin{cases} 1 & \#\{dist(I(x_i), B_k(x_i)) < R(x_i)\} < \#_{min} \\ 0 & \text{else} \end{cases} \quad (2)$$

Here, $F = 1$ implies foreground. It can thus be seen, that the decision making involves two parameters: (1) The distance threshold $R(x_i)$, which is defined for each pixel separately and which can change dynamically; and (2) the minimum number $\#_{min}$, which is a fixed global parameter.

3.2. Update of Background Model

Updating the background model B is essential in order to account for changes in the background, such as lighting changes, shadows and moving background objects such as trees.

Since foreground regions cannot be used for updating, the background model is only updated for those pixels that are currently background (i.e. $F(x_i) = 0$). Updating means that for a certain index $k \in 1 \dots N$ (chosen uniformly at random), the corresponding background model value $B_k(x_i)$ is replaced by the current pixel value $I(x_i)$. This allows the current pixel value to be "learned" into the background model. This update, however, is only performed with probability $p = 1/T(x_i)$. Otherwise no update is carried out at all. Therefore, the parameter $T(x_i)$ defines the update rate. The higher $T(x_i)$, the less likely a pixel will be updated.

We also update (with probability $p = 1/T(x_i)$) a randomly chosen neighboring pixel $y_i \in \mathcal{N}(x_i)$. Thus, the background model $B_k(y_i)$ at this neighboring pixel is replaced by its current pixel value $V(y_i)$. This is contrary to the approach in [1] where $B_k(y_i)$ is replaced by the pixel value $I(x_i)$ of the current pixel (called "diffusion" in their approach).

In general, a pixel x_i is only updated, if it is classified as background. However, a neighboring pixel y_i, which might be foreground, can be updated as well. This means that certain foreground pixels at the boundary will gradually be included into the background model. With this method, every foreground object will be "eaten-up" from the outside after a certain time, depending on the update parameter $T(x_i)$. The advantage of this property is that erroneous foreground objects will quickly vanish. Obviously this will also include slowly moving foreground objects into the background. Therefore, in Section 3.4 we present a dynamic adaptation of the update parameter $T(x_i)$, such that big objects are only "eaten-up" a little bit, while small erroneous blobs are "eaten" completely.

3.3. Update of the Decision Threshold $R(x_i)$

In a video sequence, there can be areas with high background dynamics (i.e. water, trees in the wind, etc.) and areas with little to no changes (i.e. a wall). Ideally, for highly dynamic areas, the threshold $R(x_i)$ should be increased as to not include objects to the foreground. For static regions, $R(x_i)$ should be low, such that small deviations lead to a decision for foreground. Thus, the threshold $R(x_i)$ needs to be able to automatically adapt accordingly. To allow for these changes, there needs to be a measure of background dynamics, which is done as follows:

First of all, besides saving an array of recently observed pixel values in the background model $\mathbf{B}(x_i)$, we also create an array $\mathbf{D}(x_i) = \{D_1(x_i), \ldots, D_N(x_i)\}$ of minimal decision distances. Whenever an update of $B_k(x_i)$ is carried out, the currently observed minimal distance $d_{min}(x_i) = \min_k dist(I(x_i), B_k(x_i))$ is written to this array: $D_k(x_i) \leftarrow d_{min}(x_i)$. Thus, a history of minimal decision distances is created. The average of these values $\bar{d}_{min}(x_i) = 1/N \sum_k D_k(x_i)$ is a measure of the background dynamics.

For example, assuming a completely static background, $\bar{d}_{min}(x_i)$ will be zero. For more dynamic background, there will always be a (small) deviation of the currently observed value to the previously seen ones, and thus $\bar{d}_{min}(x_i)$ will be higher.

With this estimate of the background dynamics, the decision threshold can be dynamically adapted as follows:

$$R(x_i) = \begin{cases} R(x_i) \cdot (1 - R_{inc/dec}), & \text{if} \quad R(x_i) > \bar{d}_{min}(x_i) \cdot R_{scale} \\ R(x_i) \cdot (1 + R_{inc/dec}), & \text{else} \end{cases} \tag{3}$$

Here, $R_{inc/dec}, R_{scale}$ are fixed parameters. This can be seen as a dynamic controller for the state variable $R(x_i)$. For a constant $\bar{d}_{min}(x_i)$, the decision threshold $R(x_i)$ approaches the product of $\bar{d}_{min}(x_i) \cdot R_{scale}$. Thus, a (sudden) increase in background dynamics leads to a (slow) increase of $R(x_i)$ towards a higher decision threshold $R(x_i)$.

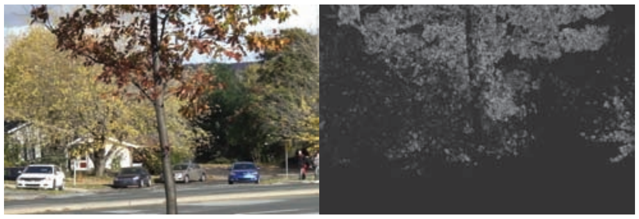
Figure 2: Example for the local distribution of $R(x_i)$.

Controlling the update rate in the way presented above leads to robust handling of varying amounts of background dynamics. An example picture with high dynamic background showing the spatial distribution of the state variable $R(x_i)$ is depicted in 2. In this, brighter pixel values indicating a higher value for $R(x_i)$.

3.4. Update of the Learning Rate $T(x_i)$

As mentioned in section 3.2, independent of the foreground state $F(x_i)$, eventually, every object will be merged into the background depending on the learning parameter $T(x_i)$. To alleviate the problem, the idea is to introduce a (second) dynamic controller for $T(x_i)$, such that the probability of background learning is (slowly) increased when the pixel is background and (slowly) decreased when the pixel is foreground. A problem of this is, that wrongly classified foreground is only slowly learned into the background and thus remains foreground.

It can be assumed that pixels are mostly wrongly classified as foreground in areas of high dynamic background. Thus, the strength of the adjustment in the controller can be adapted using the dynamic estimator $\bar{d}_{min}(x_i)$. We define:

$$T(x_i) = \begin{cases} T(x_i) + \frac{T_{inc}}{\bar{d}_{min}(x_i)}, & \text{if } F(x_i) = 1 \\ T(x_i) - \frac{T_{dec}}{\bar{d}_{min}(x_i)}, & \text{if } F(x_i) = 0 \end{cases} \tag{4}$$

Here, T_{inc}, T_{dec} are fixed parameters. There are different parameters for the two cases, because we assume that most of the time the pixels are background, which is true in most cases. Choosing independent parameters for background and foreground therefore leads to a balanced regulation of $T(x_i)$. Furthermore we define an upper and lower bound $T_{lower} < T < T_{upper}$, such that values cannot go out of a specified bound. The above controller ensures, that in case of highly dynamic background (i.e. big $\bar{d}_{min}(x_i)$), the learning parameter $T(x_i)$ stays constant or only slightly changes. In this case of highly dynamic background, erroneously detected foreground will not remain for long, because the probability to update $p = 1/T(x_i)$ does not reach zero too fast. In the other ideal case of a fully static background, a classification as foreground is quite solid, hence

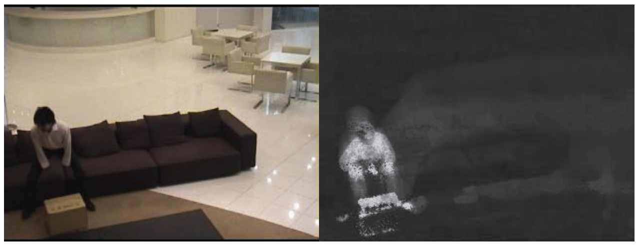

Figure 3: Example for the local distribution of $T(x_i)$.

$T(x_i)$ rapidly increases, validating the background model, in the way that it retains less updates.

Figure 3 depicts the update parameter $T(x_i)$ in case of a slowly moving, but significantly large foreground object. For the pixels corresponding to the person, a high $T(x_i)$ indicates low update probability. Assume a background pixel just outside of the person's silhouette. This pixel can update a foreground silhouette boundary pixel by means of the neighbor update rule. Thus, a boundary pixel can become background and therefore the silhouette is "eaten-up" from the outside. The low update probability at this pixel, however, will avoid further shrinking of the silhouette, such that the majority of the silhouette is still detected as foreground.

3.5. Implementation Details

In practice, the input image $I(x_i)$ is a three channel color image. In our approach, we treat each channel independently and run our algorithms in three parallel threads. The final segmentation $F(x_i)$ results from a bit-wise OR operation of the three segmentations $F^R(x_i), F^G(x_i), F^B(x_i)$.

For each color channel, in addition to the pixel value, we also use gradient magnitudes. Thus, the input $I(x_i) = \{I^v(x_i), I^m(x_i)\}$ consists of the pixel value $I^v(x_i)$ itself and the gradient magnitude $I^m(x_i)$ at the pixel. Consequently, each element of the background history $B_k(x_i) = \{B_k^v(x_i), B_k^m(x_i)\}$ (in Eq. 1) also consists of two corresponding entries. For the distance calculation in Eq. 2, we use the following equation:

$$dist(I(x_i), B_k(x_i)) = \frac{\alpha}{\overline{I^m}} \cdot |I^m(x_i) - B_k^m(x_i)| + |I^v(x_i) - B_k^v(x_i)| \quad (5)$$

Here, $\overline{I^m}$ is the average gradient magnitude over the last observed frame. Thus, the fraction $\frac{\alpha}{\overline{I^m}}$ weighs the importance of pixel values against the gradient magnitude.

4. Evaluation

We evaluate our approach on the database provided for the Change Detection Challenge [3]. This database features 31 videos from six categories including scenarios with indoor views, outdoor views, shadows, camera jitter, small objects, dynamic background and thermal images. Human-annotated ground truth is available for all scenarios and is used for performance evaluation. Thus, exhaustive competitive comparison of methods is possible on this database.

4.1. Parameter Settings

Our methods consists of a multitude of tunable parameters, which have to be adjusted for optimal system performance. Since we evaluate on the database of the Change Detection Challenge, a multitude of possible scenarios are covered. We seek one unique optimal set of parameters which gives the best performance on the complete dataset. Never the less, for certain applications, parameters could be fine tuned for a specific need. Overall, the 9 parameters, detailed below, need to be tuned:

(a) $\mathbf{N = 35}$: N is the number of components of the background model. It can be seen that "easy" scenarios such as baseline, shadow and thermal require as little as 10 background components. At $N = 35$, performance saturates for all scenarios. Increasing N further only increases memory and computational complexity.

(b) $\mathbf{\#_{min} = 2}$: The number of components that have to be closer than $R(x_i)$ in order to set the pixel to be background. An optimum is found at $\#_{min} = 2$. The same optimal value has been found in [1] and in [9].

(c) $\mathbf{R_{inc/dec} = 0.05}$: The rate, at which the decision threshold $R(x_i)$ is regulated by the controller. Recognition rate is not very sensitive to this value. Camera jitter and dynamic background seem to perform best for low regulation.

(d) $\mathbf{R_{lower} = 18}$: Lower bound of the decision threshold. Setting it too low leads to ghosts and false positives (thus, low precision), too high will lead to misses (and thus low recall).

(e) $\mathbf{R_{scale} = 5}$: Scaling factor in the controller for the decision threshold. This controls the equilibrium value $\bar{d}_{min}(x_i) \cdot R_{scale}$ and thus low values lead to low precision, while high values lead to low recall.

(f) $\mathbf{T_{dec} = 0.05}$: If x_i is background, T_{dec} is the rate at which $T(x_i)$ is decreased (i.e the rate at which the probability of background update is increased).

(g) $\mathbf{T_{inc} = 1}$: If x_i is foreground, T_{inc} is the rate at which $T(x_i)$ is increased (i.e the rate at which the probability of background update is decreased). Because a priori, foreground is less likely than background, a lower adaptation for foreground is beneficial (e.g. to keep standing foreground objects).

(h) $\mathbf{T_{lower} = 2}$: Lower bound of $T(x_i)$. Quite constant in the tested range. In case of intermittent objects and camera jitter there is a slight decrease for high values.

(i) $\mathbf{T_{upper} = 200}$: The upper bound may not be chosen too low. Values higher than 200 lead to good results. This concludes that a minimum update probability of $1/200$ is required.

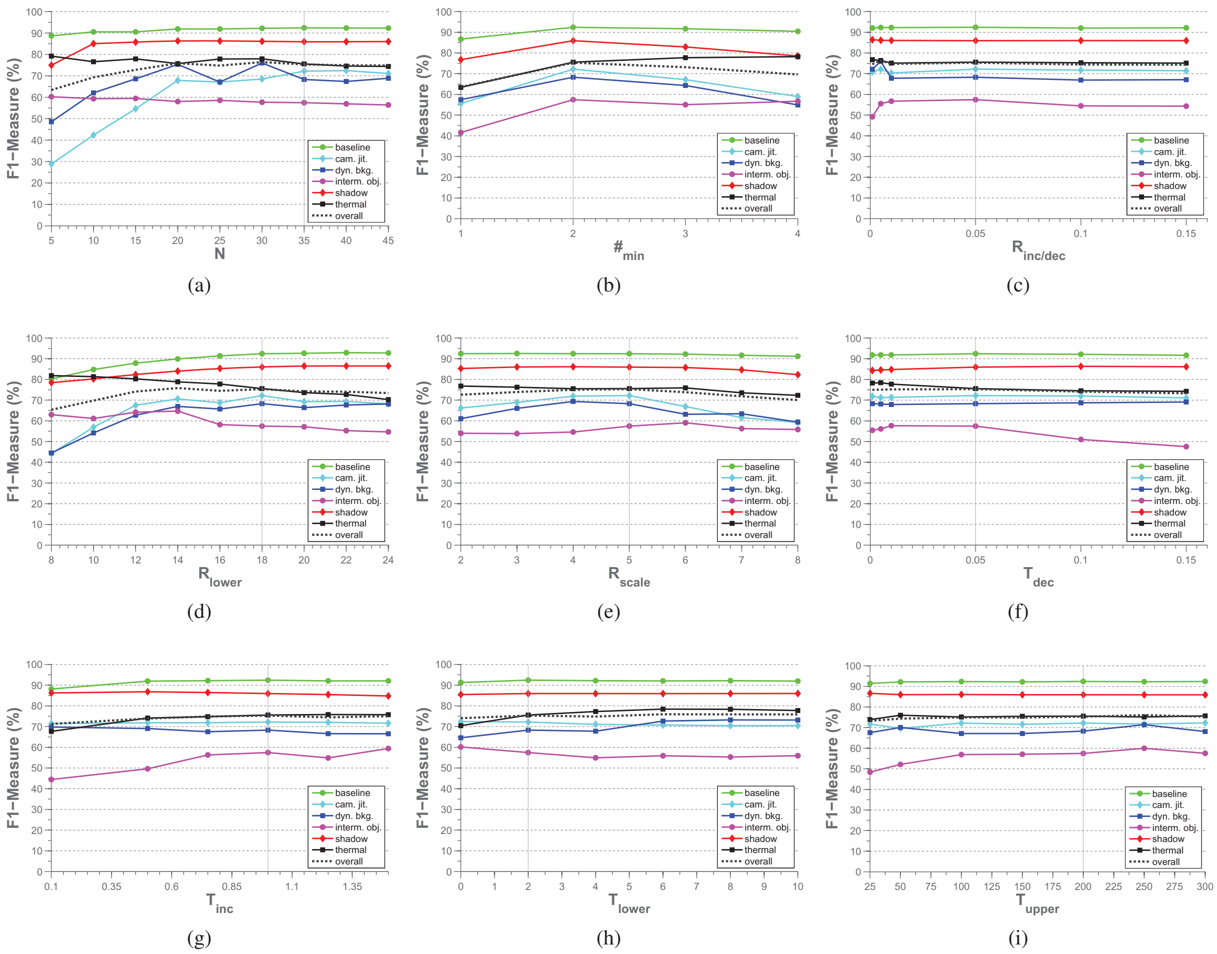

Figure 4: F1 performance of PBAS on each category as well as overal performance with changing parameter settings.

4.2. Post-Processing with Median Filter

Our proposed method is a pixel based method, which means that the segmentation decision is made independently for each pixel. The resulting output can thus benefit from spatial smoothing, which is done using simple median filtering. Table 1 depicts results for different median filter sizes. It can be seen that strong median filtering leads to better results. It is important to note that while median filtering reduces noise, it also smoothes the blobs and leads to less sharp boundaries. Thus, we decided to use a 9×9 median filters for all our experiments.

Table 1: Overall results in F1 and PBC measure, with different sizes of median filter for post processing.

	none	3x3	5x5	7x7	9x9	11x11
F1	0.7185	0.7351	0.7446	0.7500	0.7531	0.7542
PBC	2.1011	1.9393	1.8559	1.8037	1.7699	1.7525

4.3. Influence of Gradient Magnitudes

As described in Section 3.5, we added gradient magnitudes to our input features. As an optimal weighting parameter we used $\alpha = 10$ (in Eq. 5) for all our comparative evaluations. At this setting, we get an F1 of 0.7532 and a PBC of 1.7693. Setting $\alpha = 0$, disables gradient magnitudes and only uses pixel values. Here, we get an F1 of 0.7333 and a PBC of 1.8985. Therefore, we can observe a slight performance gain when adding gradient magnitudes.

4.4. Performance Evaluation

In Table 2 results using the seven proposed performance measures are shown for all six scenarios. Our method performs best for the categories *baseline*, *shadow* and *thermal*. In Table 3, the results of our PBAS is compared to several state of the art methods (those that were available prior to the change detection workshop). It can be seen that at the chosen optimal operating point, PBAS greatly outperforms

Table 2: Results of PBAS on all six scenarios using all seven measures.

Scenarios	Recall	Specificity	FPR	FNR	PBC	F1	Precision
Baseline	0.9594	0.9970	0.0030	0.0021	0.4858	0.9242	0.8941
Camera Jitter	0.7373	0.9838	0.0162	0.0100	2.4882	0.7220	0.7586
Dynamic Background	0.6955	0.9989	0.0011	0.0045	0.5394	0.6829	0.8326
Intermittent Object Motion	0.6700	0.9751	0.0249	0.0222	4.2871	0.5745	0.7045
Shadow	0.9133	0.9904	0.0096	0.0039	1.2753	0.8597	0.8143
Thermal	0.7283	0.9934	0.0066	0.0104	1.5398	0.7556	0.8922
Overall PBAS	0.7840	0.9898	0.0102	0.0088	1.7693	0.7532	0.8160

Table 3: Comparison of PBAS to several state-of-the-art methods using all seven proposed performance measures.

	Recall	Specificity	FPR	FNR	PBC	F1	Precision
SOBS [5]	**0.7882**	0.9818	0.0182	0.0094	2.5642	0.7159	0.7179
GMM — KaewTraKulPong [4]	0.5072	**0.9947**	**0.0053**	0.0291	3.1051	0.5904	**0.8228**
ViBe [1]	0.6821	0.983	0.017	0.0176	3.1178	0.6683	0.7357
KDE [2]	0.7442	0.9757	0.0243	0.0138	3.4602	0.6719	0.6843
GMM — Stauffer & Grimson [8]	0.7108	0.986	0.014	0.0202	3.1046	0.6623	0.7012
GMM — Zivkovic [10]	0.6964	0.9845	0.0155	0.0193	3.1504	0.6596	0.7079
our PBAS	0.7840	0.9898	0.0102	**0.0088**	**1.7693**	**0.7532**	0.8160

most other methods in F1 measure as well as in ”percentage of bad classification” (PBC). These two measures seem to be best for a balanced comparison of methods. Also in the other measures, PBAS outperforms most other methods, except [4], which seems to be tuned towards high precision at the cost of low recall and low F1.

5. Outlook and Conclusion

We have presented a highly efficient background modeling method. The basic idea behind this method is to use two controllers with feedback loops for both the decision threshold as well as for the learning parameter. Tuning the resulting parameters leads to results that outperform the state of the art.

Future work will focus on reducing the number of necessary parameters, as well as on further research on the underlying system dynamics. Currently, no explicit shadow modeling is performed which could be addressed in future.

References

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

[2] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *FRAME-RATE WORKSHOP, IEEE*, pages 751–767, 2000. 6

[3] P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. *Change Detection Challenge*, 2012. http://www.changedetection.net. 1, 4

[4] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. *2nd European Workshop on Advanced Video Based Surveillance Systems,*, 2001. 6

[5] L. Maddalena and A. Petrosino. A self-organizing approach to background subtraction for visual surveillance applications. *Image Processing, IEEE Transactions on*, 17(7):1168 –1177, july 2008. 6

[6] M. Piccardi. Background subtraction techniques: a review. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 4, pages 3099 – 3104 vol.4, oct. 2004. 1

[7] R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: a systematic survey. *Image Processing, IEEE Transactions on*, 14(3):294 –307, march 2005. 1

[8] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 2 vol. (xxiii+637+663), 1999. 1, 6

[9] H. Wang and D. Suter. A consensus-based method for tracking: Modelling background scenario and foreground appearance. *Pattern Recognition*, 40(3):1091–1105, 2007. 1, 4

[10] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28 – 31 Vol.2, aug. 2004. 6