AMBER: ADAPTING MULTI-RESOLUTION BACKGROUND EXTRACTOR

Bin Wang and Piotr Dudek

The University of Manchester, School of Electronic and Electrical Engineering
Manchester, M13 9PL, United Kingdom
bin.wang-2@postgrad.manchester.ac.uk; p.dudek@manchester.ac.uk

ABSTRACT

In this paper, a fast self-adapting multi-resolution background detection algorithm is introduced. A pixel-based background model is proposed, that represents not only each pixel's background values, but also their efficacies, so that new background values always replace the least effective ones. Model maintenance and global control processes ensure fast initialization, adaptation to background changes with different timescales, restrain the generation of ghosts, and adjust the decision thresholds based on noise levels. Evaluation results indicate that the proposed algorithm outperforms most other state-of-the-art algorithms not only in terms of accuracy, but also in terms of processing speed and memory requirements.

Index Terms— motion detection, background subtraction, video analytics, surveillance

1. INTRODUCTION

Background detection is widely used in various computer vision tasks, in particular, automated video surveillance. The aim is to separate the foreground, i.e., objects of interest (e.g. cars, people) from the background (e.g. motorways, corridors). In such applications, both the accuracy of the background detection and the computational efficiency of the implementation are of importance. A fast background subtraction algorithm can decrease the cost of the system by allowing, for example, a greater number of cameras to be served by a single computer. Furthermore, it could enable the transfer of the computations to embedded processors in "smart cameras", for an efficient, distributed computing system [1].

A number of background subtraction algorithms have been recently proposed [2-6] and their performance has been compared in [7]. In this paper, a fast and effective Adapting Multi-resolution Background ExtractoR (AMBER) algorithm is introduced. It is a pixel based background subtraction algorithm, and its salient novel feature is that the background model 'templates' represent not only past background values for each pixel, but also their efficacies (indicating the frequency with which the pixels are classified as matching each individual background value). Utilizing these efficacy scores, the background model maintenance process ensures that newly qualified templates replace only the least efficient ones (rather than the oldest or randomly picked ones). To improve the detection accuracy and the robustness against different types of noise, the proposed algorithm also employs distinct long-term and short-term template update strategies, detection on multiple spatial scales, detection of noisy pixels and decision threshold adaptation procedures. At the same time, the computations are kept simple and the algorithm is optimized to exploit pixel-level parallelism, in order to achieve high processing speeds (thousands of frames per second on a modern PC), while requiring a moderate amount of memory, making it suitable for both centralized multi-camera setups as well as embedded applications. A set of established benchmarks [8] has been used to evaluate the algorithm. Based on the evaluation results, the proposed algorithm outperforms most state-of-the-art background detection algorithms not only in terms of accuracy, but also processing speed.

2. AMBER

An overview of the proposed algorithm can be found in Fig.1. The algorithm is pixel-parallel, the computations described below are carried out independently for each pixel of a video frame (unless indicated otherwise, for global parameters). For each pixel, the background model is composed by K+1 templates: a "long-term" template (T_0) and K "short-term" templates ($\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3 \ldots \mathrm{T}_K$). Each template (T_k) consists of a background value B_k representing the centre point of a cluster of pixel values that are considered to represent background, and an efficacy C_k representing the 'hit counter' for that template.

$$\mathrm{T}_k = \{\mathrm{B}_k, \mathrm{C}_k\},\ k = 1,2, \ldots K. \quad (1)$$

2.1. Background Detection and Model Adaptation

The background templates are ordered according to their temporal characteristics, so that template T_0 always represents the background value that has been present at the corresponding pixel for the longest time. We will consider this the most effective template. Templates $\mathrm{T}_1, \mathrm{T}_2 \ldots \mathrm{T}_K$ represent K different background values that are likely to

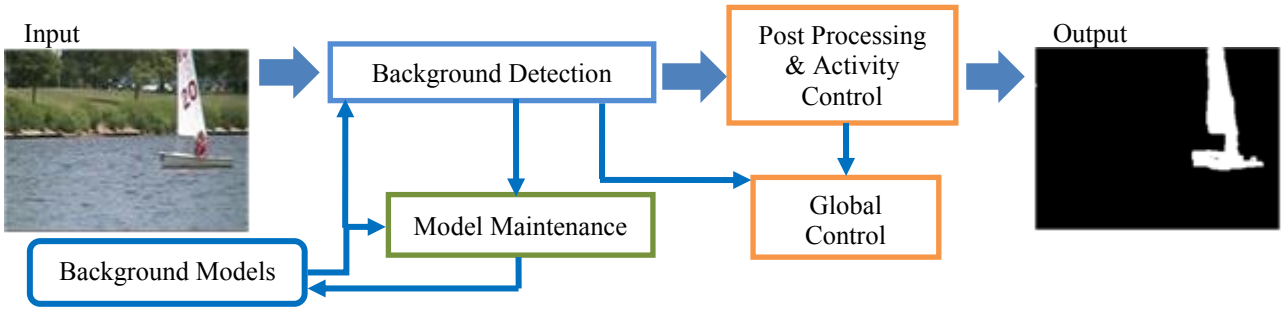

Figure 1: An overview of the proposed background detection algorithm.

appear at that particular pixel on shorter time scales. The efficacy (as indicated by the corresponding C_k value) decreases from T_1 to T_K. In the experiments presented in this paper, K is set to 3 to obtain the balance between memory requirements, processing speed and detection accuracy. At a particular time, for each pixel, some of the K templates may be inactive (denoted by $C_k = 0$).

For each pixel in the current frame, the pixel value P is compared with the background value B_k of the corresponding template, for all active templates (C_k>0). If the distance between these two values is smaller than the decision threshold ε,

$$||P - B_k|| < \varepsilon \quad (2)$$

this pixel is classified as a background pixel, and the corresponding background value B_k stored in that template will be updated (with learning rate α) towards the current pixel value P.

$$B_k \leftarrow (1 - \alpha)B_k + \alpha P \quad (3)$$

For each template, its efficacy C_k will be updated as well according to the detection result using that template. The efficacy of the first template that classified the pixel as a background pixel will be incremented (+1), while the efficacies of other templates will be decremented (-1). If C_k becomes 0, template T_k becomes "inactive", and is removed from further processing, ready to be replaced by a new template. The maximum values stored in C_0 and C_1 are saturated at L_0 and L_1 respectively.

The detection process starts by analyzing template T_0 (i.e. the most effective one), and progresses though the other K templates in sequence. The detection result is stored in a "mask" M_F. Initially, all pixels are marked as foreground (M_F=1). When the pixel is classified as background by any template T_k, this pixel will be marked as background (M_F=0).

In order to remove single-pixel noise, and improve robustness against minor high-frequency changes of the background (e.g. such as caused by waving trees or rippling water), a low-resolution detection process is also implemented. Both the current frame and the templates T_0 and T_1 are down-sampled by averaging array values in a block. In experiments presented in this paper, 4x4 blocks are used. The reason T_0 and T_1 are chosen here is that these two templates always contain the most effective background values for each pixel.

The detection rules used in the lower resolution detection are the same as those used in the full resolution detection, however, the B_k and C_k updating is not carried out. During the detection process, the classification results of all blocks in the down-sampled array are saved into a binary detection mask. A dilation is performed on this mask, which expands the foreground region to prevent the misclassifications on the edges of the foreground objects in full resolution, and a full resolution mask M_D is generated by up sampling the low resolution mask. The final detection result R for each pixel is computed as the logic *AND* between masks M_F and M_D.

2.2. Background Model Maintenance

Although the cluster updating process given by (3) ensures that the algorithm has the ability to adapt to gradual background changes, an additional mechanism is required to prevent permanent misclassification errors, e.g. everlasting ghosts. The proposed method introduces a template replacing procedure, which maintains the order of the templates according to their efficacies, replaces clusters with low efficacy with new clusters, and upgrades clusters from short-term background templates to the long-term background template.

Short-term Template Ordering: The K short-term templates are sorted according to their efficacy, so that the most effective cluster is always stored in T_1 and the least effective cluster is always stored in T_K. Recall that the efficacy C_k is changing according to the detecting history of T_k. The ordering process starts with comparing the efficacy of T_K withT_{K-1}. If C_K is bigger than C_{K-1}, the data stored in T_K and T_{K-1} are swapped. The ordering procedure compares all other short term templates in turn, and follows the same process to upgrade more effective background values to a higher order template.

Template Replacement: The template replacement process monitors the foreground pixels generated by the detection process, in order to find the foreground pixels that stay motionless for a long time, and replace the cluster stored in T_K (i.e. the least effective template for each pixel) with a new qualified template.

To realize this function, a special accumulation template (T_A) is maintained. T_A includes a prospective

background cluster value B_A and an efficacy C_A, just like other templates used in the detection step. When a new foreground pixel is detected, the distance between this foreground pixel value P and the cluster centre of the corresponding accumulation template B_A is calculated. The efficacy C_A is updated according to this distance, i.e. for each pixel, if this distance is smaller than the decision threshold ε, C_A is incremented; otherwise, C_A is decremented. For the templates with C_A equal to 0, if the corresponding pixel on the current frame is a foreground pixel, the value of this foreground pixel will be loaded into B_A, and C_A will be set to 1. If C_A is bigger than a preset threshold θ_H, the values and efficacies stored in T_A and T_K will be swapped.

Long-term Template Update: Another important function of the background model maintenance process is to exchange the long-term template with the most persistent short-term template.

The long-term template, T_0, contains the stationary background values that are present in the frame for a long time. If a background value stored in T_1 appears in the frame long enough, which is determined by a threshold θ_L ($C_1 > \theta_L$), and the corresponding background value in T_0 happens to be less frequent ($C_0 < \theta_L$), the background value stored in template T_1 is swapped with T_0, and C_0 is reset to $\gamma\theta_L$. Hence, parameter γ controls the time that the newly updated long-term background value will remain in the long-term template, regardless of any subsequent background changes. A relatively large γ (e.g. $\gamma = 3$) will restrain the generation of ghosts.

2.3. Post-processing and global control procedure

The post-processing procedure of the proposed algorithm is composed of filtering with basic binary morphologic operations (opening and closing), followed by an 'Activity Control' process designed to suppress the detection noise. The global control procedure provides detection threshold adaptation.

Activity Control: To suppress false positive errors in noisy environments, an activity control process which monitors the detection result is proposed. The activity control process maintains an activity level measure A_N for each pixel. We define a 'noise-pixel' as a pixel that has been classified as a foreground pixel during the full resolution detection process, i.e. $M_F = 1$, however, after the low resolution detection and the filtering steps, it has become a background pixel. In a noise-pixel mask, M_N, the identified noise-pixels are set to 1 while other pixels are 0. The activity control process also stores the mask for the previous frame. If the noise-pixel mask of the current frame, $M_N[n]$, is different to previous frame, $M_N[n-1]$, that pixel is classified as a "blinking" pixel, and its activity level A_N is increased by β_{inc}, otherwise it is decreased by β_{dec}. The activity level is limited between 0 and β_{MAX}. Foreground pixels having an activity level bigger than β_{TH}(i.e. noise-pixels that are constantly switching between foreground and background), are eliminated from the final detection result.

Global Control: A global process is implemented in the proposed algorithm to automatically adapt the decision threshold ε (which is the parameter that has the most direct effect on the quality of the classification process) according to the noise level of the input frames. The threshold adaptation process is as follows: the number of the noise-pixels N_{NP} for each input frame is obtained at the end of the detection process. This number, indicating the noise level of the input frame, provides a way to tune the decision threshold ε. For each frame, if the number of noise pixels is bigger than a predefined level N_{max} (e.g. 3% of the input range), the decision threshold ε will increase by $\Delta\varepsilon$; if the number of noise pixels is smaller than another predefined level N_{min} (e.g. 0.4% of the input range), the decision threshold will decrease by $\Delta\varepsilon$. To constrain the adaptation range, ε is limited to $[\varepsilon_{min}, \varepsilon_{max}]$.

3. EXPERIMENTAL RESULTS

The proposed algorithm was evaluated using the Change Detection Challenge database and methodology [8]. This database, consisting of 31 video sequences grouped into 6 categories, covers different challenges in background detection scenarios, e.g. shadows, ghosts, background changes, and camera jitter in both outdoor and indoor environments. The manually labeled ground truth is available for all the video sequences. Seven measurements defined in [8]: Recall, Specificity, False Positive Rate (FPR), False Negative Rate (FNR), Percentage of Wrong Classifications (PWC), F-Measure (F-1) and Precision, are provided to evaluate different aspects of each algorithm, and the "average ranking" and "average ranking across categories" are two final benchmarks. This database has been used recently to evaluate more than 20 state-of-the-art background detection algorithms [7].

In the experiments, pixel values are represented as three dimensional vectors in the YC_bC_r colour space. Assuming pixels values $P = \{P^Y, P^{Cb}, P^{Cr}\}$, background cluster centre values $B_k = \{B_k{}^Y, B_k{}^{Cb}, B_k{}^{Cr}\}$, and detection thresholds vector $\varepsilon = \{\varepsilon^Y, \varepsilon^{Cb}, \varepsilon^{Cr}\}$, the distance threshold condition (2) is calculated as a conjunction across all individual dimensions.

$$(P^Y - B_k{}^Y < \varepsilon^Y) \wedge (P^{Cb} - B_k{}^{Cb} < \varepsilon^{Cb}) \wedge (P^{Cr} - B_k{}^{Cr} < \varepsilon^{Cr}) \quad (6)$$

The parameters used are as follows:

$\varepsilon^Y \in [17.5, 80]$; $\varepsilon^{Cb} = \varepsilon^{Cr} \in [9, 12]$; $\Delta\varepsilon^Y = 2.5$; $\Delta\varepsilon^{Cb} = \Delta\varepsilon^{Cr} = 1$; $\alpha = 0.01$; $\theta_H = 400$; $\theta_L = 2500$; $\gamma = 1.5$; $L_0 = 6000$; $L_1 = 4000$; $\beta_{max} = 75$; $\beta_{inc} = 6$; $\beta_{dec} = 1$; $\beta_{TH} = 60$; $N_{max} = 3\%$; $N_{min} = 0.04\%$.

Table 1 Average performance comparison of AMBER and eight other state-of-the-art algorithms with the highest ranking according to [7].

Algorithm	Average Ranking	Recall	Specificity	FPR	FNR	PBC	F1	Precision
SGMM-SOD [5]	4.71 (1st)	0.7681	0.9924	0.0076	0.2319	1.5730	0.7624	0.8351
AMBER (proposed)	**5.00 (2nd)**	**0.8039**	**0.9888**	**0.0112**	**0.1961**	**1.8568**	**0.7835**	**0.8095**
PBAS [3]	5.71 (3rd)	0.7840	0.9898	0.0102	0.2160	1.7693	0.7532	0.8160
DPGMM [6]	6.00 (4th)	0.8275	0.9855	0.0145	0.1725	2.1159	0.7763	0.7928
Vibe+ [9]	8.14 (5th)	0.6907	0.9928	0.0072	0.3093	2.1824	0.7224	0.8318
PSP-MRF [10]	9.57 (6th)	0.8307	0.9830	0.0170	0.1963	2.3937	0.7372	0.7512
SC-SOB[4]	10.00 (9th)	0.8017	0.9831	0.0169	0.1983	2.4081	0.7283	0.7315

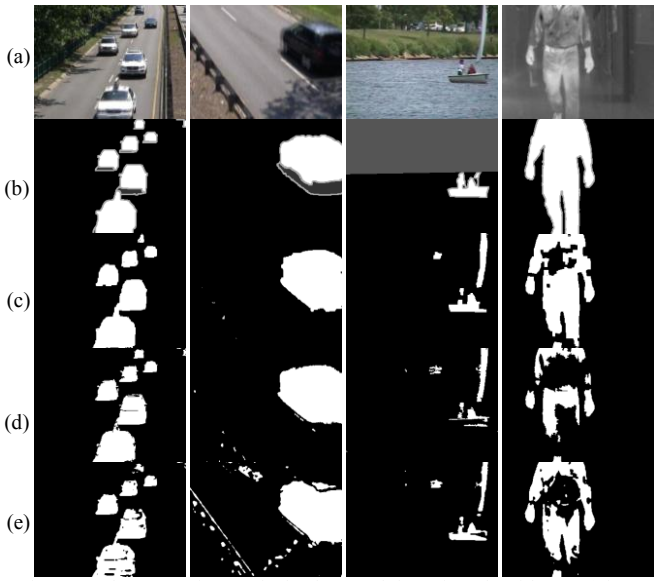

Fig.2 Result Comparisons of 3 background detection algorithms. (a) input frames; (b) corresponding ground truths [7]; (c) detection results generated by the proposed algorithm; (d) detection results produced by ViBe+ [9]; (e) detection results produced by GMM_Zivkovic[11];

Table 2. Speed and accuracy comparisons of AMBER and other three algorithms

	320x240 video processing time	False Positive pixels Number	False Negative Pixels Number
AMBER	3517 FPS	357137	245036
ViBe[2]	755 FPS	548470	279251
GMM[12]	90 FPS	3102943	311465
Bayesian[13]	53 FPS	11424806	88191

The evaluation results for all the six input categories are summarised in Table 1. Note that all the algorithms in Table 1 include post-processing steps. It can be seen that the proposed algorithm performs constantly well across all seven measurements, and particularly well in Recall, FNR, PBC and F1. Also, it achieves the second-best average ranking, as compared with other 26 algorithms considered in [7]. The detection results are illustrated in Fig.2.

In order to compare the processing speed of the implementation of the proposed algorithm with other algorithms, some simplifications are made: gray-level video sequences are considered, the background value used in the detection algorithm is the pixel intensity; and all post-processing steps are disabled (for AMBER, post processing and all global control processes are disabled). The speed comparison requires fully optimized implementations which are difficult to obtain, the comparisons presented here include only ViBe [2], GMM [12] and Bayesian background subtraction algorithm [13]. Note that SGMM-SOD [5] is a more complex algorithm than GMM [12]. The proposed algorithm is written in C++ using OpenMP and SSE, and compiled using Intel Compiler XE. The ViBe program used in this experiment has been fully optimized by the authors of [2] and can return the precise processing time (compiled code has been obtained via private communication). The GMM and Bayesian background subtraction algorithms are build-in functions in the Intel Integrated Performance Primitives (IPP) library. The platform of this experiment is a PC with an Intel Core2 Q9600 processor (running @2.66 GHz) and 4 GB DDR2 memory (400 MHz, 6-6-6-18). All four algorithms are processed completely by the CPU, there are no GPU computations involved. The resolution of the test video is 320x240 and it consists of 1700 frames. From the experimental result shown in Table 2 it can be clearly seen that the processing speed of the proposed algorithm is over four times higher than ViBe, and significantly higher than the other algorithms while at the same time the proposed algorithm provides better detection accuracy than all these algorithms.

4. CONCLUSIONS

A fast adaptive multi-resolution background detection algorithm (AMBER) has been introduced. The algorithm adapts to changing background through a number of background model update and parameter adaptation mechanisms, and implements simple but effective noise suppression strategies. In the background model maintenance process, AMBER introduces new background values to replace the *least effective* background values (rather than replacing the oldest or randomly chosen ones, which is done in other algorithms). Furthermore, it has the ability to initialize from only one frame, recover from misclassification, and quickly adapt to sudden background changes. The proposed algorithm is designed to be simple, and pixel-parallel as far as possible, which makes it easy to implement on processors with different architectures, especially those involving parallel (SIMD) units. According to the evaluation results, the proposed algorithm outperforms most of the state-of-the-art background detection algorithms in detection accuracy, while providing a significantly shorter execution time.

5. REFERENCES

[1] M. Bramberger, A. Doblander, A. Maier, B. Rinner and H. Scheabach, “Distributed Embedded Smart Cameras for Surveillance Applications”, Computer, Volume 39, Issue 2, pp. 68-75, 2006;

[2] Barnich and M. V. Droogenbroeck, "ViBe: A Universal Background Subtraction Algorithm for Video Sequences", IEEE Transactions on Image Processing, Vol. 20, No. 6, June 2011;

[3] M. Hofmann, P. Tiefenbacher, G. Rigoll, "Background Segmentation with Feedback: The Pixel -based Adaptive Segmenter", IEEE Workshop on Change Deteciton, The 25th Conference on Computer Vision and Pattern Recognition, 2012;

[4] L. Maddalena, A. Petrosino, "A Self-Organizing Approach to Background Subtraction for Visual Surveillance Applications", IEEE Transactions on Image Processing, Vol. 17, No. 7, July 2008;

[5] Rubén Heras Evangelio and Thomas Sikora "Complementary Background Models for the Detection of Static and Moving Objects in Crowded Environments". In Proceedings of the IEEE Int. Conference on Advanced Video and Signal Based Surveillance, 2011;

[6] T. D. F. Haines and T. Xiang “Background Subtraction with Dirichlet Processes”, European Conference on Computer Vision 2012;

[7] “ChangeDetection.net Video Database and evaluation result.” Internet: www.changedetection.net, 2012;

[8] N. Goyette, P. Jodoin, F. Porikli, J. Konrad and P. Ishwar, "changedetection.net: A New Change Detection Benchmark Dataset", IEEE Workshop on Change Deteciton, The 25th Conference on Computer Vision and Pattern Recognition, 2012;

[9] M. Van Droogenbroeck, O. Paquot, "Background Subtraction: Experiments and Improvements for ViBe", IEEE Workshop on Change Detection, CVPR 2012;

[10] Schick, M.Bäuml, R.Stiefelhagen "Improving Foreground Segmentations with Probabilistic Superpixel Markov Random Fields", in proc of IEEE Workshop on Change Detection, 2012

[11] Z. Zivkovic, "Improved adaptive Gaussian mixture model for back-ground subtraction," in Proc. Int. Conf. Pattern Recognition, pp. 28-31, IEEE, Piscataway, NJ 2004

[12] P. KaewTrakulPong, R. Bowden, “An Improved Adaptive Background Mixture Model for Real-Time Tracking with Shadow Detection”, Proc. 2nd European Workshop on Advance Video-Based Surveillance System, 2001;

[13] L. Li, W. Huang, I. Gu, Q. Tian, “Foreground Object Detection from Videos Containing Complex Background”, Proc. ACM Multimedia Conference, Berkley, 2003;