A Fast Self-tuning Background Subtraction Algorithm

Bin Wang
School of Electronic and Electrical Engineering
The University of Manchester
bin.wang-2@postgrad.manchester.ac.uk

Piotr Dudek
School of Electronic and Electrical Engineer
The University of Manchester
p.dudek@manchester.ac.uk

Abstract

In this paper, a fast pixel-level adapting background detection algorithm is presented. The proposed background model records not only each pixel's historical background values, but also estimates the efficacies of these values, based on the occurrence statistics. It is therefore capable of removing the least useful background values from the background model, selectively adapting to background changes with different timescales, and restraining the generation of ghosts. A further control process adjusts the individual decision threshold for each pixel, and reduces high frequency temporal noise, based on a measure of classification uncertainty in each pixel. Evaluation results based on the ChangeDetection.net database are presented in this paper. The results indicate that the proposed algorithm outperforms the majority of earlier state-of-the-art algorithms not only in terms of accuracy, but also in terms of processing speed.

1. Introduction

Background detection is widely used in various computer vision tasks, in particular, automated video surveillance, crowd and traffic monitoring, to separate the foreground, i.e. objects of interest, from the background. In the past years, many background subtraction algorithms have been proposed. The detection accuracy remains the researcher's main focus, but even though it can be improved, it is done at the expense of increased computational power. The processing speed of the majority of state-of-the-art background subtraction algorithms is less than 24 fps (frames per second) for images with modest resolutions, e.g. 240×320, on a commercial desktop CPU [1]; sometimes a highly optimised GPU implementation is required to process images at typical video frame rates. Considering that separating foreground objects is clearly only the first task in an overall computer vision system, this speed disadvantage of many background subtraction algorithms limits their usage in many applications, especially those which operate with large amounts of data obtained by distributed cameras. The aim of our work is to design a background subtraction algorithm with state-of-the-art detection accuracy, but at the same time one that is computationally simple and efficient, making it capable of real-time performance on a variety of hardware platforms, especially those with limited computational power and on-board memory, e.g. smart cameras. The algorithm and the results of evaluating its performance on the ChangeDetection.net database 2014 [2] are presented in this paper.

2. Algorithm

The proposed algorithm, which is an extended version of the AMBER algorithm presented in [8], follows the general scheme of a pixel-based background detector [3-7], where a background model for each pixel consists of a number of background values, and a classification process is based on matching the background model templates with the current pixel values. Our main innovation is the introduction of a simple yet effective scheme that allows us to perform relatively robust classifications using only a small set of adaptive templates (in contrast to [6, 7] that use a large number of samples that model underlying distribution of background values). An easily calculated estimate of the template "efficacy" allows us to discard the least useful templates (rather than automatically discarding the oldest [4] or randomly chosen [6, 7] ones), and replace them with new ones. The template ordering process, based on efficacy, facilitates the operation on multiple time scales. Furthermore, we introduce simple yet effective control processes to eliminate noise and adjust detection thresholds, based on an estimate of classification uncertainty.

The proposed algorithm is designed to exploit pixel-level parallelism. It is suitable both for general-purpose software implementation, and for implementing on a variety of hardware platforms, from GPUs and DSP vector processors, to specialized parallel processors and FPGA-based hardware accelerators on board embedded camera systems. The computations described below are carried out independently for each pixel of a video frame (unless indicated otherwise, for global parameters).

2.1. Background Model and Detection Process

The background model of each pixel is composed by K+1 templates: a "long-term" template (T_0) and K "short-term" templates ($T_1, T_2, T_3 \ldots T_K$). Each template (T_k)

contains a background value B_k and an efficacy counter C_k. The background templates are ordered by the model maintenance process so that the efficacy (as indicated by the corresponding C_k value) decreases from T_0 to T_K. Template T_0 always contains the background value that has been present at the corresponding pixel for the longest time. Templates $T_1, T_2 \ldots T_K$ store K different background values that are likely to appear at that particular pixel on shorter time scales. At a particular time some of the K templates may be inactive (denoted by $C_k = 0$).

2.2. Classification and Adaptation

a) Classification: The detection process compares the pixels in the current frame and the background values stored in active templates. The process starts from template T_0 (i.e. the most effective template), and progresses though the other K templates in sequence. Initially, all pixels are considered as foreground; when the pixel is matching any template T_k (judging with the decision threshold ε) this pixel will be marked as background.

b) Template Update: To adapt to gradual background changes, if a pixel is classified as a background pixel by a template, the corresponding background value B_k will be updated towards the current pixel value using a running average function with a learning rate α. For each template, its efficacy counter C_k is updated as well. The efficacy of the first template that classified the pixel as background pixel will be increased by one, while the efficacies of other templates will be decreased by one. 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.

c) Low-resolution detection: To enhance the robustness against minor high-frequency changes in the environment, a low-resolution detection process is also implemented. The current frame and the templates (only T_0 and T_1) are down-sampled by averaging array values in a $N \times N$ block. The detection rules used in lower resolution detection are the same as used in full resolution detection, except that B_k and C_k are not updated during detection process. The detection result generated by low-resolution detection is then up sampled and combined with the full resolution detection result, so that only pixels marked as foreground in both detection results are considered the foreground.

2.3. Background Model Maintenance

Using the updating process for the background value as described above provides the ability for the proposed algorithm to adapt to gradual background changes. However, to provide the ability to adapt to sudden background changes, and to eliminate permanent misclassification errors (e.g. ghosts), the algorithm needs to have the ability to insert new background values into the background model, and remove unused ones. In this proposed algorithm, the selection of templates to be replaced is based on their efficacies.

a) Template Ordering: This procedure is used to order background model's templates according to their efficacy, and ensure the most effective background value is always stored in T_0 and the least effective background value is always stored in T_K. A straightforward bubble sort process is used to order the templates T_1 to T_K. To ensure the quality of the background values contained in the long-term template T_0, a more strict update policy is used. 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 template T_1 is swapped with T_0, and C_0 is set to $\gamma \times \theta_L$. The newly updated background value will stay in the long-term template for a time defined by the parameter γ, even though the updated background value is absent at that particular pixel. A relatively large γ will restrain the generation of ghosts.

b) Template Replacement: This process is used to selectively include new background values into the model (e.g. to adapt to long-term changes in the scene). It locates the foreground pixels that have not changed for a relatively long time, and selectively replaces the background value stored in T_K (which contains the least effective background value for each pixel) with a new qualified background value. To realize this function, the replacement process utilizes an additional accumulation template, which contains a potential background value B_A and an efficacy counter C_A. When a new foreground pixel is detected, the distance between this pixel value and B_A is calculated, and if this distance is smaller than the decision threshold ε then C_A is increased by 1, otherwise C_A is decreased by 1. For the pixels with $C_A = 0$, if the current frame pixel has been classified as foreground, its value will be loaded into B_A and C_A will be set to 1. If C_A is greater than a preset threshold θ_A, the value stored in B_A is considered to be a potential new background value. If it was directly incorporated into the background model, all the long-term changes in the frame would eventually be adapted to by the

Table.1 Parameters of the proposed algorithm used in the evaluation part

Sections	Parameters		
2.1	K=3		
2.2 a)	$\varepsilon^Y \in [18,80]$	$\varepsilon^C \in [8,12]$	
2.2 b)	$\alpha = 0.01$	$L_0 = 6000$	$L_1 = 4000$
2.2 c)	N=4		
2.3 a)	$\theta_L = 2500$	$\gamma = 3$	
2.3 b)	$\theta_A = 200$		
2.4 a)	$\beta_{MAX} = 80$	$A_{INC} = 6$	$\beta_{TH} = 70$
2.4 b)	$\beta_{INC} = 50$	$\beta_{DEC} = 20$	$\delta_{INC}^{Cb} = 1$
	$\delta_{DEC}^{Cb} = 0.125$	$\delta_{INC}^{Y} = 20$	$\delta_{DEC}^{Y} = 0.125$
	$\delta_{INC}^{Cr} = 1$	$\delta_{DEC}^{Cr} = 0.05$	

background model, which may be unacceptable in some applications. To preserve temporarily static foreground objects, a selective process is implemented here. The potential background value with the efficacy counter $C_A > \theta_A$ will be inserted into the background model only if the value of that pixel B_A is already contained in at least one of its neighbors' background model, judging by the decision threshold ε.

2.4. Global Control

The global control process introduced here provides the proposed algorithm with two main functions: activity control and pixel level decision threshold adaptation. These two functions are realized by utilizing an uncertainty measure (referred to as activity level) at each pixel.

a) Activity Control: This process is introduced to suppress false positives in heavily noisy environments. Comparing with the "blinking" detection process described in [7], which records the activity (i.e. changing frequency between background and foreground) of all the pixels on the frame and deletes pixels with high activity from the detection results, the proposed process only monitors a subset of pixels, identified as noise-pixels, to reduce false negatives. Here, noise-pixels are defined as the pixels that have been marked as foreground during the full resolution detection process, however, after low resolution detection, they have been changed to background. If a noise-pixel appears or disappears (i.e. blinks) between successive frames, the activity level A of this pixel increases by A_{INC}, while A decreases by 1 for non-blinking pixels. Activity level is limited in a range between 0 and β_{MAX} and pixels with activity greater than β_{TH} are eliminated from the detection result. In this way, continuously blinking noise-pixels will be eliminated from the detection results, preventing the generation of false positives.

b) Pixel-level Decision Threshold Adaptation: The decision threshold ε is the parameter that has the most direct effect on the quality of the detection results. A self-tuning pixel level decision threshold (based on the noise level at each pixel) may help to generate more accurate detection results, however, this benefit comes at a cost of extra memory requirements and increased computational complexity. Instead of building an additional model for determining the decision threshold, the proposed pixel-level decision threshold adaptation process utilizes the already calculated activity level A of each pixel (since it is a measure of noise level). After the pixel-level decision threshold ε is initialized, it changes according to the following rules: if A is bigger than a threshold β_{INC}then ε increases by δ_{INC}; if A is smaller than a threshold β_{DEC} then ε decreases by δ_{DEC}. To avoid ε oscillating between two values, β_{INC} is set to a much larger value than β_{DEC}. The adaptation range of ε is limited between ε_{MIN} and ε_{MAX}.

2.5. Parameters

In this paper, pixel values are represented as three dimensional vectors in the YC_bC_r color space. Corresponding parameters (thresholds, limits) are also three dimensional vectors. A single set of parameters, as shown in Table.1, is used in all experiments. This set of parameters has been determined using a combination of heuristics and optimisations on a dataset, and is producing accurate detection results in various scenarios. As many parameters are related to the dynamics of adaptation, the algorithm's performance is not particularly sensitive to the exact values of these parameters.

3. Results

The proposed algorithm was evaluated using the 2014 ChangeDetection.net database and methodology [2]. This database, containing 53 video sequences (in 11 categories), covers various challenges in background detection scenarios, e.g. shadows, ghosts, background changes, camera jitter and movement in both outdoor and indoor environments. Seven measurements, including Recall, Specificity, False Positive Rate, False Negative Rate, Percentage of Wrong Classifications (PWC), F-Measure (also referred as F-1) and precision, are used in [2] to determine comprehensive evaluations on different features of each algorithm. Two final benchmarks, *average ranking* and *average ranking across categories*, are used to give final ranks according to each algorithm's performance.

Detailed evaluation results of the proposed algorithm for each category of the 2014 database are included in Table.2. Some detection results obtained by the algorithm are shown in Fig.1.

4. Discussion

The evaluation results indicate that the proposed algorithm gives a relatively even performance for all the categories of the database. Generally speaking, the noisy regions of the input frames are clean in the detection results, and the detected objects are very close to the objects in the ground truth.

Comparing to other categories in the evaluation database, the proposed algorithm is less effective for category "PTZ". That's mainly because we make an assumption of a static camera. Due to the considerations of increasing processing speed and maintaining pixel-level parallelism (avoiding long-distance communications between pixels), the proposed algorithm is not suitable for camera pan/tilt/zoom scenarios. This is consistent with our main motivation, to design a simple, fast and efficient background subtraction algorithm with state-of-the-art performance for all platforms, especially these platforms with limited computational power and memory.

Average performance of the proposed algorithm

Table.2 Detailed evaluation results of the proposed algorithm for each category of the evaluation dataset

Scenarios	Recall	Specificity	FPR	FNR	PWC	F-1	Precision
PTZ	0.5162	0.8808	0.1192	0.4838	12.4397	0.1575	0.2085
Bad Weather	0.6782	0.9989	0.0011	0.3218	0.6379	0.7698	0.9010
Baseline	0.8784	0.9973	0.0027	0.1216	0.9233	0.8813	0.8980
Camera Jitter	0.6505	0.9938	0.0062	0.3495	1.9125	0.7107	0.8493
Dynamic Background	0.9177	0.9956	0.0044	0.0823	0.4837	0.8436	0.7990
Intermittent Object Motion	0.7617	0.9866	0.0134	0.2383	2.7784	0.7211	0.7530
Low Frame Rate	0.4727	0.9935	0.0065	0.5273	2.5607	0.4338	0.5943
Night Vision	0.6498	0.9469	0.0531	0.3502	5.9872	0.3593	0.3150
Shadow	0.8298	0.9914	0.0086	0.1703	1.7537	0.8128	0.8098
Thermal	0.7071	0.9940	0.0061	0.2929	1.6264	0.7597	0.8514
Turbulence	0.7508	0.9997	0.0003	0.2492	0.1936	0.8175	0.9018

Table.3 Average performance comparisons of 7 background subtraction algorithms

Algorithms	Recall	Specificity	FPR	FNR	PWC	F-1	Precision
Proposed Algorithm	**0.7103**	**0.9799**	**0.0201**	**0.2897**	**2.8450**	**0.6606**	**0.7165**
KNN	0.6650	0.9802	0.0198	0.3350	3.3200	0.5937	0.6788
GMM|Stauffer&Grimson	0.6846	0.9750	0.0250	0.3154	3.7667	0.5707	0.6025
Mahalanobis distance	0.1644	0.9931	0.0069	0.8356	3.4750	0.2267	0.7403
KDE-ElGammal	0.7375	0.9519	0.0481	0.2625	5.6262	0.5688	0.5811
GMM|Zivkovic	0.6604	0.9725	0.0275	0.3396	3.9953	0.5566	0.5973
Euclidean Distance	0.6803	0.9449	0.0551	0.3197	6.6542	0.5161	0.5480

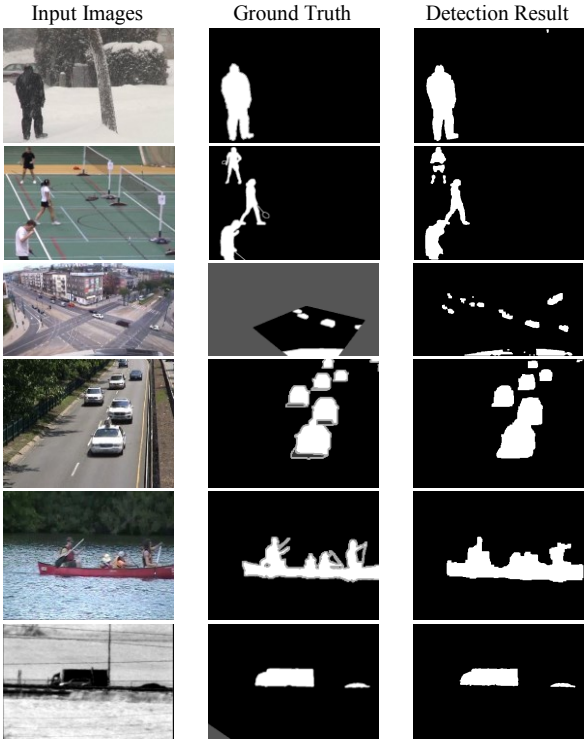

Fig.1 Detection Results generated by the proposed algorithm

comparing with six other algorithms is shown in Table.3. It can be seen that the proposed algorithm outperforms the other algorithms in most measurements, especially in PWC (percentage of wrong classification) and F-1 (F-Measure) which offer overall evaluations of the detection results.

References

[1] P. M. Jodin, E. Porikli, J. Konrad and P. Ishwar, ChangeDetection.net Video Database 2012 and evaluation result, (28/03/2014), Internet: www.changedetection.net;

[2] N. Goyette, P. M. Jodoin, F. Porikli, J. Konrad and P. Ishwar, "Changedetection.net: A New Change Detection Benchmark Dataset", in Proc. IEEE Workshop on Change detection (CDW-2012) at CVPR-2012, Providence, RI, 16-21 Jun., 2012;

[3] C. Stauffer and W. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Jun.23-25, 1999;

[4] A. Elgammal, D. Harwood and L. Davis. "Non-parametric model for background subtraction." Computer Vision—ECCV 2000. Springer Berlin Heidelberg, 2000. 751-767;

[5] K. Kim, T. H. Chalidabhongse, D. Harwood and L. Davis, "Real-time Foreground Background Segmentation using Codebook Model", Real-Time Imaging, Volume 11, Issue 3, pp. 167-256, June 2005;

[6] O. Barnich and M. Van Droogenbroeck, "Vibe: A Universal Background Subtraction Algorithm for Video Sequences", IEEE Transactions on Image Processing, 20(6):1709-1724, June, 2011;

[7] M.Hofmann, P. Tiefenbacher and G. Rigoll, "Background Segmentation with Feedback: The Pixel-based Adaptive Segmenter", Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on. IEEE, 2012;

[8] B. Wang and P. Dudek, "AMBER: Adapting Multi-Resolution Background Extractor", IEEE Internatinoal Conference on Image Processing, ICIP 2013, Melbourne, Australia;