IPSJ Transactions on Computer Vision and Applications Vol.7 31–34 (Feb. 2015)

[DOI: 10.2197/ipsjtcva.7.31]

Express Paper

Block-Propagative Background Subtraction System for UHDTV Videos

Axel Beaugendre[1,a)] Satoshi Goto[1]

Received: July 28, 2014, Accepted: November 20, 2014, Released: February 16, 2015

Abstract: The process of Ultra High Definition TV videos requires a lot of resources in terms of memory and computation time. In this paper we consider a block-propagation background subtraction (BPBGS) method which spreads to neighboring blocks if a part of an object is detected on the borders of the current block. This allows us to avoid processing unnecessary areas which do not contain any object thus saving memory and computational time. The results show that our method is particularly efficient in sequences where objects occupy a small portion of the scene despite the fact that there are a lot of background movements. At same scale our BPBGS performs much faster than the state-of-art methods for a similar detection quality.

Keywords: block propagation, UHDTV, object detection, foreground, BPBGS

1. Introduction

Foreground detection is a key task for multiple computer vision systems as it provides valuable information about objects' position and shape in the scene. The two major applications which can benefit from it are in the fields of augmented reality systems and in the field of video surveillance for object tracking, behavior recognition, person counting, etc. This subject has been studied for the last thirty years, however the resolution of the video used for testing algorithms has been confined to standard definition (SD) of 640×480 pixels at most. With the increase of mobile devices and the appearance of Ultra High Definition TV (UHDTV), there is a need for more efficient algorithms able to deal with very big video sources at a acceptable computing cost. Indeed, the 8K UHDTV [3] which has a definition of 7680×4320 pixels, will make its start in 2020 and will require adapted detection systems. In a recent synthesis work of Sobral [5], almost thirty background subtraction (BGS) systems have been compared on their speed, memory consumption and detection quality. Some of the most noticeable algorithms are the Pixel-Based Adaptive Segmenter (PBAS) [1] introduced by Hoffmann et al. in 2012, the Multi-layer BGS [8] of Yao and Odobez in 2007, the Gaussian average [7] of Wren et al. in 1997 and the Mixture of Gaussian [2] proposed by Kaewtrakulpong and Bowden in 2001. Three out of four of those algorithms are statistical methods based on one or multiple Gaussians and also based on color and texture features. The PBAS distinguishes itself by being a non-parametric approach.

We propose here a Block-Propagative Background Subtraction (BPBGS) method which, starting from a seed-block, will propagate the detection to neighboring blocks in the cases where there are foreground pixels detected on one or multiple borders of the current block. The seeds are the positions of the detected objects (or blobs) from the previous frame. The major asset of this work is that it focuses only on areas where objects are most likely located and the areas without objects are not processed at all, thus saving a great amount of computational time and memory resources. This propagative approach allows us to deal with ultra high definition images at a low cost since only the blocks around the seeds will be processed. First we present our block-propagative background subtraction method in Section 2. Then we will compare our algorithm with some state-of-art works in Section 3. Finally conclusions are given in Section 4.

[1] Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, Fukuoka 808–0135, Japan

[a)] axel.beaugendre@fuji.waseda.jp

2. Block-Propagative Background Subtraction

2.1 Block Detection

In order to work reliably, a full frame object detection is required to obtain the first foreground objects (also called blobs) which will be used as starting seeds in the block-propagative mode. Later, this full detection will occur every Δ_r frames in order to potentially detect new objects and avoid to have too much discrepancies. The refreshing period Δ_r should be set in adequacy with the frame rate (*fmr*) of the video, a too small value might be unproductive as the state of blobs will be updated needlessly. Empirically we set the refreshing period to $\Delta_r = 2/3 \times fmr$.

For the rest of the frames, the blobs B_{t-1} obtained from the previous frames f_{t-1} will become seeds for the block propagation. The frame is virtually cut out into $\mathbf{W} \times \mathbf{H}$ blocks of equal dimensions $width \times height$. The size of the block has a big influence on the propagation. If the block is too big then we might process more than enough area and, on the contrary, if the blocks are too small then there is a chance to not detect some parts because of fragments which would be in a different block. For each

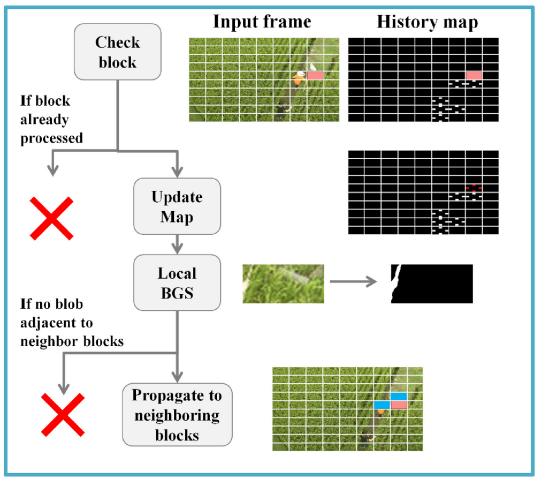


Fig. 1 Block detection flowchart. The current block iteration stops if the block has already been processed and it does not propagate further if no blob is on a border.

blob seed S of center (x_c, y_c), we compute the closest seed-block coordinates $B_s(x_b, y_b)$ following the next equation:

$$B_s(x_b, y_b) = \left\lfloor \frac{S(x_c, y_c)}{(width, height)} \right\rfloor (width, height). \quad (1)$$

For all blobs B_{t-1}, once the seed block coordinates are set, the local detection on the seed blocks begins. First the region $R(x_b, y_b)$ where the block is located needs to be checked. For that purpose, we create an history map which will save the blocks already processed. The map is an image of the same size as the input image and will be kept and updated during all the current frame process. One pixel on each border (top, bottom, right, left) is enough to determine if a block has been already processed. If it is the case then the detection process stops right there for this block. If on the contrary the block has not been processed yet, we update the history map with the current block and the detection can continue. The block is used as a region of interest which is first extracted from the input image. Then a classic static background subtraction is effectuated followed by morphological operations (open and close) and by a label of connected components. **Figure 1** presents the flowchart of the block detection.

2.2 Block-propagation

The main asset of the proposed method is its recursive aspect. Indeed, the next step is the propagation of the detection to neighboring blocks. If there are objects adjacent to one or multiple borders, then it means that it is very likely that the object is truncated and that the rest of the object is on the other side of the border. Depending on the situation, the block can propagate the detection in a maximum of four directions: top, bottom, right and left. For each direction D in which the block will propagate, the position of the next block $B^D(x, y)$ is calculated and the call for the next block detection is launched. The neighboring blocks' processes are parallelized to converge more quickly. As seen previously, the region of the block is verified through the history map before actually processing the block, therefore there cannot be multiple instances processing the same block at the same time. Also the history map is updated by all instances and it is protected during the updating process. An example of the propagation can be seen in **Fig. 2**. The detection expands little by little to neighboring blocks until there is no foreground pixel on any block border.

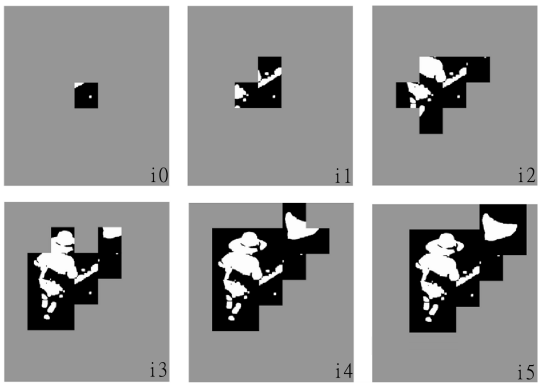


Fig. 2 Example of block-propagation iterations from the seed block i0 to the final result i5. Unprocessed areas are labeled in gray.

3. Experimental Results

We tested our method with an 8K UHDTV sequence from Ref. [3], the scene of the field. It is a 704 frames (12960–13664) long video in which children run through a rice field. The video is subject to a heavy background noise due to the crops' movements. The 8K ground truth frames are available at [*1]. In order to compare our work to state-of-art methods, we used the BGSLibrary [4]. We chose four methods among the best methods mentioned in Ref. [5]: the PBAS, the MultiLayerBGS, the DPWrenGABGS and the MixtureOfGaussianV1BGS.

The parameters used are as follow: $\Delta_r = 40$, $T = 30$, $(W_{270} = 16, H_{270} = 9)$, $(W_{1080} = 32, H_{1080} = 18)$, $(W_{4K} = 48, H_{4K} = 27)$, $(W_{8K} = 32, H_{8K} = 18)$. We developed in C++ with the OpenCV library [*2] and OpenMP 2.0 [*3], the computer used is a Quadcore i7@2.83 GHz with 4 Go of RAM. In order to compare the different methods, we evaluated the performance with the number of foreground pixels classified as foreground also called True Positive (TP), the number of background pixels classified as background or True Negative (TN), the number of False Positive (FP) which are background pixels classified as foreground and the number of foreground pixels classified as background or False Negative (FN). To measure the static quality metrics we computed different metrics: the Recall or detection rate which focuses on missed detection or false negative

$$\text{Recall} = \frac{TP}{TP + FN}, \quad (2)$$

the positive prediction or Precision (Pre.)

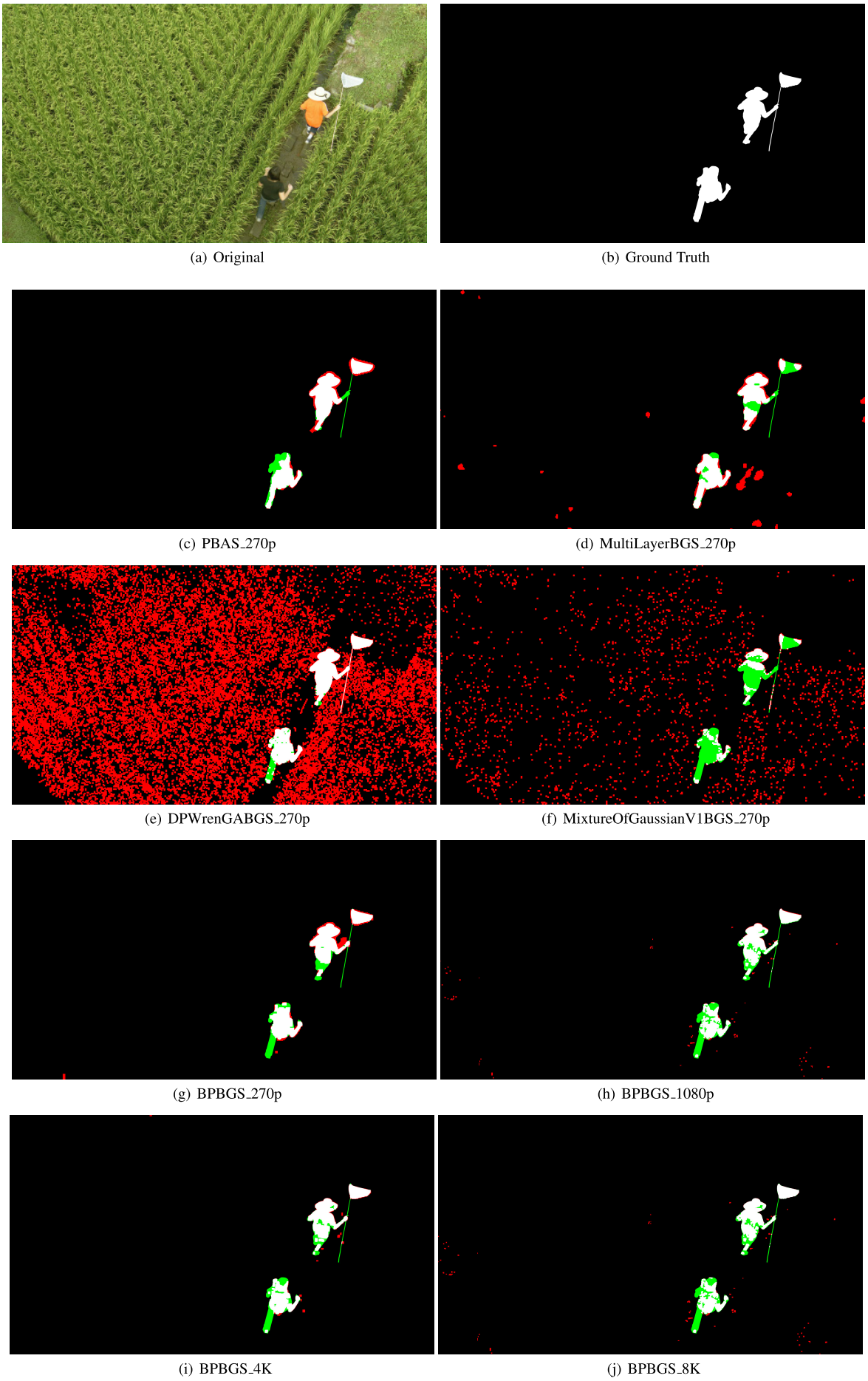
$$\text{Precision} = \frac{TP}{TP + FP} \quad (3)$$

which takes into account the background noise and incorrect detection or false positive. Moreover, the Similarity measure considers both incorrect and miss detections:

$$\text{Similarity} = \frac{TP}{TP + FN + FP}, \quad (4)$$

We measured another metric which uses the pixels metrics, the

[*1] https://sites.google.com/site/uhdtvcomputervision/downloads
[*2] http://opencv.org
[*3] http://www.openmp.org



(a) Original (b) Ground Truth

(c) PBAS_270p (d) MultiLayerBGS_270p

(e) DPWrenGABGS_270p (f) MixtureOfGaussianV1BGS_270p

(g) BPBGS_270p (h) BPBGS_1080p

(i) BPBGS_4K (j) BPBGS_8K

Fig. 3 Foreground mask obtained from the different BGS algorithms. Color map: TP-white, TN-black, FP-red, FN-green.

Table 1 Quality comparison of our BPBGS with Refs. [1], [2], [7], [8]. Best scores are in bold.

Method ID	TP	FN	FP	TN	Recall	Pre.	Sim.	F-score	PSNR	SSIM	fps
PBAS_270p [1]	429,020	64,518	797,432	31,886,628	0.705	0.680	0.594	**0.687**	20.929	0.976	2.44
MultiLayer_270p [8]	382,563	110,976	383,752	32,300,307	0.623	0.570	0.454	0.576	20.255	0.976	2.69
DPWrenGA_270p [7]	469,339	24,200	12,113,649	20,570,410	**0.776**	0.038	0.038	0.072	10.742	0.610	0.12
MOGV1_270p [2]	218,769	274,770	1,592,199	31,091,860	0.375	0.124	0.102	0.179	17.809	0.935	1.30
BPBGS_270p_16x9	385,351	108,188	66,027	32,618,032	0.638	0.693	0.560	0.660	23.627	0.993	**24.00**
BPBGS_2K_32x18	380,820	112,719	27,288	32,656,771	0.633	0.758	**0.600**	0.686	**25.964**	**0.994**	7.58
BPBGS_4K_48x27	314,695	178,843	13,831	32,670,229	0.534	**0.778**	0.519	0.627	24.856	0.992	5.69
BPBGS_8K_32x18	350,927	142,611	30,748	32,653,311	0.586	0.752	0.553	0.654	24.376	0.992	1.03

F-score which is the weighted harmonic mean of Precision and Recall is defined by

$$\text{F-score} = \frac{2 * Precision * Recall}{Precision + Recall}. \quad (5)$$

Additionally we compute the perceptual measure SSIM (Structural SIMilarity) [6]:

$$\text{SSIM(S,G)} = \frac{1}{n}\sum_{i=1}^{n}\frac{(2\mu_{S_i}\mu_{G_i} + c_1)(2cov_{S_iG_i} + c_2)}{(\mu_{S_i}^2 + \mu_{G_i}^2 + c_1)(\sigma_{S_i}^2 + \sigma_{G_i}^2 + c_2)}, \quad (6)$$

in which σ_{S_i}, σ_{G_i} are the standard deviations, μ_{S_i}, μ_{G_i} the means and $cov_{S_iG_i}$ the covariance. The different values used in the literature are $c_1 = (0.01 \times L)^2$ and $c_2 = (0.03 \times L)^2$ in which L is the dimension size and $L = 255$ for gray-scale images. Finally, the Peak Signal-Noise Ratio (PSNR) is calculated by:

$$\text{PSNR} = \frac{1}{n}\sum_{i=1}^{n} 10\log_{10}\frac{m}{\sum_{j=1}^{m}\|S_i(j) - G_i(j)\|^2}. \quad (7)$$

The computational speeds of the methods which include the label of connected components is measured in frames per second (fps).

Table 1 presents the comparison of our BPBGS to the state-of-art algorithms using the metrics we just introduced. In order to show the efficiency of our work we tested our method with multiple scales: 270 p, 1080 p, 4K and 8K. However, the other algorithms we tested on the sequence could not support scale above 270 p. Indeed, above this size the algorithms were either unable to process due to lack of memory or the processing time was unreasonable. For example the MultiLayerBGS was processing a single 1080 p frame in more than 15 min (0.001 fps). The sequence being very subject to background noise, the precision which takes into account the number of pixels incorrectly detected as foreground is very important. We can see in Table 1 that our proposals, which do not process the entire frame, are much less affected by the background noise. On the contrary, the Recall rate is lower for our BPBGS, especially for higher scales. The reason lies in the fact that some fragments might appear if the missing junction is on a border, meaning no foreground pixels on the border of a block, then the detection is not propagated. If we look at the F-score which takes into account both miss and incorrect detection rates, we can see that our BPBGS is head-to-head with the PBAS and above the MultiLayerBGS. We also obtained better results by the perceptual measure SSIM with all the different scales used. Last and most importantly, our proposals perform much faster than the state-of-art algorithms. Except for the original UHDTV scale of our BPBGS which runs at 1 fps, our proposals are much faster than the 270 p version of the other methods. We even reach real-time despite including the label process which is essential to applications such as tracking. **Figure 3** shows a visual comparison of the state-of-art methods and our BPBGS. Despite having some remaining background noise, our proposals of HD and UHDTV scales fit better the ground truth shape than the other works and detect the correct shape of the arm of the child on the right of the image (Fig. 3 (h)–3 (j)).

4. Conclusions

The block-propagative background subtraction method proposed in this paper is a fast and efficient way to detect foreground objects in high and ultra high definition videos. By recursively propagating the local block-detection to neighboring blocks, it is possible to avoid processing most of the unnecessary areas. The method obtains similar quality results compared to state-of-art background subtraction algorithms while being much faster than them. It can reach the speeds of 24 fps for the 270 p scale from the re-scale to the connected component label. For the process of the original 8K scale, the BPBGS performs at a speed of 1 fps, which is already faster than the speed of the best state-of-art algorithm on a 1080 p video. For the future work, we would like to extend this block-propagation method with more advanced and stronger background subtraction algorithms without losing too much speed in the process.

References

[1] Hoffmann, M., Tiefenbacher, P. and RIgoll, G.: Background Segmentation with feedback: The pixel-based adaptive segmenter, *Computer Vision and Pattern Recognition Workshops* (*CVPRW*), pp.38–43 (2012).

[2] Kaetrakulpong, P. and Bowden, R.: An improved adaptive background mixture model for realtime tracking with shadow detection, *European Workshop on Advance Video Based Surveillance Systems* (*AVSS*) (2001).

[3] Shishikui, Y., Fujita, Y. and Kubota, K.: Super Hi-Vision - the star of the show!, *EBU Technical review* (2009).

[4] Sobral, A.: BGSLibrary: An OpenCV C++ Background Subtraction Library, *IX Workshop de Visão Computacional* (*WVC'2013*), Rio de Janeiro, Brazil (2013).

[5] Sobral, A. and Vacavant, A.: A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos, *Computer Vision and Image Understanding*, Vol.122, No.0, pp.4–21 (2014).

[6] Wang, Z., Bovik, A., Sheikh, H. and Simoncelli, E.: Image quality assessment: From error visibility to structural similarity, *IEEE Trans. Image Process.*, Vol.13, No.4, pp.600–612 (2004).

[7] Wren, C., Azarbayejani, A., Darrell, T. and Pentland, A.: Pfinder: Real-time tracking of the human body, *IEEE Trans. Pattern Ana. Mach. Intell.*, Vol.19, No.7, pp.780–785 (1997).

[8] Yao, J. and Odobez, M.: Multi-layer background subtraction based on color and texture, *IEEE Computer Vision Recognition Conference* (*CVPR*) (2007).

(Communicated by *Hondong Li*)