

PIXEL-BASED FOREGROUND DETECTION IN REPETITIVE TIME-SERIES REGION

Wirat Rattanapitak¹, Somkiat Wangsiripitak²

^{1,2}Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok 10520
Thailand

E-mail: rattanapitak@it.kmitl.ac.th¹, somkiat@it.kmitl.ac.th²

DOI: <https://doi.org/10.22452/mjcs.sp2019no2.4>

ABSTRACT

Currently, many state-of-the-art background subtraction techniques cannot deal properly with the area of periodic changing background, while some continue classifying them as foreground at intervals, others simply mask that area as a non-region of interest. To cope with this issue, a novel method of detecting repetitive temporal patterns based on the image sequences was proposed in this paper. The main emphasis of the proposed approach is on classifying those pixels as a background and identifying foreground objects in their relevant areas. As for the foreground detection, a model of time series pattern found in each pixel is individually built first; and then, any changes beyond the allowance of model periodicity are then determined as foreground objects. The proposed method could be used and run in parallel with any state-of-the-art background subtraction technique, allowing more accurate foreground-background segmentation. Experimental results showed that using Y channel, the proposed method of detecting time-series background area could achieve 92.9% of recall rate with less than 1% false positives. The recall of foreground detection in an area of repetitive time-series pattern was about 87%; while F-measure was about 0.73 on average. The false positives of foreground detection were also less than 1%. Accordingly, the proposed time-delay detection technique could significantly help to suppress the foreground error on time series background area, especially during the change from one sub-pattern to another which causes a camera sensor to capture both sub-pattern values in one frame. Performance comparison with state-of-the-art methods showed that our proposed method was able to reduce 80% of the average false alarm and improve F-measure to 28% while the computational efficiency was reduced by only 1%.

Keywords: *subtraction techniques, change detection algorithms, periodic structures, pattern matching, time series analysis.*

1.0 INTRODUCTION

In the recent past, most video surveillance works needed manual monitoring by humans. Humans were needed to observe, analyse and decide on the observed event. However, humans may not be fully capable of handling all of the situations fast or well enough. Today, rapid and valid responses to some situations are of the highest priority; therefore, computer vision has been developed to augment human's efficiency, especially on works related to monitoring and security. The main objective of computer vision is to synthesize important data for further use by another process. For example, in the past, crime prevention work with the closed-circuit camera was done manually by human observers monitoring suspicious behaviour of the certain individual through a closed-circuit camera system. Nowadays, computer vision can filter all of the recorded video data down to only images of an individual with abnormal behaviour. In order to be able to do this, primarily computer vision has to be able to discriminate the differences between visual objects.

A video surveillance system sometimes encounters a problem of misclassification in foreground-background segmentation. It is more severe particularly in an urban area where a repetitive time-series pattern is visible in the scene; this periodic changing background is misclassified as a foreground object. A digital billboard is shown in Fig. 1 is one example of a time-series background.

Many methods have been proposed (e.g. [1, 2, 3, 4, 5]). Some of them (e.g. [6, 1, 2]) addresses a problem of dynamic backgrounds such as swaying trees, small waves on the surface of lakes, a water fountain—to the best of our knowledge, none has considered the issue of repetitive time-series pattern. We propose the method which identifies an area of time series pattern. The method of foreground segmentation in that area is also proposed in this paper. Our proposed methods will be detailed after some related works on background subtraction and time-series

detection are described in the next section. Experimental results and the performance comparison of the proposed method with state-of-the-art methods will be shown and discussed next. The conclusion is drawn at the end.

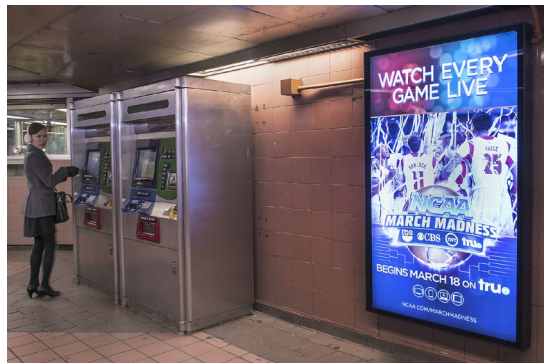


Fig. 1. An example of repetitive temporal pattern (digital advertising board) in daily life. (Metropolitan Transportation Authority of the State of New York, <https://www.mta.info>)

2.0 RELATED WORKS

Many background subtraction techniques that address a dynamic background problem have been proposed— details of those methods can be found in a review paper such as [7] and [8]. In [7], more than 300 papers were studied and categorized into two groups; one uses a traditional background model, the other uses a more recent background model. The key to success in foreground detection based on these background subtraction techniques is the model itself. Many researchers, therefore, dedicate their time and effort to robust background modelling.

One approach is to use a statistical model based on Gaussian distribution (e.g. [9] and [10]) to handle the background dynamics. However, a rapid change in the background makes this type of model far from ideal. A combination of Dirichlet mixture model and probabilistic regularization is therefore used (in [11]) so that dynamic backgrounds can be accurately modelled and continuously updated. A non-parametric approach has also been proposed to solve the problem of dynamic background. One of them, the background model based on local binary pattern (e.g. [6]), uses the texture around the pixel for background modelling. Its deficiencies are the detection problem in the foreground object with uniform intensity and the sensitivity to noise. Another non-parametric method is ViBe (visual background extractor) [1]. The random selection of previously observed pixel values for background modelling and the spatial propagation to its neighbour pixels make it more robust to dynamic backgrounds; shadows and frequent changes in the background are however unsolved. PBAS (pixel-based adaptive segmenter) [2] is also non-parametric. The background dynamics are continuously estimated for each pixel— gradual changes in the background are, consequently, properly handled.

Features and colours are other properties used in background model— [12] and [13] are examples based on local binary similarity pattern (LBSP). The use of color-LBSP representation in SuBSENSE (of [13]) helps detect subtle changes related to foreground objects and keep the irrelevant motion as the background. Some methods create more than one background model, e.g. a dual model in [5] and a multi-model in [3] and [4]. The former has two models; a self-model for the background of its pixel location, a neighbourhood-model for neighbour pixels around itself. The latter consists of long-term and short-term models; it uses a multi-resolution approach to remove noises. These multi-model methods employ many models for capturing background dynamics in the scene. The technique mentioned above is similar to that reported in [14]. That research work has developed a background subtraction technique that uses a set of multiple background images to construct a background model for detecting foreground objects in a dynamic background image. The developed technique uses data at the level of image space to resolve the issue of camera jitter which is often found in a video clip shot with a portable camera. Moreover, this work has also developed a method for reducing image noise based on colour space. The efficiency of this method is enhanced by using a hash-table lookup instead of a comparison of threshold values. Another similar technique is that reported in [15] which uses multiple BG models and megapixels to construct a background model. It improves the discrimination accuracy in identifying the foreground object from background image based on colour space. Namely, under a low light condition, identification is done on RGB and Y channels whereas it is done on Cr and Cr colour channels under a high brightness condition. Another study [16] proposes the use of difference discriminator for constructing the background model and Gaussian distribution for updating the threshold value for subtracting

foreground pixel from the background image. That study also proposes three heuristics for increasing the efficiency of model adaptability. To classify the area of repetitive temporal patterns as backgrounds, the recurrence of time series must be detected. Many methods have been proposed for temporal pattern retrieval (e.g. [17] and [18]), but none is applied to the detection of repeated temporal pattern in image sequences—the image region of those time series pattern is often misclassified as the foreground. Some methods of finding periodic movement patterns in spatio-temporal data, as surveyed in [19], are similar to our approach. However, they mainly focus on a trajectory of the object which is successively used for analysing behaviours of moving objects. The periodic pattern on electronic billboards, the main challenge of our proposed method, has no spatial movement; the pixel just changes its value from one to another and continues in a cyclic manner.

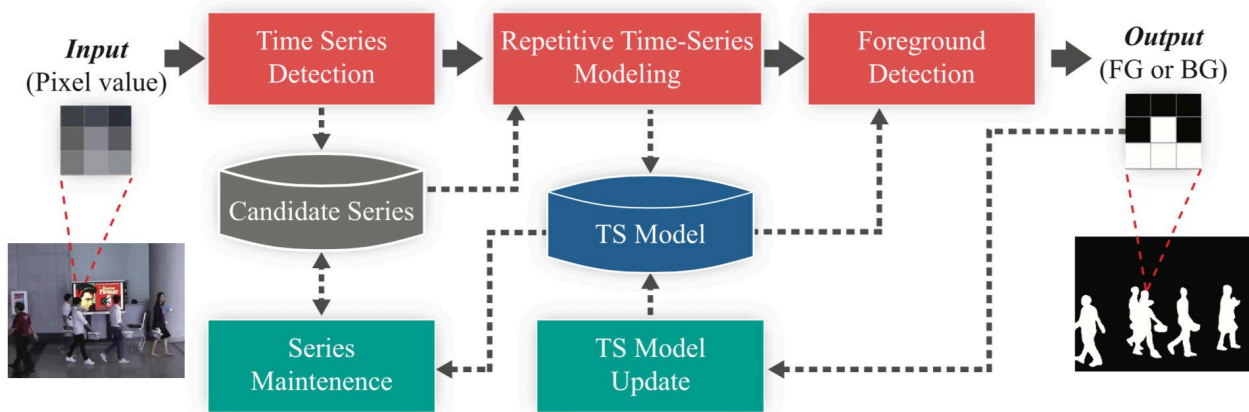


Fig. 2. An overview diagram of the proposed algorithm.

3.0 CHANGE DETECTION ON REPETITIVE TIME SERIES

Most previous works on background subtraction technique, although deal with some degrees of random changes in the scene, do not address the problem of repetitive time series appearing in some regions of image sequences. Such regions are, therefore, detected as foreground objects every time the next value of time series is observed, and remain as foregrounds for a while; they are then absorbed into the background model during the process of model updating, making them part of the background—this cycle is repeated every time the content displayed in those image areas changes from current value to the next value of the time series. This phenomenon makes the detection of the foreground on such image area infeasible. We proposed the method which is capable of detecting the repeated temporal pattern in image sequences and building the model—on a per-pixel basis—for such background region; the model is then used to find a foreground object which does not follow the pattern of periodic change, allowing the foreground detection in those image regions. An overview diagram of the proposed algorithm is shown in Fig. 2. Note that the proposed method can run individually on each pixel, and therefore it allows the implementation of per-pixel processing in separate threaded processes.

3.1 Notation

This section provides the notational conventions used throughout this paper. The value of pixel observed at location (i, j) at time t is expressed by $v_{i,j}(t)$ or v for short. The number of consecutive frames (denoted by p^h) in which the same value (or approximately the same value) is displayed at location (i, j) will be kept in the vector $\mathbf{P} = [p^v \ p^h]$ —we call this vector as ‘sub-pattern’ where p^v is an accumulated pixel value of the sub-pattern. Note that \mathbf{P}_k denotes the k^{th} sub-pattern. If a series of sub-patterns $[\mathbf{P}_0 \ \mathbf{P}_1 \dots \ \mathbf{P}_{n-1}]$ is observed repetitively and successively at the same location (i, j) , it establishes a ‘repetitive time-series pattern’ or ‘TS pattern’ for short. The TS pattern is represented by vector $\mathbf{S} = [\mathbf{P}_0 \ \mathbf{P}_1 \dots \ \mathbf{P}_{n-1}]$. The region of an image in which the TS pattern is observed is called ‘TS region’—we propose the method of building the background model for each pixel in TS region; this model is called ‘TS model’ for short. Table 1 summarizes most of the notational conventions used throughout this paper.

Table 1. Notations used in this paper. They are per pixel, unless stated otherwise.

Symbol	Explanation
$v_{i,j}(t)$ or v	Current value of pixel (i, j) observed at time (t)
$\mathbf{P}_k = [p_k^v p_k^h]$	k^{th} sub-pattern vector
p_k^v	Accumulated pixel value of k^{th} sub-pattern
p_k^h	The number of successive frames by which p_k^v is accumulated
$\mathbf{S} = [\mathbf{P}_0 \mathbf{P}_1 \dots \mathbf{P}_{n-1}]$	Repetitive time series pattern (one cycle) with n sub-patterns
$\mathbf{P}'_k = [p_k^{v'} p_k^{h'}]$	k^{th} candidate sub-pattern vector
$p_k^{v'}$	Accumulated pixel value of k^{th} candidate sub-pattern
$p_k^{h'}$	The number of successive frames by which $p_k^{v'}$ is accumulated
$\mathbf{S}' = [\mathbf{P}'_0 \mathbf{P}'_1 \dots \mathbf{P}'_{c-1}]$	List of c candidate sub-patterns (if TS pattern exists, it will be detected when $c = 2n+1$)
ε^v	Tolerance for pixel value similarity
ε^h	Minimum length of candidate sub-pattern that will not be merged (or ignored)
ε^c	Allowance for length difference of two candidate sub-patterns being compared

3.2 Preprocessing

In order to take into account, the pixel values of neighbourhoods, we apply the averaging filter of size 3×3 to all image frames before starting the detection of repetitive time series pattern in each pixel. This preprocessing helps improve the performance of time-series detection, as will be shown in experimental results.

3.3 Detecting a Repetitive Time Series Pattern

We proposed a method of detecting the repetitive time series pattern in each image pixel so that those pixels could be classified as a background—as a result, foreground detection becomes feasible in the area of those pixels. The detection of TS pattern is the first part shown on the left of Fig. 2. Algorithm 1 and Fig. 3 summarize the detection process which is run concurrently on each pixel; details of the method are as follows.

3.3.1 Building a list of candidate sub-patterns

As to each pixel location in image sequences, e.g. at the pixel location (i, j) , the historical values v of a pixel will be examined; a new candidate sub-pattern vector $\mathbf{P}' = ([p^{v'} p^{h'}])$ is successively built for each new value of v which significantly differs from previous value (or the current sub-pattern vector is updated if v is approximately equal to the previous value)— $p^{h'}$ is the number of consecutive frames in which the same (or approximately the same) pixel value is observed, and $p^{v'}$ is the accumulated value of those pixel values in $p^{h'}$ frames. As a result, the list of these candidates is constructed and incrementally expanded; it is denoted by $\mathbf{S}' = [\mathbf{P}'_0 \mathbf{P}'_1 \dots \mathbf{P}'_{c-1}]$. Lines 5–10 of Algorithm 1 are the code snippets which gradually build up the list of candidate sub-patterns. More details of them are as follows.

When the next value v of pixel at location (i, j) is obtained (line 4), its value is compared to the last candidate sub-pattern \mathbf{P}'_{c-1} , i.e.

$$\left| v - \frac{p_{c-1}^{v'}}{p_{c-1}^{h'}} \right| < \varepsilon^v, \quad (1)$$

here, $p_{c-1}^{v'} / p_{c-1}^{h'}$ is the averaged pixel value of the last sub-pattern (line 5). If the difference is less than a predefined similarity threshold ε^v , the last candidate sub-pattern is updated—its accumulated value $p_{c-1}^{v'}$ is added by v and the number of frames $p_{c-1}^{h'}$ is incremented by 1 (line 6),

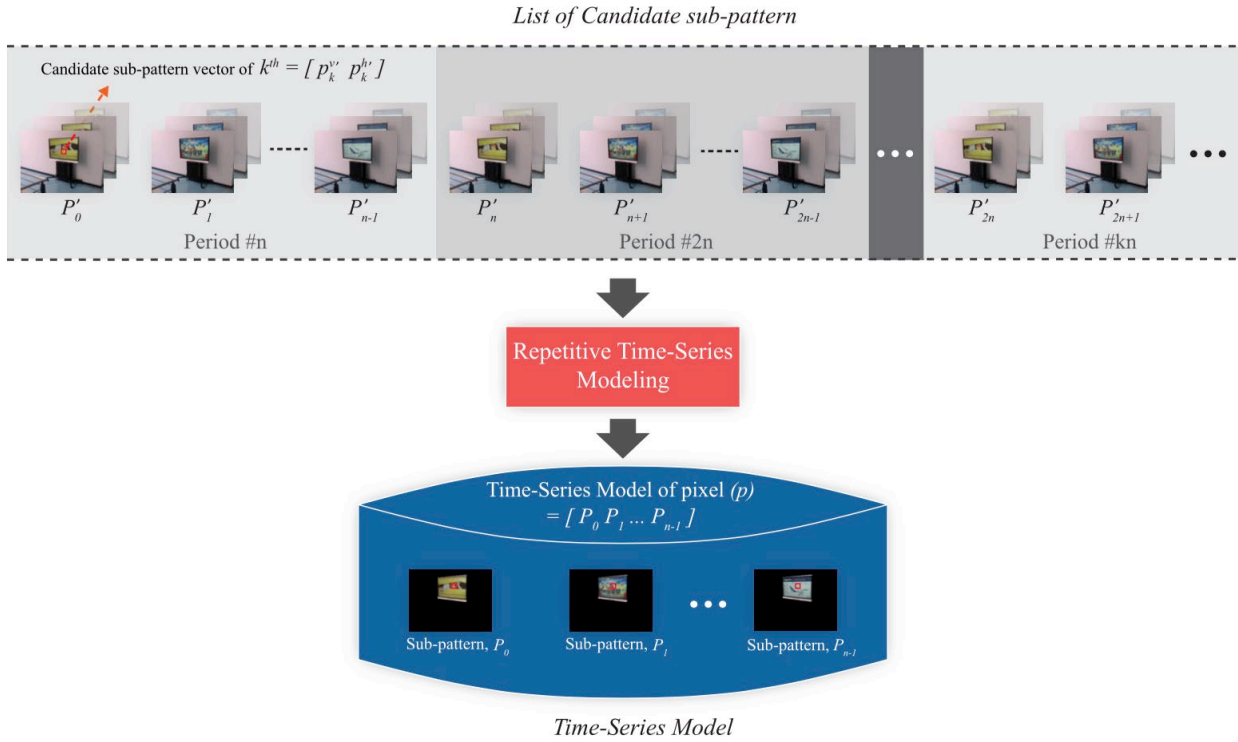


Fig. 3. Detection of repetitive time series pattern.

$$(p_{c-1}^{v'}) += v \text{ and } (p_{c-1}^{h'}) += 1. \tag{2}$$

In case that the incoming v value differs greatly from previous candidate sub-pattern, i.e.(1) is false, it is assumed that a new sub-pattern starts to display itself--new *candidate* sub-pattern $\mathbf{P}' = [p^{v'} p^{h'}]$ (where $p^{v'} = v$ and $p^{h'} = 1$) is constructed and added at the end of candidate list \mathbf{S}' (line 8). As a result, this new sub-pattern becomes the last candidate sub-pattern in the list; and the number of sub-patterns in the list (denoted by c) is incremented by 1.

3.3.2 Merging short candidate sub-patterns

Some candidate sub-patterns may have a very small value of consecutive hit count $p^{h'}$ —it happens when the period of those sub-patterns is actually short or there is a short sudden change of pixel value because of illumination shift and/or instability associated to image sensors, etc. These short candidates are not desirable because they will cause an error in detection of TS pattern. Here, they are merged into either left or right candidate sub-pattern depending on which one has more similar pixel value (line 9).

For instance, if the candidate sub-pattern \mathbf{P}'_q appears for a very short period—i.e. $p_q^{h'} < \epsilon^h$ where ϵ^h is the minimum length of *candidate* sub-pattern that will not be merged—it will be merged into the adjacent candidate sub-pattern of which averaged pixel value is more similar to its pixel value,

$$\mathbf{P}'_q += \begin{cases} \mathbf{P}'_{q-1}, & \text{if } d'_{q-1} \leq d'_{q+1} \\ \mathbf{P}'_{q+1}, & \text{otherwise,} \end{cases} \tag{3}$$

where

$$d'_a = \left| \left(\frac{p_a^{v'}}{p_a^{h'}} \right) - \left(\frac{p_q^{v'}}{p_q^{h'}} \right) \right|. \tag{4}$$

3.3.3 Finding a periodic repetition

While the list of *candidate* sub-patterns \mathbf{S}' is being built and successively grown, it is continually examined in order to find if a series of sub-patterns is periodically repeated (i.e. whether or not the repetitive temporal pattern occurs)—this is performed concurrently for each pixel location. We implement the process of verifying a periodic repetition of *candidate* sub-patterns in the same manner as searching for two identical sub-strings in the searched string. Our verification method is, however, different in many aspects, as follows.

The verification of repetitive TS pattern is an incremental process which attempts to continually check each incoming pixel value in order to find if there is a series of candidate sub-patterns repeated—the number of repeated sub-patterns are gradually increased as they are incrementally found, allowing the repetitive TS pattern to be found as quickly as possible.

This verification process, first, attempts to find the first repeated pixel value. It starts when the incoming pixel value v_i is determined to be (the first value of) the new *candidate* sub-pattern. It will find the previously created candidate sub-pattern \mathbf{P}'_k of which averaged pixel value ($p^{v'}/p^{h'}$) is the same or approximately the same as this pixel value v_i , i.e. searching for \mathbf{P}'_k that satisfies

$$\arg \min_{\mathbf{P}'_k} \left| v_i - \frac{p^{v'}}{p^{h'}} \right| < \varepsilon^v, \quad (5)$$

where ε^v is a similarity tolerance (the same as used in (1)). If the pixel value v_i is a repeat of the previous *candidate* sub-pattern \mathbf{P}'_k (i.e. the candidate sub-pattern that satisfies (5) is found); the hit count h_k is assigned a value of 1. In that case, the process successively checks each next value of that pixel—as soon as it appears in successive frame—against the averaged pixel value of matched sub-pattern (here, \mathbf{P}'_k); the hit count h_k is incremented by 1 every time they are successfully and continuously matched.

During the ongoing process of confirming the repetition in the same sub-pattern \mathbf{P}'_k , if the hit count h_k is approximately equal to the number of frames in corresponding sub-pattern \mathbf{P}'_k , i.e.

$$|h_k - p_k^{h'}| < \varepsilon^c, \quad (6)$$

the incoming pixel value $v_i (j > i)$ will be compared to the averaged pixel value of both current candidate sub-patterns \mathbf{P}'_k and next candidate \mathbf{P}'_{k+1} (here ε^c is the length difference allowance of sub-pattern \mathbf{P}'_k and the current sub-pattern (of v) which are being compared). Successfully matching with either one of both sub-patterns, i.e.

$$\left| v_j - \frac{p_k^{v'}}{p_k^{h'}} \right| < \varepsilon^c \text{ or } \left| v_j - \frac{p_{k+1}^{v'}}{p_{k+1}^{h'}} \right| < \varepsilon^c, \quad (7)$$

allows the process to continue verifying the ongoing repetition.

In case that the incoming pixel value v_j is successfully matched with the next candidate \mathbf{P}'_{k+1} (i.e. the right of (7) is true), it means that the next pixel value v_{j+1} and hereafter must be compared to the averaged pixel value of sub-pattern \mathbf{P}'_{k+1} . The new hit count h_{k+1} becomes active and is used for verifying the repetition of the candidate sub-pattern \mathbf{P}'_{k+1} —the value of h_{k+1} is initialized to be 1. The same ‘incremental verification’ process is performed against the active sub-pattern \mathbf{P}'_{k+1} .

The repetitive time series pattern is assumed to appear on the man-made device such as electronic billboard; the time series pattern, denoted by $\mathbf{S} = [\mathbf{P}_0 \mathbf{P}_1 \dots \mathbf{P}_{n-1}]$, will ordinarily repeat itself in a consecutive manner. Therefore the confirmation of repetitive TS pattern has to repeat the above-detailed verification—which is executed for each subsequent candidate sub-pattern—until the repeated series of *candidate* sub-patterns in the list \mathbf{S}' is found and the second series must immediately follow the first series that it repeat. For example, under the assumption that

- the list of c candidate sub-patterns being built and grown is $\mathbf{S}' = [\mathbf{P}'_0 \mathbf{P}'_1 \dots \mathbf{P}'_{b-1} \mathbf{P}'_b \mathbf{P}'_{b+1} \dots \mathbf{P}'_{c-2} \mathbf{P}'_{c-1}]$,
- there exists the series of candidate sub-patterns in the list \mathbf{S}' that is repeated one after the other,
- the first sub-pattern that is found repeated is \mathbf{P}'_0 ,
- its counterpart which repeats \mathbf{P}'_0 is \mathbf{P}'_b ,

- the next matched pair is \mathbf{P}'_1 and \mathbf{P}'_{b+1} and so on,

the last sub-pattern in the series of TS pattern must be \mathbf{P}'_{b-1} —in order to have the repeated TS pattern immediately follow each other. In this case, $[\mathbf{P}'_0 \mathbf{P}'_1 \dots \mathbf{P}'_{b-1}]$, becomes the candidate of repetitive TS pattern.

The candidate TS pattern will be established after the counterpart of \mathbf{P}'_{b-1} is successfully identified—normally at one image frame after the end of prospective counterpart which is denoted by sub-pattern \mathbf{P}'_{c-2} . We must check the validity of candidate TS pattern by verifying if the length of last sub-pattern \mathbf{P}'_{b-1} is approximately equal to that of \mathbf{P}'_{c-2} . It is done by detecting the next sub-pattern \mathbf{P}'_{c-1} , which help confirm the repeat of last sub-pattern in the candidate TS pattern. That is why the whole candidate TS pattern will be detected—if it exists—when the list \mathbf{S}' has $c = 2n + 1$ candidate sub-patterns (where n is the number of sub-patterns in the TS pattern). Note that line 11 of Algorithm 1 embeds all process explained above.

It is worthwhile to mention that the merging of short candidate sub-patterns, however, is not to be done during the process of finding a periodic repetition (matching incoming pixel values to previous sub-patterns); this is to avoid any delay in the detection of TS pattern. Here, we just ignore them (the short candidate sub-pattern) if it is too short (its period is less than ε^h). In other words, when the discrepancy between the incoming pixel value and the averaged pixel value of corresponding sub-pattern is found, we will not immediately cancel the ongoing verification. The process allows some matching failures to occur consecutively; if the incoming pixel values do not repeat their corresponding sub-pattern for ε^h consecutive frames, the ongoing verification will be cancelled. However, the next verification will be restarted promptly. It will try finding another first pair of sub-patterns and its repeated pixel value; and resuming the subsequent verification if the new first match was found. This iterative process will continue until the TS pattern is found or it runs out of time (i.e. the verification has been done for more than T_{ts} frames) (line 12).

3.4 Building a Repetitive Time Series Background Model

The repetitive time series background model (TS model) is built for each pixel location if the ‘candidate’ TS pattern was found. The model is initially built by averaging the ‘candidate’ TS pattern (e.g. $[\mathbf{P}'_0 \mathbf{P}'_1 \dots \mathbf{P}'_{n-1}]$) and the list of sub-patterns (e.g. $[\mathbf{P}'_n \mathbf{P}'_{n+1} \dots \mathbf{P}'_{2n-1}]$) that repeats the ‘candidate’ TS pattern. Therefore, the repetitive TS background model is

$$\mathbf{S} = [\mathbf{P}_0 \mathbf{P}_1 \dots \mathbf{P}_{n-1}], \quad (8)$$

where

$$\mathbf{P}_i = \frac{\mathbf{P}'_i + \mathbf{P}'_{n+i}}{2}. \quad (9)$$

Note that the accumulated (background) pixel value p_k^v and the number of accumulate frames p_k^h of each sub-pattern \mathbf{P}_i in TS model \mathbf{S} are updated by weighted averaging with the next cycle which successfully matches with the model (i.e. no foreground is detected).

3.5 Detecting a foreground on the region of TS pattern

The foreground detector proposed in this paper is responsible for detecting a foreground on the region of repetitive time series pattern. Based on elapsed time from the last known sub-pattern, it predicts which sub-pattern of TS model will appear on each pixel of time series region; then compares the value of each pixel to the averaged pixel value of corresponding sub-pattern in TS model. The same principle used in ‘finding a periodic repetition’ (explained previously) is utilized in the foreground detection. The pixel of which value is judged to be approximately the same as its corresponding sub-pattern—by using the left equation of (7) (and the right equation if necessary)—will be classified as the background (strictly speaking, it is the repetitive time series background); the TS model is then updated by the ‘TS model update’ module (Fig. 2). Those pixels that fail in matching will be classified as the foreground. As the duration of each sub-pattern in TS model may differ a bit from that of the same pattern appearing hereafter, the pixel value predicted from TS model around the transition point between two adjacent sub-patterns is allowed to be either one of the two sub-patterns. Strictly speaking, around the time of

changing from one sub-pattern to the next, the pixel value v of current pixel is classified as background if it is similar to either the averaged pixel value of predicted

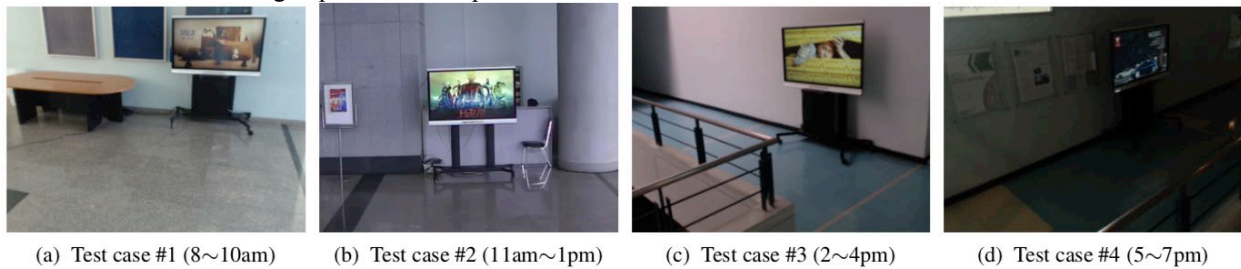


Fig. 4. Four experimental video scenes.

sub-pattern \mathbf{P}_k or that of the next sub-pattern \mathbf{P}_{k+1} . Note that this is done by using (7). The process of foreground detection allows some failure in matching of current pixel value and the predicted sub-pattern—this is also done in ‘finding a periodic repetition’. The proposed method employs a ‘time-delay detection’ which will judge that there is an object (foreground) on the pixel if and only if the matching fails consecutively for ε^h frames.

3.6 Post-processing

There is a trade-off between a high true positive rate (TPR) and a low false positive rate (FPR)—the robust background model successfully detects a lot of foreground objects, on the other hand, tends to mistakenly verify some background pixels as the foregrounds. We opt for the high detection rate; many noises (false positives) inevitably appear as a result of detection. An illumination change is one of the causes of those noises. On the other hand, some foreground pixels similar to the background model are sometimes erroneously classified as the background—they become the hole inside the foreground region. The post-processing is done here to suppress those noises, and to fill the hole inside and the gap between the foregrounds. Morphological filters—opening followed by closing—are used in our methods.

4.0 EXPERIMENTAL RESULTS AND DISCUSSION

The video of size 320×240 pixels is used in our experiments; they were captured at 30 fps. Four places in the building were used in the experiments, each taken at different time (morning, noon, afternoon, and evening)—some snapshots are shown in Fig. 4. As shown in the bottom left image of Fig. 2 (the same scene as Fig. 4b), the scene consists of (i) static background (a wall, a floor, a chair, and a bulletin board); (ii) repetitive time series background (the display of electronic billboard); and (iii) foreground objects (a walking human). There are 5–8 advertising images which are being displayed repeatedly, in the same order, and by the same interval. Many people walk past in front of the billboard; sometimes one by one, occasionally walk in group. Experiments were done in two steps; (i) the classification of time series background; and (ii) the detection of foreground. We demonstrate the result mainly in the region of periodic time series background, sometimes in the remaining area in order to check if and how many there are false positives. The objective evaluation is measured in terms of seven metrics (as described in [8])—recall (Re), precision (Pr), specificity (Sp), F-measure (FM), false positive rate (FPR), false negative rate (FNR), and percentage of wrong classifications (PWC). The image results of foreground segmentation are also shown as subjective evaluation of our proposed method.

4.1 Background Classification (Time Series Background and Other Background)

Ground truths and image results of background classification on some test cases (here, scene #1 and #4) are shown in Fig. 5 and Table 2–3. Fig. 5 shows some (input) image frames from scene #1 (top row) and #4 (bottom row) in the 1st column. We expect to classify the area of electronic billboard (TV display) as a ‘time series background area’—its ground truth is shown as white pixels in the 3rd column images. Other areas—most of them are static background—are shown in grey colour. The results of detecting the area of periodic time series pattern in scene #1 and #4—using our proposed method with four image components (hue (H), saturation (S), value (V), and

illumination (Y) channels) and various tolerances ε^v for pixel similarity measurement—are shown in Table 2 and 3 respectively.

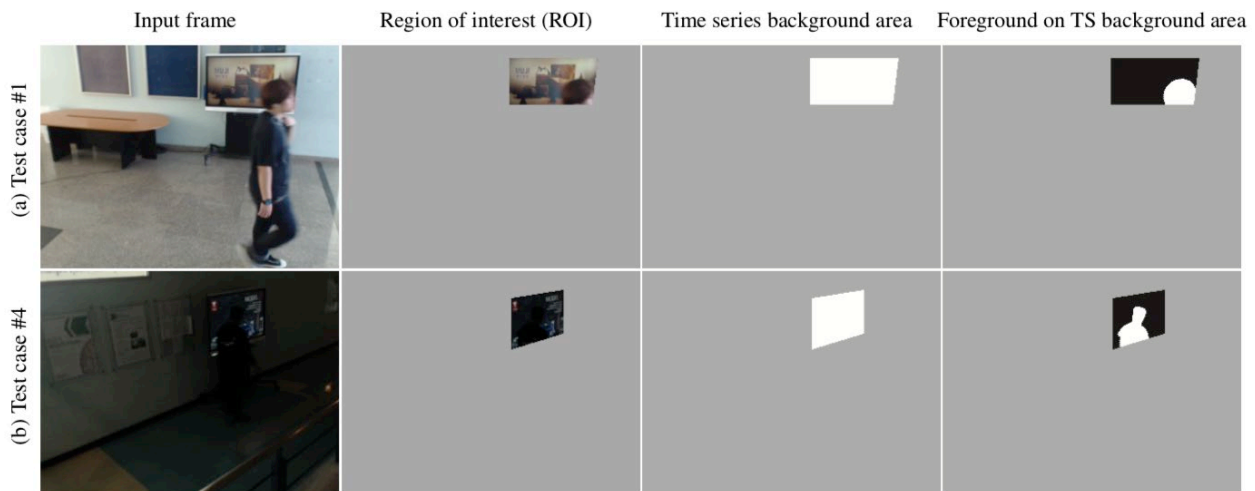


Fig. 5. Some image frames (1st column) from test cases (scenes) #1 (top row; bright illumination) and #4 (bottom row; quite dark). Expected results of time series background area detection (white pixels of the images in the 3rd column) and foreground detection on that area (white pixels of the images in the 4th column) are also shown with the prospective ROI (2nd column) of our method. Other areas which are not parts of time series background are shown in grey colour. Note that the images shown on the 1st column are from frame #4256 and #7254 of test cases #1 and #4 respectively.

According to the results shown in Table 2 and 3, the detection of TS background area on V and Y channels provides better results than H and S channels. Both V and Y channels obtain the best result of 98.4% TPR in the TS background area and 0.9% FPR in the other areas when $\varepsilon^v=12$. Less FPR in non-TS background area could be obtained with higher ε^v , but the TPR in TS background area also drops approximately at the same ratio.

Table 4 shows the averaged TPR of repetitive TS background detection on all scenes using our proposed method. Results of applying the proposed method to Y channel outperforms those of other image components (H, S, and V channels)—regardless of ε^v (tolerance for pixel similarity) value. It is worthwhile to mention that the averaged TPR becomes the highest value when applying the proposed detection method on Y channel of the scene with $\varepsilon^v=16$.

We also examined how fast the proposed method will correctly detect the pixels on TS background area. Fig. 6 shows the accumulated number of pixels (averaged over all scenes) which are correctly detected as TS background when the method is applied on V channel (Fig. 6a) and H channel (Fig. 6b). When the method is implemented without pre-processing step (shown in blue line), the detection is slower—especially with V channel—compared to the method that employs the pre-processing explained earlier. The speeds of successfully detecting 80% of TS pixels using V and H channel do not, however, differ much—they could cumulatively detect 80% of the TS background area at frame #3199 and #3206 respectively.

In principle, the proposed method can detect most pixels in TS background area when those pixels show their TS patterns twice and the first sub-pattern once more. However, in practice, as shown in Fig. 6, the method starts perceiving many TS pixels around frame #3000 which is delayed about one period of sub-pattern. One possible cause of the delay is the discarding of first ‘candidate’ sub-pattern. We throw it away because it may not be a full period of the sub-pattern—its existence will cause an error in TS background detection.

Table 2. Experimental results of time series background detection on test case #1. Four image components (hue (H), saturation (S), value (V), and illumination (Y) channels) and various tolerances ε^v for pixel similarity on each channel have been examined on TS background detection.





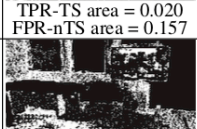
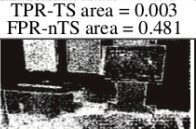
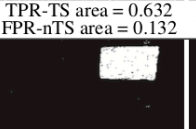
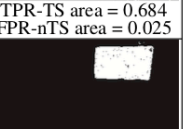
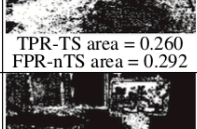
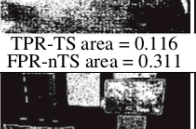
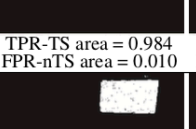
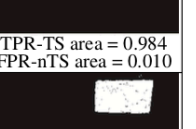
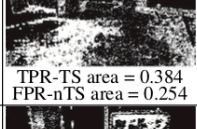
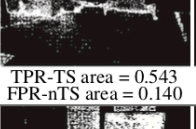
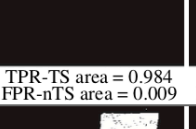
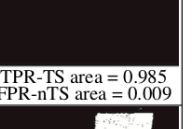
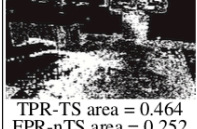
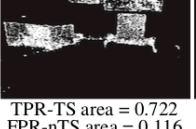
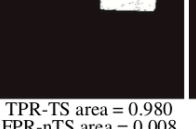
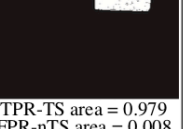

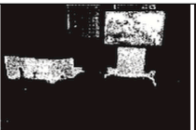


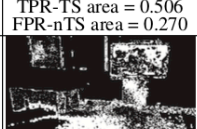
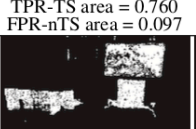
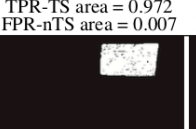
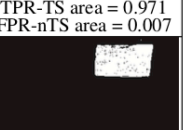
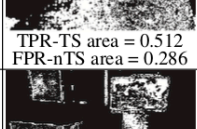
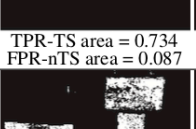
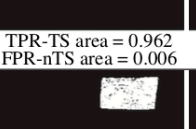
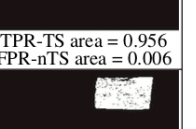
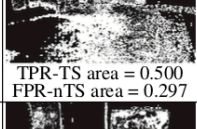
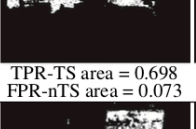
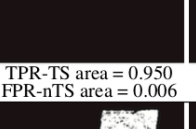
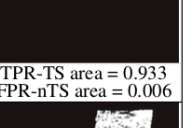
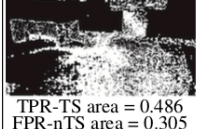
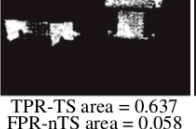
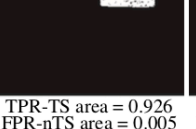
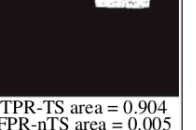
	H Channel	S Channel	V Channel	Y Channel
$\epsilon^v = 4$	 TPR-TS area = 0.020 FPR-nTS area = 0.157	 TPR-TS area = 0.003 FPR-nTS area = 0.481	 TPR-TS area = 0.632 FPR-nTS area = 0.132	 TPR-TS area = 0.684 FPR-nTS area = 0.025
$\epsilon^v = 8$	 TPR-TS area = 0.260 FPR-nTS area = 0.292	 TPR-TS area = 0.116 FPR-nTS area = 0.311	 TPR-TS area = 0.984 FPR-nTS area = 0.010	 TPR-TS area = 0.984 FPR-nTS area = 0.010
$\epsilon^v = 12$	 TPR-TS area = 0.384 FPR-nTS area = 0.254	 TPR-TS area = 0.543 FPR-nTS area = 0.140	 TPR-TS area = 0.984 FPR-nTS area = 0.009	 TPR-TS area = 0.985 FPR-nTS area = 0.009
$\epsilon^v = 16$	 TPR-TS area = 0.464 FPR-nTS area = 0.252	 TPR-TS area = 0.722 FPR-nTS area = 0.116	 TPR-TS area = 0.980 FPR-nTS area = 0.008	 TPR-TS area = 0.979 FPR-nTS area = 0.008
$\epsilon^v = 20$	 TPR-TS area = 0.506 FPR-nTS area = 0.270	 TPR-TS area = 0.760 FPR-nTS area = 0.097	 TPR-TS area = 0.972 FPR-nTS area = 0.007	 TPR-TS area = 0.971 FPR-nTS area = 0.007
$\epsilon^v = 24$	 TPR-TS area = 0.512 FPR-nTS area = 0.286	 TPR-TS area = 0.734 FPR-nTS area = 0.087	 TPR-TS area = 0.962 FPR-nTS area = 0.006	 TPR-TS area = 0.956 FPR-nTS area = 0.006
$\epsilon^v = 28$	 TPR-TS area = 0.500 FPR-nTS area = 0.297	 TPR-TS area = 0.698 FPR-nTS area = 0.073	 TPR-TS area = 0.950 FPR-nTS area = 0.006	 TPR-TS area = 0.933 FPR-nTS area = 0.006
$\epsilon^v = 32$	 TPR-TS area = 0.486 FPR-nTS area = 0.305	 TPR-TS area = 0.637 FPR-nTS area = 0.058	 TPR-TS area = 0.926 FPR-nTS area = 0.005	 TPR-TS area = 0.904 FPR-nTS area = 0.005
$\epsilon^v = 36$	 TPR-TS area = 0.463 FPR-nTS area = 0.308	 TPR-TS area = 0.581 FPR-nTS area = 0.038	 TPR-TS area = 0.898 FPR-nTS area = 0.004	 TPR-TS area = 0.871 FPR-nTS area = 0.004
$\epsilon^v = 40$	 TPR-TS area = 0.454 FPR-nTS area = 0.303	 TPR-TS area = 0.507 FPR-nTS area = 0.021	 TPR-TS area = 0.857 FPR-nTS area = 0.004	 TPR-TS area = 0.823 FPR-nTS area = 0.003

Table 3. Experimental results of time series background detection on test case #4. Four image components (hue (H), saturation (S), value (V), and illumination (Y) channels) and various tolerances ϵ^v for pixel similarity on each channel have been examined on TS background detection.

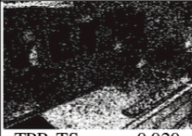
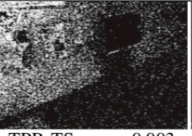














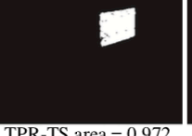
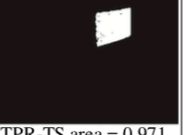


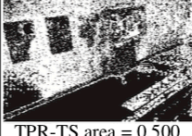
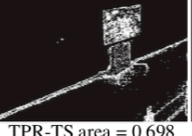
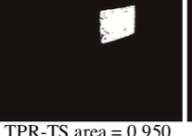

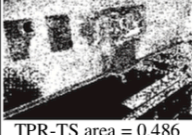
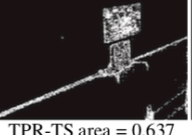
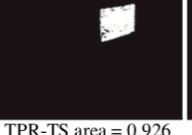
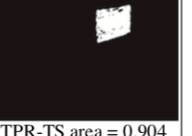
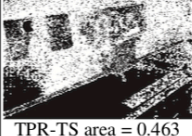
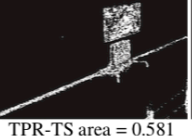
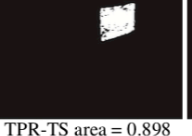
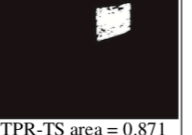
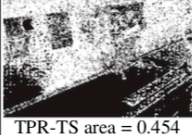
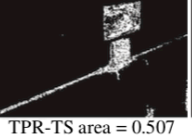
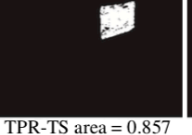
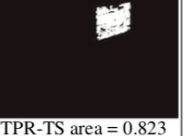
	H Channel	S Channel	V Channel	Y Channel
$\epsilon^v = 4$	 TPR-TS area = 0.020 FPR-nTS area = 0.157	 TPR-TS area = 0.003 FPR-nTS area = 0.481	 TPR-TS area = 0.632 FPR-nTS area = 0.132	 TPR-TS area = 0.684 FPR-nTS area = 0.025
$\epsilon^v = 8$	 TPR-TS area = 0.260 FPR-nTS area = 0.292	 TPR-TS area = 0.116 FPR-nTS area = 0.311	 TPR-TS area = 0.984 FPR-nTS area = 0.010	 TPR-TS area = 0.984 FPR-nTS area = 0.010
$\epsilon^v = 12$	 TPR-TS area = 0.384 FPR-nTS area = 0.254	 TPR-TS area = 0.543 FPR-nTS area = 0.140	 TPR-TS area = 0.984 FPR-nTS area = 0.009	 TPR-TS area = 0.985 FPR-nTS area = 0.009
$\epsilon^v = 16$	 TPR-TS area = 0.464 FPR-nTS area = 0.252	 TPR-TS area = 0.722 FPR-nTS area = 0.116	 TPR-TS area = 0.980 FPR-nTS area = 0.008	 TPR-TS area = 0.979 FPR-nTS area = 0.008
$\epsilon^v = 20$	 TPR-TS area = 0.506 FPR-nTS area = 0.270	 TPR-TS area = 0.760 FPR-nTS area = 0.097	 TPR-TS area = 0.972 FPR-nTS area = 0.007	 TPR-TS area = 0.971 FPR-nTS area = 0.007
$\epsilon^v = 24$	 TPR-TS area = 0.512 FPR-nTS area = 0.286	 TPR-TS area = 0.734 FPR-nTS area = 0.087	 TPR-TS area = 0.962 FPR-nTS area = 0.006	 TPR-TS area = 0.956 FPR-nTS area = 0.006
$\epsilon^v = 28$	 TPR-TS area = 0.500 FPR-nTS area = 0.297	 TPR-TS area = 0.698 FPR-nTS area = 0.073	 TPR-TS area = 0.950 FPR-nTS area = 0.006	 TPR-TS area = 0.933 FPR-nTS area = 0.006
$\epsilon^v = 32$	 TPR-TS area = 0.486 FPR-nTS area = 0.305	 TPR-TS area = 0.637 FPR-nTS area = 0.058	 TPR-TS area = 0.926 FPR-nTS area = 0.005	 TPR-TS area = 0.904 FPR-nTS area = 0.005
$\epsilon^v = 36$	 TPR-TS area = 0.463 FPR-nTS area = 0.308	 TPR-TS area = 0.581 FPR-nTS area = 0.038	 TPR-TS area = 0.898 FPR-nTS area = 0.004	 TPR-TS area = 0.871 FPR-nTS area = 0.004
$\epsilon^v = 40$	 TPR-TS area = 0.454 FPR-nTS area = 0.303	 TPR-TS area = 0.507 FPR-nTS area = 0.021	 TPR-TS area = 0.857 FPR-nTS area = 0.004	 TPR-TS area = 0.823 FPR-nTS area = 0.003

Table 4. Averaged TPR (true positive rate) of time series background detection when the proposed method is applied to all scenes. Results of detection on H, S, V, and Y channels are shown with various tolerances (ϵ^v) of pixel

value similarity. The best results of each image channel—when ε^v is varied—are shown in bold face; while the best channel that provides the highest TPR for each value of ε^v are shown in blue.

ε^v	Image channel			
	H	S	V	Y
4	0.017	0.001	0.510	0.611
8	0.165	0.141	0.735	0.793
12	0.252	0.321	0.843	0.902
16	0.301	0.428	0.901	0.929
20	0.323	0.508	0.912	0.929
24	0.325	0.565	0.899	0.907
28	0.319	0.580	0.864	0.874
32	0.311	0.574	0.819	0.832
36	0.300	0.541	0.763	0.788
40	0.280	0.501	0.719	0.737

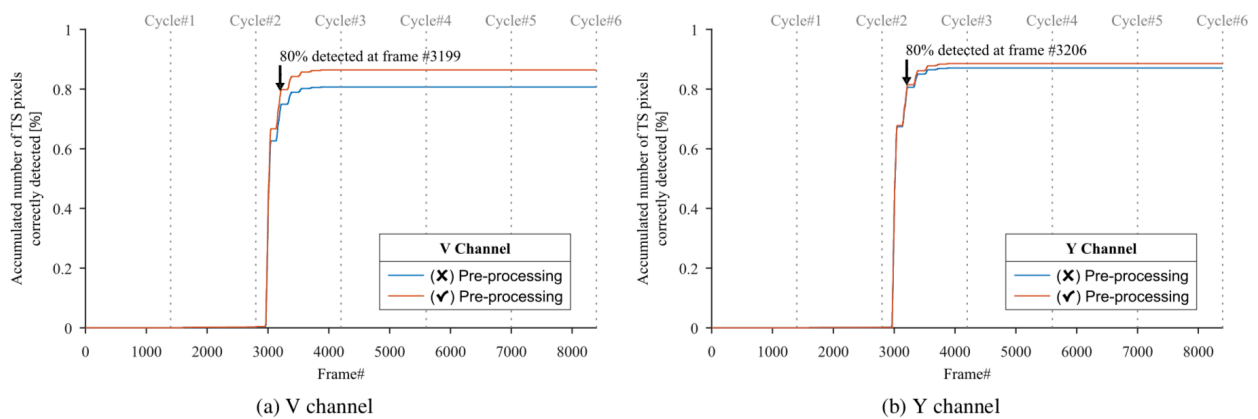


Fig. 6. Accumulated number of pixels which are correctly classified as time series background—when the TS background detection is performed on V and Y channels. The blue line shows the result of detection without pre-processing; while the red line with pre-processing. Note that the results shown here are the average of detection results from all scenes.

4.2 Foreground Detection

Image results of foreground detection (or change detection) on the image frames of Fig. 5a (test case #1) and Fig. 5b (test case #4) are shown in Table 5 and Table 6 respectively. Ground truths are shown in the 4th column of Fig. 5—foreground in white, TS background in black, and others in grey colour. The experiments show the results from four image components (hue (H), saturation (S), value (V), and illumination (Y) channels) with various tolerance ε^v for pixel similarity measurement.

According to the results, the detection of foreground on V and Y channels is more accurate than H and S channels in the area of TS background—there are also less false positives in other areas. The post-processing applied for noise suppression helps decrease the scattered small dots of noise; and also fill the gap/hole. The best value of F-measure (about 0.8) is obtained when using V and Y channels with $\varepsilon^v=12-20$ and utilizing the post-processing in foreground detection on test case #1 (the bright scene). Lower value of ε^v , around 8–12, is required to obtain the best F-measure (about 0.9) in case of test case #4 (quite dark scene). From these results, the value of tolerance ε^v has to change according to the overall illumination in the scene—this kind of adaptiveness is required in order to keep the good detection result. Figs.7 and 8 show the average/min/max F-measure (FM) and average false positive rate (FPR) of foreground detection on time series background area with various tolerances ($\varepsilon^v= \pm 4 \sim \pm 40$) for pixel value similarity. Good average F-measures are obtained when detecting foreground on V (shown in green) and Y (shown in red) channels, especially with pre- and post-processing (Fig. 7d) and when $\varepsilon^v= \pm 12 \sim \pm 24$. As for V and Y channels, the higher the ε^v value is, the smaller the F-measure variance of foreground detection becomes. On the

other hand, average FPR is the lowest when detecting foreground on H (shown in blue) and S (shown in black) channels, with $\epsilon^v = \pm 4$ (Fig. 8). A trade-off between the high FM versus the low FPR suggests— from Figs. 7 and 8—that foreground detection is to be done on V or Y channel with pre- and post-processing and using the tolerance ϵ^v of $\pm 16 \sim \pm 24$.

Table 5. Experimental results of foreground detection on the image frame of Fig. 5a (test case #1). Four image components (hue (H), saturation (S), value (V), and illumination (Y) channels) and various tolerance ϵ^v for pixel similarity on each channel have been examined on foreground detection. The results without post-processing (morphological opening followed by closing) and with post-processing are shown in the column named ‘Unfiltered’ and ‘Filtered’ respectively. The pixels verified as a foreground are shown in white; a TS background in black; and other background area in grey colour.

	H Channel		S Channel		V Channel		Y Channel	
	Unfiltered	Filtered	Unfiltered	Filtered	Unfiltered	Filtered	Unfiltered	Filtered
$\epsilon^v = 4$								
	FM = 0.026	FM = 0.000	FM = 0.010	FM = 0.000	FM = 0.539	FM = 0.555	FM = 0.519	FM = 0.526
$\epsilon^v = 8$								
	FM = 0.406	FM = 0.584	FM = 0.242	FM = 0.388	FM = 0.684	FM = 0.706	FM = 0.664	FM = 0.689
$\epsilon^v = 12$								
	FM = 0.485	FM = 0.624	FM = 0.371	FM = 0.500	FM = 0.752	FM = 0.795	FM = 0.749	FM = 0.794
$\epsilon^v = 16$								
	FM = 0.526	FM = 0.749	FM = 0.448	FM = 0.541	FM = 0.755	FM = 0.806	FM = 0.748	FM = 0.799
$\epsilon^v = 20$								
	FM = 0.519	FM = 0.702	FM = 0.450	FM = 0.544	FM = 0.739	FM = 0.791	FM = 0.728	FM = 0.798
$\epsilon^v = 24$								
	FM = 0.499	FM = 0.701	FM = 0.442	FM = 0.544	FM = 0.716	FM = 0.752	FM = 0.708	FM = 0.775
$\epsilon^v = 28$								
	FM = 0.489	FM = 0.671	FM = 0.411	FM = 0.580	FM = 0.699	FM = 0.751	FM = 0.678	FM = 0.762
$\epsilon^v = 32$								
	FM = 0.462	FM = 0.731	FM = 0.331	FM = 0.545	FM = 0.659	FM = 0.755	FM = 0.652	FM = 0.722
$\epsilon^v = 36$								
	FM = 0.421	FM = 0.640	FM = 0.239	FM = 0.463	FM = 0.618	FM = 0.713	FM = 0.606	FM = 0.700
$\epsilon^v = 40$								
	FM = 0.382	FM = 0.665	FM = 0.154	FM = 0.218	FM = 0.582	FM = 0.673	FM = 0.560	FM = 0.672

Table 6. Experimental results of foreground detection on the image frame of Fig. 5b (test case #4). Four image components (hue (H), saturation (S), value (V), and illumination (Y) channels) and various tolerance ϵ^v for pixel similarity on each channel have been examined on foreground detection. The results without post-processing (morphological opening followed by closing) and with post-processing are shown in the column named ‘Unfiltered’ and ‘Filtered’ respectively. The pixels verified as a foreground are shown in white; a TS background in black; and other background area in grey colour.

	H Channel		S Channel		V Channel		Y Channel	
	Unfiltered	Filtered	Unfiltered	Filtered	Unfiltered	Filtered	Unfiltered	Filtered
$\epsilon = 4$								
	FM = 0.120	FM = 248	FM = 0.002	FM = NaN	FM = 0.816	FM = 0.885	FM = 0.908	FM = 0.909
$\epsilon = 8$								
	FM = 0.352	FM = 0.548	FM = 0.051	FM = 0.002	FM = 0.917	FM = 0.920	FM = 0.915	FM = 0.918
$\epsilon = 12$								
	FM = 0.453	FM = 0.806	FM = 0.092	FM = 0.060	FM = 0.919	FM = 0.928	FM = 0.887	FM = 0.900
$\epsilon = 16$								
	FM = 0.497	FM = 0.804	FM = 0.198	FM = 0.397	FM = 0.905	FM = 0.917	FM = 0.732	FM = 0.801
$\epsilon = 20$								
	FM = 0.503	FM = 0.822	FM = 0.451	FM = 0.816	FM = 0.848	FM = 0.874	FM = 0.652	FM = 0.646
$\epsilon = 24$								
	FM = 0.511	FM = 0.750	FM = 0.600	FM = 0.836	FM = 0.719	FM = 0.773	FM = 0.592	FM = 0.571
$\epsilon = 28$								
	FM = 0.540	FM = 0.769	FM = 0.608	FM = 0.809	FM = 0.652	FM = 0.654	FM = 0.558	FM = 0.557
$\epsilon = 32$								
	FM = 0.556	FM = 0.821	FM = 0.579	FM = 0.842	FM = 0.603	FM = 0.618	FM = 0.519	FM = 0.549
$\epsilon = 36$								
	FM = 0.527	FM = 0.764	FM = 0.560	FM = 0.848	FM = 0.563	FM = 0.625	FM = 0.481	FM = 0.531
$\epsilon = 40$								
	FM = 0.515	FM = 0.806	FM = 0.510	FM = 0.861	FM = 0.545	FM = 0.560	FM = 0.444	FM = 0.503

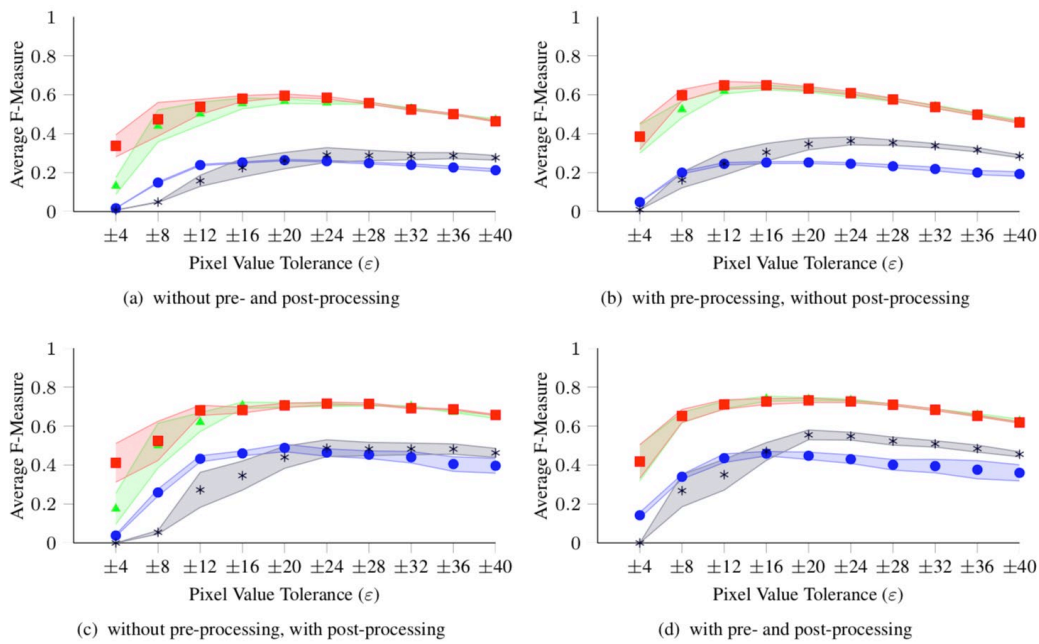


Fig. 7. Average F-measure (FM) of foreground detection on TS background area with various tolerances ($\epsilon^v = \pm 4 \sim \pm 40$) for pixel value similarity. Four results are shown— (a) without pre- and post-processing; (b) with pre-processing but without post-processing; (c) without pre-processing but with post-processing; and (d) with pre- and post-processing. The average/min/max F-measures of foreground detection on H, S, V, and Y channels are shown in blue circle, black asterisk, green triangle, and red square respectively.

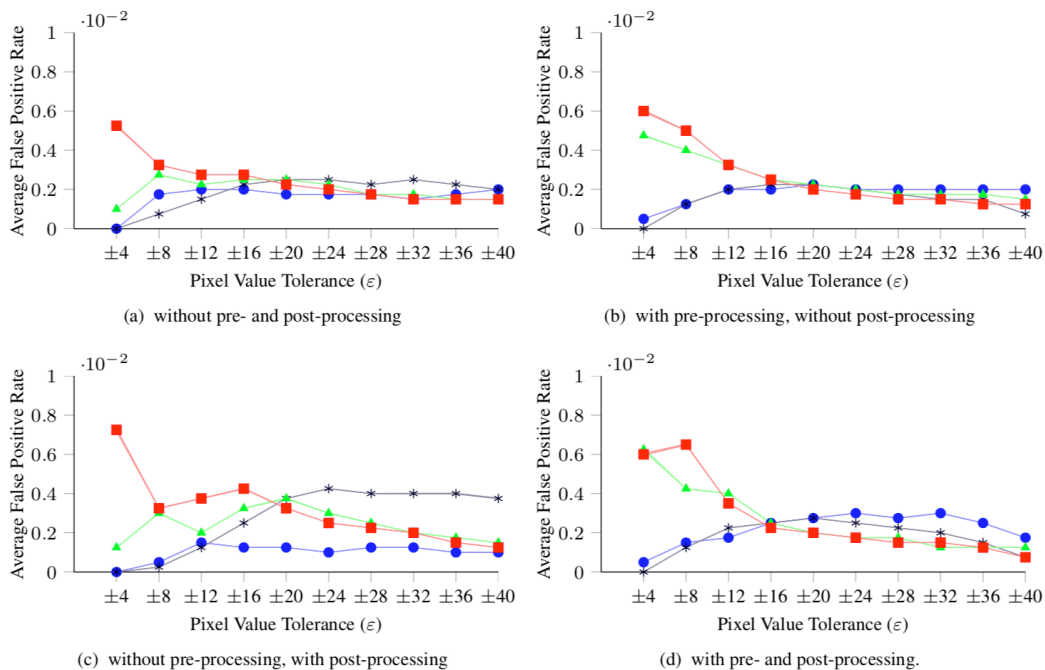


Fig. 8. Average false positive rate (FPR) of foreground detection on TS background area with various tolerances ($\epsilon^v = \pm 4 \sim \pm 40$) for pixel value similarity. Four results are shown—(a) without pre- and post-processing; (b) with pre-processing but without post-processing; (c) without pre-processing but with post-processing; and (d) with pre- and post-processing. The average FPR of foreground detection on H, S, V, and Y channels are shown in blue circle, black asterisk, green triangle, and red square respectively.

Table 7 shows average performance of our proposed foreground detection. Recall (Re), precision (Pr), specificity (Sp), F-measure (FM), false positive rate (FPR), false negative rate (FNR), and percentage of wrong classification (PWC) are evaluated for all frames in the video—four image components (H, S, V, and Y channels) are examined while the pixel value tolerance is varied from $\pm 4 \sim \pm 40$. The best results for each metric are shown in blue; most of them (Re, Pr, FNR, and PWC) are with Y channel. Other two metrics (Sp and FPR) are just 0.001 worse than the best value. The last metric (FM) is 0.009 lower than the best value. Therefore, Y channel is a good choice for foreground detection. The best value of ε^p for all metrics in Y channel is difficult to be determined as there is a trade-off among them. However, the value between ± 8 to ± 28 looks quite promising in terms of objective and subjective results obtained so far.

4.2.1 Foreground detection around the frame of sub-pattern transition

Most background subtraction techniques have difficulties handling a repetitive time series background—particularly around the frame of transition from one sub-pattern (advertising image in case of electronic billboard) to another. The evaluation of FPR when applying the proposed foreground detection method on the area of time series background in each frame is shown in Fig. 9 (top row). As expected, the FPR around the frame that is about to display next sub-pattern, which sometimes shows the blending image of two sub-patterns, is significantly high (spikes in the graph, about 35%)—this happens when the proposed foreground detection method is implemented without ‘time-delay’. One example of such blending image is shown in Fig. 9—the image displayed in TS background area at frame #6290 (middle column of 2nd and 3rd row) is the blending of two sub-pattern images (one shown on the left, the other on the right). Without ‘time-delay’, many pixels (around 37.15%) in the TS background region at frame #6290 are mistakenly detected as foreground (middle image of 4th row). When the ‘time-delay detection’ is employed, the FPR in TS area of frame #6290 decreases to 0.19% (middle image of last row)—approximately equals to FPR in the same area of normal frame (images on the left and the right of last row).

4.2.2 Processing time

The processing time required for running our foreground detection method has been measured and shown as the number of frames that our foreground detection can process in one second (fps). The measurement has been done on many scenes—the size of time series background area in the scene is varied from 160×120 , 320×240 , 640×480 , 720×480 , 960×720 to 1280×720 pixels. Fig. 10 shows the measurement results of implementing the algorithm in one computer process (without multithreading or other optimization) and running that process on an Intel Pentium Core i7 2.40 GHz Processor. As expected, the number of frames that our method can process in one second is inversely proportional to the number of time series pixels in the scene; because the total processing time for one image frame is, in principle, equal to the number of TS pixels multiplied by the time required for processing one TS pixel.

As shown in the results, the process can run at 25–30 fps or more when the size of time series background area is not larger than 640×480 pixels. If the size of TS background is the same as high-definition television (HDTV), i.e. 1280×720 pixels, the process can run at 15 fps.

Table 7. Average performance of our foreground detection method when applied to all video scenes with various tolerances ϵ^v for pixel similarity measurement. Four image components (H, S, V, and Y channels) have been examined. The best results of each image channel—when ϵ^v is varied—are shown in bold face; while the best results of all image channel are shown in blue.

	ϵ^v	Re	Pr	Sp	FM	FPR	FNR	PWC
H	4	0.028	0.012	1.000	0.142	0.001	0.972	0.067
	8	0.254	0.102	0.999	0.340	0.002	0.746	0.156
	12	0.363	0.112	0.998	0.435	0.002	0.637	0.187
	16	0.422	0.091	0.998	0.458	0.003	0.579	0.207
	20	0.465	0.148	0.997	0.448	0.003	0.535	0.256
	24	0.457	0.247	0.997	0.430	0.003	0.543	0.240
	28	0.447	0.133	0.997	0.401	0.003	0.553	0.237
	32	0.439	0.163	0.997	0.395	0.003	0.561	0.240
	36	0.405	0.047	0.998	0.376	0.003	0.595	0.214
	40	0.350	0.086	0.998	0.360	0.002	0.650	0.178
S	4	0.000	0.000	1.000	0.000	0.000	1.000	0.029
	8	0.245	0.025	0.999	0.268	0.001	0.756	0.133
	12	0.364	0.054	0.998	0.351	0.002	0.636	0.123
	16	0.466	0.071	0.998	0.470	0.003	0.535	0.105
	20	0.569	0.086	0.997	0.555	0.003	0.431	0.099
	24	0.543	0.084	0.998	0.548	0.003	0.458	0.101
	28	0.491	0.100	0.998	0.523	0.002	0.509	0.098
	32	0.452	0.109	0.998	0.509	0.002	0.548	0.084
	36	0.377	0.120	0.999	0.484	0.002	0.623	0.061
	40	0.297	0.177	0.999	0.455	0.001	0.704	0.045
V	4	0.685	0.088	0.994	0.412	0.006	0.315	0.215
	8	0.806	0.230	0.996	0.643	0.004	0.194	0.099
	12	0.848	0.231	0.996	0.708	0.004	0.153	0.076
	16	0.804	0.353	0.998	0.741	0.003	0.196	0.044
	20	0.756	0.345	0.998	0.739	0.002	0.244	0.038
	24	0.704	0.434	0.998	0.734	0.002	0.296	0.026
	28	0.647	0.393	0.998	0.709	0.002	0.353	0.028
	32	0.597	0.226	0.999	0.684	0.001	0.403	0.025
	36	0.549	0.313	0.999	0.658	0.001	0.452	0.029
	40	0.496	0.327	0.999	0.626	0.001	0.504	0.028
Y	4	0.634	0.225	0.994	0.418	0.006	0.366	0.160
	8	0.868	0.221	0.994	0.652	0.007	0.132	0.144
	12	0.828	0.297	0.997	0.711	0.004	0.173	0.066
	16	0.777	0.410	0.998	0.726	0.002	0.223	0.043
	20	0.734	0.498	0.998	0.732	0.002	0.266	0.033
	24	0.693	0.478	0.998	0.727	0.002	0.307	0.026
	28	0.624	0.457	0.999	0.710	0.002	0.376	0.023
	32	0.581	0.479	0.999	0.684	0.002	0.419	0.020
	36	0.527	0.506	0.999	0.652	0.001	0.474	0.022
	40	0.474	0.442	0.999	0.619	0.001	0.527	0.023

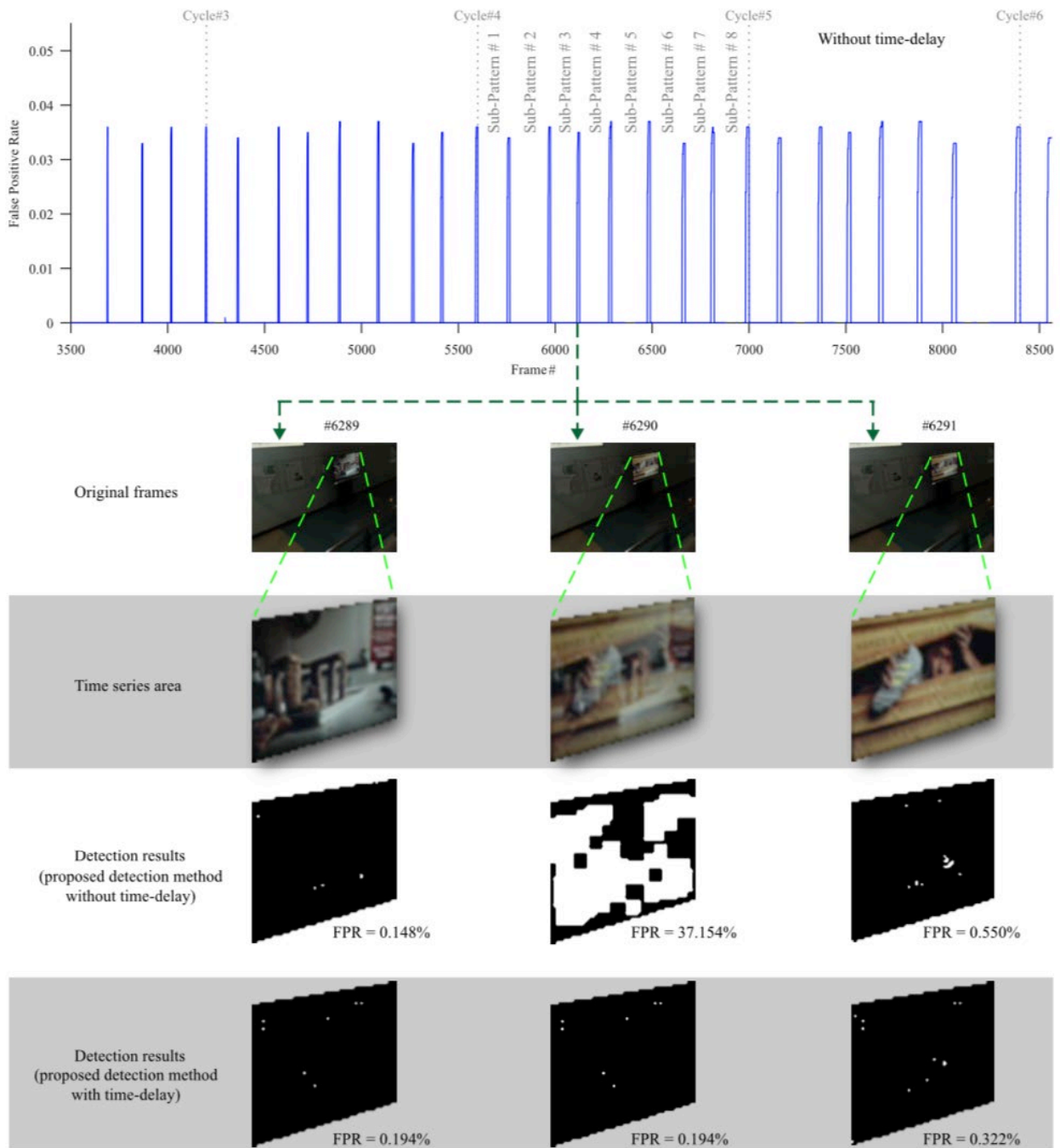


Fig. 9. (Test case #4) (top row:) False positive rate (FPR) of foreground detection on the area of repetitive time series background, particularly around the frame of sub-pattern transition. (left and right columns of 2nd and 3rd row:) Frame #6289 and #6291 display sub-patterns #3 and #4 respectively on TS background area (electronic billboard); (middle column of same rows:) while frame #6290 displays the blending of sub-patterns #3 and #4. (middle column of 4th row:) Ordinary method will detect those pixels in frame #6290 as foreground because these pixel values are not in the model. (middle column of last row:) By allowing some matching failures, the FPR in this area of the blending frame dramatically decreases.

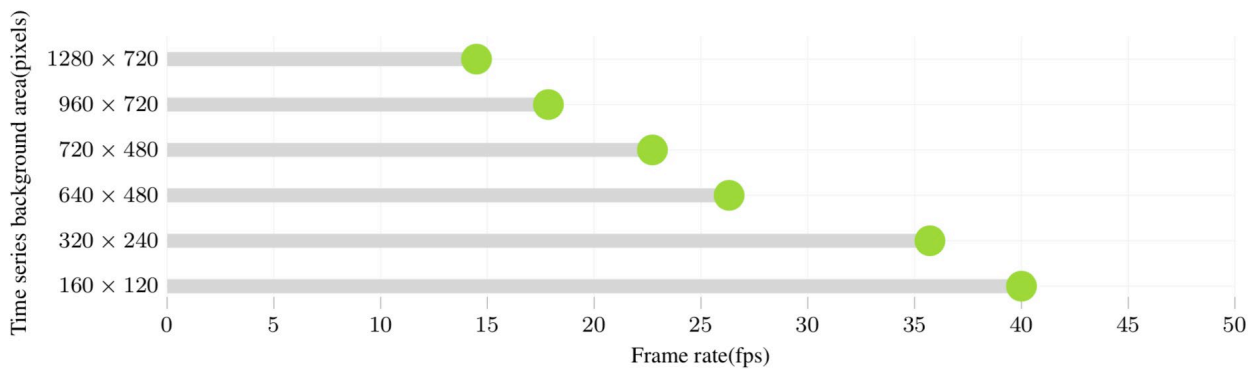


Fig. 10. The number of frames that the proposed foreground detection method can process in one second (frames per second, fps) when the image has different amounts of time series pixel.

4.3 Comparison of foreground object detection between our proposed methods combined with state-of-the-art methods and the state-of-the-art methods alone

This section presents the results of foreground object detection of images of which background had both non-time series change and time series change by our combined proposed method with five other state-of-the-art methods and the five state-of-the-art methods alone that were very capable of dealing with dynamic background issue [7, 8]: KDE [20], LBP [6], MoG [9], MultiCue [21], and ViBe[1]. The objective of this comparison was. The comparison was made on five sets of dynamic background test images from CD.NET [22] of which background had non-time series change and three sets of test images of which background had time series change mentioned in Section 4.0. Performance evaluation was done with the same seven standard evaluation metrics presented in Section 4.0. The parameters of all five state-of-the-art methods were set according to the reference values stated by the authors of the corresponding papers, while the parameters of our proposed method were set as follows: $\nu = 20$ which is the value obtained from the test mentioned in [4.1, 4.2]; YCbCr color space; and single Y channel throughout the comparison. In order to make it easier to evaluate foreground object detection subjectively both for regions with dynamic background and static background, the comparison was made on detection of ground truth image only in the region with dynamic background of time series change. Results of foreground object detection of images that had non-time series change and time series change are presented in Section 4.3.1 and 4.3.2, respectively, and the comparative results of processing time of all tested methods are shown in Section 4.3.3

4.3.1 Results of foreground object detection of images of which background had non-time series change

In foreground object detection of images of which background had no time series change shown in Table 8, our proposed hybrid method and the other five state-of-the-art methods performed equally well because the images had no time series regions hence no successful detection of foreground objects in such regions. This result was a direct result of the procedure of our proposed method which also did not produce false positive in any other types of regions.

4.3.2 Results of foreground object detection of images of which background had time series change

Even though the performance of the hybrid of our proposed method with state-of-the-art methods on detecting foreground objects in images of which background had non-time series change was comparable to those of the state-of-the-art methods alone as shown in Table 8, subjective evaluation of the images shown in Table 9 clearly shows that the state-of-the-art methods produced some false alarms in the regions with time series change, especially when there was a change to a new sub-pattern even if no foreground objects had appeared. This is because the state-of-the-art methods relied only on the current existing sub-pattern for representation of background model, so every time there was a change to the sub-pattern, these methods instantly considered the new sub-pattern to be a foreground object. Moreover, the status of this foreground object remained until the new sub-pattern was updated to the background model of these methods. Therefore, if a foreground object appeared in the image while there was a change to a new sub-pattern, foreground object detection would be falsely detected. This is shown in the results of

foreground object detection shown in Table 10. On the contrary, our proposed hybrid method not only prevented false alarms when no foreground object existed but the shapes of the foreground objects that it was able to detect were nearly identical to Table 8. Results of foreground object detection by all tested methods on images of which background had both Non-time series change and Time series change. The best result of each method is shown in bold face. (note: ++ signifies the results from the hybrid of our proposed method with state-of-the-art methods)

Foreground detection on Non-time series change background evaluation							
Methods	Re	Pr	Sp	FPR	FNR	PWC	FM
KDE	0.876	0.044	0.803	0.197	0.124	0.194	0.200
KDE ++	0.876	0.044	0.803	0.197	0.124	0.194	0.200
LBP	0.391	0.211	0.995	0.005	0.609	0.011	0.435
LBP ++	0.323	0.176	0.997	0.003	0.479	0.011	0.394
MoG	0.699	0.096	0.957	0.044	0.301	0.048	0.365
MoG ++	0.699	0.096	0.957	0.044	0.301	0.048	0.365
MultiCue	0.616	0.384	0.953	0.047	0.384	0.052	0.399
MultiCue ++	0.616	0.384	0.953	0.047	0.384	0.052	0.399
Vibe	0.552	0.188	0.990	0.010	0.448	0.014	0.509
Vibe ++	0.552	0.188	0.990	0.010	0.448	0.014	0.509
Foreground detection on Time series change background evaluation							
KDE	0.981	0.010	0.226	0.774	0.019	0.769	0.220
KDE ++	0.841	0.042	0.974	0.026	0.159	0.026	0.607
LBP	0.723	0.016	0.611	0.389	0.277	0.389	0.276
LBP ++	0.662	0.270	0.993	0.007	0.338	0.009	0.620
MoG	0.863	0.114	0.898	0.102	0.137	0.102	0.620
MoG ++	0.753	0.229	0.994	0.006	0.247	0.007	0.721
MultiCue	0.889	0.010	0.446	0.554	0.111	0.551	0.251
MultiCue ++	0.763	0.050	0.982	0.018	0.237	0.019	0.601
Vibe	0.353	0.015	0.895	0.105	0.647	0.110	0.288
Vibe ++	0.327	0.129	0.999	0.001	0.673	0.006	0.442
Overall Foreground detection evaluation							
KDE	0.929	0.027	0.515	0.485	0.071	0.482	0.210
KDE ++	0.859	0.043	0.889	0.111	0.141	0.110	0.403
LBP	0.557	0.113	0.803	0.197	0.443	0.200	0.355
LBP ++	0.527	0.240	0.994	0.006	0.473	0.010	0.528
MoG	0.781	0.105	0.927	0.073	0.219	0.075	0.493
MoG ++	0.726	0.163	0.975	0.025	0.274	0.027	0.543
MultiCue	0.753	0.197	0.700	0.300	0.247	0.302	0.325
MultiCue ++	0.690	0.217	0.968	0.032	0.310	0.036	0.500
Vibe	0.453	0.102	0.942	0.058	0.547	0.062	0.398
Vibe ++	0.439	0.158	0.994	0.006	0.561	0.010	0.475

those of the actual foreground objects in the ground truth image as shown in Table 10. Comparative results of F-measure and false positive rate on all test images are shown in Fig. 11. It can be seen that in the case of images that had time series change, our proposed method was able to reduce 96% of false positive while improve 45% of F-measure. On the average among all of the methods tested, our proposed method was able to reduce 80% of false alarm while improve 28% of F-measure.

4.3.3 Comparative results of processing time

As for the comparative increase in processing time that our proposed method used, the increase was only less than one frame per second. To put it in terms of actual use, our proposed method was able to operate in real-time at an average of 16 frames per second. All of the comparative results for all tested methods are shown in Fig 12.

Table 9. Results of foreground object detection by all tested methods on images of which background had time series change and no change in sub-pattern as well as no existing foreground objects in the images. The pixels verified as a foreground are shown in white; a TS background in black; and non-ROI in grey colour. (note: ++ signifies the results from the hybrid of our proposed method with state-of-the-art methods)

	Test case#1		Test case#2		Test case#3	
	Frame #3309	Frame #3310	Frame #3674	Frame #3675	Frame #4364	Frame #4365
Ground truth						
KDE						
KDE ++						
LBP						
LBP ++						
MoG						
MoG ++						
MultiCue						
MultiCue ++						
ViBe						
ViBe ++						

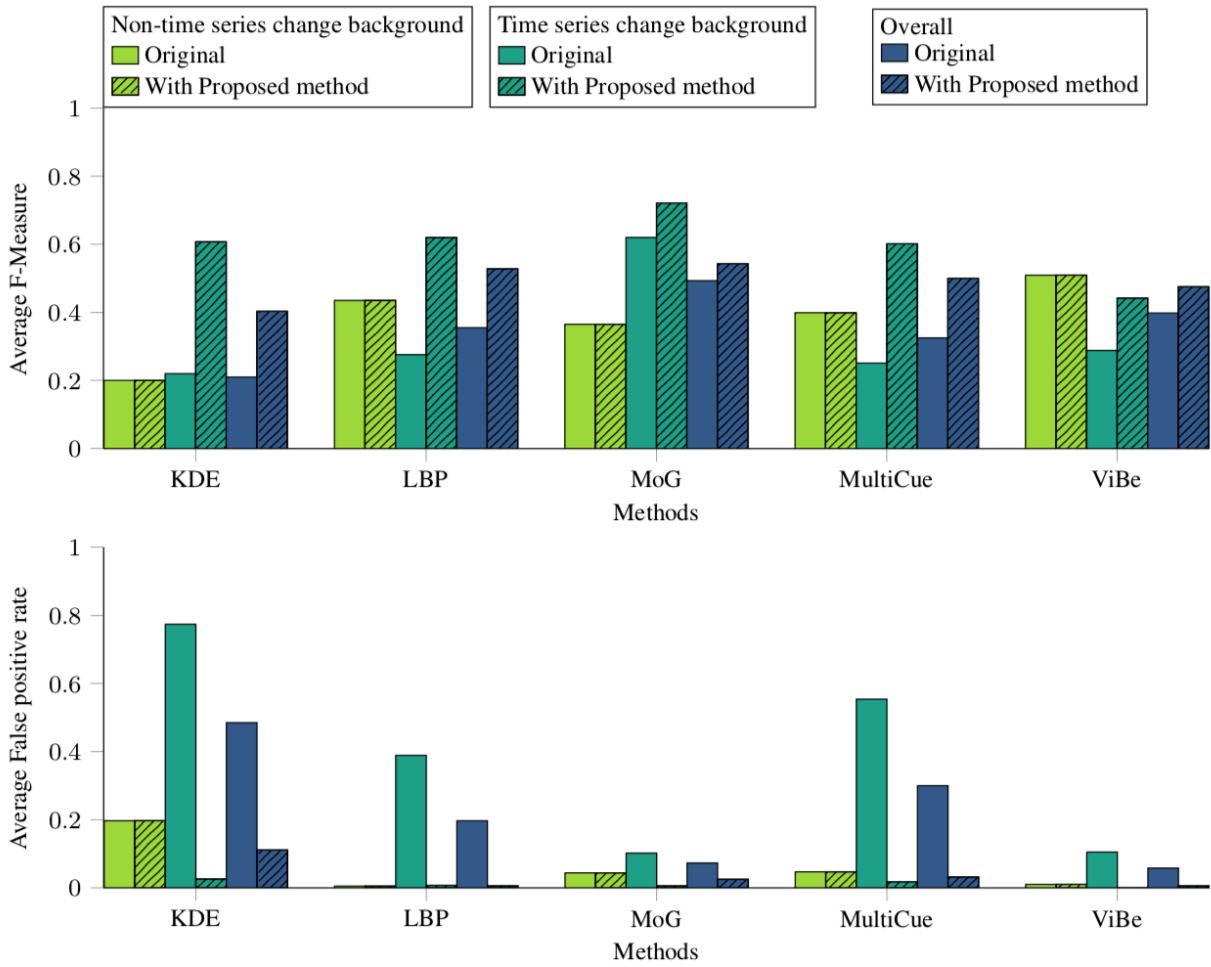


Fig. 11. Comparative results of average F-Measure(a) and average false positive rate(b) by all tested methods on all test images

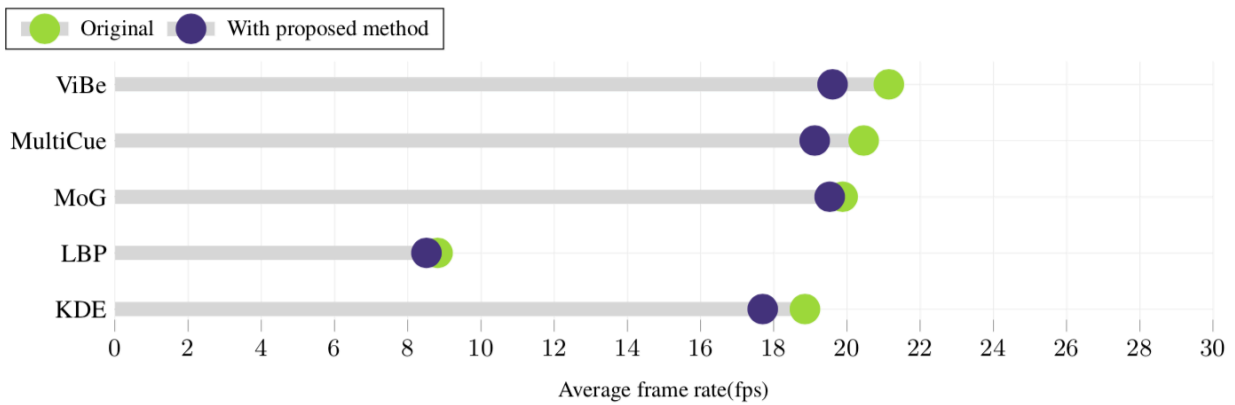
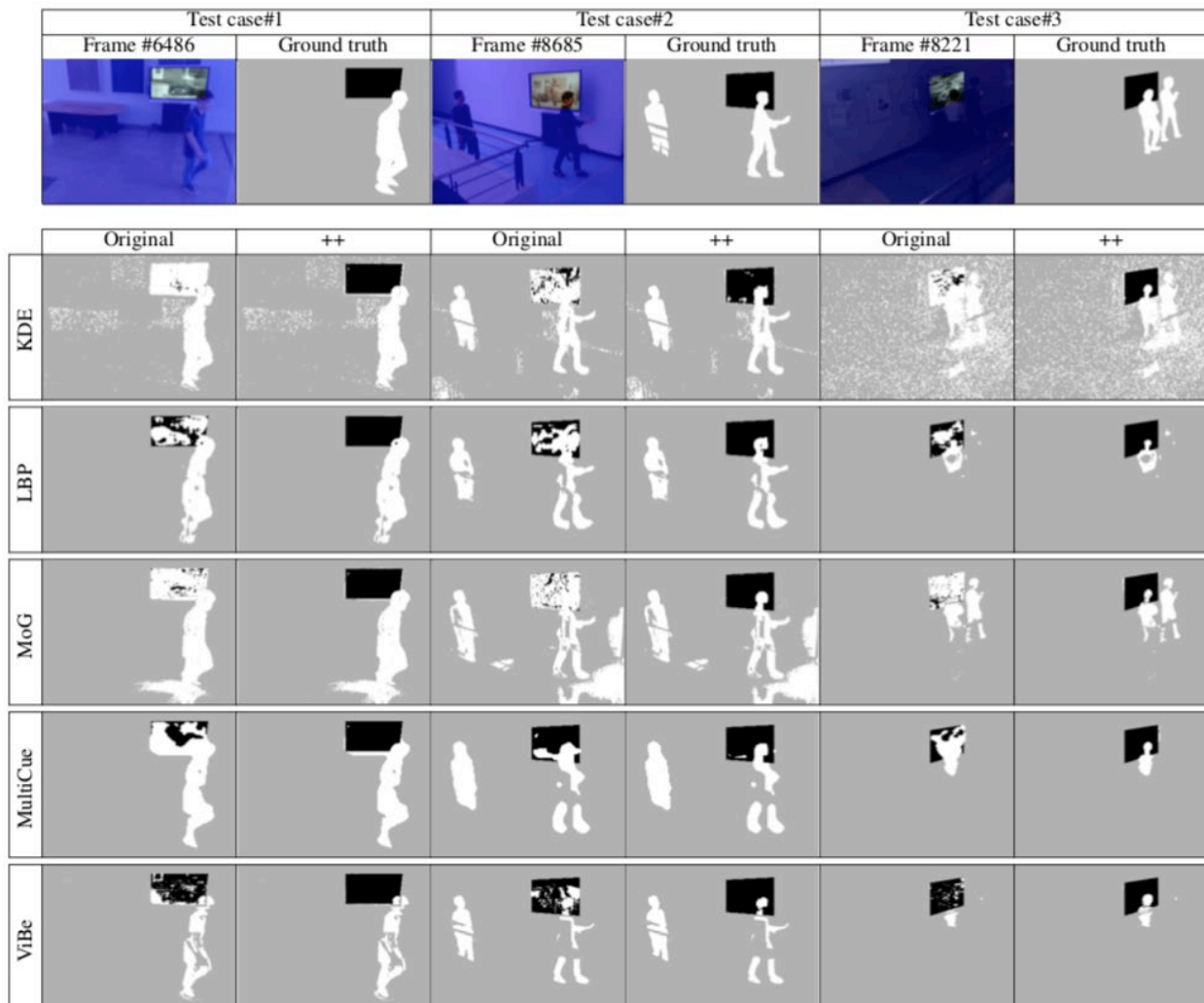


Fig. 12. Comparative results of average processing time of all tested methods

Table 10. Results of foreground object detection by all tested methods on images of which background had time series change and no change in sub-pattern as well as some foreground objects in the images. The pixels verified as a foreground are shown in white; a TS background in black; and non-ROI in grey colour. (note: ++ signifies the results from the hybrid of our proposed method with state-of-the-art methods)



5.0 CONCLUSION

Vision-based foreground detection based on background subtraction approach always encounters the problem of dynamic background. Most state-of-the-art approaches have been proposed such that some degree of change in the background, e.g. swaying trees, is handled properly; the repetitive time series background, however, had not been addressed. To suppress an erroneous detection of those repeated temporal background as a foreground, some simply mark those area as “not a region of interest”; this method however creates a non-monitoring area in the surveillance system, which potentially becomes a serious problem with high security area.

We proposed the method of time series background detection so as to overcome this problem. The method of foreground detection on those periodic changing background areas was also presented. Both methods can run concurrently on each pixel, and in parallel with any other state-of-the-art background-subtraction methodology which could perform well only with traditional background. Although the pre- and post-processing—the suppression of noise—cannot run per pixel, the proposed methods are able to run in real-time.

The proposed method has been proved that it could accurately detect the area of repetitive time series background with only few numbers of false positives, especially when the V or Y channel of image is used in the detection. Average true positive rate (or recall) is 92.9% and average F-measure is 0.92 when the detection is done on Y

channel. Our proposed method could detect about 80% of the time series background area in the scene as soon as the time series pattern repeats itself and the first sub-pattern is shown once more. The tolerance used in judgment of pixel similarity (for verifying if it repeats previous sub-pattern) has been investigated; we found that the darker the scene is, the smaller the tolerance value should be used; otherwise false positives will increase. Therefore, this tolerance must be adaptive, i.e. change its value proportionately to the brightness of the scene.

The proposed method of foreground detection in time series background area is also examined. The V and Y channels provide better results than H and S components—average F-measure and recall are about 0.70 ~ 0.85 depending on the value of pixel similarity tolerance. There are only few false positives; the false positive rate is less than 1%. Erroneous foreground detection on the frame in which two advertising images are captured and blended, although not always happens, causes a false positive from time to time. We proposed a ‘time-delay detection’ which helps suppress this kind of foreground detection error—the false positive of foreground detection on the blending image (normally not kept in background model) reduces from 35% to the same rate as of other normal image (sub-pattern), i.e. less than 1%.

Although the proposed method was implemented in and experiments shown earlier were done by a single-threaded process, it can run at 25–30 fps when the quantity of pixels in time series background area is not larger than 640×480 pixels. In case that there are 1280×720 pixels of time series background (as large as HD size), the method is still able to run at 15 fps. In addition, only the colour feature on a single channel of input frame was processed. Our proposed method outperformed the state-of-the-art methods tested in terms of processing time.

Objective evaluations of foreground object detection capabilities of our proposed method and five state-of-the-art methods that are very good at dealing with dynamic background issue show that our proposed method was able to reduce over 80% of false positive in detection of foreground objects in images of which background had time series change and able to improve more than 20% of F-measure over the state-of-the-art methods. Since the state-of-the-art methods relied on only the current existing sub-pattern for representation of background model, every time there was a change to the sub-pattern, these methods instantly considered the new sub-pattern to be a foreground object thus creating a false alarm. In contrast, the proposed method did not produce false positive in the regions that were not time series region. Subjectively, it can be clearly seen that the detected foreground objects by our proposed method were nearly identical to those in the ground truth image. All of these positive results were obtained by using nearly the same processing time that the five state-of-the-art methods used. Moreover, it also did not produce false positive in the regions that were not time series region. Subjectively, it can be clearly seen that the detected foreground objects by our proposed method were nearly identical to those in the ground truth image. All of these positive results were obtained by using almost no different processing time from those used by the five state-of-the-art methods.

REFERENCES

- [1] O. Barnich and M. V. Droogenbroeck, “ViBe: A Universal Background Subtraction Algorithm for Video Sequences,” *IEEE Trans. Image Process.*, vol. 20, no. 6, pp. 1709–1724, Jun. 2011.
- [2] M. Hofmann, P. Tiefenbacher, and G. Rigoll, “Background segmentation with feedback: The Pixel-Based Adaptive Segmenter,” in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Jun. 2012, pp. 38–43.
- [3] B. Wang and P. Dudek, “AMBER: Adapting multi-resolution background extractor,” in *2013 IEEE International Conference on Image Processing*, Sep. 2013, pp. 3417–3421.
- [4] “A Fast Self-Tuning Background Subtraction Algorithm,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Jun. 2014, pp. 401–404.
- [5] B. Yin, J. Zhang, and Z. Wang, “Background segmentation of dynamic scenes based on dual model,” *IET Comput. Vis.*, vol. 8, no. 6, pp. 545–555, 2014.
- [6] M. Heikkila and M. Pietikainen, “A texture-based method for modeling the background and detecting moving objects,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 657–662, Apr. 2006.

- [7] T. Bouwmans, "Traditional and recent approaches in background modeling for foreground detection: An overview," *Computer Science Review*, vol. 11-12, pp. 31–66, May 2014.
- [8] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, pp. 4–21, May 2014.
- [9] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 2, 1999, p. 252 Vol. 2.
- [10] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2, Aug. 2004, pp. 28–31 Vol.2.
- [11] T. S. F. Haines and T. Xiang, "Background Subtraction with DirichletProcess Mixture Models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 670–683, Apr. 2014.
- [12] G. A. Bilodeau, J. P. Jodoin, and N. Saunier, "Change Detection in Feature Space Using Local Binary Similarity Patterns," in *2013 International Conference on Computer and Robot Vision*, May 2013, pp. 106–112.
- [13] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin, "SuBSENSE: A Universal Change Detection Method with Local Adaptive Sensitivity," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 359–373, Jan. 2015.
- [14] H. Lee, H. Kim, and J. I. Kim, "Background Subtraction Using Background Sets with Image and Color Space Reduction," *IEEE Trans. Multimed.*, vol. 18, no. 10, pp. 2093–2103, Oct. 2016.
- [15] H. Sajid and S. C. S. Cheung, "Universal Multimode Background Subtraction," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3249–3260, Jul. 2017.
- [16] X. Zang, G. Li, J. Yang, and W. Wang, "Adaptive difference modelling for background subtraction," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, Dec. 2017, pp. 1–4.
- [17] S. Benabderrahmane, R. Quiniou, and T. Guyet, "Evaluating distance measures and times series clustering for temporal patterns retrieval," in *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)*, Aug. 2014, pp. 434–441.
- [18] J. Grabocka, M. Wistuba, and L. Schmidt-Thieme, "Scalable Classification of Repetitive Time Series Through Frequencies of Local Polynomials," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 6, pp. 1683–1695, Jun. 2015.
- [19] D. Zhang, K. Lee, and I. Lee, "Periodic Pattern Mining for Spatio-Temporal Trajectories: A Survey," in *2015 10th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, Nov. 2015, pp. 306–313.
- [20] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric Model for Background Subtraction," in *Computer Vision — ECCV 2000*, ser. *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, Jun. 2000, pp. 751–767.
- [21] S. Noh and M. Jeon, "A New Framework for Background Subtraction Using Multiple Cues," in *Computer Vision – ACCV 2012*, ser. *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, Nov. 2012, pp. 493–506.
- [22] Y. Wang, P. M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, "CDnet 2014: An Expanded Change Detection Benchmark Dataset," in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Jun. 2014, pp. 393–400.