Efficient and accurate background subtraction using multilevel background model

Hossein Soleimani

1Department of Electrical and Computer engineering, Isfahan university of technology, Isfahan, Iran
Email: h.soleimani@ec.iut.ac.ir

ABSTRACT
Background subtraction is an essential step of intelligent video surveillance and has got a lot of interest among researcher community in recent years. It has a critical impact on the performance of object tracking and activity analysis. In this paper we propose a new multi-level background modeling to overcome dynamic background problem. At first, an image is segmented to foreground or background in larger window(12 × 12) level and results are refined by smaller windows(6 × 6 and 3 × 3) level and pixel based operation. Our pixel based segmentation section uses VIBE method, which is fast and needs less memory to conserve background model components, with some modification to detect shadows. Subtraction is done in coarse levels firstly, and resulted foreground are investigated more by smaller windows(fine levels). This makes the algorithm to be more efficient. A new once-off background changing detection and model updating is proposed to make our algorithm as accurate as possible. The last part of our algorithm is enhancement where, we have used morphological operators in order to improve the subtraction quality. The approach provides us with many advantages compared to the state-of-the-art. Experimental results clearly justify our strategy.

KEYWORDS
background subtraction, background modeling, moving object detection, shadow detection, model updating.

1. Introduction

Background subtraction is an essential issue in visual surveillance, human motion analysis and human-machine interaction. It deals with detecting instances of moving object of various classes (such as humans, animals, buildings, or vehicles) in digital images or frame sequences [1], [2]. Background usually contains nonliving objects that remain passive in the scene. The background objects can be stationary objects, such as walls, doors and room furniture, or non-stationary objects such as wavering bushes or moving escalators, waving tree or changing lights. Difficult issue in background subtraction is that the background is usually non-stationary. Moreover, when moving objects are involved in a scene, there might be some shadows cast or changes in the lighting, which could result in incorrect detections [3], [4]. The appearance of background objects have changes over time, e.g., the changes in brightness caused by changing weather conditions or the switching on/off of lights [5], [6]. Therefore, a good background subtraction algorithm should be robust against, gradual and once-off background changing and also handle shadows and non-stationary backgrounds [7]. In the state-of-the-art algorithms, lots of false alarms or undesired pixels were detected as foreground or background due to environmental changes or dynamic background or sudden illumination change.

Our main contributions are:(1) a new multi-level background modeling is proposed to overcome dynamic background problem which is a challenging issue and most of the existing algorithms fail to address it completely. At first, an image is segmented to foreground or background in larger window level and results are refined by smaller windows and pixel based operation. This makes the algorithm to be more efficient and applicable for real time application. Our pixel based segmentation section uses VIBE [8] method, which is fast and needs less memory to conserve background model components, with some modification to detect shadows. (2) A new once-off background changing detection and model updating is proposed to make our algorithm as accurate as possible. Finally, an enhancement is done where, we have used morphological operators in order to improve the subtraction quality.

2. Related works

Background subtraction purpose is to generate a reliable background model to detect moving objects efficiently [9], [10]. Some background subtraction methods include simple background subtraction (SBS), running average (RA), Σ − Δ estimation (SDE), multiple Σ − Δ estimation (MSDE) [11], cosine
transform (DCT) domain [12], and temporal median filter (TMF) [13]. Most of these basic subtraction algorithms have difficulty with non-stationary backgrounds and cannot adapt to fast backgrounds changing. We continue with some state-of-art and popular background subtraction methods.

Most foreground detection methods are pixel-based, and one of the preferred one is the mixture of Gaussian (MOG) [14]. Stauffer-Grimson [15] proposed the MOG by using multiple Gaussian distributions to represent each pixel in background modeling. This background pixel model is able to cope with the multimodal nature of many practical situations and leads to good results when repetitive background motions, such as tree leaves or branches, are encountered. Since its introduction, the model has gained vast popularity among the computer vision community [16], [17]. The advantage is to overcome non-stationary background and thus provides better adaptation for background modeling. Yet, it has some drawbacks. One of which is that if the standard deviation is too small, a pixel may easily be detected as foreground, and vice versa. Another defect is that it cannot remove shadows.

Li et al. [18] proposed a statistical modeling to handle complex background. They constructed three different static-color, static-gradient and dynamic model to static and dynamic background to make foreground detection as efficient as possible. A Bayes decision rule is derived for background and foreground classification based on the statistics of principal features. Although this approach removes some of drawbacks of previous method, it is time consuming and cumbersome task.

Kim et al. [19] presented a real-time algorithm for foreground detection, which samples background pixel values and then quantizes them into codebooks (CBs). This approach can improve the processing speed by compressing background information. Moreover, two features, layered modeling/detection and adaptive CB updating, are presented for further improving the algorithm. Guo et al. [20] used the concept of block-based CBs to construct two different background models. Codebooks are constructed to be able to capture background motion over a long period of time with a limited amount of memory. Therefore, codebooks are learned from a long training sequence and a codebook update mechanism (described in [19]) helps the algorithm to evolve with the lighting conditions after training phase. However, it should be noted that the codebook algorithm does not create new code-words (which is necessary if a region of background replaced by new background), and this may introduce some errors if permanent structural changes occur in the background (for example, in the case of newly freed parking spots in urban outdoor scenes).

Lin [21] has proposed a background subtraction model for complex background over spatial and temporal domain. At any location of the scene, they extract a sequence of regular video bricks, i.e., video volumes spanning over both spatial and temporal domain. For each sequence of video bricks, they pursued the subspace by employing the auto regressive moving average model that jointly characterizes the appearance consistency and temporal coherence of the observations. Hati [22] has proposed an intensity range based method for pixel location in the background in order to handle issues related to change in illumination as well as motion in the background.

VIBE [8] is another state-of-art algorithm, differs from those methods which are based on the classical assumption that the oldest values should be replaced first. When a pixel is found to be part of the background, its value is propagated into the background model of a neighboring pixel too. This approach will be described in details in the next sections.

Considering all drawbacks of previous methods, we propose a new fast and accurate background subtraction scheme that extracts the foreground in different levels. In each level, Image is divided to non-overlapping windows with constant size and for each window a background model is constructed through initial frames. The organization of the proposed method to find foreground pixels is as follows. In first level, image is divided to non-overlapping 12 × 12 windows and image segmentation (background or foreground) is done for each window separately. Foreground windows are analyzed again in four (6 × 6) windows separately. Again we separate foreground window (6 × 6) to four (3 × 3) windows and do segmentation. Finally, window (3 × 3) which is classified as foreground is investigated by pixel based segmentation. This strategy helps algorithm to be applicable in real time applications. Also window based background subtraction is a good approach to deal with non-stationary backgrounds [19]. Our pixel based segmentation uses VIBE[8] idea which needs less memory to save previous frames information. This method has difficulty with shadows and non-stationary backgrounds. Using image texture we proposed a new shadow detection method which was missed by most of stat-of-art algorithms. Finally a new once-off background changing procedure is applied to update background model in all levels if a region of background is replaced by a new structure. There is a different and independent model for all blocks and pixels of image. Segmentation in each level is done by checking if a window or pixel fits the corresponding model or not. Comparing to other method, ours proved to be of higher efficacy. This was indicated by both qualitative and quantitative results through analysis using a range of natural video sequences.

The remainder of our paper is organized as follows. Window-based and pixel-based background model construction are discussed in section 3. In section 4, updating, searching, matching and adaptive thresholding are described. Section 5 explains some final refinements which foreground mask needs and some quantitative and visual experiments are presented in section 6. Finally, the conclusion is drawn in section 7.

3. Background model construction

Most of background subtraction algorithms build a model using K initial frames. The goal is to construct and maintain a statistical representation of the scene that the fixed-camera sees. A model contains, feature vectors and any other information (like importance of feature vectors and their last updating time) extracted from previous frames to judge current pixel or win-
In this section we describe background model construction procedure and introduce any features needed to be extracted. To solve non-stationary background problem, window-based model is applied. Non-stationary backgrounds contain many motions which results significant color difference in pixel locations. It behaves like noise and yields to different intensities for a pixel. To remove noise from a signal or an image, filters should be adapted. Here, we apply mean filter to overcome dynamic backgrounds. Although different intensities appear in a pixel location, their mean value is almost constant in a region (or window). For example, leaves of a tree in windy condition always exist in part of image and mean of intensities in that area is constant. Therefore, mean value of intensities in a window is a suitable feature to tackle dynamic backgrounds. Now the question is what size is appropriate for a window? Smaller windows would not be able to handle this problem (non-stationary background) and leads to errors. Conversely, larger windows do segmentation in coarse level and may miss some details of moving objects (real foreground). Hence it is reasonable to do segmentation firstly in coarse levels (by larger windows) and refine it by smaller windows and finally in pixel level. We used different strategy to build window based and pixel based models. It is described in following subsections.

### 3.1 Window-based model construction

Using first \( K = 50 \) frames of video sequence, a multilevel background model is built. Incoming frame \( F_t \) is divided to non-overlapping \( 12 \times 12, 6 \times 6 \) and \( 3 \times 3 \) windows and for each obtained window, mean value of image is calculated in each R, G and B color channels.

\[
\mu_{Col} = \frac{1}{(M \times M)} \sum_{i,j} F_t^{Col}(i,j), \ Col = R, G, B
\]

Hence, each window \((M \times M \text{ and } M = 3, 6, 12)\) has a three dimensional feature vector.

\[
v = \{\mu_R, \mu_G, \mu_B\}
\]

Not that for frame \( F_t \) of size \( P \times Q \), we have total number of \((P/12 \times Q/12) + (P/6 \times Q/6) + (P/3 \times Q/3)\) different feature vectors. This number of models is required in a frame after the background model is constructed. A \( model^M \) contains feature vectors \( v_i, W_i \) (importance of \( i^{th} \) vector), number of feature vectors \( N^M, M = 3, 6, 12 \).

Figure 1 shows flow chart of proposed method to build a model by incoming feature vector \( v \) for a typical \( model^M \) and \( M = 3, 6, 12 \). Vector \( v \) is compared with existing vectors in model by a matching function and if it is similar to one of these vectors, say \( v_i, v_j \) and other component of the model are updated as follow:

\[
v_i = (1 - \alpha)v_i + \alpha v, \ W_i = W_i + \alpha
\]

\[
W_j = W_j - \frac{\alpha}{10}, \ j \neq i
\]

Importance of feature vectors which do not match the current vector \( v \) is reduced by rate \( \frac{\alpha}{10} \). This helps the model to lessen weights of vectors with less contribution describing the background region. If it is first frame or vector \( v \) is new for \( model^M \), (it does not match to any vectors in model) it should be added to model.

\[
N^M = N^M + 1, \ v_{N^M} = v, \ W_{N^M} = \alpha
\]

This kind of updating strategy is like estimating probability density function(pdf) of intensity variable in a window location [18]. In this paper, \( \alpha \), learning rate, set to \( 0.02 \) and \( N^M = 0 \) for all image windows \((M \times M)\) in first frame. It is seen that, the more a feature vector is repeated in a window, the higher \( W_i \) it has. Conversely, some vectors in a model are not frequently repeated and has lower \( W \) as a result.

To check whether vector \( v \) matches to vector \( v_i, i = 0, 1, ..., N^M \) from corresponding \( model^M \) or not, most of articles calculate Euclidean distance of vectors and compare it with a constant threshold. Constant threshold is not appropriate to result in good segmentation in all region of a frame. Frame contains, static and dynamic pixel with different illuminations. In general, higher threshold value create more false positives and lower values result in false negatives. To overcome this problem, we propose a new adaptive thresholding method (described further in paper) in all three color channels (R,G,B). In other word, \( model^M \) has three different adaptive thresholds, \( \lambda^M_R, \lambda^M_G, \lambda^M_B \) which are updated in each frame.

To check similarity of vectors \( v \) and \( v_i \) from \( model^M \), we use
match($v, v_i$) $= \begin{cases} 
\text{True} & \text{if} \begin{cases} 
\text{abs}(v(1) - v_i(1)) < \lambda^M \text{R} \\
\text{abs}(v(2) - v_i(2)) < \lambda^M \text{G} \\
\text{abs}(v(3) - v_i(3)) < \lambda^M \text{B}
\end{cases} \\
\text{False} & \text{otherwise.}
\end{cases}
$(5)$

where $v = \{\mu, \mu_G, \mu_B\}$ and $v(1) = \mu, v(2) = \mu_G$ and $v(3) = \mu_B$. This matching function detects any change in each three channels.

If there is only one match between $v$ and $v_i$, pixel is classified into the background. Search process starts from first vector of model, $i = 1$. Most of the time, there are no big changes between pixels located at the same place from successive images (90 percent of the pixels are in the background on average). Therefore, if vector $v_i$ matches to vector $v$, we swap vector $v_i$ and all it’s components(weight) with vector $v_0$ in order to speed up the searching process. Then, for the next frame, the algorithm has a high probability to find a match at position 0 and to skip $N^M - 1$ unnecessary tests.

After, training and construction all models, still there are two problems. 1) how many vectors should a model have or what is maximum value of $N^M$. 2) Initial $K$ frames, which were used to build background model may contain moving objects and their feature vector be added to model which is not desired.

In pixel based segmentation, 6.5 and 1 feature vector(on average) is sufficient to describe a pixel in dynamic and static background, respectively[19]. Therefore, it is reasonable to restrict $N^M$ in order to reduce complexity and memory we need to save model components. It is clear that window of image, needs less feature vectors respect to pixel to be described in model and also larger windows require more less vectors to indicate mean value of image in a region. Experimentally, we limited $N^3, N^6, N^{12}$ to $N^3_{\text{max}} = 8, N^6_{\text{max}} = 6, N^{12}_{\text{max}} = 4$ respectively. For next frames after training phase, if we decide to add a vector to model$^M$ and $N^M$ exceeds number of $N_{\text{max}}$, vector with minimum importance is replaced by new vector.

Moving objects remain in a block of image for a fewer frames. Hence, their feature vectors have less importance value in corresponding model. Strictly speaking, their importance together(vectors which belong to moving objects) is less than 20 percent of whole importance of feature vectors in a model. From constructing model approach, we know that whole importance of feature were zero at first and it increments by $\alpha = 0.02$ in each frame. Since whole frames in training phase is $K = 50$, whole importance of vectors in a model is approximately equal to 50 * 0.02 $\approx 1$. Therefore, effect of moving objects is less than 0.2. To address moving object problem, we sort vectors in descending order based on their importance and do the following refinement to each model

$$N^{M}_{\text{optimal}} = \text{argmin}_n (\sum_{i=1}^n |W_i| > 0.8), n <= \text{min}(N^M_{\text{max}}, N^M)$$
$(6)$

Remove vectors $v_i, i > N^M_{\text{optimal}}$ and set$N^M = N^M_{\text{optimal}}$.

If there are many moving object during training phase, many redundant vectors may be added to model. In this case we can decrease 0.8 to 0.7 or lesser in equation $(6)$ to remove more un-useful feature vectors. Also If there are almost no moving objects during the background model construction, then this threshold has no effect on the vectors number. These modified window-based models will be applied to segment background from foreground in next frames.

### 3.2 Pixel-based background construction

During training phase, for each pixel a model is built. Constructing pixel model is different from that with window-based. As it was mentioned above, our pixel based segmentation uses VIBE method[8]. This method models each background pixel with a set of samples instead of with an explicit pixel model. Consequently no estimation of the pdf of the background pixel is performed, and so the current value of the pixel is compared to its closest samples within the collection of samples. This is an important difference of VIBE in comparison with other algorithms. A new pixel intensity is compared to background samples and should be close to some of the sample values instead of the majority of all values. The underlying idea is that it is more reliable to estimate the statistical distribution of a background pixel with a small number of close values than with a large number of samples. VIBE only saves some samples intensity for each pixel as background model and it does not require importance of a sample, updating time or any other background component.

Like window-based construction, pixel-based model is constructed using initial $k = 50$ frames. For a pixel in location $x$ in a frame and with intensity value $v(x)$, background is modeled by a collection of $N$ background sample values,

$$M(x) = \{v_1, v_2, v_3,...,v_N\}$$
$(7)$

taken in previous frames. $N$ is maximum number of samples in each model and it is 20 in original paper. In frames 1 to $N = 20$, all pixels value in location $x$ is added to model and no matching function is applied. In next frames, no sample is added to model but, one of samples in model may be replaced by current sample value. To update model or classify pixel, intensity of pixel, $v(x)$ is compared to samples in model and if number of matches is more than a threshold, model is updated. In other word, it is compared with the closest values within the set of samples by defining a sphere $SR(v(x))$ of radius $R$ centered on $v(x)$.

The pixel value $v(x)$ is then classified as background or capable to be added to model if the cardinality, $\sharp$, of the set intersection of this sphere and the $M(x)$ is larger than or equal to a given threshold $\sharp_{\text{min}} = 2$, see figure.2. Since, we only interested in few matches, $\sharp_{\text{min}} = 2$, algorithm can be stopped once $\sharp_{\text{min}}$ matches are found. Using proposed strategy for window based model updating, we swap first $\sharp_{\text{min}}$ samples of model with those samples which have matched to current sample. This is done to reduce unnecessary tests and make the algorithm much faster.

Accuracy of this algorithm depends on $R$ and $\sharp_{\text{min}}$. $R$ is a constant threshold by which samples similarity is judged. As
it was mentioned above, constant threshold does not work well for all region of image with different illuminations and static or dynamic pixels. Therefore, we applied adaptive threshold in three color channel separately and use equation (5) as match function to see if \( v(x) \) is similar to \( v_i \) or not. Using Euclidean distance, to check similarity, needs calculating one distance (3 subtractions, 3 multiplies, 2 summations) and one comparison while our proposed methods only needs 3 comparisons. It should be noticed that proposed algorithm needs much memory to save adaptive thresholds for each pixel. Original VIBE needs to save \( N \) intensities in three channels (R,G,B) and our approach needs to conserve \( N \) intensities and an adaptive threshold value.

For pixel in location \( x \) if \( \bar{z}_{\min} = 2 \) samples matches to \( v(x) \), pixel is known as background and it should be added to model. Since all \( N = 20 \) locations of model filled in previous frames, one of the samples in model should be replaced by new pixel value \( v(x) \). The classical approach to the updating of the background history is to discard and replace old values after a number of frames or after a given period of time; or, as it is pointed in ground history is to discard and replace old values after a number of frames. This component is updated in each frame in which model suppose to be updated. First, background mean \( bm^M = \{ bm^M_R, bm^M_G, bm^M_B \} \) vector is calculated in each model, using feature vectors. As it mentioned before, background feature vectors show background history and contain almost all intensities of scene in corresponding window or pixel location.

\[
\begin{align*}
    bm^M & = \begin{cases} 
        \frac{\sum_{i=1}^{N} v_i \cdot w_i}{\sum_{i=1}^{N} w_i} & M = 3, 6, 12 \text{ and } i = 1, 2, ..., N^M \\
        \frac{\sum_{i=1}^{N} v_i}{N} & M = 1 \text{ and } i = 1, 2, ..., N
    \end{cases} 
\end{align*}
\]

Note that for \( M = 1 \), pixel model, all samples have same importance and there is no weight vector. After that, absolute difference of \( bm^M \) and current vector \( v \) is obtained as \( \Delta = | bm^M - v | = \{ \Delta_R, \Delta_G, \Delta_B \} \).

Finally, we update \( \lambda^M \) showing intensity variance of each channel in a block or pixel location during time.

\[
\lambda^M = \begin{cases} 
    (1 - \alpha) \lambda^M + 3 \alpha \Delta & \text{if } \Delta < \frac{bm^M}{4} \\
    \lambda^M & \text{otherwise}
\end{cases}
\]

In order to further extend the size of the time window covered by a pixel model of a fixed size, VIBE resorts to random time sub-sampling. Authors of VIBE do not update model in each frame. They select some frames randomly and apply updating procedure in those frames. By making the background update less frequent, the expected lifespan of the background samples are extended. To do this they randomly choose an integer form \( \{0, \varphi - 1 \} \), \( \varphi = 16 \), and if the selected number is equal to 0, they update model. This strategy may not be appropriate for dynamic backgrounds but, since, in our case, dynamic background is removed in window-level, it does not make problem and decreases updating complexity.

Our window-based subtraction approach suppose to remove non-stationary backgrounds which leads to less dynamic backgrounds in pixel-based subtraction. Therefore we need less samples to completely describe a pixel and we can reduce \( N \). To make our algorithm as simple as possible, we set \( N \) to 10 and \( \bar{z}_{\min} \) to 1 which is best choice for static backgrounds [8].

### 3.3 Adaptive thresholding

Generally, illumination of scene changes over time and different regions of image have different backgrounds and lightening condition. Constant threshold may not cover all of these variations. To overcome this we suggest to use adaptive thresholds. This threshold should reflect amount of intensity variation in each image region.

A new component \( \lambda^M_1 \) is added to each background model \( M = 1, 3, 6, 12 \) as threshold, representing background intensity variation of a block or pixel of image through recent frames. This component is updated in each frame in which model suppose to be updated. First, background mean \( bm^M = \{ bm^M_R, bm^M_G, bm^M_B \} \) vector is calculated in each model, using feature vectors. As it mentioned before, background feature vectors show background history and contain almost all intensities of scene in corresponding window or pixel location.

\[
\lambda^M = \begin{cases} 
    (1 - \alpha) \lambda^M + 3 \alpha \Delta & \text{if } \Delta < \frac{bm^M}{4} \\
    \lambda^M & \text{otherwise}
\end{cases}
\]

If one of \( \Delta_R, \Delta_G \) or \( \Delta_B \) is greater than a threshold (experimentally \( \frac{bm^M_R}{4}, \frac{bm^M_G}{4}, \frac{bm^M_B}{4} \)), adaptive threshold is not updated. This high value shows a great change in window of current frame and this means that it may not belong to background pixel, i.e. it is
because of moving object. In first frame $\lambda^M$ is set to 10. These adaptive thresholds may increase during time and get high values. This can be because of wrong segmentation or existence of many moving objects in a region for a long time. As a result, the corresponding window or pixel always will be considered as background. To address this concern we restrict all three elements of $\lambda^M$ to $[5,15]$ and $[7,25]$ in window and pixel level operations, respectively.

$\lambda^W_G$, adaptive threshold of green color of a sample frame is shown in figure 3. As it was expected, most bright pixels (value $=25$), belong to dynamic regions (leaf waves). In contrary to dynamic pixels, stationary regions take low threshold values.

4. Background subtraction

Our proposed subtracting method is to handle non-stationary backgrounds and reduce computation complexity. Also, it rejects shadows created by moving objects and detects once-off background changing. After training phase, we have four level models, $M = 12, 6, 3, 1$, describing background scene. All window based models are pruned by restricting maximum vector number and discarding feature vector with less importance value which may belong to moving objects. These models are utilized for the introduced multilevel background subtraction. First, image is investigated by window-based models, i.e., all pixels of image in a specific window are judged by corresponding window model. If that window is classified as background, no further process is needed which boosts algorithm speed. Window whose feature vector does not match to any vector, $v^i$, in model, is processed by smaller windows and finally pixel based models. The pixel-based models at the end of the proposed system can also classify the pixels into tree types, background, foreground and shadow. In following sections we explain window-based and pixel based background subtraction in detail.

4.1 subtraction

Whole schema of proposed subtraction in window based level, is shown in figure 4.

Initially, similar to the construction of the window-based background model, incoming frame $F_t$, $t = K + 1, K + 2, ...$ is divided to $12 \times 12$ non-overlapping windows and each window is processed separately. As it is seen in algorithm flowchart, for a window $12 \times 12$, feature vector $v$ is obtained by equation (1). Using matching function, equation (5), this vector is compared with vectors, $v^i (i = 1, 2, 3, ..., N^12)$, in corresponding model. If it matches to one of $v^i$s, the whole window is classified as background and the first $v^i$ which is similar to vector $v$ is updated by equation (3). Algorithm stops processing once first $v^i$ met all three adaptive thresholds $(\lambda^R_{12}, \lambda^G_{12}, \lambda^B_{12})$ condition which will decrease processing time. This means that it is not necessary to check all $v^i$s and find the most close match to judge the window. Note that before classifying this window, adaptive thresholding procedure, described in section 3.3., is applied to update $(\lambda^R_{12}, \lambda^G_{12}, \lambda^B_{12})$ values and it does not matter whether window is background or foreground.

If feature vector $v$ did not match to any $v^i$s in $Model^{12}$, window is passed to second level and it is processed in $6 \times 6$ window level. Window is divided to four $6 \times 6$ non-overlapping windows and equation (1) is applied to obtain vector of each window. Vector $v$ belonging to window $6 \times 6$ compared to corresponding model vectors $(v^i, i = 1, 2, ..., N^6)$ and is classified as background or foreground by procedure explained for a window $12 \times 12$ in first level (see flowchart in figure 4). Again, if window $6 \times 6$ was known as foreground it is passed to third level to be judged by finer window level, i.e., $3 \times 3$. In this way, the window-based stage can remove most of the noise and dynamic backgrounds; however, it has low precision. This low accuracy is because of window size. The smaller the window is, the more accuracy and precision the algorithm has. Here, the smallest window size is $3 \times 3$ and as a result the detected foreground regions are composed of $3 \times 3$ windows. In other word the boundary of these regions are not extracted accurately. Therefore, this regions should be refined in pixel level. Result of applying these three window level is depicted in figure 5 and as it is seen most of background area, including static and dynamic, is subtracted in first level, by $12 \times 12$ window.

The main contribution of the window-based operation is to reduce the redundant foreground detection operations and to reduce the noise in the dynamic background. This is because of the mean value which is considered as feature in the block-based stage. Mean feature is a robust feature against noise or dynamic background which shows noise behavior. In addition, applying windows in descending order, based on their size, increases the processing speed. The result from the window-based stage is then fed into the pixel classification, introduced in the following section. Pixel-based operation is not applied while a window has been clarified as background during the window-based subtraction; which makes the proposed method less time consuming.

As it was mentioned above, if a window is segmented as background, it is not evaluated by smaller windows or pixel-based operation. Now if there is no moving objects for a number frames, for example 10 frames, all region in frame will be segmented as background by $12 \times 12$ window models and only statics of these models will be updated. This means that information being necessary to detect moving objects in next frames by smaller windows and pixel level is neglected. Therefore,
to conserve our models in all levels to be representative, we count number of times that a window is classified as background consecutively, see figure 4,\( B_{12} \text{ count}, B_{6} \text{ count}, B_{3} \text{ count} \).

If these values,\( B_{12} \text{ count}, B_{6} \text{ count}, B_{3} \text{ count} \), exceed update threshold \( Thr \), statics of all models belonging to that window should be updated. For example, if \( B_{12} \text{ count} \) is equal to \( Thr \), models of all four \( 6 \times 6 \), sixteen \( 3 \times 3 \) windows and 144 pixels in a \( 12 \times 12 \) window should be updated. The number \( Thr \) represents that the system will update the window models, with size smaller than \( M \), and pixel information to ensure that no data are missed after there are \( Thr \) successive matches with a block of size \( M \times M \). A smaller \( Thr \) means that less information is missed, yet it also leads to a higher computational complexity. Depending on background properties, having more dynamic or static regions, \( Thr \) can take small or large values, respectively. We set \( Thr \) to 5 in this paper.

4.2 Pixel based subtraction

Pixel level segmentation is adapted to refine output of window-based segmentation. Intensities in channels (R,G,B) are consid-
which has passed all four levels and has not been considered as shadow or foreground in first stage and only those pixels are passed to final shadow detecting algorithm. We set threshold values of foreground and background, is processed to check if this pixel belongs to real moving object (foreground) or shadow.

Shadows in images have long been disruptive to certain computer vision applications such as edge detection, image segmentation, object recognition, video surveillance, and stereo registration. Shadows occur when direct light from a light source is partially or totally blocked by an object. One of the simple assumptions that can be used for detecting shadows is that pixel intensity decreases in the shadow regions since they are blocked directly from the illumination source [23]. When a shadow appears in a background region, intensity in all three channels decreases by almost same amount. In other word, for pixel located in shadow we have:

\[ r \cong g \cong b \]

\[ r = v(1) - v_i(1), g = v(2) - v_i(2), b = v(3) - v_i(3) \]

where \( r, g, b < 0 \) and \( v(1), v(2), v(3) \) are image intensities of pixel \( x \) in channels R, G, B, respectively. But using only this feature is not a reliable method for shadow detection. However it can be used as a first stage to reject some non-shadow regions and reduce the algorithm complexity. If all three differences, \( d_1 = |r - g|, d_2 = |r - b|, d_3 = |g - b| \), among \( r, g \) and \( b \) are less than threshold \( S_{low} \), more likely pixel \( x \) belongs to background. On the other hand if one of \( d_1, d_2, d_3 \) is greater than a threshold \( S_{high} \), it can confidently be said that \( x \) belongs to moving object (foreground). Pixels not satisfying these two conditions are passed to final shadow detecting algorithm. We set \( S_{high} \) and \( S_{low} \) to 20 and 5 respectively. Our experiments showed that more than 80 percent of pixels which did not match our model, considered as shadow or foreground in first stage and only 20 percent of them need to be checked by shadow detection algorithm. This clearly shows how the suggested conditions reduce algorithm complexity and speed it up.

To avoid complexity, we applied a simple shadow detection algorithm proposed in [24]. Assuming \( B \) is background image in grayscale and \( I_t \) is grayscale version of frame \( F_t \), a region \( R \) with \( 3 \times 3 \) pixels centered at each shadow pixel candidate \( x \) with coordinate \( (i, j) \) is considered and pixel is classified as a shadow pixel if:

\[ \text{std}_R(x) = \sqrt{\frac{1}{9} \sum_{n=1-j}^{i+1} \sum_{m=1-j}^{j+1} \left( \frac{I_t(n,m)}{B(n,m)} - \mu \right)^2} < L_{std} \]

\[ \text{and } L_{low} < \left( \frac{I_t(i,j)}{B(i,j)} \right) < 1 \]

where \( \mu \) is mean value of \( \frac{I_t(n,m)}{B(n,m)} \) in region \( R \) and \( L_{low} = 0.5, L_{std} = 0.05 \). In [24], background image \( B \) is constructed by temporal median filter. This means that, \( B(i,j) \) is median value of frames in location \( (i, j) \). Hence we can calculate \( B(i,j) \) as median value of conserved samples value in pixel model at \( (i, j) \). Note that our pixel models contains samples value which their values are not updated and only replaced by new matched sample value. Result of pixel based background subtraction and shadow removing for sample image (see figure 5) is illustrated in figure 6.

### 4.3 Once-off background changing detection

When a new object is added to background or one of background objects leaves the scene, once-off background change (sudden change) happens and the new background appearance becomes dominant soon after the change. Similar to moving object, sudden background changing feature vector does not match to model and it is detected as foreground until its feature vector is added to model. For instance, moving car which is parked in scene, becomes a stationary background. Assuming that a moving object does not stand in a location for more than a time period (let say 10 second or 100 frames for a video with 10 fps), sudden changes in background can be detected. To adapt this situation, we count number of times \((F_{12\_count}, F_{6\_count}, F_{3\_count}, F_{1\_count})\) that a window (in all three levels) or a pixel is classified as foreground successively. If the number is greater than a threshold \( D_M \) (\( M = 12, 6, 3, 1 \)), the feature vector \( v \) is added to corresponding model. For pixel-based models, one of samples in model is removed randomly and new sample is replaced while for window-based model, we find feature vector
We collect a number of challenging videos to validate our approach, which are publicly available or from real surveillance systems. Four of them (Bootstrapping, Gradual Light, Waving Trees, Camouflage) are from Wallflower database [27] and the rest including Meeting room, Campus, Water surface, Airport and Fountain are from star dataset, available at [28]. Also we used Camp4outdoor’ available at [31] to show how our algorithm works in shadowed regions. Most of the videos include thousands of frames, and some of the frames are manually annotated as the ground-truth provided by the original databases.

In the experiments, we use the first 50 frames of each testing video to initialize our system (i.e. to perform the initial learning), and keep model updated in the rest of sequence. All other competing approaches are executed with the same setting as our approach.

The performance of the proposed method is evaluated with respect to various criteria. Three criteria, including F-measure ($FM$), true positive rate (or Recall) ($R$) and precision ($PR$) [26], are employed.

$$FM = \frac{2TP}{2TP + FP + FN}, R = \frac{TP}{TP + FN}, PR = \frac{TP}{TP + FP}$$

(13)

where $TP$ is true positives (real foreground pixels), $FN$ is false negatives (false background pixels) and $FP$ indicates false positive (false foreground pixels).

6. EXPERIMENTS

6.1 Experimental Results

We compared our method, multilevel background model (MBM), with five state-of-the-art online background subtraction algorithms including Gaussian Mixture Model (GMM) [14] as baseline, Improved Gaussian Mixture Model (IGMM) [25], VIBE [8], Code Book (CB) [19] and Bayesian [18]. We implemented Bayesian and CB by our self and used available codes for other methods. Implementation of GMM and IGMM are available in [29] and VIBE code can be find in [30].

A number of sampled results of background subtraction are exhibited in figure 8. All images are $160 \times 128$. First row are original images (selected frames), second row shows hand-segmented foreground masks and the other rows are results of different methods. Compared with all mentioned five methods, GMM, IGMM, CB, VIBE and Bayesin, the proposed algorithm provides better performance. The most significant feature of MBM is superior performance in highly dynamic backgrounds. This is seen in Campus and Waving Trees, which suffers from a serious dynamic background, frame sample in figure 8. For Water surface frame sample, VIBE and proposed MBM efficiently removed all background pixels while VIBES has some false negative errors, see feet of the body in the picture. This is because of constant threshold ($R = 20$) which VIBE uses for segmentation. Converse to VIBE, MBM uses adaptive thresholding procedure and produces a different threshold values for each model in background.

Last column in figure 8 shows algorithm’s results in a stationary background with shadows. As it is illustrated, all five methods, GMM, IGMM, CB, VIBE and Bayesin, detected foreground pixels, near to Ground Truth, while they introduced...
some false positives in shadowed regions. Contrary to these algorithms, MBM removed shadow pixel using proposed shadow detection strategy. To illustrate the efficiency of shadow detection, FM measure, resulted by our algorithm, is depicted for video sequence ‘Shadow’ in figure 7 with and without shadow detection (frames 631 to 640 of this video are segmented by hand as ground truth).

The F-scores (%) over all 10 videos are reported in Table 1, where the last column reports results of our method. As it is seen in all videos with dynamic backgrounds, like ‘Waving trees’ and ‘Meeting Room, or shadow, our method outperforms any other algorithm. Also, it shows better performance in gradual lightening changing. It is because of the shadow detection algorithm adapted to our method. As it was referred in section 4.2 all three (R,G,B) colors change by same amount(r ≃ b ≃ g) when the light decreases(by shadow for example) or increases(by sunrise for example). This kind of changes is detected by shadow detection approach helping the algorithm not to introduce false positive errors.

On of the drawbacks of proposed algorithm can be find in videos ‘Airport’ and ‘Bootstrapping’. These videos contain moving objects with small sizes. Most of small objects are detected as background in first level (12 ∗ 12 window) which leads to low accuracy in this videos. Since these objects completely have been removed from foreground mask, they are not reconstructed in final refinement part, section 5, of algorithm. Therefore, detecting all backgrounds, dynamic and static, has the cost of some FN errors. As a result proposed algorithm has much less FP in respect to FN errors.

To compare the MBM with other algorithms and investigate more regarding its performance, we summarized R and PR scores in Tables 2 and 3. In general, like F − Score, these criteria emphasis on superiority of our algorithm too. In average, Bayesian and GMM have the best R − Score, less FN, while they perform poor in dynamic backgrounds and produce high number of FPs. On the other hand, proposed MBM shows the best results by PR-score and has minimum FPs.

6.2 Discussion

Except for the reliability discussed in previous section, the processing time and algorithm speed is also an important issue in real time applications. Comparing with VIBE which is one of the rapid algorithms and can process 200 FPS at a resolution of 320 ∗ 240 [8], MBM has lower speed and process 70 FPS in average. This value is more than satisfactory and easily meet requirements to be applied in real time video surveillance applications. Our experiments showed that average speed of GMM, IGMM and CB for such resolution is less than 20 FPS. All experiments were done by a desktop compute where its hardware architecture is Intel(R) Core(TM)2QUAD, (2.83 GHz) CPU and 4GB RAM.

Thr is a parameter that confines algorithm’s speed. This threshold value is used to update smaller window models while the larger window, containing all those smaller windows, classified as background for several sequential times (see section 4.1). The larger Thr lets the algorithm to process more frames per second, less update process is needed, while it may result in some errors in next frames. Probability of these errors is more in dynamic backgrounds. This means that, we can increase Thr values for static backgrounds and reach better speed with high reliability.

It seems that the window size should be adapted to the image resolution to reach a higher reliability. As it was explained in Experimental result section, larger window has capability of removing highly dynamic backgrounds while it introduces some FNs. Also, the larger window means that smaller number of models is needed to be constructed which yields in less computation during updating and searching processes. This speeds up the algorithm more. On the other hand, smaller windows do classification more precisely with less FNs while they have difficulty with dynamic background (produce more FPs) and make the algorithm to be slower. In our experiments, all images were 160 ∗ 128 and regarding FN and FP errors, algorithm speed and how much a background is dynamic, we selected window size as 12 ∗ 12, 6 ∗ 6 and 3 ∗ 3. For higher resolutions or highly dynamic backgrounds, this window size can be change to (5 ∗ 5, 10 ∗ 10 and 20 ∗ 20) or (4 ∗ 4, 8 ∗ 8 and 16 ∗ 16). Also, for images with low resolution window size should be changed to 2 ∗ 2, 4 ∗ 4 and 8 ∗ 8.

Regarding memory issue, MBM needs less memory to store models. The maximum number of components per pixel (NCPP) which MBM needs to store background model is as follow:

\[
NCPP = \left( \frac{2 \cdot N_{\text{max}}^3}{3^2} + \frac{2 \cdot N_{\text{max}}^6}{6^2} + \frac{2 \cdot N_{\text{max}}^{12}}{12^2} \right) \times 3 + (N + 1) \times 3
\] (14)

Note that it uses three channels, (R,G,B) and also stores one adaptive threshold value for each channel in a model. For a window model, vector number N_M, importance value (W_i) and mean value (μ_i), i = 1, 2, ..., N_M, are stored. We already discussed that N_{max}^3 = 8, N_{max}^6 = 6, N_{max}^{12} = 4 and N = 10 in our algorithm.

Figure 7. F-score of MBM with and without shadow detection obtained for ‘Shadow’ video sequence.
Table 1. Quantitative Results and Comparisons on The Some Complex Videos Using The F-score (%) Measurement. Best Performance for Each Video Had Been Bold-ed.

<table>
<thead>
<tr>
<th>Videos</th>
<th>GMM</th>
<th>IGMM</th>
<th>CB</th>
<th>VIBE</th>
<th>Bayesian</th>
<th>MBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waving Trees</td>
<td>51.34</td>
<td>57.51</td>
<td>81.32</td>
<td>80.22</td>
<td>56.47</td>
<td><strong>85.92</strong></td>
</tr>
<tr>
<td>Camouflage</td>
<td>67.98</td>
<td>65</td>
<td>71.21</td>
<td>72.52</td>
<td>63.48</td>
<td><strong>82.71</strong></td>
</tr>
<tr>
<td>Airport</td>
<td>43.37</td>
<td>44.56</td>
<td>56.72</td>
<td><strong>73.21</strong></td>
<td>52.63</td>
<td>68.98</td>
</tr>
<tr>
<td>Meeting room</td>
<td>62.83</td>
<td>71.73</td>
<td>68.91</td>
<td>65.29</td>
<td>69.37</td>
<td><strong>75.38</strong></td>
</tr>
<tr>
<td>Fountain</td>
<td>35.76</td>
<td>38.91</td>
<td>75.23</td>
<td><strong>79.36</strong></td>
<td>40.19</td>
<td>74.95</td>
</tr>
<tr>
<td>Bootstrapping</td>
<td>74.33</td>
<td>76.71</td>
<td>80.49</td>
<td><strong>83.13</strong></td>
<td>78.93</td>
<td>81.64</td>
</tr>
<tr>
<td>Water</td>
<td>79.43</td>
<td>85.78</td>
<td>87.45</td>
<td>88.93</td>
<td>78.24</td>
<td><strong>92.14</strong></td>
</tr>
<tr>
<td>Campus</td>
<td>43.23</td>
<td>47.79</td>
<td>63.25</td>
<td>68.11</td>
<td>40.17</td>
<td><strong>71.49</strong></td>
</tr>
<tr>
<td>Shadow</td>
<td>80.57</td>
<td>85.23</td>
<td>86.11</td>
<td>88.19</td>
<td>84.58</td>
<td><strong>89.78</strong></td>
</tr>
<tr>
<td>Gradual Light</td>
<td>53.55</td>
<td>51.08</td>
<td>68.45</td>
<td>70.37</td>
<td>57.12</td>
<td><strong>76.58</strong></td>
</tr>
</tbody>
</table>

Table 2. Results and Comparisons Using The R-score (%) Measurement. Best Performance for Each Video Had Been Bold-ed.

<table>
<thead>
<tr>
<th>Videos</th>
<th>GMM</th>
<th>IGMM</th>
<th>CB</th>
<th>VIBE</th>
<th>Bayesian</th>
<th>MBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waving Trees</td>
<td>98.5</td>
<td>97.43</td>
<td>97.12</td>
<td>92.16</td>
<td>99.46</td>
<td><strong>99.51</strong></td>
</tr>
<tr>
<td>Camouflage</td>
<td>98.92</td>
<td>94.12</td>
<td>95.01</td>
<td>90.21</td>
<td>96.78</td>
<td>97.39</td>
</tr>
<tr>
<td>Airport</td>
<td>97.91</td>
<td>96.76</td>
<td>97.18</td>
<td>97.8</td>
<td><strong>98.68</strong></td>
<td>93.63</td>
</tr>
<tr>
<td>Meeting room</td>
<td>91.03</td>
<td>88.98</td>
<td>85.67</td>
<td>82.11</td>
<td><strong>92.36</strong></td>
<td>89.47</td>
</tr>
<tr>
<td>Fountain</td>
<td><strong>88.65</strong></td>
<td>84.56</td>
<td>81.23</td>
<td>85.32</td>
<td>87.71</td>
<td>83.54</td>
</tr>
<tr>
<td>Bootstrapping</td>
<td>93.81</td>
<td>92.35</td>
<td>90.67</td>
<td><strong>93.51</strong></td>
<td>92.67</td>
<td>86.59</td>
</tr>
<tr>
<td>Water</td>
<td>98.43</td>
<td>97.18</td>
<td>99.52</td>
<td>95.41</td>
<td>96.12</td>
<td><strong>99.63</strong></td>
</tr>
<tr>
<td>Campus</td>
<td>87.34</td>
<td>85.03</td>
<td>93.24</td>
<td>82.87</td>
<td><strong>92.43</strong></td>
<td>85.78</td>
</tr>
<tr>
<td>Shadow</td>
<td>87.27</td>
<td>93.45</td>
<td>95.44</td>
<td>95.19</td>
<td>96.08</td>
<td><strong>96.18</strong></td>
</tr>
<tr>
<td>Gradual Light</td>
<td>96.54</td>
<td>95.32</td>
<td>96.92</td>
<td>94.53</td>
<td><strong>97.27</strong></td>
<td>96.58</td>
</tr>
</tbody>
</table>

Table 3. Results and Comparisons Using The PR-scor (%) Measurement. Best Performance for Each Video Had Been Bold-ed.

<table>
<thead>
<tr>
<th>Videos</th>
<th>GMM</th>
<th>IGMM</th>
<th>CB</th>
<th>VIBE</th>
<th>Bayesian</th>
<th>MBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waving Trees</td>
<td>34.71</td>
<td>40.79</td>
<td>69.94</td>
<td>71.01</td>
<td>39.42</td>
<td><strong>86.64</strong></td>
</tr>
<tr>
<td>Camouflage</td>
<td>51.78</td>
<td>49.64</td>
<td>56.94</td>
<td>60.63</td>
<td>47.22</td>
<td><strong>71.87</strong></td>
</tr>
<tr>
<td>Airport</td>
<td>27.85</td>
<td>28.94</td>
<td>40.04</td>
<td><strong>58.5</strong></td>
<td>35.88</td>
<td>54.6</td>
</tr>
<tr>
<td>Meeting room</td>
<td>47.96</td>
<td>60.08</td>
<td>57.63</td>
<td>54.18</td>
<td>55.54</td>
<td><strong>65.12</strong></td>
</tr>
<tr>
<td>Fountain</td>
<td>22.39</td>
<td>25.26</td>
<td>70.05</td>
<td><strong>74.17</strong></td>
<td>26.0</td>
<td>67.96</td>
</tr>
<tr>
<td>Bootstrapping</td>
<td>61.54</td>
<td>65.6</td>
<td>72.36</td>
<td>74.82</td>
<td>68.73</td>
<td><strong>77.22</strong></td>
</tr>
<tr>
<td>Water</td>
<td>66.57</td>
<td>76.77</td>
<td>77.99</td>
<td>65.96</td>
<td>85.69</td>
<td><strong>92.14</strong></td>
</tr>
<tr>
<td>Campus</td>
<td>28.72</td>
<td>33.23</td>
<td>47.85</td>
<td>57.81</td>
<td>25.66</td>
<td><strong>61.28</strong></td>
</tr>
<tr>
<td>Shadow</td>
<td>74.82</td>
<td>78.33</td>
<td>78.44</td>
<td>82.14</td>
<td>75.53</td>
<td><strong>84.17</strong></td>
</tr>
<tr>
<td>Gradual Light</td>
<td>37.05</td>
<td>34.88</td>
<td>52.90</td>
<td>56.04</td>
<td>40.43</td>
<td><strong>63.44</strong></td>
</tr>
</tbody>
</table>
Figure 8. Sampled results of background subtraction generated by our approach (MBM) and other competing methods.
Therefore, \textit{NCPP} of MBM is less than 35.48 component per pixel. In comparison with VIBE which needs 60 component for a pixel, \( N = 20 \) in original VIBE, or CB which its \textit{NCPP} approximately is \( 10 + 10 = 100 \)(each code book contains approximately 10 codewords and each codewords has 10 components), MBM’s memory using is more efficient.

\section{7. Conclusion}

This paper studies an effective method for background subtraction, addressing the all challenges in real surveillance scenarios including, shadows, background changing and stationary backgrounds. Proposed method is based on multilevel window and pixel model handling dynamic backgrounds. It is applicable in real time applications and uses less memory in comparison to state-of-art algorithms. However, because of using window model, suggested algorithm introduces some false negative errors especially for videos containing small moving objects. The results of our approach were analyzed both quantitatively and qualitatively in a range of natural video sequences. These analyses illustrated the efficacy of our proposed motion detection approach as not only did the accuracy rates of our procedure exceed those of other methods but also the resulting visual performance was more pleasing. Generally, the proposed method is a good candidate for intelligent moving object detection. In the future, we plan to improve this method in two aspects. First, some efficient tracking algorithms can be adapted into the algorithm to better distinguish the moving objects. Second, temporal information form next frames should be added to model in order to make the algorithm more accurate.

\section{References}


