A Novel Approach in Video Scene Background Estimation

Hamidreza Baradaran Kashani, Seyed Alireza Seyedin and Hadi Sadoghi Yazdi

Abstract—**This paper presents a novel method for background estimation in a video sequence from the function estimation point of view. The proposed algorithm, called Kernel-based Background Learning (KBL), is designed based on kernel machine joint with learning schemes. In order to estimate background using KBL algorithm, we first interpret foreground samples as outliers relative to the background ones and so propose an Outlier Separator (OS). Then, the obtained results of OS algorithm are employed in the KBL method in order to train and estimate background in each pixel. Experimental results show the high accuracy and effectiveness of the proposed method in background estimation and foreground detection for the scenes including moving backgrounds, camera shakes, and non-empty backgrounds.**

Index Terms—**Background estimation, outlier separator, kernel-based background learning.**

I. Introduction

Background estimation in video streams is often the first crucial step of information extraction in many of computer vision applications such as automatic video surveillance, people tracking, and so on. It includes the removal of moving regions as foreground objects from the background layers and maintenance of changes relevant to the background.

Many algorithms for performing background estimation/modeling have been proposed in the recent literature (see surveys in [1]–[3]). A common principle of these algorithms is to construct a model of the static or dynamic background in order to compare this model to each new frame for distinguishing the regions of unusual motion. Some of the commonly used and efficient techniques are described below.

The methods based on filtering concept are the primary methods to model or estimate background pixels. For example, median filter in [4], Σ-Δ filter in [5], [6], minimum-maximum filter [7], Wiener filter [8], single Gaussian [9] and Kalman filtering [10], [11] for updating the model presented in [9] are some approaches of this class. Although they have fast computation, ghost effects in the estimated background, lack of robustness to the non-stationary backgrounds, and crowded scenes are main disadvantages of these filter-based approaches.

Manuscript received September 8, 2009.

Hamidreza Baradaran Kashani is with the Electrical Department, Ferdowsi University of Mashhad, Iran. Postal Code: 9177948944, P.O. Box:1111, fax:+985118763301 (email: hamidreza.baradaran@gmail.com).

Seyed Alireza Seyedin is with the Electrical Department, Ferdowsi University of Mashhad, Iran. Postal Code: 9177948944, P.O. Box:1111, fax:+985118763301(email: seyedin@um.ac.ir).

Hadi Sadoghi Yazdi is with the Computer Department, Ferdowsi University of Mashhad, Iran. Postal Code: 9177948944, P.O. Box:1111, fax:+985118763301 (email: h-sadoghi@um.ac.ir)

Since a single Gaussian assumption will not hold for the probability density function (pdf) of the pixel intensity values in multimodal scenes, a mixture of Gaussians (MoG) modeling technique was proposed in [12]–[14] to solve this problem. However, there are several shortcomings for mixture learning methods. First, fast variations of background cannot be accurately modeled with only a few Gaussians. Second, even using incremental expectation maximization (EM) algorithm, parameter estimation and its convergence are noticeably slow where the Gaussians adapt to a new cluster [15]. Finally, it has relatively high computational complexity and parameters should be tuned carefully.

To address the issues of parametric methods like MoG, a nonparametric kernel density estimation (KDE) method for pixel-wise background modeling was proposed in [16]. Memory and time consuming, choosing a proper kernel bandwidth for each pixel, and the model convergence in situations where the illumination suddenly changes are major issues to be addressed in these methods. A modified version of [16] as an adaptive kernel density estimation (AKDE) algorithm was also presented in [15]. Against conventional KDE in [16], the new algorithm considered dependencies between the pixel features and applied an independent threshold to each pixel instead of a global threshold for all pixels. A good discussion of kernel estimation techniques exists in [18].

In [19], [20] Hidden Markov Models (HMM) were proposed to handle the environments such as night/day and sunny/cloudy. The model was able to learn sudden illumination changes. However, slow model training speed and sensitivity to model selection and initialization are problems of these methods [21].

Recently, background modeling based on fuzzy concepts has been presented. In [22], fuzzy background subtraction and fuzzy running average algorithms were proposed instead of classic versions of them. A novel fuzzy background subtraction based on cellular automata (CA) was also presented in [23]. In that approach, each frame sequence was considered as a 2D cellular space, and so each pixel as a cell of cellular automata. Based on this assumption, each frame sequence was modeled by a cellular automata and specific cellular automata rules applied to pixels. However, both approaches in [22] and [23] applied only to traffic scene sequences, and their performance on scenes including moving backgrounds is problematic.

After study of the aforementioned works, the issue of background estimation in this paper is considered from the viewpoint of a function estimation problem. We first suggest a new function estimator which includes a weighted type of kernel least mean square algorithm and adopts the concepts of fuzzy modeling approaches. Using the presented estimator, we then propose a new method, called kernel-based background learning (KBL) with the purpose of background estimation and foreground detection in video scenes. The presented method can remove the influences of moving objects in generated background perfectly. It can also retain the dynamic properties of the background as much as possible. Most of the ghost effects and noises are reduced in the proposed algorithm. No requirement to empty background (non-empty background problem), high accuracy, and fair computational complexity are other features of the proposed method.

This paper is organized as follows: Section II provides a brief introduction to the Kernel Least Mean Square (KLMS) algorithm. Section III explains the proposed kernel-based estimator and two examples on function estimation with noisy data and outliers. Proposed background estimation approach namely KBL using suggested estimator is described in the Section IV. Experimental results are shown in Section V. Finally, conclusions and future work are given in Section VI.

II. Kernel Least Mean Square Algorithm

The basic idea in KLMS algorithm is to perform the linear Least Mean Square (LMS) algorithm [25] in the kernel space. This space is often called feature space where the inner products can be calculated using a positive definite kernel function satisfying Mercer's conditions [24]:

$$K\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)=\left\langle \boldsymbol{\Phi}\left(\boldsymbol{x}_i\right),\boldsymbol{\Phi}\left(\boldsymbol{x}_j\right)\right\rangle \tag{1}$$

where K is a kernel function, $\boldsymbol{\Phi}$ is a mapping, $\boldsymbol{x}_i$ is a data, and $\langle .,.\rangle$ represents the inner product in the kernel Hilbert space. Without loss of generality, in this paper we will only consider the translation-invariant radial basis (Gaussian) kernel, which is the most greatly used Mercer kernel.

$$K\left(\boldsymbol{x},\boldsymbol{y}\right)=exp\left(-\frac{\left\|\boldsymbol{x}-\boldsymbol{y}\right\|^2}{2\sigma^2}\right) \tag{2}$$

where σ is the kernel bandwidth parameter. Fig. 1 shows the input vector $\boldsymbol{u}_n$ being transformed to the infinite feature vector $\boldsymbol{\Phi}(\boldsymbol{u}_n)$, whose components are then linearly combined by the infinite dimensional weight vector $\boldsymbol{\Omega}_n$ to generate output y_n. In this figure, d_n is the desired output, $e_n = d_n - y_n$ and is the error at time n. According to Fig. 1 and detail presented in [26], the final equation of KLMS algorithm for the output at n is expressed as

$$y_n=\mu\sum_{i=0}^{n-1} e_i\,K\left(\boldsymbol{u}_i,\boldsymbol{u}_n\right) \tag{3}$$

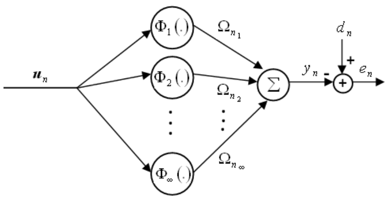

Fig. 1. Filtering scheme in the feature mapped space. Note that the feature space is infinite dimensional when Gaussian kernel is used.

where μ is the step size. More detail about KLMS algorithm can be found in [26].

III. The Proposed Kernel-based Estimator

In the proposed approach, the concept of the TSK fuzzy modeling approaches [28] is adopted. Generally, a TSK fuzzy model consists of a set of if-then rules which is obtained by applying a learning procedure to an input data set. Then, by applying fuzzy rules to each test sample and combining the results of each rule at the sample, the sample's final outcome is acquired. In fact, in TSK fuzzy model the global behavior of a system is described by using combination of simple submodels (more detail about TSK model is available in [28]). With such concept, in the proposed approach, several kernel-based models (KMs) constructed by the KLMS learning algorithm are combined to describe the global behavior of the system with a weight assigning mechanism. We can say that the proposed kernel-based estimator (Fig. 2) includes three main stages: data partitioning, learning the estimator in each partition, and estimating using a weight assigning mechanism to the learnt submodels. These three stages are stated in detail as follows.

First, we should divide the original training data data into training subsets. We can use an unsupervised learning algorithm like fuzzy c-means (FCM) clustering algorithm for this division. Let $\{(x_i, y_i), i = 1, \ldots, N\}$ be input training data set. By applying FCM algorithm to the data of x_i ($i = 1, \ldots, N$), a matrix of membership values is obtained as below

$$\boldsymbol{U}=\left[u'_{ij}\right],\quad i=1,\ldots,N\,,\ j=1,\ldots,C \tag{4}$$

where C is the number of clusters or partitions, u'_{ij} is the degree of membership of x_i in the cluster j, and x_i is the ith training data (for more detail about FCM algorithm, see [28]). Based on the obtained membership values, input data points are split into training subsets φ_k by

$$\varphi_k=\Big\{\left(x_i,y_i\right)\Big|\ k=arg_j\ max\ u'_{ij}\ \text{and the corresponding output}\ y_i\Big\} \tag{5}$$
for $i=1,\ \ldots,\ N$, $j=1,\ \ldots,\ C$.

Also we consider normal probability distribution for samples x_i of each training subset φ_k.

Second, each available training subset for any partition is an input for a KM. The output of any KM is a function which can be expressed as the equation below

$$KM^{j} = \mu \sum_{l=1}^{n_j} e_l \; K(x, x_l), \quad x \in \varphi_j, \; j = 1, \dots, C \tag{6}$$

where n_j denotes the number of training data in the jth training subset and $\{e_l \mid l = 1, \dots, n_j\}$ is a set of the resulted error values in during training procedure for partition jth. These values are obtained for each partition based on "(7)," below

$$e_l = y_l - \hat{y}_l, \text{where}$$
$$\hat{y}_i = \mu \sum_{l=1}^{i-1} e_l \, K(x_i, x_l) \text{ and } (e_1 = y_1) \tag{7}$$

It can be said that after performing the two above stages, a set of if-then rules similar to TSK fuzzy model is available.

Finally, we should compute the weights of every partition ($w_j(x)$, for $j = 1, \dots, C$) at each test sample x. By applying fuzzy rules to any test sample, corresponding weights to the sample are obtained which are computed normal pdfs of each partition at the sample. In the end, the overall output is obtained by mixing the kernel-based functions via the weights $w_j(x)$. The kernel-based functions similar to "(6)," are expressed at each test sample x as follows

$$f^{j}(x) = \mu \sum_{l=1}^{n_j} e_l \, K(x, x_l), \quad j = 1, \dots, C \tag{8}$$

Notice that in the above equation, the sets of error values $\{e_l\}$ and learning samples $\{x_l\}$ where $l = 1, \dots, n_j$, for each partition are available from the two pervious stages. Therefore, the overall output of the proposed method is calculated as

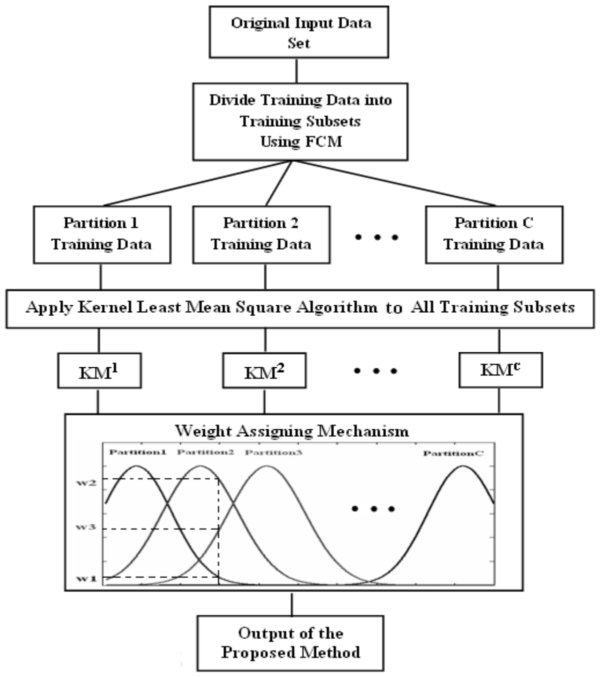
Original Input Data Set
Divide Training Data into Training Subsets Using FCM
Partition 1 Training Data
Partition 2 Training Data
Partition C Training Data
Apply Kernel Least Mean Square Algorithm to All Training Subsets
KM¹
KM²
KMᶜ
Weight Assigning Mechanism
Partition1
Partition2
Partition3
PartitionC
w2
w3
w1
Output of the Proposed Method

Fig. 2. The proposed kernel-based estimator.

$$\hat{f}(x) = \frac{\sum_{j=1}^{C} w_j(x) f^{j}(x)}{\sum_{j=1}^{C} w_j(x)} \tag{9}$$

Here, two examples are illustrated to show the performance of the proposed estimator in approximation of functions. In Fig. 3, 501 data points are generated from "(10)," with signal-to-noise ratio (SNR) = 10 dB. We observe that the proposed method has estimated the main signal well, despite noisy data as learning samples. In this case, the number of the used test data is 2001, $\sigma = 1$, and $C = 4$. The main function is defined as

$$y = exp(-x/3).sin(2x) \qquad \text{with} \quad x \in [0, 10] \tag{10}$$

In the literature (e.g., [17] and [27]), σ is proposed as (0.1 ~ 0.5) * range(x), where range(x) represents the input range of the training/test data. Note that, based on various experiments, a proper value for μ lies in "0.1 to 0.4". We use the value 0.25 for the parameter μ in all experiments of this paper.

In the second example, in Fig. 4, despite generating 55 data points from "(11)," five artificial outliers are also added into the learning sample data pool. Notice that we here suppose that there is a process which is capable of separating outliers from normal samples (we will present an outlier separator in Section IV which is applicable to background estimation).In fact, with this example, we would show if outliers are removed from the learning data, a good estimation of the main signal is resulted. This issue will be realized for our main purpose i.e., background estimation, in the following section. In this example, the number of the used test data is 181, $\sigma = 0.6$, and $C = 3$. The main function for this example is defined as

$$y = cos(3x) \qquad \text{with} \quad x \in [-3, 3] \tag{11}$$

IV. The Proposed Background Estimation Approach

In this section, we proposed our proposed method namely KBL using the presented kernel-based estimator from Section III. As it can be seen from the pervious section, the proposed kernel-based estimator suppresses effects of the noisy samples and outliers and generates desirable estimations of them. This issue motivates using this estimator in the estimation of background function from observations of each pixel. For this purpose, we interpret foreground samples as outliers relative to the background ones. Additionally, the estimator would also consider most of the background observations as the noisy samples. So, it's also expected that applying the estimator to the pixel observations results in estimating desirable values of foregrounds using background samples and suppressing of flicker noises and dynamic properties in the background ones.

The proposed procedure of our estimator in Fig. 2 is applied to a window with the width of N video frames.

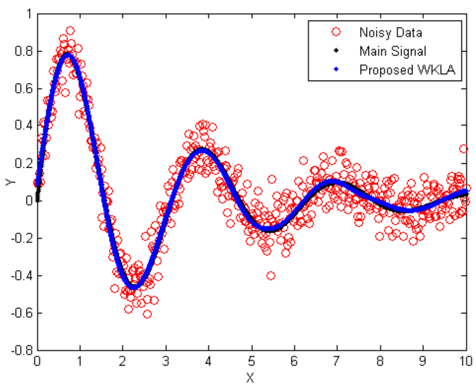

Fig. 3. Obtained result for noisy data of the function given in (10).

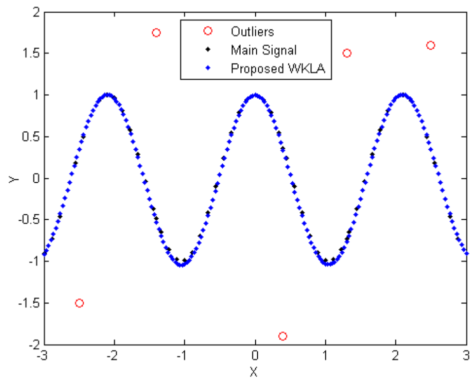

Fig. 4. Obtained result for data of a function given in (11) including outliers.

Also, we use pairs (t, x_{ij}^t) as input to our background estimation algorithm where x_{ij}^t is an observation (intensity value or RGB vector) of pixel (*i, j*) in frame (time) *t*. Throughout the paper, for simplicity, we usually remove subscript *ij* for sets, sequences, and arrays related to the pixel (*i, j*). The proposed background estimation algorithm (i.e., KBL) is shown in pseudo-code format in Fig. 5. The most important part of the algorithm is the training step and we start our discussion with it. In addition, we explain the initialization of the parameters in the part where they are employed.

A. *Training Step*

This step consists of two stages: first, applying the proposed OS to *N* observations of each pixel and then implementing stages 1 and 2 of the estimator for training of background model in each pixel.

1) *Outlier separator in background estimation*

As already mentioned, in order to achieve a desirable estimation using the suggested estimator shown in Fig. 2, it must be trained with normal samples not outliers. Thus, we need to have an outlier separation algorithm before training of the estimator which adapts to our interpretation of outliers. Notice that, for simplicity, we present our approach in this section based on intensity observations of each pixel. However, extension of the approach to the RGB color space is straightforward and will express in the subsequent section. OS algorithm itself includes four subsections which are expressed as follows.

a) *Applying a fuzzy 3-means clustering algorithm to pixel observations*

We first divide pixel observations x_{ij}^t (t = 1,…, N) into three clusters. In fact, we believe that there is usually one cluster including only background samples, static or dynamic. The two other clusters can also consist of static/dynamic background observations or/and foreground ones. Moreover, the two clusters may have less intensity values or/and more intensity ones than the first cluster. We use a fuzzy 3-means clustering algorithm for this division and denote three obtained clusters by C_k (k = 1, 2, 3).

b) *Calculation of a defined temporal criterion for each cluster*

We extract two features as size and maximum negative run length (MNRL) from each cluster. The first feature is clearly equivalent to the number of samples of each cluster and the second one is defined as the maximum interval of time that the cluster has not recurred during the period of *N* frames. These are denoted as n and λ, respectively. The latter has been used in [29] as an extracted feature of each codeword considered for each pixel. Based on the fact that a pixel is most of the times enumerated as a background pixel, the cluster having a large size and a small MNRL is mostly relative to a background event. Also, one having a small size and large MNRL could be a foreground event. So, we define a temporal criterion in the form of λ / n. This criterion is calculated for each cluster.

c) *Determining the two sets S_{normal} and $S_{outlier}$*

In this stage of OS algorithm, three obtained clusters are classified into two more general sets S_{normal} and $S_{outlier}$ including normal samples and outliers, respectively, based on their temporal criterions. Since, it's necessary to have some normal samples for training of the estimator; we first specify the cluster with minimum criterion of λ / n as an element of set S_{normal}. So, we have

$$S_{normal} = \left\{ C_k \mid k = \arg_m \min \lambda_m / n_m \right\}, \; m = 1, 2, 3 \qquad (12)$$

where λ_m / n_m is the temporal criterion corresponding to the cluster m. Clearly, $N = n_1 + n_2 + n_3$. Then it can be determined which of the two remaining clusters may include outliers. Thus, we define a model for the cluster consisting of outliers as

$$S_{outlier} = \left\{ C_m \mid \lambda_m \geq T_\lambda \wedge n_m \leq T_n \wedge C_m \notin S_{normal} \right\}, \quad m = 1, 2, 3 \qquad (13)$$

We set T_λ and T_n equal to half the number of frames in a window of the estimator, i.e., N/2. So, "(13)," can be rewritten as

$$S_{outlier} = \left\{ C_m \mid \lambda_m / n_m \geq 1 \wedge C_m \notin S_{normal} \right\}, m = 1, 2, 3 \qquad (14)$$

This means any of the remaining clusters matching the model, should be considered as a cluster including outliers

IACSIT International Association of Computer Science and Information Technology WWW.IACSIT.ORG

and be classified as an element of set $S_{outlier}$, otherwise it should be clearly merged into the set S_{normal}.

```
1. Initialization $(N, \sigma, C, \alpha_1, \varepsilon_1, \varepsilon_2)$
2. Training step
        process on each pixel
                $[S_{normal_{ij}}, S_{outlier_{ij}}, \boldsymbol{M}_{ij}] \leftarrow Outlier\ Separator\ (\boldsymbol{X}_{ij});$
                    $[e_{Training_{ij}}, T_{Training_{ij}}] \leftarrow Background\ Training\ (T_{normal_{ij}}, S_{normal_{ij}});$
3. Testing step
        process on each frame[1]
                process on each pixel
                        $[B^t_{f_{ij}}] \leftarrow Background\ Estimation\ (t, x^t_{ij}, m^t_{ij}, e_{Training_{ij}}, T_{Training_{ij}});$
                    $[Mask^t_{ij}] \leftarrow Foreground\ Detection\ (x^t_{ij}, B^t_{f_{ij}}, m^t_{ij});$
4. Adaptation step
        Shift the window of the estimator;
        Return to step 2;
```

Fig. 5. The proposed background estimation algorithm.[1]

d) Assigning of vector consisting of membership values ($\boldsymbol{M}_{ij}$)

To take into account the uncertainty of the fuzzy clustering algorithm, we propose to assign a vector of membership values to each pixel as $\boldsymbol{M}_{ij}$ below

$$\boldsymbol{M}_{ij} = \left[m^1_{ij}, m^2_{ij}, \ldots, m^N_{ij} \right] \tag{15}$$

In the above equation, m^t_{ij} is expressed as below

$$m^t_{ij} = u'_{tl} \tag{16}$$

where $l = \arg_k \max u'_{tk}$ and $C_k \in S_{normal}$ for $k = 1,2,3$ and $t = 1, \ldots, N$.

In other words, m^t_{ij} (called as membership value) is the degree of membership of sample x^t_{ij} in the closest cluster of set S_{normal} to this sample. Notice that the values u'_{tk} have been obtained after performing the first stage of the OS approach.

2) Training of background using normal samples

In this stage, training of background is performed for each pixel using its normal samples (we name this stage as "Background Training" in pseudo-code in Fig. 5). In fact, this stage is the same as stages 1 and 2 of the estimator which were presented in Section III.

It's important to note that, few samples of foreground (few outliers) may lie in the background cluster (the cluster of normal samples) even after implementing the OS approach and result in undesirable background estimation. Based on the fact that the outliers usually take membership values (m^t_{ij}) less than the normal samples, we define a subset $S_{Training}$ of set S_{normal} as below

$$S_{Training} = \left\{ x^t_{ij} \mid x^t_{ij} \in S_{normal} \wedge m^t_{ij} \geq Th_{ij} \right\} \tag{17}$$

where $S_{Training}$ is a refined version of set S_{normal} and Th_{ij} is the used threshold to choose the proper samples in training of the background for the pixel (i, j) and takes a value "0 to 1". We use an adaptive threshold for each pixel as

$$Th_{ij} = median\left(m^t_{ij}\right), \quad \text{for} \quad t \in T_{normal} \tag{18}$$

where T_{normal}, is the set of observation times corresponding to the samples of set S_{normal} and *median* represents the median value of the membership values m^t_{ij} for $t \in T_{normal}$. Notice that the set $S_{Training}$ is used along with the set of its corresponding observation times i.e., set $T_{Training}$ in training process.

It's supposed that outliers occur for a relatively long time. If many partitions are used, it is possible that a partition is placed on the time interval of occurrence of outliers. In this case, there are not any normal samples in the interval and thereby the accurate estimation of background is not obtained. So, we can use a few partitions like 2 or 3. Moreover, the kernel bandwidth parameter (σ) is also selected based on the term represented in Section III (i.e., (0.1 ~ 0.5) * range(x)). According to this term, the kernel bandwidth parameter is appropriately selected to reflect the input range of training/test data. Now, in our background estimation problem, if this range is considered equal to the range of the training data, it's probable that inaccurate estimations of the background are obtained in the intervals of occurrence of outliers, especially, when they occur for a relatively long time. So, we choose the range(x) equal to the width of the window of the proposed estimator. Clearly, since this value (i.e., the width of the window) is fixed, it is independent of the time

[1] All frames of current window of the estimator should be processed in the testing step which have not been estimated until this stage.

interval of occurrence of outliers. According to the above description, the problem of existing of foregrounds at the start of the estimator window or non-empty background problems can be addressed by the proposed approach as it can be seen in Fig. 6.

The necessary outputs of this stage include sequences/sets of training errors and observation times which were denoted as $e_{Training}$ and $T_{Training}$, respectively, in pseudo-code in Fig. 5. These outputs along with vector $\boldsymbol{M}_{ij}$ obtained by OS algorithm are the main inputs of subsequent step (i.e., testing step).

B. *Testing Step*

In this step, for each frame, a testing procedure including estimation of background and detection of foreground is performed on its pixels. These two stages are called "Background Estimation" and "Foreground Detection" in Fig. 5 which are explained as follows.

1) *Estimation of background*

First, by applying stage 3 of the proposed estimator to the pixel in the current frame and its output sequences of training step, background value of the pixel is estimated which is denoted as $B^t_{KBL_{ij}}$. As we already mentioned, besides suppressing of foreground samples (outliers) by the estimator, it would also suppress background samples. Apparently, the suppressing of foregrounds is desirable, but it's not good for backgrounds, because it causes the simultaneous detection of moving background pixels and foreground ones in the foreground detection stage. To overcome this problem, we propose to use "(19)," as the final equation of background estimation for all observations in the estimator window.

$$B^t_{f_{ij}} = m^t_{ij}\left(\left(1-\alpha_l\right)B^t_{KBL_{ij}} + \alpha_l x^t_{ij}\right) + \left(1-m^t_{ij}\right)B^t_{KBL_{ij}} \qquad (19)$$

where $B^t_{f_{ij}}$ is the final value of the estimated background in pixel (i, j) in frame t, m^t_{ij} is obtained by OS algorithm and according to "(16)," and α_l is a learning coefficient as a constant value in "0 to 1". With some attention to "(16)," it can be found that m^t_{ij} takes the relatively large value (close to 1) for normal samples and small value for those which are in set $S_{outlier}$. In [30], a fuzzy approach for background subtraction has been proposed. In that work, membership value of each pixel to the foreground and background class was used to update the background value of the pixel.

Although using membership values improves the estimated background, proper choosing of learning coefficient also leads to a more accurate estimation of the background and decreasing errors of the resulted foreground. Based on "(19)," using a very large value of α_l may make some foreground samples appear in the estimated background. On the other hand, a very small value for this parameter ensures suppressing of dynamic backgrounds in the estimated background. Since the background is usually static in indoor scenes, we use a relatively small value of α_l in such environments. This certifies a good removing of moving foregrounds. However, removing of moving objects is an essential task in background estimation, but maintaining moving backgrounds in dynamic scenes is important too. So, a larger value of the learning coefficient is chosen in these scenes. In other words, a proper value for the parameter α_l is dependent on the scene being modeled. According to the experiments, values "0.6 to 0.7" of this parameter give good results for all of our test sequences with dynamic background. The proper value of the parameter is "0.1 to 0.2" in the scenes with little amounts of changes of background and indoor environments.

2) *Foreground detection*

In our approach, an incoming pixel must meet two conditions until it is classified as foreground: (1) The absolute value of the difference between pixel value in current frame and estimated background image is more than the detection threshold ε_1 and (2) the membership value for the pixel in current frame is less than the detection threshold ε_2. These thresholds take constant values in "0 to 1". Obviously, each of above conditions is not satisfied, the respective pixel is classified as background. We show the foreground map in pixel (i, j) of frame t as $Mask^t_{ij}$.

$$Mask^t_{ij} = \begin{cases} 1 & \text{if } \left|x^t_{ij} - B^t_{f_{ij}}\right| > \varepsilon_1 \ \wedge\ m^t_{ij} < \varepsilon_2 \\ 0 & \text{otherwise} \end{cases} \qquad (20)$$

In all experimental results of Section V, we will choose ε_1 from "0.02 to 0.03". Note that we first map the possible values "0 to 255" of the above absolute value to values "0 to 1", and then use threshold ε_1 from its proposed interval. Also, we have tested different values of ε_2 and, by experimentation, best value for this parameter is "0.6 to 0.8".

C. *Adaptation Step*

We suppose that the scene illumination may undergo gradual changes all the time. So, in order to update the model for each pixel to adapt the system to gradual changes, we slide the window of the estimator on video frames such that each two consecutive windows have some overlaps on each other. In all experiments, we shift the window as much as two thirds of the width of the window (i.e., $2N/3$). In other words, by considering $N = 300$, every 200 frames, each pixel in the scene undergoes a re-training process. This re-training process similar to the original training can be performed with the samples of $S_{Training}$ in the window. When the re-training is completed its results are used in the testing step. However, in order to decrease the cost of the computation, this adaptation process is not performed at each frame.

V. EXPERIMENTAL RESULTS

The proposed method has been tested over different video sequences.[2] Three examples of the standard sequences namely airport (AP), water ripples (WR), and campus (CAM) are tested in this paper. Various dynamic backgrounds and busy scenes with several moving objects are two main features of these videos. However, in the chosen sequence

[2] All the experiments have been performed on a Pentium 4 PC, 2.54 GHz, using Matlab 6.5.

IACSIT
International Association of Computer Science and Information Technology
WWW.IACSIT.ORG

WR, the partial shakes of camera also exist.

All results of this section are obtained by implementing the proposed method to the color pixel observations. For this purpose, x_{ij}^{t} is considered as a 3-dimentional vector with color components "R, G, and B". So, FCM algorithm is applied to the array of 3-dimentional vector observations. That is, clustering of observations of each pixel is performed in RGB color space instead of intensity space. Also, it should be considered that the error data for each sample has a vector form. However, since kernel calculation is done on time axis, it still takes a scalar form as when intensity values are used.

For each sequence, background is estimated over some randomly selected frames using the proposed method. Moreover, we compare the performance of our method in detection of foreground objects to the method of Stuffer and Grimson (i.e. MoG) [14] in both qualitative results and quantitative evaluations. MoG is a widely-used adaptive background subtraction method. It performs quite well for both stationary and non-stationary backgrounds among the existing methods in [8]. We evaluate these methods using the quantitative measure used in [31]. If F is a detected region and G the corresponding "ground truth," then the similarity measure between regions F and G is defined as

$$S(F,G) = (F \cap G)/(F \cup G) \qquad (21)$$

This measure is increasing with the similarity of the detected region to the "ground truth," with values "0 to 1". It reaches the maximum value of 1 if these two regions are the same. Additionally, for a fair comparison, the number of components in MoG is fixed to 3 in all experiments. Also, the same post-processing (i.e., the morphological smoothing and small region elimination) is applied to two methods.

In the figures of this section, pictures are arranged in columns. In each column, the images from top to bottom are the input frame, the background image estimated by the proposed method at the moment, the "ground truth," the results of the detected foreground by the proposed method and MoG. Except sequence WR, the two remaining of test sequences, and the "ground truths," of the sample frames are available in [32]. For water ripples video, a sequence including 500 frames with 25 frames per second is available for us. We manually generated some "ground truths," for this sequence.

The first example in Fig. 6 displays an indoor busy scene which includes many people walking most of the times. This sequence is an example of non-empty background problems. From the results, one can see that the proposed method has removed the effect of all persons from the background. Moreover, it has obtained the satisfactory results in foreground detection apart from where the parts of the shadows have been detected in this environment. Contrary to the proposed method, in MoG not only shadows of people were mis-detected as foregrounds, many parts of humans' bodies were also missed.

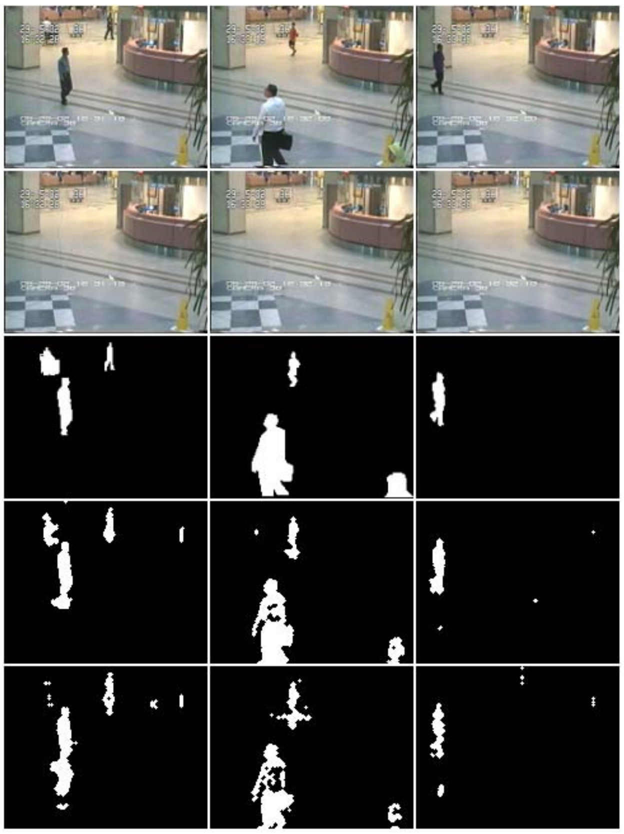
Fig. 6. Experimental results on a sequence of an airport (AP). They are frames 656, 1180 and 1289.

In the second example, a sequence including water ripples as a moving background and a ship and two small boats as foreground objects is tested like Fig. 7. In this sequence, background has a non-periodic and irregular motion. Besides, the intense noise and small vibrations of the camera also exist. As expected, the proposed method would suppress water ripples. But, by considering membership vector $\boldsymbol{M}_{ij}$ and choosing a proper value for learning coefficient α_l, this suppression is weakened. Moreover, by comparing the results of detection with "ground truth," one can see that accurate shape information of the foreground objects has been obtained by the proposed method. It is evident that the nominal motion of the camera causes substantial degradation in performance of MoG, despite using a three-component mixture model and a relatively high learning rate of 0.05.

The last example displayed in Fig. 8 comes from a video containing moving tree branches. The great motion of tree branches was caused by strong winds which can be observed from the waving yellow flag at the left of the images. Also, the three example frames contain the people with few pixels relative to the total size of image and vehicles with different colors. So, it can be said that this sequence is an almost complex scene from the background modeling point of view. But, according to the results, the proposed method has estimated the background relatively well. Also the humans and vehicles have been almost perfectly detected. However, few pixels of moving flag and branches were considered as foreground pixels. In addition, due to sudden illumination changes over the tree at the left of images, some pixels of the tree were mis-detected as foreground.

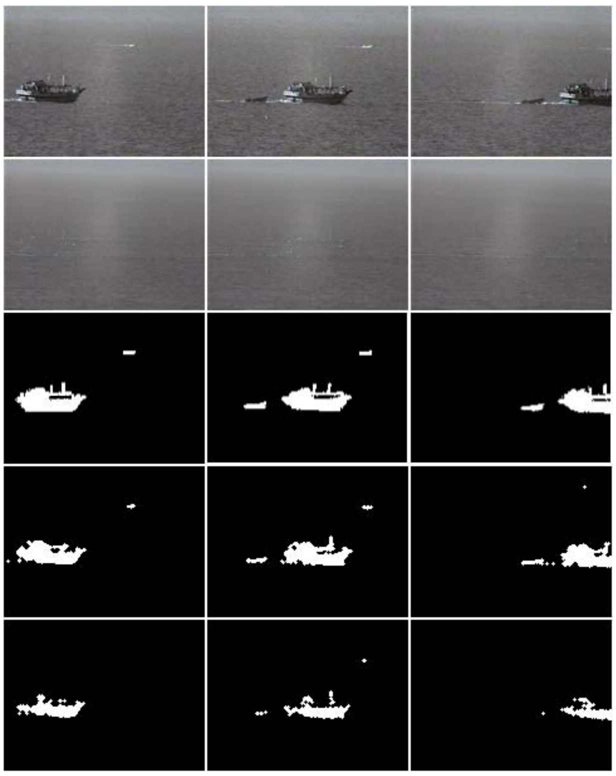
Fig. 7. Experimental results on a sequence of water ripples (WR). They are frames 139, 241 and 359.

Also, since the coverage of background colors for moving trees are very large, MoG with three components cannot model this dynamic background well. So, many parts of the foreground objects were mis-classified as the background. Clearly, MoG also fails when sudden illumination changes occur.

From the comparisons with MoG in the examples shown in Figs. 6–8, it can be found that the proposed method has outperformed the MoG method in these selected situations.

Table I shows the average of the similarity measures using "(21)," which is evaluated on 20 randomly selected frames of each sequence. The quantitative evaluation agrees with the conclusions from the visual observation of the experimental results.

According to the quantitative results in Table I, we can see that the propose method (i.e., KBL) has given the larger values of similarity measure compared to the MoG method for all tested sequences. It is evident that the obtained quantitative value for sequence AP is lower than WR, since in sequence AP, the detection of objects' shadows in foreground mask makes the similarity between detected foreground and the corresponding "ground truth," be reduced. Also, due to the occurrence of local sudden illumination changes in CAM, the similarity value for this sequence is lower than WR.

TABLE I. COMPARISON OF SIMILARITY VALUES OBTAINED BY THE PROPOSED METHOD AND MOG [14]

	AP	WR	CAM
PROPOSED	0.7032	0.7847	0.6955
MOG	0.4826	0.6259	0.4031

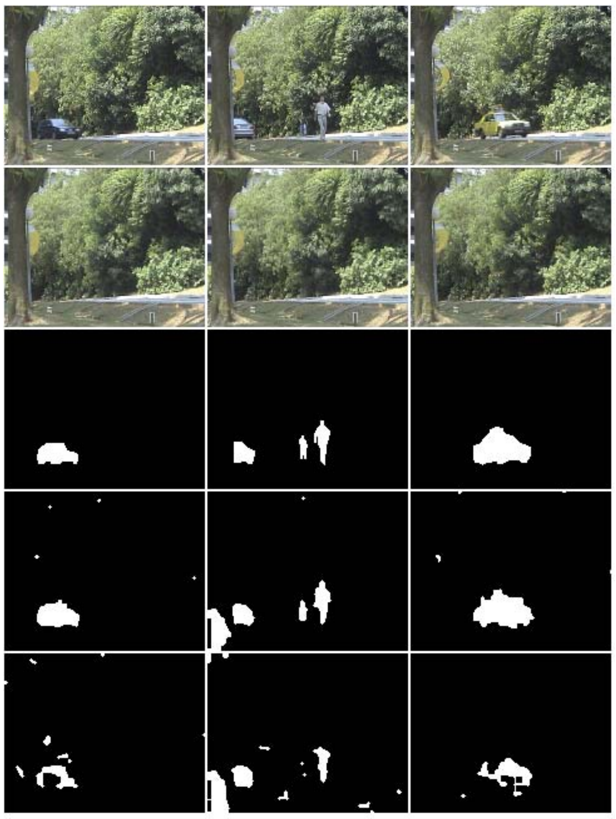
Fig. 8. Experimental results on a sequence of a campus (CAM). They are frames 695, 831 and 1019.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a new method in video background estimation based on a kernel-based background leaning (KBL) algorithm. It concluded that KBL algorithm must be trained only with normal samples to result in an accurate estimation. So, we first proposed an outlier separator (OS) by interpreting foreground samples as outliers relative to the background ones. Besides separating outliers set from the normal samples one, OS results in a defined vector of membership values for each pixel. Background estimation is then achieved by applying the results of OS to the proposed estimator. We also used an approach based on two complementary conditions in foreground detection. We employed membership values and a learning coefficient in the final estimation to improve the estimated background. The qualitative and quantitative results on different sequences demonstrated the effectiveness of the proposed approach.

Future directions of this work are stated as follows. We are investigating a more efficient outlier separator which increases the possibility of identifying real outliers and works with less computational time. Also we plan to improve the method to handle the scenes including sudden illumination changes and situation where foreground objects may remain in that for a long time.

REFERENCES

[1] S. Cheung and C. Kamath, "Robust background subtraction with foreground validation for urban traffic video," Journal of Applied Signal Process., Sept. 2005.

IACSIT
International Association of Computer Science and Information Technology
WWW.IACSIT.ORG

[2] M. Piccardi, "Background subtraction techniques: a review," in Proc. IEEE Int. Conf. Systems, Man, Cybernetics, 2004, pp. 3099–3104.
[3] S. Elhabian, K. El-Sayed, and S. Ahmed, "Moving object detection in spatial domain using removal techniques - state-of-art," Recent Patents on Computer Science, vol. 1, no. 1, Jan. 2008, pp. 32–54.
[4] R. Cucchiara, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no. 10, Oct. 2003, pp. 1–6.
[5] N. J. B. McFarlane and C. P. Schofield, "Segmentation and tracking of piglets in images," Mach. Vision Application, vol. 8, 1995, pp. 187–193.
[6] F. C. Cheng and Y. K. Chen, "Effective Σ–Δ background estimation for video background generation," IEEE Asia-Pacific Services Computing Conf., 2008.
[7] I. Haritaoglu, D. Harwood, and L. Davis, "W4: real-time surveillance of people and their activities," IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 8, Aug. 2000, pp. 809–830.
[8] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance," in Proc. IEEE Int. Conf. Computer Vision, Sept. 1999, pp. 255–261.
[9] C. Wren, A. Azarbaygaui, T. Darrell, and A. Pentland, "Pfinder: real-time tracking of the human body," IEEE Trans. Pattern Anal. Machine Intell., vol. 19, July 1997, pp. 780–785.
[10] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russel, "Towards robust automatic traffic scene analysis in real-time," in Proc. ICPR, Oct. 1994, pp.126–131.
[11] K. Karmann and A. von Brandt, "Moving object recognition using an adaptive background memory," Time-Varying Image Process. Moving Object Recognit., 1990, pp. 289–296.
[12] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in Proc. IEEE Conf. Vision and Pattern Recognit., 1999, pp. 246–252.
[13] N. Friedman and S. Russell, "Image segmentation in video sequences: a probabilistic approach," Proc. Conf. Uncertainty in Artificial Intell., 1997, pp. 175–181.
[14] C. Stauffer and W. Grimson, "Learning patterns of activity using real-time tracking," IEEE Trans. Pattern Anal. Machine Intell., vol. 22, Aug. 2000, pp. 747–757.
[15] A. Tavakkoli, M. Nicolescu, and G. Bebis, "Automatic statistical object detection for visual surveillance," in Proc. of IEEE Southwest Symposium on Image Analysis and Interpretation, 2006, pp. 144–148.
[16] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," Proc. IEEE, vol. 90, no. 7, 2002, pp. 1151–1163.
[17] V. Cherkarsky and Y. Ma, "Practical selection of SVM parameters and noise estimation for SVM regression," Neural Netw., vol. 17, no. 1, Jan. 2004, pp. 113–126.
[18] D. W. Scott. Multivariate Density Estimation. New York: Wiley Interscience, 1992.
[19] J. Rittscher, J. Kato, S. Joga, and A. Blake, "A probabilistic background model for tracking," in Proc. 6th Eur. Conf. Computer Vision, 2000, pp. 336–350.
[20] C. Montacie, M. J. Caraty, and C. Barras, "Mixture splitting technic and temporal control in a HMM-based recognition system," Proc. (ICSLP), 1996, pp. 977–980.
[21] A. Tavakkoli, M. Nicolescu, and G. Bebis, "Non-parametric statistical background modeling for efficient foreground region detection," 2008, pp. 1–16.
[22] M. Sigari, N. Mozayani, and H. Pourreza, "Fuzzy running average and fuzzy background subtraction: concepts and application," (IJCSNS), vol. 8, no. 2, Feb. 2008, pp 138–143.
[23] M. Shakeri, H. Deldari, H. Foroughi, H. Saberi, and A. Naseri, "A novel fuzzy background subtraction method based on cellular automata for urban traffic applications," ICSP2008, pp. 899–902.
[24] V. Vapnik, Statistical Learning Theory. NewYork: Wiley Interscience, 1998.
[25] B. Widrow, Adaptive filters I. Fundamentals (TR 6764-6), Stanford Electronics Laboratories, Stanford, CA, 1966, Technical Report.
[26] W. Liu, P. P. Pokharel, and J. C. Principe, "Kernel least mean square algorithm," IEEE Trans. Signal Process. vol.56, no. 2, Feb 2008, pp. 543–554.
[27] C. C. Chuang, "Fuzzy weighted support vector regression with a fuzzy partition," IEEE Trans. On Systems, Man, And Cybernetics–Part B: Cybernetics, vol. 37, no. 3, June 2007, pp. 630–640.
[28] C. T. Lin and C. S. George Lee, Neural Fuzzy Systems. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[29] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. S. Davis, "Real-time foreground-background segmentation using codebook Model," Real-Time Imag., vol. 11, 2005, pp. 172–185.
[30] F. El Baf, T. Bouwmans, and B. Vachon, "A fuzzy approach for background subtraction," ICIP, 2008, pp. 2648–2651.
[31] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian, "Statistical modeling of complex backgrounds for foreground object detection," IEEE Trans. Image Process., vol. 13, no. 11, Nov. 2004, pp. 1459–1472.
[32] http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html.

Hamidreza Baradaran Kashani was born in Mashhad, Iran in 1984. He received to B.S. degree in Robotic Engineering from Shahrood University, Iran, 2007. He pursues M.S. position in Electrical Engineering in Ferdowsi University of Mashhad. His research interests include image and video processing, and pattern recognition.
Email: hamidreza.baradaran@gmail.com

Seyed Alireza Seyedin was born in Mashhad. He received the B.S. degree in Electronics Engineering from Isfehan University of Technology, Isfehan, Iran in 1986, and the M.E. degree in Control and Guidance Engineering from Roorkee University, Roorkee, India in 1992, and the Ph.D. degree from the University of New South Wales, Sydney, Australia in 1996. He has been an Associate Professor with the Department of Electrical Engineering, the University of Mashhad (Ferdowsi), Mashhad, Iran. His research interest includes image processing, computer vision, signal processing, and pattern recognition. In these fields, specially, he is interested in image analysis, motion detection and estimation in image sequences, Autonomous vehicles, and diverse applications of The Radon transform. email: seyedin@um.ac.ir

Hadi Sadoghi Yazdi received the B.S. degree in electrical engineering from Ferdowsi University of Mashad in 1994, and then he received to the M.S. and PhD degrees in electrical engineering from Tarbiat Modarres University of Iran, Tehran, in 1996 and 2005 respectively. He works in Computer Department as an assistant professor at Ferdowsi University of Mashhad. His research interests include adaptive filtering, image and video processing, and optimization in signal processing. email: h-sadoghi@um.ac.ir, sadoghi@sttu.ac.ir