Adaptive Learning of Multi-Subspace for Foreground Detection under Illumination Changes

Y. Dong, G. N. DeSouza*

Electrical and Computer Engineering Department

University of Missouri,

Columbia, MO, USA

* Corresponding author: DeSouzaG@missouri.edu

Ph: (573) 882-5579 - Fax: (573) 882-0397

Abstract

We propose a new adaptive learning algorithm using multiple *eigen subspaces* to handle sudden as well as gradual changes in background due for example to illumination variations. To handle such changes, the feature space is organized into clusters representing the different background appearances. A local principle component analysis transformation is used to learn a separate eigen subspace for each cluster and an adaptive learning is used to continuously update the eigenspaces. When the current image is presented, the system automatically selects a learned subspace that shares the closest appearance and lighting condition with the input image, which is then projected onto the subspace so that both background and foreground pixels can be classified. To efficiently adapt to changes in lighting conditions, an incremental update of the multiple *eigen subspaces* using synthetic background appearances is included in our framework. By doing so, our system can eliminate any *noise* or distortions that otherwise would incur from the foreground objects, while it correctly updates the specific eigen subspace representing the current background appearance. A forgetting factor is also employed to control the contribution of earlier observations and limit the number of learned subspaces. As the extensive experimental results with various benchmark sequences demonstrate, the proposed algorithm outperforms, quantitatively and qualitatively, many other appearance-based approaches as well as methods using Gaussian Mixture Model (GMM), especially under sudden and drastic changes in illumination. Finally, the proposed algorithm is demonstrated to be linear with the size of the images d, the number of basis in the local PCA m, and the number of images used for adaptation n: that is, the algorithm is $O(dmn)$ and our C++ implementation runs in real time – i.e. at frame rate for normal resolution (VGA) images.

Keywords: multiple *eigen subspace*, Local PCA, incremental learning, illumination invariance.

1 Introduction

Foreground object detection is an essential task in many image processing and image understanding algorithms, in particular for video surveillance. Background subtraction is a commonly used approach to segment out foreground objects from their background. In one way or another, background subtraction consists of modeling and storing away the background so it can be later compared to newly observed images. The difference image obtained by this comparison is then thresholded so the foreground objects can be segmented out. Despite the simplicity of this concept, in real world applications, temporal and spacial changes in pixel values such as due to shadows, gradual/sudden changes in illumination, etc. make modeling backgrounds a quite difficult task. For that reason, most systems have focused on either capturing the temporal or the spacial changes, and only a few systems have expanded one approach into the other: i.e. from a temporal approach into spacial or vice versa.

In order to capture changes in the background over time, different approaches have been proposed in the past decades. In [1] and [2], for example, the system recursively updated the background model using adaptive filters. In both cases, the method could only accommodate gradual illumination variations and it often failed in the presence of sudden illumination changes. A widely used technique, Gaussian Mixture Models (GMM), has been employed by many systems, such as in [3, 4, 5, 6, 7, 8, 9]. In all these cases, the basic principle was to model the pixel intensity at each location and capture the pixel statistics over time with a mixture of Gaussians. More recently, improved approaches using multi-modal techniques such as: 1) an adaptive GMM [5, 4]; 2) layers of GMM for each pixel [8]; 3) feature vectors consisting of color and texture [9, 10]; etc. have achieved not only better performance, but also the ability to handle gradual changes in illumination. However, GMM-based approaches do not perform well when the pixels change drastically or over long periods of time. Besides, GMM alone cannot capture the spatial relations among pixels, which is an important requirement for a coherent foreground segmentation. This problem also appears in other non-parametric and kernel-based approaches, such as [11] and [12], and had to be handled in a separate and

computationally intensive step, currently preventing frame rate performances [13].

In that regard, a system known as Wallflower [14] was one of the first systems to propose a combined temporal and spacial framework using three major components: 1) the pixel-level component, which uses a Wiener filter to create a linear predictor of the pixel intensity values given the pixel history; 2) the region-level component, which groups homogeneous pixels based on their spatial relations; and finally 3) the frame-level component, which adapts to the gradual and sudden changes of the background. The combination of these three levels of processing achieved good results, but the complexity added prevented its use with real-time performance. Other GMM-based approaches were recently proposed to address, for example, illumination invariance and foreground fragmentation – e.g. [6]. However, typical problems of GMM, such as the use of global learning rates for the background update and slow convergence, were not addressed by this method. So, in [7], the authors proposed locally adaptive learning rates for each Gaussian distribution. The rates were based on the most likely Gaussians over several continuous frames. That approach improved convergence significantly, but the spatial relationships between pixels were still ignored and therefore foreground fragmentation still occurred. Finally, other statistical approaches to model dynamic scenes were also proposed. But they led to either computationally complex solutions [15] or to restrictions on the relative dynamic behavior of background and foreground [16].

On the other side of the coin, many approaches based on subspace learning have been proposed [17, 18, 19, 20, 16] to capture spacial relations between pixels,. In the celebrated *EigenBackground*, [17], for example, the system built a PCA feature space to describe the appearances of the learned backgrounds. Newly observed images were then projected onto this eigenspace and the foreground objects were segmented out by a thresholding based on the Euclidean distance between the reconstructed (projected) images and the original images. Due to the nature of PCA, this method exploited extensively the spatial relations between pixels and thus it often resulted in homogeneously coherent regions for each object in the foreground. Additionally, the off-line learning characteristic of the method made it fast and

easy to implement. More recently, temporal changes in background appearance – e.g. due to illumination, dynamic backgrounds, etc. – were incorporated into eigen-based approaches, as in [20] and [16], to capture not only the pixel spacial relations inherent to PCA approaches, but also their temporal relations. Also, to add adaptation over time, an incremental PCA (or adaptive PCA – APCA) was employed by a few systems [16, 18]. However, as our results demonstrate, by employing multiple subspaces, our approach can more accurately capture the appearance of the background in the presence of, for example, sudden illumination changes. Finally, other appearance based methods were proposed to handle changes in the background, but their main focuses were on: a) subdividing the background image into blocks in order to capture their appearance despite small translations of these blocks [21]; b) the use of tensors for the calculation of a global PCA that could more efficiently capture the similarities between groups of pixels in the background [22]; and c) the efficient calculation of a global PCA [23].

Motivated by the above limitations, we propose a new algorithm that offers four major advantages over current approaches: 1) it uses multiple feature subspaces to quickly capture and learn different backgrounds and lighting conditions; 2) it provides a clustering scheme to initialize the subspaces; 3) it employs an adaptive and efficient (i.e. linear) learning scheme that assures real-time performance while dealing with continuous and/or abrupt changes of illumination; and 4) it relies on synthesized backgrounds to prevent foreground objects from *corrupting* the on-line learning. As we demonstrate in the next sections and in Appendix A, the proposed use of adaptive and local PCAs to build q multiple m-dimensional local subspaces leads to a much better performance when compared to other approaches using a single and global $m \times q$-dimensional space. In other words, our results show that by letting each subspace continuously learn and adapt to different illumination condition – or even a different appearance of the background (e.g. a door in the back of the room is opened/closed) – a newly observed image can be projected onto the most representative subspace with a smaller error than if using a single, large-dimensional space. This smaller error leads to a

much more coherent and robust foreground segmentation.

The rest of this paper is organized as follows: first, our proposed adaptive learning of multi-subspace using local PCA is described in Section 2. Next, in Section 3, we compare a pure PCA, an adaptive PCA, and an adaptive GMM-based approach, to our previous Local PCA approach without adaptation [24], and our new proposed method with adaptation (ALPCA). Future work and final conclusion are given in Section 4.

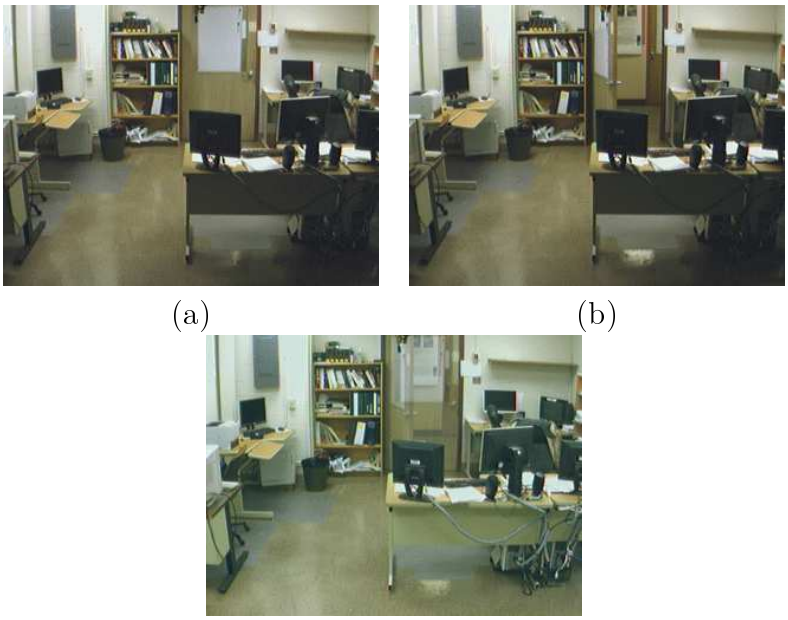
2 Proposed Framework using Adaptive Local PCA

In order to understand our framework, we must first revisit the original idea of background subtraction using appearance method: that is, EigenBackground.

2.1 EigenBackground Revisited

The EigenBackground method [17] consists of two steps: background learning and foreground classification. In the first step, a set of static images (the training set) is collected to form a high dimensional feature space. That is, each image in the set is regarded as a vector comprised of the pixels intensities in all three color channels in sequence. Then, the statistics of the set – mean and covariance – are used to determine the major axes of the sample distribution – that is, the principal eigenvectors or components of the distribution. In step two, background classification, the observed image (query) is projected onto the feature subspace defined by the above principal eigenvectors. Since the training images did not contain any foreground objects, it is expected that the projection of the query image will reconstruct only the static elements in the scene, i.e., the background. The actual foreground objects can be extracted by thresholding the Euclidean distance between the reconstructed and the observed images.

The main reason for the success of this approach is that the learned eigenspace represents the probability distribution of the background, which models a range of possible appearances



(a) (b)

(c)

Figure 1: Representing two different clusters of background appearances using one single PCA space: (a) sample image from one of the clusters; (b) sample image from the second cluster; (c) reconstruction of the average image representing both clusters by a single PCA space. As we can see, aspects of both clusters are captured, combined, and can be observed in the reconstructed average image (c) – i.e. both the closed door and the hallway behind it can be seen in the image.

of that same background. This allows for the subtraction of the background to be insensitive to outliers caused by random noise, camera vibration, reflectance etc. Furthermore, regions of the foreground obtained by the above process are homogeneously coherent, since all pixels in the image are used in the classification, rather than individual pixels. That is, the eigenspace exploits the spatial relationship among neighboring pixels through their learned cross correlations.

Unfortunately, this same property of EigenBackground presents drawbacks when in the presence of multiple appearances of the background – e.g. due to different lighting conditions. Since the principal components analysis (PCA) preserves the leading eigenvectors corresponding to the largest variances of the data set, a single feature subspace is not capable of modeling desired changes of the background accurately. In other words, the variances captured by the largest eigenvectors do not necessarily represent the probability distribution of the background appearances for any one condition, but for all of the conditions combined. This idea, for the case of a door both opened and closed in the background, is illustrated in Figure 1, where both clusters of background appearances are *merged* in a single PCA space. Later in the paper, we will show that such characteristic of single PCA spaces leads to poorer results – whether the algorithm is made adaptive or otherwise. The same principle presented in Figure 1, though harder to be appreciated visually, applies to difference in appearances due to lighting conditions. In the end, this poor representation of the range of appearances of the background leads to a degraded performance for either gradual or sudden changes in the dataset, even when these changes occur in reasonably large numbers and are used to continuously update the PCA space.

2.2 Proposed Method

Although many factors may cause a significant change in the appearance model – e.g. camera motion, shape deformation, gradual illumination changes, etc. – one of the main problems in background subtraction comes from sudden illumination changes. In light of these facts, our

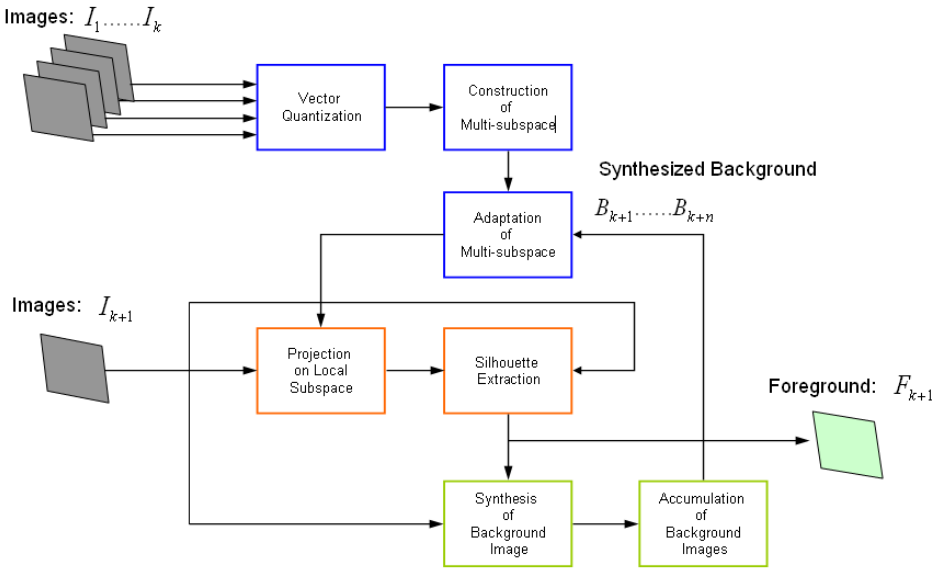


Figure 2: Proposed Framework

proposed method is fourfold: 1) the use of multiple feature subspaces to quickly capture and learn different lighting conditions; 2) a clustering scheme to initialize the subspaces; 3) the incorporation of an adaptive learning scheme to improve the performance when dealing with continuous changes of illumination; and 4) a smart learning scheme using synthesized backgrounds to prevent foreground objects from *corrupting* the on-line learning. The proposed framework is depicted in Figure 2. In the next sections we explain in detail this framework, beginning with a summary of the background subtraction using incremental learning of subspace [25]. Then, we present our new incremental learning scheme using multiple subspaces, including a discussion of the benefits of our background synthesis and a performance improvement using a motion hypothesis test.

2.2.1 Incremental Learning in a Single Feature Subspace

In real world scenarios, changes of illumination cause pixel intensities to change drastically and thus, potential misdetection of foreground becomes quite likely. In order to make any solution to background subtraction effective, we must incorporate some continuous adapta-

tion to illumination. That is, an ability to incrementally learn with new observations and to immediately adapt within a temporal window. In the case of a single subspace, this adaptation is carried out by the computation of a single extended matrix which encompasses the changes in the subspace due to the last few observations. These changes are incorporated into the subspace and the process is restarted. That is, new observations lead to the calculation of another extended matrix which is again used to iteratively modify the subspace – the blue squares in Figure 2.

In order to explain this process in more details, let A be the current block of observations, that is, $A = [\mathbf{x}_1, \ldots, \mathbf{x}_k]$, with an eigen basis U and singular values D. This feature space can be obtained from the singular value decomposition (SVD) of A, where $\mathbf{x}_j$ denotes the zero-mean d-dimensional vector representation of the image I_j[1]. As we will show next, the subspace of this block of images A will be iteratively updated, eliminating the need to collect off-line training images with a static background. That is, for a new block of observations $B = [\mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+n}]$ to be incorporated into this feature space, all that needs to be done is to incrementally estimate the eigen basis U' and the singular values $D'$of the new combined block of observations $[A,\ B]$. That is, for a temporal window with size n:

- First, we generate a zero-mean matrix of new observations $\hat{B} = \left[(\mathbf{x}_{k+1} - \mathbf{r}_b)\,,\ \ldots\ ,\ (\mathbf{x}_{k+n} - \mathbf{r}_b)\ ,\ \sqrt{\frac{\kappa n}{\kappa + n}}\,(\mathbf{r}_b - \mathbf{r}_a)\right]$ where $\mathbf{r}_a$, $\mathbf{r}_b$ are the sample means of A and B respectively, and κ is initially set equal to k;

- Next, we update, $\mathbf{r}_a$, with the mean of the concatenation of A and B, that is:

$$\mathbf{r}_a = \frac{f\kappa}{f\kappa + n}\mathbf{r}_a \ + \ \frac{n}{f\kappa + n}\mathbf{r}_b \tag{1}$$

where $f \in [0,\ 1]$ is the forgetting factor;

- We compute the QR decomposition of the residue of $\hat{B}$ with respect to its projection onto the current subspace, that is: $\tilde{B} = \mathsf{qr}(\hat{B} - UU^T\hat{B})$ and we form a matrix R given

[1]All algorithms described here and in the result section were applied to all three channels (R, G, and B) separately.

$$\text{by } R = \begin{bmatrix} fD & U^T B \\ 0 & \tilde{B}\left(\hat{B} - UU^T\hat{B}\right) \end{bmatrix};$$

- Finally, we compute the SVD of R to obtain $\tilde{U}\tilde{D}\tilde{V}^T = \mathsf{svd}\,(R)$;

- The updated subspace is given by

$$U' = \left[U,\, \tilde{B}\right]\tilde{U} \qquad and \qquad D' = \tilde{D} \tag{2}$$

where a new κ is updated by $\kappa = f\kappa + n$.

The forgetting factor f plays an important role in adjusting the weight between old and new observations. In order words, depending on the image sequence and the application for which one needs to subtract the background, it may be desirable to quickly incorporate the changes into the subspace. On other scenarios, however, it may be desirable to maintain the learned background for a longer period of time. While we will show how to adjust this forgetting factor in Section 3, one should keep in mind that, with each iteration, the importance of any observation degrades by an additional factor of f^2 [25]. Similarly, the size of the temporal window can also affect the performance of the background subtraction. The choice of this parameter n is also discussed in Section 3.

It should also be noted that the appearance model of the background may become corrupted due to outliers in the calculation of $\mathbf{r}_a$. That is, the update of the subspace representing the background may be polluted with foreground objects present in the new observations. In order to solve this problem, we propose a solution that guarantees a clean adaptation of the appearance model by updating the subspace with synthetic background images. This idea will be explained in details in Section 2.2.3.

The actual background subtraction is achieved by extracting foreground silhouettes as new images arrive. For that, a foreground mask $\mathbf{s}$ of the newly observed image I, over locations x, must be determined using the following union operation:

$$\mathbf{s}(x) = \cup_{c=r,g,b} \left(||\mathbf{y}_c(x) - \hat{\mathbf{y}}_c(x)|| \geq thresh_c \right) \quad (3)$$

where $||\mathbf{y} - \hat{\mathbf{y}}||$ denotes the Euclidean distance between the vector representation y of the image I and its projection $\hat{y}$ onto the subspace. Also $thresh_c$ is the threshold for each corresponding color channel c.

2.2.2 Using Multiple Feature Subspaces

As we will show in the results section, the above method works reasonably well under gradual illumination changes, but its major disadvantage comes from the use of a single subspace. As we already explained, one reason for this poor performance is because a single subspace must capture all the lighting conditions combined. But another reason is because the adaptation of the subspace can be slow compared to the illumination change. That is, changes in illumination may lead to a large number of misdetections over long periods of time. To address this problem, we propose the use of multiple subspaces formed by a local principal component analysis [26], so that each subspace can learn a specific lighting condition. This strategy allows for a quick response to previously observed lighting conditions without involving long learning processes. That does not mean to say that our approach cannot adapt to new conditions. That only means that initially learned lighting condition can be locally stored as a subspace so that the system can quickly respond to similar lighting conditions. However, the use of multiple subspaces raises a few questions: 1) how to initialize the subspaces; 2) how to adapt each subspace; and 3) how to subtract the background using the subspace that best represents the current background. Please, refer back to Figure 2 for a visualization of these steps within the framework.

Initialization

In order to initialize the subspaces, our ALPCA handles each subspace separately. For example, the current and new blocks of observations are now divided into $A^{(i)} = \left[\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_k^{(i)}\right]$,

and $B^{(i)} = \left[\mathbf{x}_{k+1}^{(i)}, \ldots, \mathbf{x}_{k+n}^{(i)}\right]$, where i is the index of the subspace clustered by the ALPCA. The clustering of these observations is obtained by Vector Quantization (VQ) with reconstruction error as the distortion measurement. We chose this criterion because instead of a simple Euclidean distance, the reconstruction error is equivalent to a Mahalanobis distance, which preserves the different *scales* (eigenvalues) of the PCA. That is, as shown in [26], by taking the reconstruction error as the distortion measure, we preserve the weight of the information along each leading direction of the principal components. Next, we explain our clustering method through Vector Quantization.

Given the matrix $A = \bigcup_{i=1}^{q} A^{(i)} = [\mathbf{x}_1, \ldots, \mathbf{x}_k]$ with dimension $d \times k$, for each subspace i, m leading eigenvectors $U^{(i)} = \left[\mathbf{e}_1{}^{(i)}, \cdots, \mathbf{e}_j{}^{(i)}, \cdots, \mathbf{e}_m{}^{(i)}\right]$ are retained for each cluster $C^{(i)}$. Then, the projection of the image vector $\mathbf{x}$ onto $C^{(i)}$ is given by

$$\mathbf{z} = \sum_{j=1}^{m} \mathbf{e}_j^{(i)T} \left(\mathbf{x} - \mathbf{r}^{(i)}\right) = U^{(i)T} \left(\mathbf{x} - \mathbf{r}^{(i)}\right) \tag{4}$$

where $\mathbf{r}^{(i)}$ is the reference vector or the mean vector of the subspace defined by the cluster $C^{(i)}$. The reconstruction of the image vector $\mathbf{x}$ is

$$\hat{\mathbf{x}} = \mathbf{r}^{(i)} + U^{(i)}\mathbf{z} \tag{5}$$

The reconstruction error can thus be calculated by

$$d(\mathbf{x}, \mathbf{r}^{(i)}) = \|\mathbf{x} - \mathbf{r}^{(i)} - U^{(i)}\mathbf{z}\|^2 \tag{6}$$

Using an iterative partitioning method, we first initialize the reference vectors $\left\{\mathbf{r}^{(i)}\right\}_{i=1}^{q}$ by randomly choosing q vectors from A. Also we initialize the covariance matrix of each subspace $\left\{D^{(i)}\right\}_{i=1}^{q}$ with the identity. Usually, $q \in [2, 5]$, but this value can be adjusted according to the specific image sequence, as explained in Section 3.

After the initialization of these parameters, the partitioning proceeds to separate the training data into the q clusters, where

$$C^{(i)} = \left\{ \mathbf{x} \mid d(\mathbf{x}, \mathbf{r}^{(i)}) \leq d(\mathbf{x}, \mathbf{r}^{(j)}); \forall\, i \neq j \right\} \tag{7}$$

with $d(\mathbf{x}, \mathbf{r}^{(i)})$ defined by equation (4).

Both the reference vector and the subspace covariance matrix in each iteration will be updated according to the following rules:

$$\mathbf{r}^{(i)} = \arg\min_{\mathbf{r}} \frac{1}{N^{(i)}} \sum_{\mathbf{x} \in C^{(i)}} d(\mathbf{x}, \mathbf{r}) \tag{8}$$

$$D^{(i)} = \frac{1}{N^{(i)}} \sum_{\mathbf{x} \in C^{(i)}} (\mathbf{x} - \mathbf{r}^{(i)})(\mathbf{x} - \mathbf{r}^{(i)})^T \tag{9}$$

where $N^{(i)}$ is the number of data points in cluster $C^{(i)}$. The leading eigenvectors of each subspace covariance matrix are also computed iteratively. The iteration terminates when the fractional change of the reference vector is below some predefined threshold. Due to the large dimension of the image vector $\mathbf{x}$, only a subset of the eigenvectors for each subspace will be retained. As we explained earlier, the major advantage of our approach is that it achieves a quick convergence while it preserves local adaptation. At the same time, images under newly observed illumination will not disturb the previously learned subspaces, given an appropriate size of q. If similar lighting conditions are observed, our framework will quickly respond with the selection of the correct subspace, while the incremental learning scheme is able to capture and learn new lighting conditions.

Adaptation

The adaptation of multiple subspaces is done in a similar fashion with the single subspace. The major difference is that it must be carried out on each subspace separately. That is, all the equations in Section 2.2.1 must extended to now use $A^{(i)} = \left[\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_k^{(i)}\right]$, and

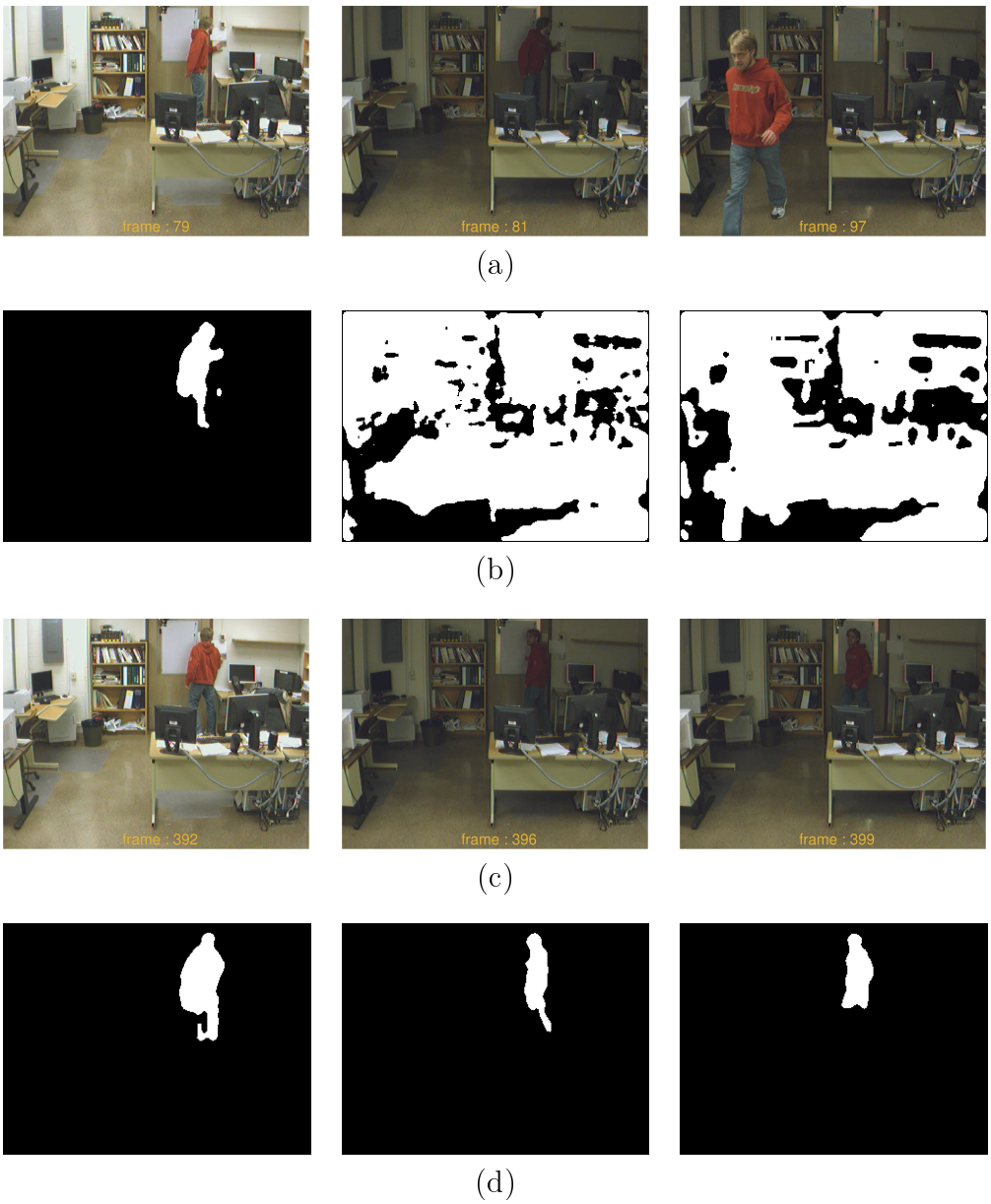


Figure 3: The first two rows show the response of the system to the first time a sudden illumination change is observed (frames 79 to 97). The last two rows show a very similar illumination change, but this time after the system had adapted to the different subspaces (frames 392 to 399).

$B^{(i)} = \left[\mathbf{x}_{k+1}^{(i)}, \ldots, \mathbf{x}_{k+n}^{(i)}\right]$, with i as the index of the subspace. Also, it is important to notice that the forgetting factor affects each subspace separately since the image used for adaptation is classified by the algorithm before it is used to update its own subspace.

As we mentioned in the beginning of this section, the purpose of the initialization described above is to allow our system to respond correctly from the very beginning of the image sequence. However, the system is also capable of learning and adapting to new conditions. This idea is depicted in Figure 3 where we present two new, but similar changes of illumination – that is, that specific lighting condition was not learned during initialization. As the Figure 3a-f shows, at the first time the system observes this change, the response is not ideal, since the subspace had not been formed yet. However, at the second time a similar transition occurs (Figure 3g though l), the system quickly responds based on the subspaces learned during adaptation.

In other words, whether the multiple subspaces are perfectly initialized or not, the adaptation allows the eigenvectors defining the multiple subspaces to eventually move towards the clusters. Obviously, the required number of subspaces depends on the sequence and how different its backgrounds are. In Section 3, we will discuss the selection of the optimum number of subspaces.

Background Subtraction

Since we now have multiple subspaces representing different background appearances, when it comes to actually extracting the foreground from the background, the system must find the subspace that best represents the current background – the red squares in Figure 2. We assume that the best subspace is the closest one, and by *closest* we mean the subspace with the set of eigen features that provides the minimum reconstruction error for the new image. This set of eigen features are then used in equation 3 to obtain the foreground mask $\mathbf{s}$ as explained in Section 2.2.1.

(a) Original input images for learning

(b) Learned *polluted* background due to foreground objects

(c) Poor foreground extraction due to polluted background model

(d) Learned background after eliminating foreground objects

(e) Clean foreground extraction obtained by our method

Figure 4: Two examples of foreground extraction: (a) Original Images, (b) and (c) unsuccessful foreground subtraction due to foreground objects corrupting the background model; and (d) and (e) our proposed method.

2.2.3 Learning using Synthetic Background Images

As mentioned earlier, another advantage of our method comes from the use of synthetic images during adaptation. That is, since input images may include foreground objects, our framework provides a new mechanism that prevents such objects from being learned as part of the models of the background. Intuitively, any incremental learning with non-static images may introduce foreground pixels as outliers for the background models – as shown in Figure 4. These misrepresentation of the true appearance of the background may result in future misdetection during foreground extraction. For this reason, our system has the option to update the subspaces with the synthesized background of the observed images. This option can be turned off whenever the application requires that foreground object be slowly incorporated into the background.

In order to understand how this background synthesis works, let us call $\mathbf{b}$ the synthetic background of the original image vector $\mathbf{y}$. Our framework is able to calculate $\mathbf{b}$ by means of the foreground mask $\mathbf{s}$ from the silhouette extraction. That is, synthetic background is given by:

$$\mathbf{b} = (\mathbf{s} \wedge \mathbf{r}^{(j)}) \vee (\mathbf{y} \wedge \bar{\mathbf{s}}) \tag{10}$$

That is, first we remove the foreground pixels by an *"AND"* operation of the image $\mathbf{y}$ with the complement of the mask, $\bar{\mathbf{s}}$. Then, we add (*"OR"* operation) the background pixels obtained from one of the mean vectors of the various subspaces: $(\mathbf{s} \wedge \mathbf{r}^{(j)})$. The mean vector $r^{(j)}$ is chosen as in: $j = \arg\min_i \, d(\mathbf{y}, \mathbf{r}^{(i)})$.

Our method should be contrasted with other systems in the literature that only use foreground-free images/pixels for updating the background. For example, in [27] background pixels are classified into *permanent* and *non-permanent* and only pixels observed long enough are regarded as permanent and incorporated in the background model. Also, in [28], only

blocks classified as background are used for updating. Especially in the second case, these methods can cause a large delay in the adaptation, since foreground-free images can take a long time to appear again.

2.2.4 Computational Complexity and Other Observations

The analysis of the computational complexity of our method can be divided into two parts. The first part refers to the initialization step of the algorithm and it represents the most onerous of the two complexities. The major reason for this complexity is the calculation of the singular value decomposition of each large matrix $A^{(i)}$, containing all the initial images used for the learning of the subspaces. Fortunately, since this step is executed only at the beginning of the algorithm, its computational cost does not affect the real-time performance of our method.

On the other hand, a much more efficient computational complexity comes from the second part of the algorithm, which performs the adaptation step. In this step, the SVD of the matrix $A^{(i)}$ must be updated for the newest n images observed for a same cluster i. As we explained in Sections 2.2.1 and 2.2.2, this step involves the computation of the QR decomposition of the matrix $B^{(i)}$, with dimension $d \times n$ – where d is the size of the images – and the subsequent calculation of the SVD of R, with dimension $2(d \times n)$. The QR decomposition was implemented using the modified Gram-Schmidt algorithm described in [29], which performs on the columns of $B^{(i)}$ only. Also, we used the "partitioned R-SVD" algorithm described in [29] and [30] to extract the m principal components, or first eigenvectors, of R. These components are used to adapt the original subspace $U^{(i)}$ containing the eigenvectors according to eq. (2). The complexity of the algorithm for adaptation of each subspace is therefore $O(dmn)$, [30]. Since this adaptation only needs to occur after n new images of one same subspace are observed, at which point one single subspace is updated, the final total complexity of the adaptation is also $O(dmn)$. However, in order to decide to which subspace an image belongs, the residue for each subspace needs to be computed.

This calculation takes $O(dmq)$ time, and since in general $q \ll n$, the final complexity of the entire algorithm is $O(dm(n+q))$ or simply $O(dmn)$. This complexity made it possible for a C++ implementation of the algorithm to achieve frame rate performance for normal VGA images.[2]

In order to improve even further the performance of our system, we embedded a heuristics into our implementation. We use a simple motion detection test to determine whether a drastic illumination change occurred or if a moving object is present in the scene. If only an illumination change happened, the framework forces a quick adaptation by setting the foreground mask to *zeros*. Otherwise, the foreground mask is left alone. The test is performed by analyzing two consecutive frames I_k and I_{k+1}, binarizing its temporal difference, applying a morphological opening, and thresholding the moving pixels.

Another advantage of our framework is its ability to fall back into one of the early subspace-based methods. That is, in an extreme case where the number of multiple subspaces is found to be one, our system reduces to an adaptive eigenbackground (APCA). Also, if the forgetting terms is set to zero, the behaves like the local eigenbackground without adaptation (LPCA) [24]. If both options were removed, the system would behave as a simple eigenbackground (pure PCA).

3 Experimental Results

We tested and compared our proposed ALPCA to four other approaches using six different datasets: four benchmark datasets available from the web, and two datasets that we created. The reason for our own datasets is because none of the benchmark datasets available had changes in illumination that were drastic and/or sudden enough to test our algorithm. That is, while most datesets contained gradual, *"over-the-day"* kinds of illumination changes, in order to test our algorithm, we needed the kinds of change as the ones in Figure 5e) and f).

[2]This C++ implementation is being made publicly available from our website.

Here is a short description of each dataset used. The reader can refer to Figure 5 for a few examples of the frames in each of these datasets.

The "Dance" contains more than one thousand frames of a graphically generated indoor scene with two dancing characters. While this sequence contains a somewhat sudden change of illumination, these changes are very subtle, besides this is a synthetic image sequence. This is part of the VSSN 2006 dataset[3].

The "Campus" video sequence is part of the PETS 2001 dataset[4]. We used almost four thousand of its frames, including various with gradual illumination change (over the period of a day) and people walking at a reasonably far distance from the camera.

The "Lobby" is also an outdoor video sequence with almost five hundred frames. It is part of the PETS 2004 dataset [5] and it also contains people meeting/chatting at the lobby of the INRIA Lab, in France.

The "Subway" is the last of the benchmark videos used in our tests, and it contains a mix of natural and artificial illumination sources. It is part of a dataset called PETS 2006 [6] and we used more than fourteen hundred of its frames. It was shot at a subway station and it captured people coming in and out of the station.

The "Sudden-Change" is the first of our new video sequences. It was shot inside our lab and it was created with the purpose of testing our algorithm for drastic and sudden changes in illumination. During this five-hundred-frame video sequence, half of the light fixtures are switched on-and-off separately, creating three different combinations of lighting conditions. A person moves back and forth in front of the camera. Both this and the next sequences are available from our lab website[7].

[3]http://imagelab.ing.unimore.it/vssn06
[4]http://www.cvg.cs.rdg.ac.uk/PETS2001
[5]http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/
[6]http://www.cvg.cs.reading.ac.uk/PETS2006/data.html
[7]*http://vigir.missouri.edu/~evan/backgroundsubtration*

(a) The *Dance* sequence (VSSN2006)

(b) *Campus* (PETS2001)

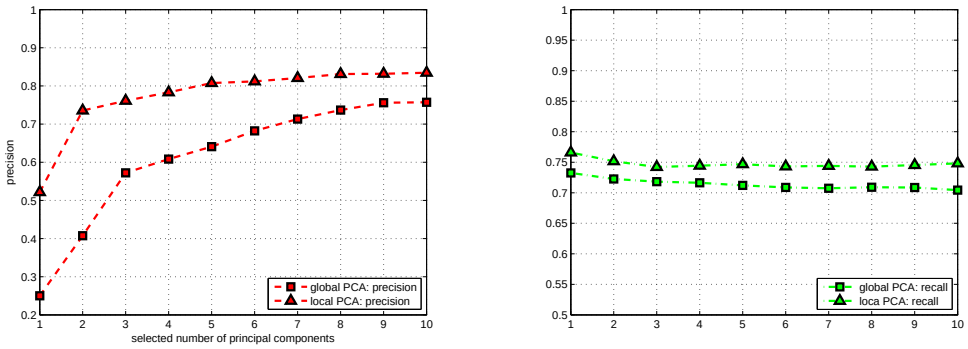
(c) *Lobby* (PETS2004)

(d) *Subway* (PETS2006)

(e) *Sudden-Change*

(f) *Sudden-Change-Door*

Figure 5: Samples from the six video sequences used for testing



(a) (b)

Figure 6: Effects of the number of eigenvectors on Precision and Recall for a typical test sequence: *"Sudden-Change-Door"*.

The "Sudden-Change-Door" is a sequence similar to the one above, with the addition of a sudden background change. That is, while some of the lights are switched off, a door in the back of the room is opened and closed. The light from the hallway floods the room, creating yet another set of combinations of lighting conditions.

3.1 Parameter Selection

As we mentioned earlier, there are three parameters that need to be selected: number of subspaces, forgetting factor, and size of the temporal window. In order to appreciate the impact of these parameters in the performance of the ALPCA, we tested each one against the video sequences above. In the next subsections, we will discuss the choices of optimum values for each one of these parameters. However, as we will demonstrate next, unlike the other systems in our comparisons, the ALPCA is not very sensitive to the choice of parameters, which makes this approach more attractive. This analysis was made using two metrics: *Precision*, $(\frac{N_{tp}}{N_{tp}+N_{fp}})$ and *Recall*, $(\frac{N_{tp}}{N_{tp}+N_{fn}})$, calculated for each sequence in its entirety.

3.1.1 Number of Eigenvectors

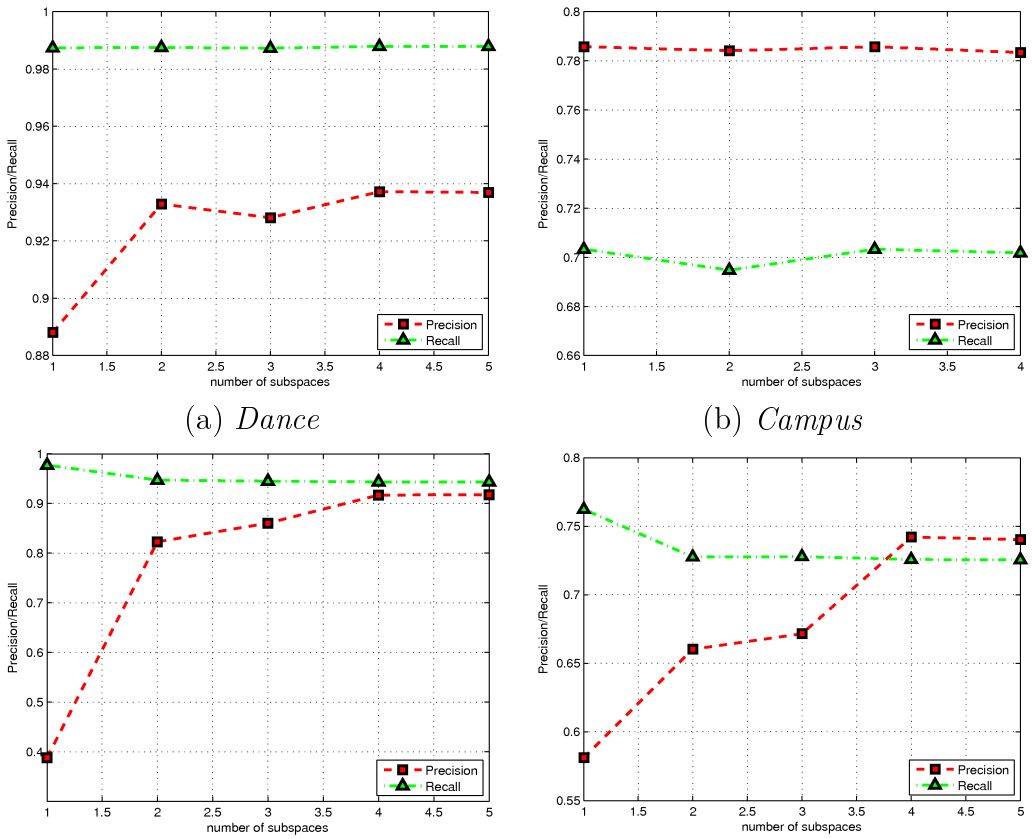
The first parameter to be determined is the number m of eigenvectors used to represent each cluster. In order to make this choice, we performed a test over the entire set of images in

each sequence and calculated the precision and recall versus the number of eigenvectors used in the construction of the subspace. As the Figure 6 (a) and (b) illustrate, the incorporation of multiple subspaces achieves superior performance over single subspace. As the readers will notice, the precision/recall becomes smooth after the fifth principal component for the multiple subspaces. In Appendix A, we discuss in more depth the implications of these results, but it is clear that a choice of five eigenvectors should be enough to capture most of the variations in the images and that was the value used in the next sections for the proposed ALPCA method.

3.1.2 Number of Subspaces

In order to determine an optimum number of subspaces q, we computed Precision and Recall as a function of that number. As Figure 7 shows, the optimum q varies from one to five. For example, for the *Campus* sequence (Figure 7b), since the change in illumination happens very slowly, any adaptive method should be able to handle such sequences. That is, there is no need to keep multiple models of the background appearances at any single moment since the adaptation itself can handle those changes. In that case, the ALPCA can be made to capture the appearance background using one single adaptive subspace, and the ALPCA is reduced to the APCA approach. Also, the performance of the two approaches should be quite similar, if not identical. On the other hand, for the *Dance* sequence, which contains a somewhat sudden change in illumination (Figure 7a) and for our two new sequences, which contain sudden as well as drastic changes of illumination (Figure 7c and d), the ALPCA required at least two subspaces. In some cases, achieving an improvement of Precision by more than 50% (e.g. Figure 7c).

In general, a q equal to four will work for most sequences. In fact, a slightly larger number, say five or six, could also be used just for "safety". That is, whenever a number of subspaces larger than the actual number of background appearances is specified, the ALPCA will either split the samples among two congruent subspaces, or into two closely adjacent



(a) *Dance* (b) *Campus*

(c) *Sudden-Change* (d) *Sudden-Change-Door*

Figure 7: Precision/Recall versus Number of Subspaces, q, for four of the six test sequences.

subspaces (with possible overlaps). In either cases, these subspaces will remain *"redundant"* until a newly observed background appearance forces them apart. This property of the ALPCA must be used carefully though. These *redundant* subspaces can cause a slight decrease in the performance of the system, as Figure 7a and b demonstrate for $q = 3$ and $q = 2$, respectively.

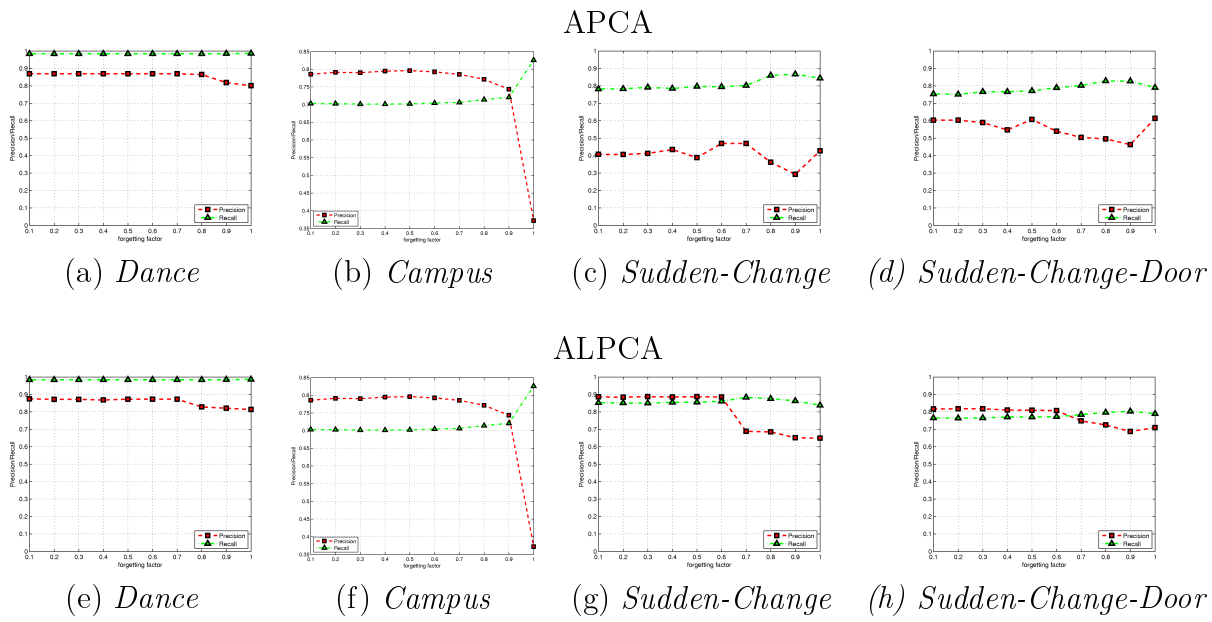
3.1.3 Forgetting Factor

Figure 8 shows the effect of the f in the APCA as well as the proposed ALPCA. As we have already mentioned, whenever ALPCA is applied to a simple sequence – i.e. with only gradual changes in illumination – the performance of the APCA and ALPCA are quite similar – e.g. the *Campus* sequence in Figures 8b and f, where we forced ALPCA to only one subspace. On the other hand, for more challenging sequences (Figures 8c, g and d, h) the improvement achieved by the multiple subspaces is quite obvious.

From eq. (1), we can see that the forgetting factor, f, controls how much of the past observations the adaptation algorithm retains. However, the new observations are always used to change the subspaces, regardless of f – i.e. the weight used for the new block of observations, B in eq. (1), is never zero. Moreover, compared to the APCA, this parameter plays a much smaller role in the proposed ALPCA – any value between 0.1 to 0.6 works for all the sequences. This should not be a surprise since the APCA has only a single subspace to learn all observed background appearances, and different values of the forgetting factor in the APCA can affect significantly the adaptation. On the other hand, for the ALPCA, the forgetting factor affects the subspaces independently, retaining all other subspaces unchanged while it adapts the current one.

3.1.4 Size of the Temporal Window

The last parameter to be tuned is the temporal window size: n. Figure 9 shows the behavior of our approach as a function of n. As before, the performances of the APCA and ALPCA



(a) *Dance* (b) *Campus* (c) *Sudden-Change* (d) *Sudden-Change-Door*

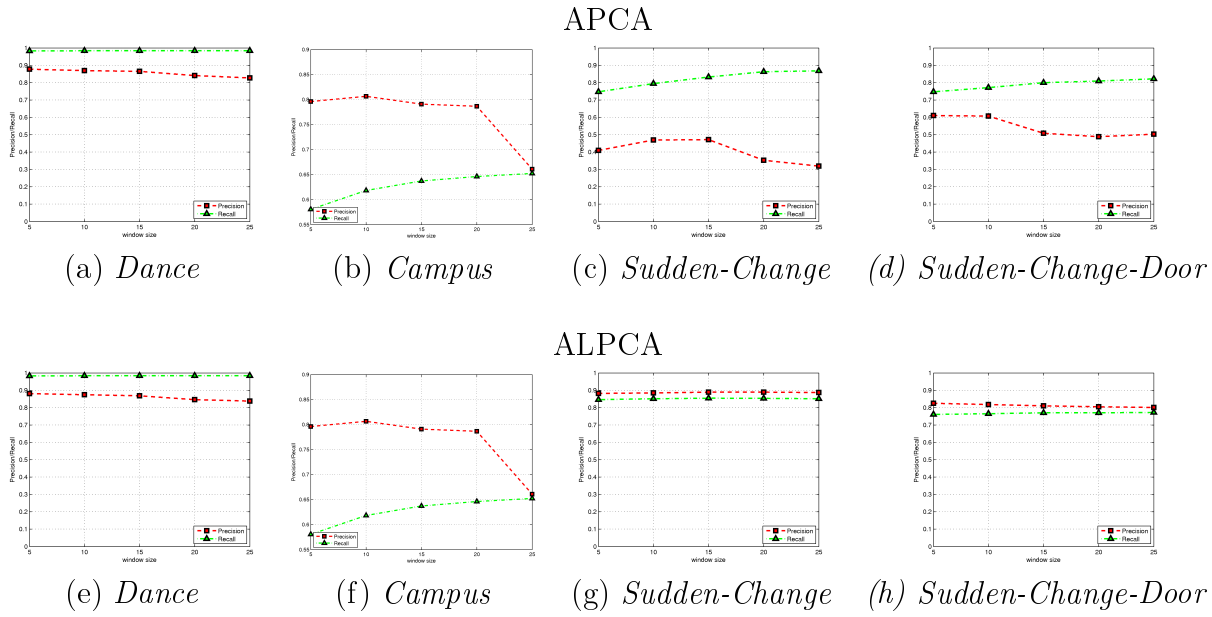
(e) *Dance* (f) *Campus* (g) *Sudden-Change* (h) *Sudden-Change-Door*

Figure 8: Precision/Recall versus Forgetting Factor. First row: APCA, and second: ALPCA.

for simple sequences are quite similar – e.g. Figures 9b and 9f, where again the changes in illumination happen so slowly that adaptation with a single subspaces ($q = 1$) is sufficient to effectively detect the background. In most cases, as with the forgetting factor, the ALPCA is quite insensitive to changes in the temporal window, but for others, a large value for the temporal window size, n, may lead to a decrease in performance. The reason is similar to what happens with the forgetting factor f, since the window size affects the adaptation in a similar fashion. That is, a large value of n may also leads to a slow adaptation. In general, the advantages of ALPCA over APCA becomes quite distinct.

Table 1: Quantitative evaluation between APCA and ALPCA

	Dance		*Campus*		*Sudden-Change*		*Sudden-Change-Door*	
	p (%)	*r (%)*	*p (%)*	*r (%)*	*p (%)*	*r (%)*	*p (%)*	*r (%)*
APCA	86.76	97.29	79.08	63.76	47.13	83.20	60.76	76.08
Proposed ALPCA	88.04	98.27	79.08	63.76	88.92	85.48	82.53	77.14



(a) *Dance* (b) *Campus* (c) *Sudden-Change* (d) *Sudden-Change-Door*

(e) *Dance* (f) *Campus* (g) *Sudden-Change* (h) *Sudden-Change-Door*

Figure 9: Precision/Recall versus Size of Temporal Window. First row: ALPCA, and second: APCA.

3.2 Quantitative Results

The results above were used to select the optimum set of parameters for both the APCA and ALPCA for all tests presented hereafter. Once we found these parameters, we run the APCA and ALPCA for all datasets and averaged *Precision/Recall* for all the frames in each sequence. Table 1 presents the results for four of those sequences: *Dance, Campus, Sudden-Change, and Sudden-Change-Door.* As expected, the performances of the two methods are quite similar for simple sequences, but under sudden changes in illumination, the ALPCA presents a much better performance.

3.3 Qualitative Results

In order to demonstrate how effective the proposed method is in terms of foreground segmentation, our algorithm was qualitatively compared it to five other approaches: an adaptive GMM-based algorithm [4], and four approaches using background appearances: pure PCA, Adaptive PCA, Local PCA, and Adaptive Local PCA. For the cases where training was

required (e.g. PCA), a small portion of the frames (~10%) were randomly selected and the remaining frames used for testing. Examples of the segmentation for the most challenging frames in each sequence are depicted in Figures 10 to 13.

From Figure 10, we notice that by simply adding the ability to handle multiple subspaces (Figure 10e) to the pure PCA approach, the LPCA can already perform much better than without it. In this test sequence (*the Dance),* sudden, but minor changes in illumination, are present, and one single subspace is not capable of capturing effectively the two different background appearances contained in the sequence. However, by learning from the beginning both appearances and storing them as separate subspaces, the LPCA can do a much better job of segmenting out the foreground. Also, for this specific sequence, the ALPCA does not offer much improvement, unless when we use the ALPCA without any prior training. In that case, the ALPCA will misdetect the foreground when it observes the change of illumination for the first time, but then, next time it will have adapted by creating another subspace and the performance will be again much better than that of any other method that uses a single subspace. This advantage of the ALPCA has already been discussed in Section 2.2.2, but Figure 11e shows yet another example of that. This time, since we used only a small portion of the frames for training of the LPCA and this method has no adaptation, the clustering formed during the training phase of the LPCA was ineffective, and the performance for those frames was not inferior with respect to the performances of the APCA and the ALPCA, which can adapt even to a bad initialization of the clusters (Figures 11d and f).

The previous benchmark sequences already demonstrate the advantages of a local treatment of the subspaces using an adaptive and local PCA. However, it is only after checking the results for our two new sequences that one can appreciate it fully. In those two cases, the ALPCA can extract the foreground almost perfectly, whether the illumination changes slowly as before, or suddenly as in the two sequences in Figures 12 and 13.

Finally, we also test the performance of the ALPCA using the other two benchmark videos: *Lobby* and *Subway.* Figure 14 shows these result. The results for all frames in all six

(a) Sample frames from the *Dance* sequence

(b) result using GMM

(c) result using PCA

(d) result using APCA

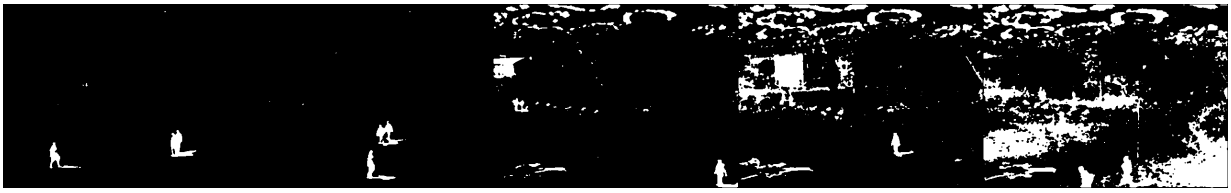
(e) result without adaptation: LPCA

(f) result using our proposed method with adaptation: ALPCA

Figure 10: Results for the *Dance* sequence.

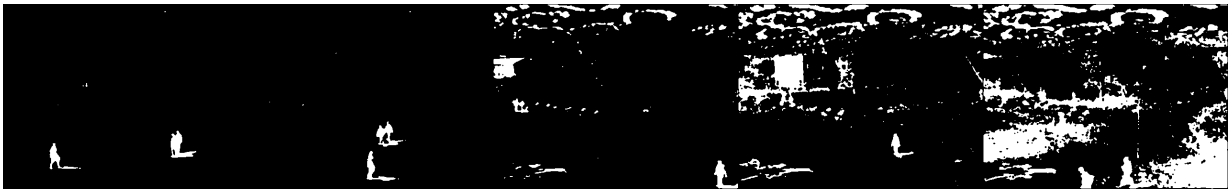
(a) Sample frames from the *Campus* sequence

(b) result using GMM



(c) result using PCA

(d) result using APCA



(e) result without adaptation: LPCA

(f) result using our proposed method with adaptation: ALPCA

Figure 11: Results for the *Campus* sequence.

(a) Sample frames from the *Sudden-Change* sequence

(b) result using GMM

(c) result using PCA

(d) result using APCA

(e) result without adaptation: LPCA

(f) result using our proposed method with adaptation: ALPCA

Figure 12: Results for the *Sudden-Change* sequence.

(a) Sample frames from the *Sudden-Change-Door* sequence

(b) result using GMM

(c) result using PCA

(d) result using APCA

(e) result without adaptation: LPCA

(f) result using our proposed method with adaptation: ALPCA

Figure 13: Results for the *Sudden-Change-Door* sequence.

(a) Original frames from the *Lobby* sequence

(b) Results from ALPCA

(c) Original frames from the *Subway* sequence

(d) Results from ALPCA

Figure 14: Two more results using ALPCA: the *Lobby* (a)-(b) and *Subway* sequences (c)-(d)

sequences can be downloaded from our website at:

*http://vigir.ee.missouri.edu/~*evan*/backgroundsubtraction.*

4 Conclusion and Future work

We proposed a new multi-subspace adaptive algorithm that can robustly and effectively extract foreground objects under various illumination conditions: despite how sudden and drastic those changes may be. If trained during initialization, the feature space was partitioned into clusters corresponding to the different lighting conditions and background appearances in the training set. Then, Local Principal Component Analysis was performed to build a set of multiple subspaces that better represent each of the many background appearances. However, even when an off-line training is not possible, the proposed approach

offered an adaptation method using synthetic background images that eventually converged to the same set of multiple subspaces. In either cases, the foreground detection was shown to be more accurate and robust than other methods in the literature. That was true not only, but especially when the backgrounds presented sudden and drastic changes of lighting conditions.

The proposed method relies on a *"K-Means like"* clustering procedure. While this may be a simple clustering procedure, the results demonstrated that the ability to do adaptation makes the initial clustering not critical via-a-vis the final overall performance of the method. However, in the classification step, the algorithm relies on fixed thresholds for both the synthesis of the background used during adaptation as well as for the proper foreground extraction. In the future, more investigation will be conducted to make these thresholds adaptive with respect to the sequences.

The proposed system developed in C++ runs on a single core 2.8GHz Pentium IV and achieves 25fps in real time. The C++ code used in this work is being made available from *http://vigir.ee.missouri.edu/~*evan*/backgroundsubtraction.*

Appendix A

Based on intuition alone, one could wrongly assume that using a single $m \times q$-dimensional PCA space will always achieve better results than building q multiple m-dimensional subspaces as we proposed in our research. In fact, as we will demonstrate in this section, the actual intuition should lead exactly to the opposite conclusion.

Let us take for example the simple case of a 3-dimensional space with two clusters representing the two possible *appearances* of the data – in the real case of background subtraction, those clusters of background appearances would be caused, for example, by different lighting conditions, or by a door that is opened/closed in the back of the room, as we will discuss next. But despite the actual reason for the clusters, since the main purpose of using PCA is

to reduce the original dimensionality of the image space, in this initial and simplified case we must assume that $m \times q$ cannot equal 3 (the full dimensionality of the data), and therefore the question becomes whether a single two-dimensional (global) PCA will always perform better than two one-dimensional (local) PCAs.

Figure 15(a) depicts the 3D sample data as well as the three corresponding eigenvector bases. That is, the figure shows in red and green the two clusters of data points with the two one-dimensional vector bases, $\mathbf{e}_1^{(1)}$ and $\mathbf{e}_1^{(2)}$, and the single two-dimensional vector basis $\mathbf{e}_{1,2}^{(s)}$.

An intuitive demonstration that the above claim is not true – i.e. that local PCAs will instead often outperform a global PCA – can start by taking a data point $\mathbf{x}$ at the center of the first data cluster. In that case, it would be easy to visualize that despite of which eigenvector(s) are chosen for the local PCA subspaces, the reconstruction of $\mathbf{x}$ using the closest local PCA $\mathbf{e}_1^{(1)}$ will always be perfect – i.e. there will be no residue in the reconstruction of such $\mathbf{x}$ with respect to $\mathbf{e}_1^{(1)}$ since the distance of $\mathbf{x}$ to $\mathbf{e}_1^{(1)}$, in that case, is zero. On the other hand, the reconstruction of $\mathbf{x}$ using the global PCA space $\mathbf{e}_{1,2}^{(s)}$ will have a residue proportional to the remaining coordinate of $\mathbf{x}$ – which had to be discarded for the construction of the two-dimensional global PCA space. In other words, the residue in the reconstruction of $\mathbf{x}$, which is the distance of $\mathbf{x}$ to the plane spanned by the basis $\mathbf{e}_{1,2}^{(s)}$ will not be zero since the center of the local cluster does not necessarily lies on that plane.

Mathematically, this advantage of a local PCA versus a global PCA is demonstrated by taking the projection of $\mathbf{x}$ onto the local PCA subspaces, i.e.

$$\mathbf{z} = \sum_{j=1}^{m} \mathbf{e}_j^{(i)T} \left(\mathbf{x} - \mathbf{r}^{(i)}\right) = U^{(i)T} \left(\mathbf{x} - \mathbf{r}^{(i)}\right) \tag{11}$$

where $\mathbf{r}^{(i)}$ is the reference vector or the mean vector of the subspace defined by the cluster $C^{(i)}$ and $\mathbf{e}_j^{(i)T}$ are the eigenvectors of $C^{(i)}$ and define the basis $\mathbf{U}^{(i)}$. The best reconstruction of the data vector $\mathbf{x}$ is obtained with respect to the closest local PCA subspace:

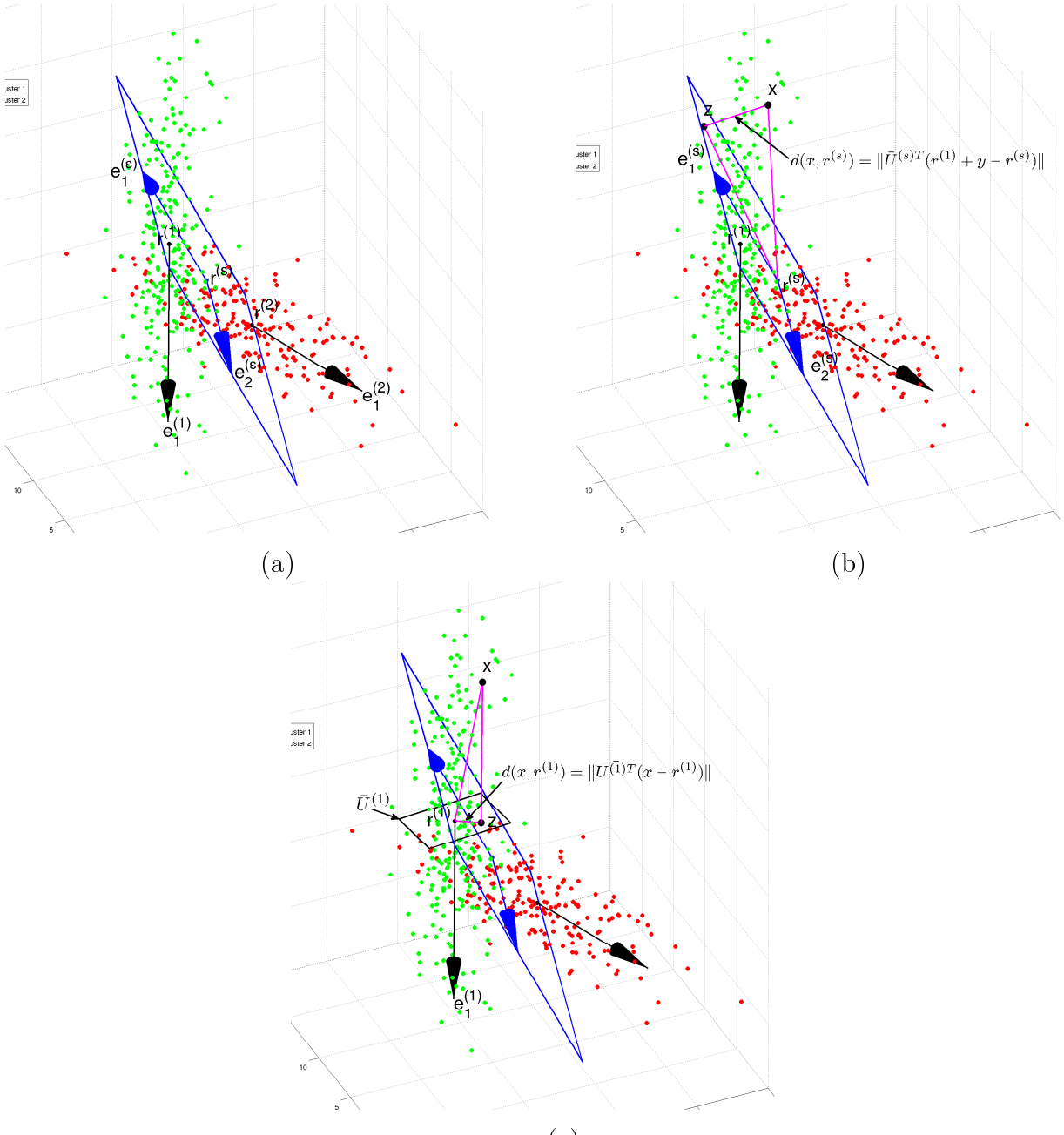


Figure 15: Single vs. multiple subspaces using 3D data: (a) Data points; (b) Single feature space captured by traditional global PCA approaches; (c) Same data represented by multiple (two) subspaces.

$$\hat{\mathbf{x}} = \mathbf{r}^{(1)} + U^{(1)}\,\mathbf{z}^T \tag{12}$$

and the reconstruction error, or residue, is given by the distance:

$$d(\mathbf{x},\, \mathbf{r}^{(1)}) \;\; = \;\; \|\mathbf{x} - \mathbf{r}^{(i)} - U^{(i)}\,\mathbf{z}^T\| \tag{13}$$

Since $\mathbf{x} = \mathbf{r}^{(1)}$, $\mathbf{z}$ will be equal to zero and so will be the residue $d(\mathbf{x}= \mathbf{r}^{(1)},\, \mathbf{r}^{(1)})$.

On the other hand, for the global two-dimensional PCA space $\mathbf{U}^{(s)}$, the non-zero reconstruction error is:

$$d(\mathbf{x},\, \mathbf{r}^{(s)}) \;\; = \;\; \|\mathbf{x} - \mathbf{r}^{(s)} - U^{(s)}\,\mathbf{z}^T\| \tag{14}$$

which, after a few simple manipulations, can be expressed as:

$$d(\mathbf{r}^{(1)},\, \mathbf{r}^{(s)}) \;\; = \;\; \|\overline{U}^{(s)T}(\mathbf{r}^{(1)} - \mathbf{r}^{(s)})\| \tag{15}$$

where $\overline{\mathbf{U}}^{(s)}$ represents the complement of vector basis $\mathbf{U}^{(s)}$ in $\mathbf{R}^3$.

For a more general case where $\mathbf{x}$ is not necessarily at the center of any of the clusters, such $\mathbf{x}$ can be expressed as $\mathbf{x} = \mathbf{r}^{(1)} + \mathbf{y}$ and the residues with respect to the local subspace $\mathbf{U}^{(1)}$ and the global space $\mathbf{U}^{(s)}$ become, after a few simple manipulations, respectively:

$$d(\mathbf{x},\, \mathbf{r}^{(1)}) \;\; = \;\; \|\overline{U}^{(1)T}\mathbf{y}\| \tag{16}$$

$$d(\mathbf{x},\, \mathbf{r}^{\,(s)}) \;\; = \;\; \|\overline{U}^{(s)T}(\mathbf{r}^{\,(1)} + \mathbf{y} - \mathbf{r}^{\,(s)})\| \tag{17}$$

Figures 15(b) and (c) show these residues as the respective distances of the point $\mathbf{x}$ to the complement spaces $\overline{\mathbf{U}}^{(1)}$ and $\overline{\mathbf{U}}^{(s)}$. Once again, for this simple 3D example, it is easy to intuitively infer the relationship between $d(\mathbf{x},\, \mathbf{r}^{\,(s)})$ and $d(\mathbf{x},\, \mathbf{r}^{\,(1)})$, that is, for which cases is $\|\overline{U}^{(s)T}(\mathbf{r}^{\,(1)} + \mathbf{y} - \mathbf{r}^{\,(s)})\| > \|\overline{U}^{(1)T}\mathbf{y}\|$, and the local PCA always better than the global PCA. For the more realistic application in question, background subtraction, our results in the previous sections already demonstrated the advantages of a local PCA. However, an analysis of the expression above can provide further insights regarding those cases. For example, since both residues depend on $\mathbf{y}$, it seems that the determinant term above is the distance between the centers of the global space and the local subspace: $\|(\mathbf{r}^{\,(1)} - \mathbf{r}^{\,(s)})\|$. That is, the further those two centers, the larger will be the residue of the reconstruction of $\mathbf{x}$ in the global PCA vis-a-vis the residue in the local one. That is exactly the case when one of the clusters is formed by a low-occurrence image, such as due to sudden illumination changes – those are like outliers of the main background appearance. Also, it is important to mention that the two residues above are calculated with respect to two different bases: $\overline{\mathbf{U}}^{(1)}$ and $\overline{\mathbf{U}}^{(s)}$, and despite the fact that the $dim(\overline{\mathbf{U}}^{(1)}) > dim(\overline{\mathbf{U}}^{(s)})$, those same dimensions are still much smaller than the actual dimension of the images. That is, usually in our experiments $dim(\mathbf{U}^{(1)}) = 5$ and the number of cluster $m = 5$, leading to the $dim(\mathbf{U}^{(s)}) = 25$, which is still much smaller than the image size $d = 640 \times 480 \times 3$.

In order to appreciate the fact that the image residue from the reconstruction of an image with respect to a global PCA is much larger than the image residue with respect to a local PCA, we present Figure 16. In that case, the same test image was reconstructed using a single, global PCA space Figure 16(d) and the closest of the two local PCA subspaces Figure 16(e). As it can be observed, the second residue is much smaller than the first, which leads

to a much better foreground segmentation, or background subtraction.

Finally, as we have already shown in Figure 6, the use of 25 dimensions for the global PCA space, i.e. $dim(\mathbf{U}^{(s)}) = 25$, is not a wise choice since the precision of the foreground detection does not increase much after 10 principal components are included – while the computational complexity certainly does.

(a)

(b)

(c)

(d)

(e)

Figure 16: Difference in the residue images for two clusters of background appearances using one single PCA space versus two local PCAs: (a) sample image from one of the clusters; (b) sample image from the second cluster; (c) test image similar to the first cluster, but with a person on the foreground; (d) residue image using a single, global PCA space; (e) residue image using the closest of two local PCA subspace.

References

[1] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell, "Toward robust automatic traffic scene analysis in real-time," in *Proceedings of the IEEE international conferences on Pattern Recognition*, 1994.

[2] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 780–785, July 1997.

[3] W. E. L. Grimson, C. Stauffer, and R. Romano, "Using adaptive tracking to classify and monitor activities in a site," in *Proceedings of the IEEE international conferences on Computer Vision and Pattern Recognition*, 1998.

[4] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings of the IEEE international conferences on Computer Vision and Pattern Recognition*, 1998.

[5] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 381–396, March 2002.

[6] Y.-L. Tian, M. Lu, and A. Hampapur, "Robust and efficient foreground analysis for real-time video surveillance," in *Proceedings of IEEE conferences on Computer Vision and Pattern Recognition*, 2005.

[7] D.-S. Lee, "Effective gaussian mixture learning for video background subtraction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, May 2005.

[8] O. Tuzel, F. Porikli, and P. Meer, "A bayesian approach to background modeling," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2005.

[9] J. Yao and J. Odobez, "Multi-layer background subtraction based on color and texture," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2007.

[10] M. Heikkila and M. Pietikainen, "A texture based method for modeling the background and detecting moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[11] A. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric background model for background subtraction," in *Proceedings of the 6th European conferences on Computer Vision*, 2000.

[12] K. A. Patwardhan, G. Sapiro, and V. Morellas, "Robust foreground detection in video using pixel layers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, April 2008.

[13] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1778–92, Nov. 2005.

[14] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintence," in *Proceedings of the IEEE international conferences on Computer Vision*, 1999, greece.

[15] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian, "Statistical modeling of complex backgrounds for foreground object detection," *IEEE Transactions on Image Processing*, November 2004.

[16] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh, "Background modeling and subtraction of dynamic scenes," in *Proceedings of IEEE international conferences on Computer Vision*, 2003, france.

[17] N. M. Oliver, B. Rosario, and A. P. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 831–843, August 2000.

[18] L. Wang, L. Wang, M. Wen, Q. Zhuo, and W. Y. Wang, "Background subtraction using incremental subspace learning," in *Proceedings of IEEE International Conference on Image Processing*, 2007.

[19] Y. T. Zhuang and C. Chen, "Efficient silhouette extraction with dynamic viewpoint," in *Proceedings of IEEE International Conference on Computer Vision*, 2007.

[20] Y. D. Zhao, H. F. Gong, L. Lin, and Y. D. Jia, "Spatio-temporal patches for night background modeling by subspace learning," in *Proceedings of IEEE International Conference on Pattern Recognition*, 2008.

[21] F. Kahl, R. Hartley, and V. Hilsenstein, "Novelty detection in image sequences with dynamic background," in *Proceedings of the 2nd Workshop on Statistical Methods in Video Processing (SMVP)*, 2004, pp. 117–128.

[22] X. Li, W. Hu, Z. Zhang, and X. Zhang, "Robust foreground segmentation based on two effective background models," in *Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval*, 2008, pp. 223–228, vancouver, British Columbia, Canada.

[23] R. Li, Y. Chen, and X. Zhang, "Fast robust eigen-background updating for foreground detection," in *Proceedings of IEEE international Conferences on Image Processing*, Oct. 2006, pp. 1833–1836.

[24] Y. Dong, T. X. Han, and G. N. **DeSouza** , "Illumination invariant foreground detection using multi-subspace learning," *Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 14, no. 1, pp. 31–41, 2010.

[25] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, May 2008.

[26] N. Kambhatla and T. Leen, "Dimension reduction by local principal component analysis," *Neural Computation*, vol. 9, pp. 1493–1516, 1997, mIT Press.

[27] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-Time Imaging*, vol. 11, no. 3, pp. 172–185, 2005, special Issue on Video Object Processing.

[28] H. H. Lin, Y. L. Liu, and J. H. Chuang, "Learning a scene background model via classification," *IEEE Transactions on Signal Processing*, May 2009.

[29] G. Golub and C. V. Loan, *Matrix Computations*. Baltimore: Johns Hopkins Univ Press, 1996.

[30] A. Levy and M. Lindenbaum, "Efficient sequential karhunen-loeve basis extraction," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, 2001, pp. 739–739.