Foreground Detection Using Spatiotemporal Projection Kernels

Yair Moshe[1] Hagit Hel-Or[1] Yacov Hel-Or[2]

[1]Department of Computer Science,
University of Haifa,
Haifa, 31905, Israel
{yair@ee.technion, hagit@cs.haifa}.ac.il

[2]School of Computer Science,
The Interdisciplinary Center,
Kanfey Nesharim St., Herzliya, 46150, Israel
toky@idc.ac.il

Abstract

In this paper, we propose a novel video foreground detection method that exploits the statistics of 3D space-time patches. 3D space-time patches are characterized by means of the subspace they span. As the complexity of real-time systems prohibits performing this modeling directly on the raw pixel data, we propose a novel framework in which spatiotemporal data is sequentially reduced in two stages. The first stage reduces the data using a cascade of linear projections of 3D space-time patches onto a small set of 3D Walsh-Hadamard (WH) basis functions known for its energy compaction of natural images and videos. This stage is efficiently implemented using the Gray-Code filtering scheme [2] requiring only 2 operations per projection. In the second stage, the data is further reduced by applying PCA directly to the WH coefficients exploiting the local statistics in an adaptive manner. Unlike common techniques, this spatiotemporal adaptive projection exploits window appearance as well as its dynamic characteristics. Tests show that the proposed method outperforms recent foreground detection methods and is suitable for real-time implementation on streaming video.

1. Introduction

Foreground detection is at the core of many video processing tasks such as tracking, video retrieval, or scene analysis. Efficient and accurate foreground detection relies on reliable background modeling, where common and expected background changes are statistically characterized. A robust background model should be capable of dealing with changes of illumination conditions, shadows, weather conditions (rain, snow), effects of moving elements of the scene (e.g., swaying trees, spouting fountains), and objects introduced or removed from the scene. Thus, it should characterize persistent and common structures or motions in the scene and efficiently differentiate between these and newly introduced objects.

A video sequence is a 3D signal with two spatial dimensions and one temporal dimension. In order to accurately characterize both the spatial and temporal structures, background modeling should describe joint space-time patches. The main difficulty with this approach is that processing overlapping space-time patches involves a huge amount of raw video data, making it impractical to perform in real-time systems. Thus, background modeling methods typically use some form of marginalization or reduction techniques.

Many background modeling techniques are performed on a pixel-by-pixel basis. The simplest such technique maintains a single background image that is computed using, for example, running average or median of the intensity of each pixel across time. Foreground objects can then be detected by subtracting the current frame from the background image. These methods perform well only in easy scenarios. A very popular approach for background modeling is to model each pixel as a mixture of Gaussians (MoG) [23]. When the assumptions imposed by parametric methods, such as MoG, fail, nonparametric approaches are more suitable. Some examples are the use of kernel density estimation (KDE) [7] or quantization of background values at each pixel into codewords [12]. Nonparametric based methods provide a flexible framework to represent multimodal densities. However, their high memory requirements and computational complexity inhibit the use of these methods in real-time applications. Predictive methods have also been used as dynamic models to predict the pixel intensity from previous observations [24].

As pixel-by-pixel background modeling techniques lack the ability to capture correlations among neighboring pixels, several authors have proposed techniques that account for spatial correlations. Spatial correlations together with some temporal consistency have been modeled using Hidden Markov Models (HMMs) [27], Markov Random Field (MRF) [21] and autoregressive models [15]. An interesting background modeling approach, first suggested in [17], uses Principal Component Analysis (PCA) to compute a small number of 'eigenbackgrounds' that capture the dominant variability of the background. Detection is performed by projecting input images onto the eigenbackgrounds and thresholding

the obtained distances. Since the basis vectors model only the background, image regions containing foreground objects are expected to be distant from their projections onto the background subspace. PCA is also used in [20] to compute a small number of eigenbackgrounds. However, in this method PCA is separately applied to each image block and then spatial correlations among blocks are exploited using co-occurrence of image variations. For more information on background modeling techniques, the reader is referred to [3, 5, 8, 19].

Background modeling using these methods gives satisfactory results on easy scenarios. However, it has been shown [4, 18] that these methods fail in the general case of scenes with dynamic backgrounds. For such scenes, more elaborate modeling of temporal frequencies is required.

In this paper, we propose a novel video background modeling technique that directly models the statistics of 3D space-time video patches, thus capturing both the appearance and dynamical characteristics of the scene. The suggested approach assumes that each background space-time patch lies within a low dimensional subspace (such as in [13]). Since real-time systems prohibit applying this reduction directly on the raw pixel data, we propose a new approach where reduction is performed successively in two stages. The first stage applies initial reduction by projecting each space-time patch onto a fixed set of basis vectors known to be efficient for natural video scenes. The second stage applies further reduction, this time adaptive, where each patch is projected into its local subspace using PCA. Since the second reduction is performed on the coefficients generated in the first stage, the complexity of the adaptive PCA is manageable and possible to apply in real-time. The suggested two-stage approach allows maintaining a compact background model using space-time patches. We show that this technique is efficient both in its run-time performance and in its discrimination power.

The remainder of this paper is organized as follows. Fast projection of space-time patches using the Gray-Code filtering scheme is described in Section 2. The proposed background modeling technique is presented in Section 3, and its results are given in Section 4. Finally, conclusions are drawn in Section 5.

2. 3D Projections using Gray Code Kernels

A common approach to obtain compact signatures for spatiotemporal video patches is to design a set of specific kernels which are efficient to apply. Studies that took this course of action include the integral image [26], which was later extended to integral video [11], summed-area tables [6], and a generalized version of these called boxlets [22]. These approaches have two main drawbacks. First, they allow only a limited set of kernels to be computed efficiently. Second, they do not form an orthogonal set of kernels, thus their extracted features may include redundant information. In this section, we propose a novel scheme for extracting a compact set of features for each spatiotemporal patch. The proposed scheme does not suffer from the above drawbacks.

An efficient pattern matching framework was recently presented in [9]. Using this framework, patterns and image windows are represented using a small set of coefficients that are calculated by projecting the pattern/windows onto a small set of Walsh-Hadamard (WH) basis vectors. For natural images, these basis vectors, ordered in increasing sequency (number of sign changes in the vector - analogous to frequency), capture a large portion of the window energy with very few projection coefficients on average. Furthermore, it is shown in [2] that the Walsh-Hadamard basis vectors are a specific case of a broader family of kernels – the Gray-Code Kernels (GCK). The GCK kernels are highly efficient for projection using only 2 operations per pixel per kernel independent of window size. We briefly review this technique. For further details, the reader is referred to [2].

Consider first the 1D case where signal and basis kernels are one-dimensional[1]. Each WH kernel in 1D can be generated by a sequence of Kronecker products of the strings $(++)$ and $(+-)$. For example:

$$(++)\otimes(++)=(++++)$$

$$(+-)\otimes(++)=(++--)$$

$$(+-)\otimes(++)\otimes(++)=(++++----)$$

Thus, the set of possible basis kernels is isomorphic to a binary string, say $(++)=1$ and $(+-)=-1$, according to their generating sequence. Each binary string in reverse order is called the *α-index* of the kernel. Thus, the α-index of the examples above is $[1\ 1]$, $[1\ -1]$ and $[1\ 1\ -1]$ respectively. Two basis kernels of similar length are defined to be *α-related* if the hamming distance between their α-index (the number of positions for which their α-indices differ) is one. Without loss of generality, let the α-indices of two α-related kernels be $(\alpha_1 \ldots \alpha_{r-1}, +1, \ldots \alpha_k)$ and $(\alpha_1 \ldots \alpha_{r-1}, -1, \ldots \alpha_k)$. We denote the corresponding kernels as v_+ and v_- respectively. It is shown in [2] that two α-related kernels always share the same prefix of length $2^{r-1}=\Delta$. Two α-related kernels also share a special relationship. Let b_+ and b_- be the two 1D signals resulting from convolving a signal x with two α-related filter kernels v_+, v_-, respectively, then it can be shown [2]:

$$\begin{aligned} b_+(i) &= +b_+(i-\Delta)+b_-(i)+b_-(i-\Delta) \\ b_-(i) &= -b_-(i-\Delta)+b_+(i)-b_+(i-\Delta) \end{aligned} \quad (1)$$

[1] We call the basis vector kernels as they are applied to the entire space-time space as convolution kernels.

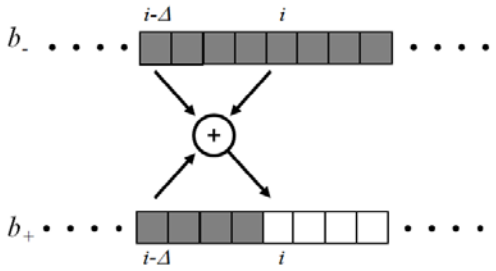


Figure 1. Given $\boldsymbol{b}_-$ (convolution of a signal with filter kernel $\boldsymbol{v}_-$), the convolution result $\boldsymbol{b}_+$ can be computed using 2 operations per pixel regardless of kernel size [2].

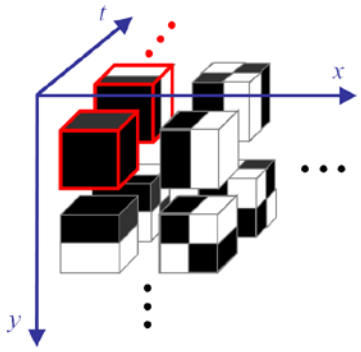


Figure 2. The set of 3D Walsh-Hadamard kernels in increasing sequency order. The projection onto kernels marked in red are stored in memory and are used for future projections.

This forms the basis of an efficient scheme for convolving a signal with a set of GCK. Given the result b_+ (b_-) of convolving the signal with the filter kernel v_+ (v_-), the result b_- (b_+) of convolving with the filter kernel v_- (v_+) requires only two operations per pixel independent of the kernel size (Figure 1). A sequence of WH kernels ordered in dyadic (or Paley) order of increasing sequency is shown to be consecutively α-related. Thus, sequentially convolving an image with kernels in the sequence requires only 2 operations per pixel per kernel regardless of signal and kernel size. For separable kernels, such as the WH kernels, the previous definitions and results can be generalized to higher dimensions. The computation cost remains at two operations per pixel per kernel regardless of the kernel size or dimension.

Based on the separability of the WH kernels, we propose to extend the fast GCK projection scheme into 3D spatiotemporal patches, as depicted in Figure 2. As in the 2D case, the 3D GCK coefficients are both informative and fast to compute. Figure 3 shows that for natural video sequences, the 3D WH projection vectors, ordered in increasing sequency, capture a large portion of the window energy with very few projection coefficients on average. The plot displays the percentage of energy accumulated by the projection coefficients. It can be seen that the first 10 (of 512) projections capture over 90

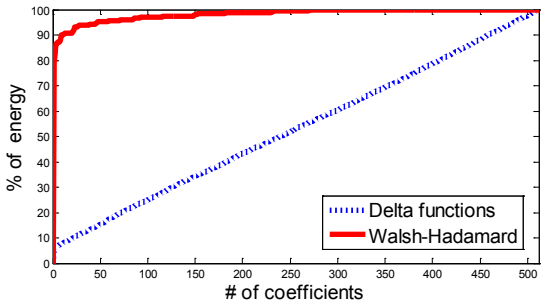


Figure 3. The percentage of space-time patch energy accumulated by projection coefficients for both the standard basis, i.e. delta functions, and for the 3D Walsh-Hadamard projections. The values displayed are averaged over 1000 8×8×8 pattern-window pairs chosen.

percent of the energy. For comparison, the percentage of accumulated energy is shown for the standard basis (delta functions), i.e. patch energy accumulated by summing squared pixels. In this case, more than 460 projections were required to capture 90 percent of the energy.

The extension of the efficient GCK projection framework into 3D is straightforward. However, the fact that the additional dimension is time and the fact that the signal is acquired incrementally (frame by frame) raises nontrivial implementation considerations. Remember that, according to Eq. (1), the efficient GCK projection scheme requires maintaining projections of Δ previous samples. In 2D this implies accessing image projections calculated to the left or above current image location. This is not a problem since the image and its previous projections are available in memory. In 3D, using Δ previous samples requires going back in time and maintaining all projection values for the last Δ frames. Frame buffering complicates the ordering of the projection kernels since Δ depends on the temporal length of the space-time patches as well as on the temporal sequency of projection kernels. Projecting onto a kernel with high temporal sequency may incur high storage complexity (requiring a larger number of previous projections to be maintained in memory).

To allow efficient buffering and low memory cost, we revise the notion of a linear sequence of α-related kernels and consider a connected graph of α-related kernels. In the suggested approach, we first compute the projections onto kernels with spatial sequency 0 (i.e. kernels that are spatial-DC kernels) across all required temporal sequencies (the kernels marked in red in Figure 2). From each of these kernels a linear sequence of α-related kernels can be formed, which follows a 'snake' order (see [16]) within the set of kernels of the same temporal sequency but having different spatial sequency. This scheme allows a significant reduction in memory usage as the only projections required to be stored in memory for the preceding video frames are those associated with the

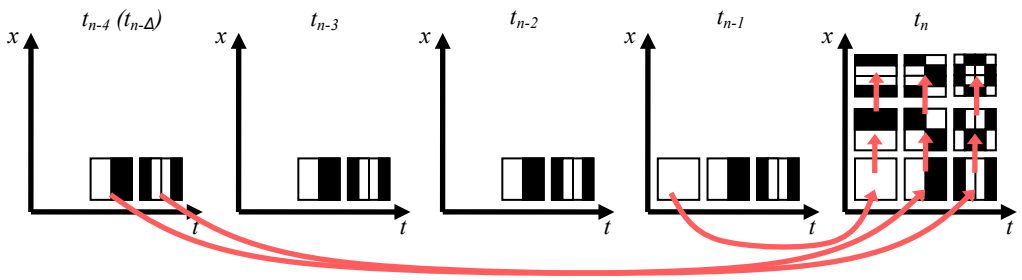


Figure 4. Proposed projection scheme for space-time patches with $\boldsymbol{p_s = 3, p_t = 3}$. To simplify illustration, the spatial domain is shown as 1D (*y*-axis). At time $\boldsymbol{t_n}$, projections are first performed onto spatial DC kernels across all required temporal sequencies (bottom row of kernels in array of $\boldsymbol{t_n}$). From each of these kernels, a sequence of projections is calculated having the same temporal sequency but having different spatial sequency (shown as red arrows in array of $\boldsymbol{t_n}$). These calculations require previously stored projections (shown as arc-arrows from the previous time frames). Only 9 (instead of 36) past frame projections are required to be maintained in memory.

spatial DC kernels alone. Thus, the projection scheme performs as follows (Figure 4). Given the *n*th video frame:

1. Calculate the spatiotemporal DC projection for the patches of the frame using the spatiotemporal DC projections of the previous frame. This can be performed with 6 operations per pixel using a separable box filter.
2. Calculate all spatial-DC projections (for all required temporal frequencies) using the fast projection approach [2]. This requires access to the spatiotemporal DC projections and spatial-DC projections computed for previous frames.
3. Free memory associated with frame number $(n - \Delta)$.
4. Calculate all remaining projections using a spatial 'snake' order (see [16]). This can be done based on the spatial-DC projections of frame *n* alone,
5. Store in memory only the spatial-DC projections of frame *n*.

Denote p_s and p_t the number of spatial and temporal sequencies of the kernels to be used. Also denote Δ_{max} the maxium Δ over all required temporal kernels. The suggested approach reduces memory requirement from $\Delta_{max} p_t p_s$ frame projection values to $\Delta_{max}(p_t - 1) + 1$ frame projection values. For example, for patches of size $8 \times 8 \times 8$ and with $p_s = 4, p_t = 4$, the number of stored frame projections reduces from 64 to 13. The GCK scheme with the suggested projection order was tested on real video sequences and patterns. The results show that the suggested projection method is highly efficient and achieves better-than-real-time performance. Table 1 summarizes running times and memory requirements for different projection configurations.

patch size (h×w×t)	# of kernels ($p_s \times p_t = m$)	run time (fps)	memory (frame proj.)
4×4×8	4×4=16	37	13
4×4×16	4×4=16	37	25
8×8×8	4×4=16	37	13
4×4×8	2×2=4	97	5
4×4×8	4×2=8	61	5

Table 1. Running times and memory requirements for projections of spatiotemporal video patches at D1 (704x576) resolution using one core of an Intel Core 2 Due @ 1.8Ghz.

3. Two-Stage Background Modeling

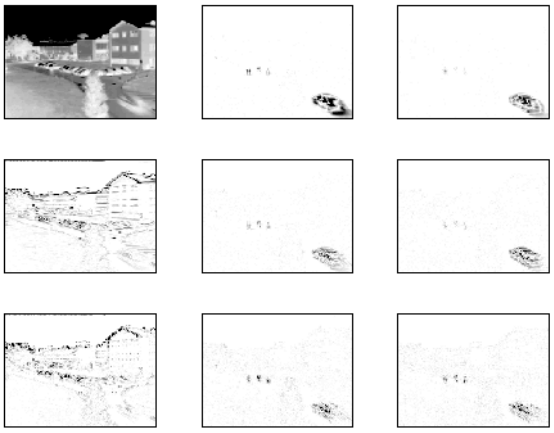
In this paper we propose a background modeling scheme that assumes each space-time background patch can be well modeled by a low dimensional linear subspace. Foreground patches are determined by their distance to the modeled subspace. However, since subspace modeling is performed on each space-time patch, performing subspace modeling on the raw data, as in [17, 20], is not feasible. Thus, we propose to apply subspace modeling (and classification) in two successive stages:

Global dimensionality reduction

The first stage reduces dimensionality of the data in a fast and efficient manner that is independent of the input content. This is performed by projecting space-time video patches onto a small set of 3D Walsh-Hadamard (WH) basis vectors that form a low-dimensional subspace. We utilize the Gray-Code filtering scheme [2] described in Section 2, which enables efficient projection using only 2 operations per window per projection, regardless of the window size. As shown earlier, for natural sequences, a few of these coefficients suffice to capture well the structure of the space-time patch.

Projecting each space-time patch using 3D GCK forms a low dimensional space that contains m WH coefficients. The kernels used and their order of sequencing is described in Section 2. Example of projection onto these kernels is shown in Figure 5. Figure 5(a) depicts a frame of a video sequence from the PETS 2001 dataset [1] which contains four foreground objects – a car, two pedestrians and a cyclist. Figure 5(b) depicts the coefficients obtained for the space-time patches associated with the pixels of this frame, for 9 different low sequency 3D WH

(a)



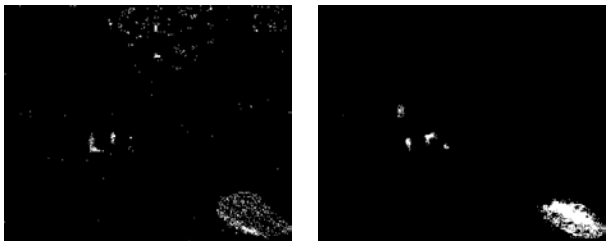
(b)

Figure 5. (a) A frame of a video sequence from the PETS 2001 dataset and (b) a set of its low sequency 3D Walsh-Hadamard projections obtained by projecting patches of size 4×4×8 (height×width×frames). Temporal frequencies of the projections increase left to right. Spatial frequencies of the projections increase top to bottom. Dark colors represent high coefficient values. Foreground objects of the scene have high values while static objects and background moving objects have low values.

projections. In the coefficient images, dark colors represent large coefficient values. It can be observed that patches associated with foreground objects have high coefficient values while static objects and background moving objects, e.g., waving trees, have low values. The figure demonstrates that by exploiting the global statistics of spatiotemporal data the set of low sequency spatiotemporal WH coefficients captures both spatial (appearance) and temporal (dynamic) frequencies in the scene.

Adaptive dimensionality reduction

The second stage of the process further reduces dimensionality by exploiting the local statistical characteristics of the space-time patches of the data which have been preserved in the first stage. The second stage builds a spatially adaptive lower dimensional subspace representing background patches. This is performed using Principal Component Analysis (PCA) [10] on the WH



(a) (b)

Figure 6. Foreground object detection results, for the frame from Figure 5(a) using (a) MoG and (b) the proposed technique.

coefficients calculated in the first stage.

Given n video frames composing the training data, we consider the n spatio-temporal patches of these frames associated with the same spatial location within all the training frames. We denote this set as a *block*. Every patch of the block is represented by m WH coefficients obtained in stage 1, thus n m-dimensional vectors are associated with each block. These are used to calculate the eigenspace that models the background at the block's spatial location. For each block, the mean of the WH vectors $\mu = E\{X_i\}_{i=1..n}$ is subtracted from the vectors to form a set of zero mean vectors $\{\hat{X}_i\}_{i=1..n}$. The $m \times m$ covariance matrix C is then computed as:

$$C = E\left\{\hat{X}_i \hat{X}_i^T\right\}_{i=1..n} \quad (2)$$

This covariance matrix can be diagonalized via eigenvalue decomposition:

$$D = \Phi C \Phi^T \quad (3)$$

where Φ is the eigenvector matrix of C and D is the corresponding diagonal matrix. In order to reduce dimensionality of the space, PCA is used to collect the top q eigenvectors (associated with the q largest eigenvalues) to form the matrix Φ_q. These q eigenvectors span the subspace representing the background block.

During classification into foreground and background, the 2-stage process is repeated. Given a new frame, the m WH coefficients are calculated for each space-time patch associated with pixels of the new frame, producing the vector Y_i. Following, the vector Y_i is projected onto the subspace defined by Φ_q by first subtracting the subspace mean μ obtaining:

$$\hat{Y}_i = Y_i - \mu \quad (4)$$

followed by a projection onto the subspace vector. The distance d between $\hat{Y}_i$ and the projected vector is computed:

$$d_i = \hat{Y}_i - \Phi_q \Phi_q^T \hat{Y}_i \quad (5)$$

To determine classification into foreground and background, a threshold operation is performed on d, assuming large values for foreground objects since they should not be modeled well by the eigenspace.

Note that the background model is constantly updated

along time to account for possible changes in the background model. The current estimate of eigenbackgrounds is updated through Incremental PCA [28] that updates with the current frame while the effect of the previous observations is exponentially reduced.

It is interesting to analyze the influence of parameters on algorithm performance. As discussed in Section 3, patch size ($h \times w \times t$) does not affect running time. Large space-time patches can better capture (spatial or temporal) frequencies while small space-time patches can better localize these frequencies. A large number, m of WH coefficients can capture more space-time frequencies but incurs high time complexity. Large number of q dominant eigenvectors can better represent patches but also incurs higher time complexity. Typical values of the aforementioned parameters will be discussed in the results section.

Finally, we consider the run time of the process. During training, applying PCA directly on the pixels of the sequence, requires $\mathcal{O}((hwt)^2 n)$ operations for each 3D patch. Using the suggested scheme only $\mathcal{O}(m^2 n)$ operations are needed for each patch since the m coefficients are calculated with only 2 operations per patch per coefficient. Typical values for hwt are in the range $[64\ 4096]$ while typical values for m are in the range $[4\ 16]$ thus, reducing the complexity of the proposed scheme by a few orders of magnitude compared to a naive scheme. Time complexity of the classification stage is dependent on the m WH projections, the projection onto the subspace defined by the q PCA coefficients and on thresholding. We have measured the running time of the proposed algorithm implemented in C and using the configurations from Table 1. It can be seen, in Table 2, that the classification stage is fast and outperforms real-time performance.

# of kernels ($p_s \times p_t = m$)	# of q largest eigenvalues	run time (fps)
2×2=4	2	39
4×2=8	2	26
4×2=8	4	19
4×4=16	2	26
4×4=16	4	18
4×4=16	8	17

Table 2. Running times for foreground detection using the proposed method at D1 (704x576) resolution using one core of an Intel Core 2 Due @ 1.8Ghz.

4. Results

The proposed method was tested on a variety of video sequences and compared with other state-of-the-art background modeling methods. In all tests, the proposed method was performed with space-time patches of size 8×8×8 and with $m = 16$ low sequency 3D WH projections that were further projected using Incremental PCA with $q = 4$ eigenvalues. Note that is this paper we concentrate on the luminance component. A chrominance component can be added independently to improve results.

Figure 6 depicts a comparison of background modeling of the frame from Figure 5(a) using the proposed method compared with the Mixture of Gaussians method (MoG) [23]. Typical to many background modeling techniques, the cloudy sky at the top of the scene is detected as foreground by the MoG. This is due to the inability of this approach to model correctly the temporal frequencies in this region that is coarsely quantized by the video codec. The proposed approach models well the dynamics of this background.

A significant advantage of the proposed background modeling method is its robustness to dynamic backgrounds. In order to further demonstrate the strengths of the proposed method, tests were performed with video sequences taken from [14]. These video sequences are difficult-to-model since they contain dynamic backgrounds (e.g., sea waves, a spouting fountain, and wavering trees). Figure 7 compares background modeling results for two frames from the Watersurface sequence and from the Airport sequence. The Watersurface sequence background contains highly dynamic sea waves. The Airport sequence contains a busy airport scene with many people crossing the camera. Both video sequences form a challenge to background modeling methods. Results are compared with the MoG [23], codebook model [12] and block-based eigenbackgrounds [17] methods. The MoG and eigenbackgrounds methods result in many false alarms. Results of the proposed method are substantially improved. Opposed to the other three methods, the

	MoG			codebook			eigenbackgrounds			proposed		
	prec.	recall	*F*	prec.	recall	*F*	prec.	recall	*F*	prec.	recall	*F*
Airport	0.71	0.32	0.44	0.41	0.67	0.51	0.65	0.51	0.57	0.67	0.71	**0.69**
Campus	0.73	0.09	0.11	0.67	0.32	0.43	0.41	0.41	0.30	0.52	0.57	**0.54**
Fountain	0.83	0.70	0.75	0.64	0.65	0.64	0.63	0.69	0.64	0.72	0.81	**0.76**
Lobby	0.85	0.35	0.41	0.68	0.36	**0.47**	0.72	0.23	0.21	0.61	0.56	**0.47**
Watersurface	0.91	0.46	0.57	0.82	0.61	0.70	0.74	0.68	0.71	0.82	0.77	**0.79**

Table 3. Quantitative evaluation results for 5 of the video sequences from [14] . The proposed method outperforms MoG [23], codebook model [12] and block-based eigenbackgrounds [17].

Figure 7: Foreground object detection for (a) two frames from the Watersurface video sequence (left) and from the Airport video sequence (right) together with their (b) ground-truth results. Object detection was performed using (c) MoG [23], (d) codebook model [12], (e) block-based eigenbackgrounds [17], and (f) the proposed method. The proposed method shows improved results compared to other techniques.

codebook model method has the advantage that it uses color information. Still, in these two video sequences, the proposed technique shows improved results compared to the codebook model technique.

For a quantitative evaluation, the performance of the proposed method was evaluated on 200 randomly selected frames from the video sequences of [14]. Like in [14], results were evaluated by comparing with the corresponding 20 ground truth frames for each sequence. Results are given in terms of *precision* (the fraction of detected pixels that are foreground pixels), *recall* (the fraction of foreground pixels that are detected) and *F-measure* [25] which combines precision and recall using even weighting:

$$F = 2\frac{precision \cdot recall}{precision + recall} \tag{6}$$

Results of several background modeling methods – MoG [23], codebook model [12], block-based eigenbackgrounds [17] and the proposed method, are

compared in Table 3. It can be observed that for these videos, background modeling using the proposed method outperforms the compared methods. In our tests, comparison with all other video sequences of [14] show similar results.

5. Conclusion

In this paper we propose a novel method for detecting foreground objects from complex environments. Unlike common foreground detection methods, the proposed method models the statistics of space-time background patches to exploit window appearance as well as its dynamic characteristics. Real-time performance is achieved in two stages of dimensionality reduction. The first stage reduces the data using fast and efficient projections of space-time video patches onto a small set of 3D Walsh-Hadamard (WH) basis vectors that utilize the Gray-Code filtering scheme. The second stage builds spatially adaptive lower dimensional subspace representing background patches by applying PCA to the WH coefficients. The suggested two-stage approach allows maintaining a compact background model that characterizes spatiotemporal structures. Tests show that the proposed method is fast and efficient and outperforms common background modeling methods in various scenarios.

Acknowledgments. This research was supported by the Chief Scientist Office at the Israeli Ministry of Industry, Trade & Labor, under the Magnet Program (VULCAN).

References

[1] Performance Evaluation of Tracking and Surveillance Workshop, PETS 2001, http://pets2001.visualsurveillance.org/.

[2] G. Ben-Artzi, et al. The Gray-Code Filter Kernels, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(3):382-393, 2007.

[3] T. Bouwmans, et al. Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey, *Recent Patents on Computer Science*, 1(3):219-237, 2008.

[4] T.H. Chalidabhongse, et al. A Perturbation Method for Evaluating Background Subtraction Algorithms, *Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, 2003.

[5] S.-C.S. Cheung and C. Kamath. Robust Techniques for Background Subtraction in Urban Traffic Video, *Proc. of SPIE*, 5308:881–892, 2004.

[6] F.C. Crow. Summed-area Tables for Texture Mapping, *Ann. Conf. Computer Graphics and Interactive Techniques*, 1984, pp. 207-212.

[7] A. Elgammal, et al. Non-parametric Model for Background Subtraction, *IEEE Int. Conf. on Computer Vision (ICCV), Frame-Rate Workshop*, 2000.

[8] S.Y. Elhabian, et al. Moving Object Detection in Spatial Domain using Background Removal Techniques - State-of-Art, *Recent Patents on Computer Science*, 1:32-54, 2008.

[9] Y. Hel-Or and H. Hel-Or. Real-time Pattern Matching Using Projection Kernels, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(9):1430-1445, 2005.

[10] I.T. Jolliffe, *Principal Component Analysis, 2nd ed.*, Springer, 2002.

[11] Y. Ke, et al. Efficient Visual Event Detection using Volumetric Features, *IEEE Intl. Conf. Computer Vision (ICCV'05)*, 2005, pp. 166-173.

[12] K. Kim, et al. Real-time Foreground-Background Segmentation Using Codebook Model, *Real-Time Imaging*, 11(3):172-185, 2005.

[13] L.J. Latecki, et al. Motion Detection Based on Local Variation of Spatiotemporal Texture, *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshop Object Tracking and Classification Beyond the Visible Spectrum*, 2004.

[14] L. Li, et al. Statistical Modeling of Complex Backgrounds for Foreground Object Detection, *IEEE Trans. on Image Processing*, 13 (11):1459-1472, 2004.

[15] A. Monnet, et al. Background Modeling and Subtraction of Dynamic Scenes, *IEEE Int. Conf. on Computer Vision (ICCV)*, 2003, pp. 1305-1312

[16] Y. Moshe and H. Hel-Or. Video Block Motion Estimation Based on Gray-Code-Kernels, *IEEE Trans. on Image Processing*, 18(10):2243-2254, 2009.

[17] N. Oliver, et al. A Bayesian Computer Vision System for Modeling Human Interactions, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):831-843, 2000.

[18] D.H. Parks and S.S. Fels. Evaluation of Background Subtraction Algorithms with Post-processing, *IEEE Int. Conf. on Advanced Video and Signal-based Surveillance (AVSS)*, 2008.

[19] M. Piccardi. Background Subtraction Techniques: A Review, *IEEE Int. Conf. on Systems, Man and Cybernetics*, 4:3099-3104, 2004.

[20] M. Seki, et al. Background Subtraction based on Cooccurrence of Image Variations, *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2003, pp. 65-72.

[21] Y. Sheikh and M. Shah. Bayesian Modeling of Dynamic Scenes for Object Detection, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(11):1778-1792, 2005.

[22] P. Simard, et al. Boxlets: A Fast Convolution Algorithm for Signal Processing and Neural Networks, *Advances in Neural Information Processing Systems*:571-577, 1999.

[23] C. Stauffer and W.E.L. Grimson. Adaptive Background Mixture Models for Real-Time Tracking, *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1999.

[24] K. Toyama, et al. Wallflower: Principles and Practice of Background Maintenance, *Int. Conf. on Computer Vision (ICCV)*, 1999, pp. 255-261.

[25] C.J. van-Rijsbergen, *Information Retireval*, Butterworths, 1979.

[26] P. Viola and M. Jones. Robust Real-Time Face Detection, *Intl. J. Computer Vision*, 57(2):137-154, 2004.

[27] D. Wang, et al. A Novel Probability Model for Background Maintenance and Subtraction, *Int. Conf. on Vision Interface*, 2002, pp. 109-117.

[28] J. Weng, et al. Candid Covariance-Free Incremental Principal Component Analysis, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(8):1034-1040, 2003.