Integrated Computer-Aided Engineering 12 (2005) 1–12 1
IOS Press

Reliability of motion features in surveillance videos

Longin Jan Latecki[a,*], Roland Miezianko[a] and Dragoljub Pokrajac[b]
[a]*CIS Department, Temple University, Philadelphia, PA 19122, USA*
E-mail: {latecki,rmiezian}@temple.edu
[b]*CIS Department and Applied Mathematics Research Center, Delaware State University, Dover, DE 19901, USA*
E-mail: dpokraja@desu.edu

Abstract. Although a tremendous effort has been made to perform a reliable analysis of images and videos in the past fifty years, the reality is that one cannot rely 100% on the analysis results. With exception of applications in controlled environments (e.g., machine vision application), one has to deal with an open world, which means that content of images may significantly change, and it seems impossible to predict all possible changes. Relying on content-based video analysis may lead to bogus results, since the observed changes may be consequence of unreliable features, and not necessarily of observed events of interest.
Our main strategy is to estimate the feature properties when the features are reliable computed, so that any set of features that does not have these properties is detected as being unreliable. This way we do not perform any direct content analysis, but instead perform unsupervised analysis of feature properties that are related to the reliability.
The solution pursuit in this paper is to monitor the reliability of the computed features using temporal changes and statistical properties of feature value distributions. Results on benchmark real-life videos demonstrate the capability of the proposed techniques to successfully eliminate problems due to change in light conditions, transition/compression artifacts and unwanted camera motions.

Keywords: Motion detection, feature reliability, real time performance evaluation, texture representation, incremental PCA

1. Introduction

The main effort in video analysis nowadays continues to be making the feature computation more reliable. However, it seems that we need to accept the fact that the computed features will never be 100% reliable. Namely, with exception of applications in controlled environments where closed world assumptions apply (such as in machine vision), the content of scene may significantly change from frame to frame, and it seems impossible to predict all possible changes. Recall that such changes may be result not of an observed phenomenon of interest (e.g., activity of people on the scene) but instead of spurious artifacts. For example, in the context of surveillance videos, the light conditions may suddenly fluctuate in parts of scene only, video compression or transmission artifacts may occur, a wind may cause a stationary camera to tremble, and so on. If we depend on video analysis to detect content changes of interest, such approach may fail since content changes due to the spurious artifacts can make such analysis unreliable. Instead of trying to distinguish between "desirable" and "parasite" changes, we propose to assess reliability of the features. A proposed system computes the feature reliability measures independent of the content and subsequently performs decisions only when features are labeled as sufficiently reliable. The guiding principle of our approach is that an intelligent system for video analysis not only should compute features but also perform instantaneous evaluation of features reliability, and adapt its behavior in accordance to the reliability. For example, if the goal of a system is to monitor motion activity and to sig-

*Corresponding author.

ISSN 1069-2509/05/$17.00 © 2005 – IOS Press and the author(s). All rights reserved

nal an alarm if the activity is high, the system may be allowed to make reliable decisions only if there exist a subset of the computed motion activity features that is sufficiently reliable. The monitoring of feature reliability and adjusting the system behavior accordingly, seems to be the only way to deploy autonomous video surveillance systems.

We begin our discussion by showing two examples of video content changes that cause the existing motion detection approaches to inaccurately detect the presence of substantial motion. Clearly, the detected motion is present in videos, but it is due to some content artifacts and not due to the actual presence of moving objects. Consequently, a human observant ignores such "motion" as irrelevant, while standard video analysis systems detect it as significant activity. We will show that the feature reliability methods proposed in this paper allow us to identify the unreliable motion features, and to ignore the irrelevant artifacts. This is possible without reducing the detection rate of real moving objects. Consequently, we eliminate false alarms without reducing the detection rate. We stress that this is obtained without any direct video content analysis (e.g., using different features), but by monitoring the reliability of computed features. Namely, direct video content analysis with further features does not solve the problem, since these features may also become unreliable.

Our first example illustrates motion artifacts in *Campus 3*[1] video introduced by some reflections in windows that are probably caused by cars passing by. In Fig. 1, we show two frames from *Campus 3* video, one showing real motion, and the second showing motion artifacts in addition to the normal motion. Our second example, in Fig. 2(a), shows motion artifacts introduced by video compression. The same scene without such artifacts is shown in Fig. 2(b). This video, which we call *Temple 2*, was recorded in real-world environment by the video surveillance system of the Campus Police Division of the Temple University.

In Section 2, we first describe a simple temporal method to determine the reliability of motion detection. In Section 3, we present a more sophisticated statistical method based on distribution analysis of feature values and information theory [21]. Both methods monitor features computed by our motion detection approach presented in [17], which we summarize in Section 4.

[1] *Campus 3* can be obtained from the Performance Evaluation of Tracking and Surveillance (PETS) repository: ftp://pets.rdg.ac.uk/PETS2002//DATASET1/TESTING/CAMERA3_JPEGS/.

Now we briefly summarize our motion detection approach.

The motion features are computed for gray level or infrared videos using 3D spatiotemporal blocks of spatial size 8×8 pixels, and temporal size of 3 frames. The blocks are disjoint in space and overlap by one frame in time. As result we obtain *motion measure* values for each 8×8 block in every video frame. The value of motion measure *mm*(*x*,*y*,*t*) at particular block (x, y) at time t describes the speed of texture change in this block. The motion measure *mm* is a function of the texture type and the speed of the moving object. Its definition is given in Section 4.3. By thresholding the motion measure values, we obtain a binary feature, called *motion detection*, with 1 standing for 'motion detected' and 0 for 'no motion detected'. Both videos as well as our motion detection results can be viewed on [12].

Based on *mm* we derive a global *motion amount* measure as sum of texture changes in a given frame t:

$$ma(t) = \sum_{x,y} mm(x, y, t)$$

In the following section we will deal with reliability assessment of the motion amount *ma*. Its reliability assessment is important, since system decisions on unusual activities (alarm situations) in videos can be based on *ma*. For example, a simple thresholding of *ma* can be used to identify time intervals with large amount of motion amount, which under certain assumptions, is a clear indicator of alarm situations. A direct, temporal reliability analysis of motion amount *ma* is introduced in Section 2. A statistical reliability analysis of motion amount *ma* is based on distribution analysis of motion measure *mm* is presented in Section 3.

2. Temporal analysis of feature reliability

In this section, we describe a simple temporal method to determine the reliability of motion amount *ma*. Our reliability assessment is based on the assumption that the speed of change function *ma* is bounded. Having detected the frames with the abrupt changes, these frames are excluded from decision making. This reliability property works under the assumption that there exists an upper bound on the size of moving objects whose motion we want to detect (measured in the number of moving blocks). This assumption holds for surveillance videos taken by a stationary camera positioned so that recorded moving objects cannot get to close to it (i.e., a roof mounted camera).

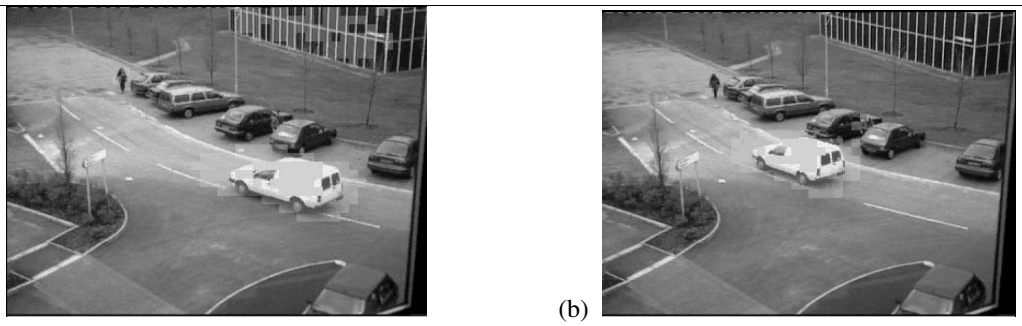


Fig. 1. Two frames from Campus 3 video with moving blocks highlighted red: (a) motion artifacts due to reflections in the windows, (b) the same scene (a few frames later) without the artifacts.

Fig. 2. Two frames from *Temple 2* video with moving blocks highlighted red: (a) motion artifacts introduced by video compression, (b) the same scene (a few frames later) without the artifacts.

We use the finite difference approximation of first derivative of *ma* to monitor the reliability of our motion detection. In simple words, if the jump in values of *ma* is above a certain threshold for a given time interval, the feature is unreliable in this interval. The threshold necessary to detect the unreliable features is not static. We propose a dynamic thresholding algorithm described in Section 2.1 to learn and vary this threshold. However, some other learning techniques could also be used.

Now we consider an example video, called *Temple 2*. It satisfies the assumption the speed of the change function is bounded. This video is recorded by a roof mounted, stationary camera, so that a certain minimal distance to moving objects is guaranteed. Typical moving objects there, humans and vehicles, cannot get arbitrarily large. Hence, the fraction of the scene occupied by a moving object is limited. Observe that the actual value of the upper bound on the size of moving objects needs not to be known, since our algorithm learns it automatically.

In Fig. 3(a), we see the graph of function *ma* for *Temple 2* video. Time intervals with significant jumps of *ma* that are correctly identified by our dynamic thresholding (applied to the derivative of *ma*) are marked with red lines in Fig. 3(b). The graph of modified feature *ma*, when *ma* was set to zero within the time intervals when motion detection was detected as unreliable is shown in Fig. 3(c). Figure 3(c) shows that the proposed method is able to identify and exclude the unreliable results of motion detection. By excluding these time intervals from further processing, we not only eliminate false alarms, but make possible to correctly detect alarm situations, although the input motion detection is not 100% reliable. For example, a significant increase in the number of motion blocks after the frame 1700 indicates an alarm situation. This is a correct prediction, since a street fight is recorded on the video after the frame 1700, see the *Temple 2* video [12].

2.1. Dynamic thresholding algorithm

Now we describe the dynamic thresholding algorithm used to detect the jumps of function *ma*. Since this algorithm can be applied to any real valued function we state it for a function f that maps discrete time units to real numbers. (In our case we either have $f = ma$ or $f = ma'$.) First we compute the initial values of mean *meanl* and standard deviation*stdl* using

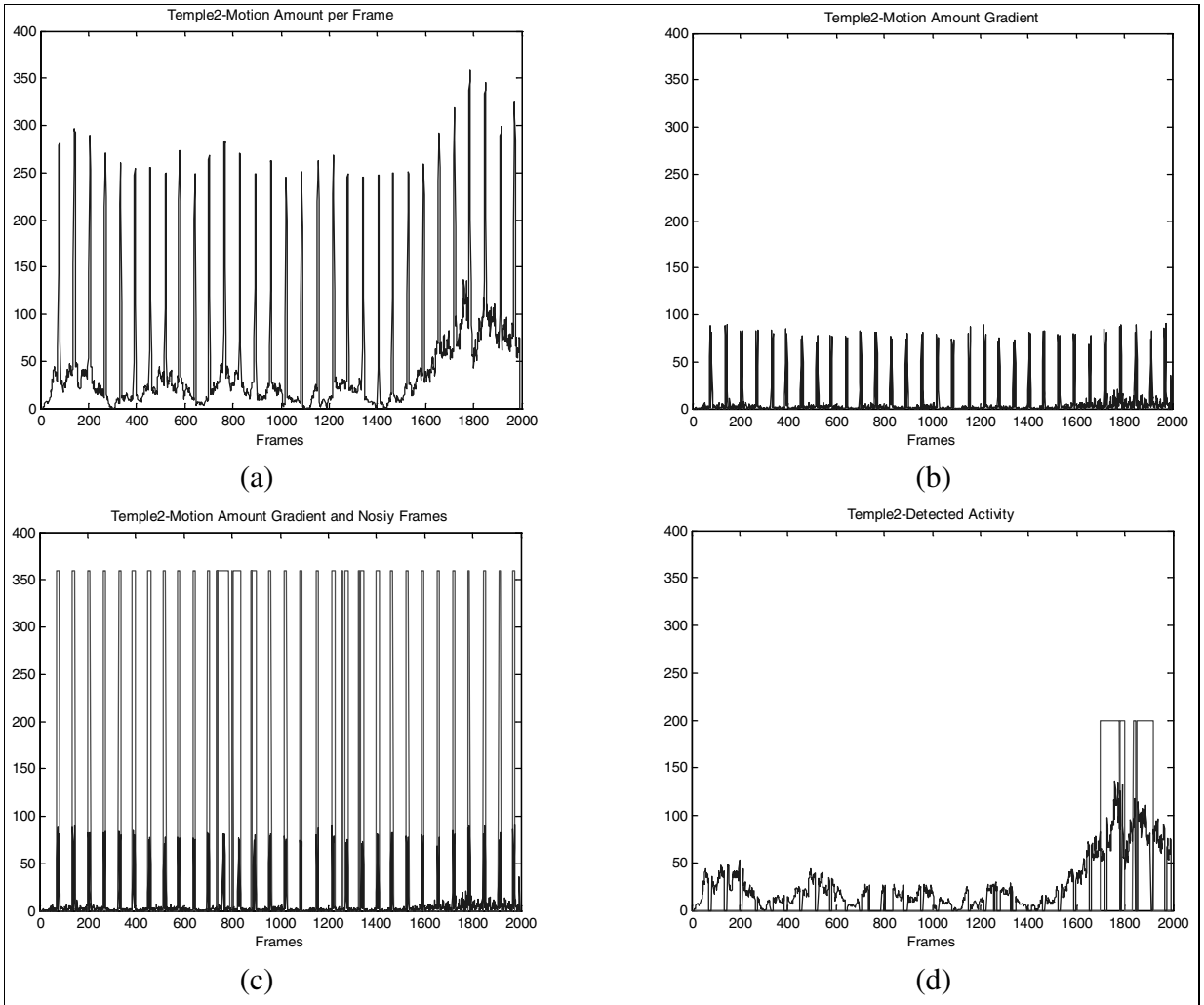


Fig. 3. (a) The graph of *ma*, which is the number of moving blocks as function of frame number t. (b) Derivative of *ma*. (c) Significant jumps of *ma* (caused by feature unreliability) correctly identified by our dynamic thresholding as applied to the derivative of *ma*. (d) The graph of *ma* padded by zeros for frames with unreliable motion detection showing increased activity detection.

all values of $f(x)$ for x =1, ..., *t*-1 and some time instance t. The actual dynamic thresholding starts at time $x = t + w$. A jump up is detected at points $x \in \{x - w, .., x - 1, x, x + 1, x + w\}$ for a window size w if

meanrw(x)– *meanl*(x)> $C1$•*stdl*(x),

where $C1$ is a constant and *meanrw* is defined below. A dynamic threshold values *meanl* and *stdl* are updated if

meanrw(x)– *meanl*(x) < $C2$• *stdl*(x),

where $C2 < C1$ is a second constant. The updated values are:

meanl(x) = u•*meanl*(x)+(1−u) • *meanrw*(x)

stdl(x) = u•*stdl* (x)+(1−u) *stdrw*(x)

where u is a learning constant (the larger u, the smaller is the influence of new values) and

$$meanrw(x) = \frac{1}{w}\sum_{\tau=1}^{w} f\,(x+\tau)$$

$$stdrw(x)=\sqrt{\frac{1}{w-1}\sum_{\tau=1}^{w}(f(x+\tau)-meanrw(x))^2}.$$

The symmetric window constant w was set to 3, giving us a sliding window of 7 frames ($2*w+1$). The learning constant was $u = 0.9$. Constants $C1$, $C2$ of function f used in detecting jumps of the *Temple 2* video were selected based on the initial running average *meanl* and *stdl*. The value of *meanl* was 10.3 and

the value of *stdl* was 7.4. Constant $C1$ was set to 15 and constant $C2$ was set to 3 providing the initial jump detected threshold to 154.5 and reset to no-jump detected threshold of 30.9.

3. Statistical analysis of feature reliability

In this section a reliability analysis of motion amount *ma* is based on distribution analysis of motion measure *mm*. We assume that a feature is more reliable, the more its distribution differs from a normal (Gaussian) distribution. This is in analogy to Information Theory stating that a feature bears more information if its distribution differs more significantly from a normal (Gaussian) distribution. Similar heuristics are used e.g., in Independent Components Analysis [20].

Our distribution assumption is valid for motion measure *mm* feature, under an additional assumption that there is at least one moving object in video. Clearly, if no object is moving in a video plane, the distribution of *mm* is Gaussian with mean being close to zero. A single object moving that has only one type of texture adds a second mode to the distribution of *mm*, since the motion measure has similar values for every block containing this object. The mean of the new mode is the speed of texture change measured by *mm*. If the moving object has more that one type of texture or if more than one object is moving, the distribution of *mm* is multimodal Gaussian; with each mode corresponding to the speed of change of each texture type (which is a function of the texture type and the speed of the moving object).

The follow-up assumption is that the feature becomes unreliable if some random noise is superimposed, which makes the distribution of such noisy feature to become more Gaussian like. The main idea here is that when we approximate the distribution of the noise with a single Gaussian (this distribution does not have to be Gaussian), its standard deviation (std) is very large in comparison to the distribution of a single moving object. Then the joint distribution of noise and background is significantly closer to a single Gaussian distribution (due to the large std) than the joint distribution of the moving object and the background.

Hence, by measuring to what extent a feature distribution differs from a Gaussian distribution, one can not only get information to what extent the feature is useful but also when such usefulness drops (e.g., due to some external and often non-observed factor). There may be many different sources of the random noise, for example, the above presented compression artifacts or reflections of moving objects in windows. The key argument in our approach is that we do not need to specify all possible situations that will make the *mm* distribution to become Gaussian. We only need to assure that the normal situations (in which we want the system to perform robustly) lead to a clear multimodal distribution.

The Information Theory proposes *negentropy* as the measure of this discrepancy. Given a probability density $p(x)$ of a feature, Differential Entropy is defined [18,19] as:

$$H(x) = \int_{-\infty}^{\infty} -p(x) \log p(x)\, dx \tag{1}$$

For a given class of distributions $p(x)$ that have the same variance σ^2, differential entropy is maximal for a Gaussian distribution where it is equal

$$H_{Gauss}\left(\sigma^2\right) = \frac{1}{2} \log 2\pi\ e\sigma^2. \tag{2}$$

Hence, a negentropy, which defined as

$$J(x) = H_{Gauss}\left(\sigma^2\right) - H(x) \tag{3}$$

or its normalized value

$$J(x) / H_{Gauss}\left(\sigma^2\right) = 1 - \frac{H(x)}{H_{Gauss}\left(\sigma^2\right)} \tag{4}$$

may be used to measure usefulness and reliability of a feature. Observe that the minimal value of negentropy is 0 (when $p(x)$ is Gaussian).

A naive approach to compute negentropy would be to employ histograms to approximate $p(x)$ with piecewise constant function $p'(x)$ such that:

$$p'(x) = Kp(x_i), x \in [x_i, x_i + \Delta x]$$

where K is a normalization constant (chosen such that $p'(x)$ is a distribution). However, as shown in [21] this non-parametric technique is very unstable since dependent on a proper choice of a histogram bin size Δx and histogram centers x_i. Hence we use parametric approach suggested in Hyvarinen's NIPS 1997 paper [18]. The main ideas of this approach are:

1) Instead of original feature x, use a standardized feature $x* = (x - mean(x))/std(x)$ that have zero mean and unit standard deviation. This way, we could directly use negentropy to compare reliability with no need to normalize with the entropy of a Gaussian.

2) Use a first-order Taylor approximation of a logarithmic function in Eq. (1) that leads to: $(1+\varepsilon)\log(1+\varepsilon) \approx \varepsilon + \varepsilon^2/2$;

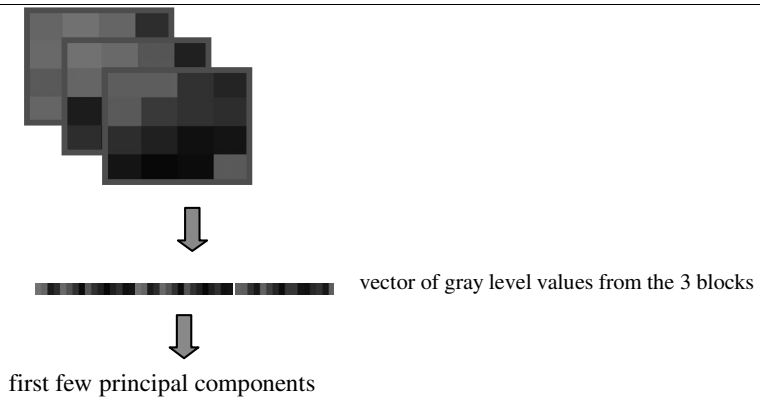


Fig. 5. Illustration of the computation process of texture vectors.

3) Use conveniently chosen set of orthogonal functions of $G_i(x)$ of a feature x to expand probability density function $p(x)$ in vicinity of a Gaussian probability density.

In practice, the choice of orthogonal functions is based on practicability and sensitivity on outliers of the computation of estimates for expectations $E(G_i(x))$, integrability of the obtained probability density function approximation and last, but not the least, the properties of non-Gaussian distributions we want to capture.

Based on such consideration [18], proposes the following two approximations of negentropy that we use in this paper:

$$J_a(x^*) = k_1 E\left(x^* \cdot e^{-x^{*2}/2}\right)^2 + k_{2a}\left(E(|x^*|) - \sqrt{\frac{2}{\pi}}\right)^2 \quad (5)$$

$$J_b(x^*) = k_1 E\left(x^* \cdot e^{-x^{*2}/2}\right)^2 + k_{2b}\left(E\left(e^{-x^{*2}/2}\right) - \sqrt{\frac{1}{2}}\right)^2 \quad (6)$$

where the coefficients are determined as:

$$k_1 = \frac{36}{8\sqrt{3}-9}, k_{2a} = \frac{24}{2-\frac{6}{\pi}}, \quad k_{2b} = \frac{24}{16\sqrt{3}-27}. \quad (7)$$

The proposed technique is applicable on any continuous feature. In this paper, we evaluate the reliability of the *motion activity* feature, defined in Section 4 (as the largest eigenvalue of texture vectors in a small time window). For each frame, we standardize the feature values $x*$, compute expectations $E(|x^*|)$, $E\left(x^* \cdot e^{-x^{*2}/2}\right)^2$ and finally compute the negentropy approximations Eqs (5), (6) per frame.

We evaluated the proposed techniques for assessing feature reliability on a set of videos [12]. This set includes infrared videos, for which the same settings of parameters as for visual light videos were used. Here we focus on our results on two video sequences from the Performance Evaluation of Tracking and Surveillance (PETS) repository: a sequence from PETS2001[2] here referred to as *Campus 1* sequence, a sequence from PETS2002, here referred to as the *Campus 3* sequence and on a *Temple 2* sequence from Temple University.

Campus 1. At the beginning of the sequence, there is no movement, so changes in the motion activity (an observed feature) are random, which reflects small negentropy values in approximately first 100 frames, see Fig. 4(a). Both negentropy approximations (Eqs (5), (6)) demonstrate strong drop between frames 1960 and 2000 which corresponds to the higher level of noise that can be visually observed between these frames. Function B (Eq. (5)) provides more stable approximation values, which makes it potentially more useful.

Campus 3. Both methods identified drop around frames 330, 660, a strong drop around 700, a drop around 720 and the relatively long-term drop between 800 and 900, see Fig. 4(b). Finally, there were some small oscillations between 1200 and 1300 and one drop around 1400. All these events correspond to frames in the video sequence when our algorithm has difficulties in properly identifying moving objects based on observed feature (e.g., due to reflections in the upper right part of the frame, cp. Fig. 1). Again Function B

[2] ftp://pets.rdg.ac.uk/PETS2001/DATASET1/TESTING/CAMERA1_JPEGS/.

Use first q values of vectors $\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(q)}$ to estimate initial values of eigenvectors and eigenvalues using standard eigenvalue decomposition of estimated covariance matrix, eq. (4.1a)
For each new vector $\mathbf{x}^{(m)}$:
Compute matrix $\mathbf{A}$ using eq. (4.9);
Compute matrix $\mathbf{B}$, eq. (4.10);
Perform eigenvalue decomposition of $\mathbf{B}$, eq. (4.11) and find p most significant eigenvalues;
Use eq. (4.13) to compute p corresponding eigenvectors;
Normalize vectors $\mathbf{u}_i^{(m)}$ to satisfy eq. (4.14).

Fig. 6. Incremental PCA algorithm.

(Eq. (5)) performed better, by having less oscillations and fluctuations.

Temple 2. On this video, there is evident instability (manifested as flicker) that can be traced to applied compression technique. The period of this disturbance, which has negative effects on motion detection, is around 62 frames. Using the proposed technique, we obtained negentropy values that reflect this periodicity. Both functions Eqs (5) and (6) have strong periodical components, see Fig. 4(c), and demonstrate oscillations which period can be correctly determined using a Fast Fourier Transform [22], as approximately 62 frames. Function Eq. (6) is again more stable and provides better automatic period estimation. The results of the statistical method agree with those of the temporal methods, cp. Fig. 3.

A common denominator of the results shown is that the proposed negentropy-based technique can help in determining frames when the observed feature is unreliable (periodic or pulse flicker, noise, etc.). Since both Eqs (5) and (6) are only relatively rough approximations of negentropy, there is no wonder they do not provide the same values, especially when a negentropy is relatively high. As expected, when a negentropy is low, the feature probability distribution is closer to a Gaussian so both approximations would give similar results. Generally, Eq. (6) provides better performance. It is more stable and has fewer fluctuations. Hence is potentially more suitable for automatic thresholding.

4. Feature generation and motion detection

We shortly describe our motion detection method proposed in [17]. It is based on change analysis of texture vectors computed for 3D, spatiotemporal (sp) blocks. In our previous paper [11] we have shown that the use of sp texture vectors of 3D blocks in the framework of Stauffer and Grimson [14] can improve the detection of moving objects while potentially cutting back the processing time due to the reduction of the number of input vectors per frame. Our experimental results in [17] (videos be viewed on [12]) show that the motion detection technique used here leads to further performance improvements.

The outline of the algorithm is as follows. For each fixed block position (x, y) in a video frame we perform computation of motion measure *mm* independent from other block video positions. We first compute spatiotemporal (sp) texture vectors for each block (x, y, t) using principal component analysis (PCA). We describe static PCA computation in Section 4.1, which is used for initialization for each block position (x, y). For actual, real time computation of sp vectors, we use incremental PCA described in Section 4.2. Finally, in Section 4.3 we describe our method to measure the speed of change of sp vectors and define the motion measure *mm*(*x*,*y*,*t*).

4.1. Video representation with spatiotemporal (sp) texture vectors

We represent videos as three-dimensional (3D) arrays of gray level or monochromatic infrared pixel values $g_{i,j,t}$ at a time instant t and a pixel location i, j. A video is characterized by temporal dimension Z corresponding to the number of frames, and by two spatial dimensions, characterizing number of pixels in horizontal and vertical direction of each frame.

We divide each image in a video sequence into disjoint $N_{BLOCK} \times N_{BLOCK}$ squares (e.g., 8×8 squares) that cover the whole image. Spatiotemporal (3D) blocks are obtained by combining squares in consecutive frames at the same video plane location. In our experiments, we used $8 \times 8 \times 3$ blocks that are disjoint in space but overlap in time (see Fig. 5), i.e., two blocks at the same spatial location at times t and $t + 1$ have one square in common.

The fact that the 3D blocks overlap in time allows us to perform successful motion detection in videos with very low time frequency, e.g., in our experimental results [12] videos with 2 fps (frames per second) are included. The obtained 3D blocks are represented as 192-dimensional vectors of gray level or monochromatic

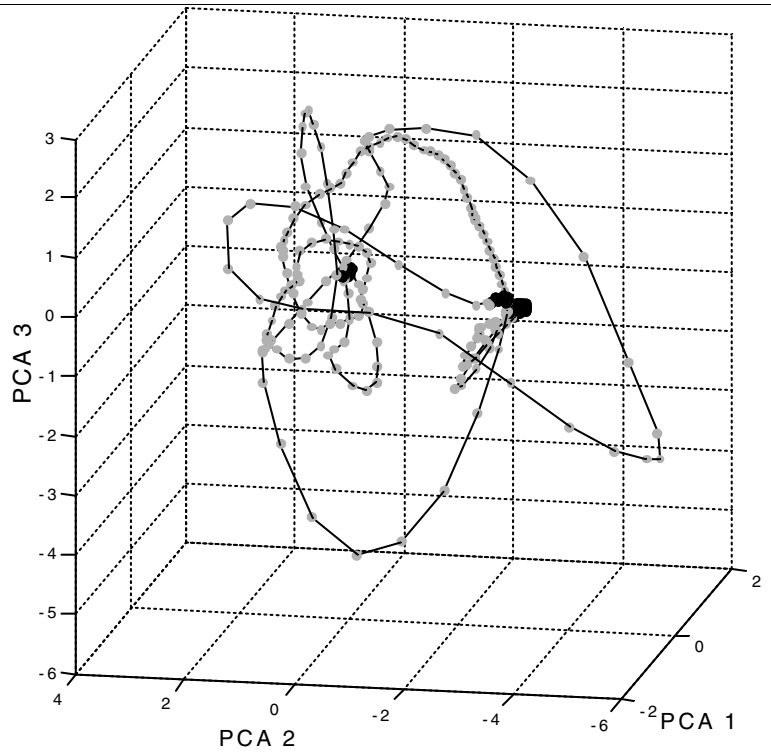


Fig. 7. Coordinates of 3-dimensional location vectors for block (24,28) of the *Campus* 1 video, Blue-gray dots correspond to the frames where the block was moving.

infrared pixel values. We then zero mean these vectors (the mean is computed over the gray values in each $8 \times 8 \times 3$ block) and project them to a lower dimensional subspace using principal component analysis (PCA), see Fig. 5. The obtained k-dimensional vectors form a compact spatiotemporal texture representation for each block. In our experiment the dimension k was between 3 and 10. The PCA projection matrices are computed separately for all video plane locations (a set of disjoint 8×8 squares in our experiments), i.e., at each block position over time.

The blocks are represented by N-dimensional vectors $\mathbf{b}_{I,J,t}$, specified by spatial indexes (I, J) and time instant t. Vectors $\mathbf{b}_{I,J,t}$ contain all values $g_{i,j,t}$ of pixels in the corresponding 3D block.

To reduce dimensionality of $\mathbf{b}_{I,J,t}$ while preserving information to the maximal possible extent, we compute a projection of the normalized block vector to a vector of a significantly lower length K<<N using a PCA projection matrix $\mathbf{P}^K_{I,J}$ computed for all $\mathbf{b}_{I,J,t}$ at video plane location (I, J). The resulting sp texture vectors $\mathbf{b}^*_{I,J,t} = \mathbf{P}^K_{I,J} \bullet \mathbf{b}_{I,J,t}$ provide a joint representation of texture and motion patterns in videos and are used as input of algorithms for detection of moving objects. We used $K = 3$ in our experiments.

To compute $\mathbf{P}^K_{I,J}$ we employ the principal values decomposition following [4,5]. A matrix of all normalized block vectors $\mathbf{b}_{I,J,t}$ at video plane location (I,J) is used to compute the N×N dimensional covariance matrix $\mathbf{S}_{I,J}$. The PCA projection matrix $\mathbf{P}_{I,J}$ for spatial location (I,J) is computed from the $\mathbf{S}_{I,J}$ covariance matrix. The projection matrix $\mathbf{P}_{I,J}$ of size N × N represents N principal components. By taking only the principal components that corresponds to the K largest eigenvalues, we obtain $\mathbf{P}^K_{I,J}$.

4.2. Incremental computation of the spatiotemporal (sp) texture vectors

The method described in Section 4.1 computes the sp vectors in a static manner. Therefore, we only use it in an initialization phase, e.g., for the first 100 frames. To allow real time computation, we use incremental estimation of PCA projection matrix described in this section.

Three different kinds of incremental PCA algorithms have been introduced in [23,27], and [28–32]. In [27] an incremental PCA algorithm that does not explicitly compute the covariance matrix is proposed. The algo-

rithm in [28] computes PCA by successive orthogonalization. It gradually increases the number of eigenvectors/eigenvalues when the current number of eigenvectors cannot satisfactorily approximate the observation vector.

The algorithm for incremental estimation of PCA coefficient we use [23] is based on well-known [24,25] algorithms for incremental estimation of parameters of multivariate Gaussian distribution and the approximation of the covariance matrix by a few most-significant eigenvalues and corresponding eigenvectors.

Consider the n-dimensional observation row vectors $\mathbf{x}^{(1)}$,$\mathbf{x}^{(2)}$,. . . ,$\mathbf{x}^{(m-1)}$,$\mathbf{x}^{(m)}$,$\mathbf{x}^{(m+1)}$,. . .

For m consecutive vectors, the mean $\mu^{(m)}$ and standard deviation $\Sigma^{(m)}$ can be estimated [4] as:

$$\mu^{(m)} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}^{(i)} \tag{8}$$

$$\Sigma^{(m)} = \frac{1}{m-1}\sum_{i=1}^{m}(\mathbf{x}^{(i)} - \mu^{(m)})^T (\mathbf{x}^{(i)} - \mu^{(m)}) \tag{9}$$

The exponentially weighted moving average approximations of Eq. (8) [23–26] update the estimates of the mean and the standard deviation in the previous instant m-1 using the current value $\mathbf{x}^{(m)}$of the observation vector as:

$$\mu^{(m)} \approx \alpha\mu^{(m-1)} + (1-\alpha)\,\mathbf{x}^{(m)}, \tag{10}$$

$$\Sigma^{(m)} \approx \alpha\Sigma^{(m-1)} + (1-\alpha)\left(\mathbf{x}^{(m)} - \mu^{(m-1)}\right)^T \left(\mathbf{x}^{(m)} - \mu^{(m-1)}\right). \tag{11}$$

Here, α is a learning rate ranging from 0 (corresponding to the estimate based *only* on the current value of the observation vector and not on historical values) to 1 (corresponding to the static estimation).

It is known [5] that $n \times n$ covariance matrix $\Sigma^{(m)}$ can be decomposed through its eigenvectors and eigenvalues such that:

$$\Sigma^{(m)} = \mathbf{U}^{(m)}\Lambda^{(m)}\mathbf{U}^{(m)^T}, \tag{12}$$

where $\mathbf{U}^{(m)}$ contains the*column* eigenvectors and $\Lambda^{(m)}$ the corresponding eigenvalues, such that

$$\mathbf{U}^{(m)} = [\mathbf{u}_1^{(m)}, \ldots, \mathbf{u}_n^{(m)}]$$
$$\Lambda^{(m)} = diag(\lambda_1^{(m)}, \ldots, \lambda_n^{(m)}). \tag{13}$$

Assume that we approximate $\Sigma^{(m)}$ by p the most significant (largest by magnitude) eigenvalues $\Lambda_p^{(m)} = diag(\lambda_1^{(m)}, \ldots, \lambda_p^{(m)})$ and the corresponding orthonormal eigenvectors $\mathbf{U}_{np}^{(m)}$ =[$\mathbf{u}_1$,. . . ,$\mathbf{u}_p$]. Similar to Eq. (3) we can write:

$$\Sigma^{(m-1)} \approx \mathbf{U}_{np}^{(m-1)}\Lambda_p^{(m-1)}\mathbf{U}_{np}^{(m-1)^T} \tag{14}$$

and hence formula(2b) becomes

$$\Sigma^{(m)} \approx \alpha\mathbf{U}_{np}^{(m-1)}\Lambda_p^{(m-1)}\mathbf{U}_{np}^{(m-1)^T} + (1-\alpha)\left(\mathbf{x}^{(m)} - \mu^{(m-1)}\right)^T \left(\mathbf{x}^{(m)} - \mu^{(m-1)}\right). \tag{15}$$

The new values for eigenvalues $\lambda_i^{(\mathrm{m})}$ and eigenvectors $u_i^{(\mathrm{m})}$ could be obtained by eigenvalue decomposition of n*n matrix $\Sigma^{(m)}$, such that:

$$\Sigma^{(m)}\mathbf{u}_i^{(m)} = \lambda_i^{(m)}\mathbf{u}_i^{(m)}, i = 1, \ldots, n. \tag{16}$$

Observe that in Eq. (15) we need only first p the most significant eigenvalues and eigenvectors. Also, it is of interest to reduce the computational effort by performing eigenvalue decomposition of a smaller matrix instead of $n \times n$ matrix $\Sigma^{(m)}$. In order to do this, first denote that we can express formula Eq, (14) as:

$$\Sigma^{(m)} \approx \mathbf{A}\mathbf{A}^T \tag{17}$$

where

$$A = [\mathbf{y}_1, \ldots, \mathbf{y}_{p+1}]$$
$$\mathbf{y}_i = \begin{cases} \sqrt{\alpha\lambda_i^{(m)}}_i\mathbf{u}_i^{(m)}, & i = 1, \ldots, p \\ \sqrt{1-\alpha}\left(\mathbf{x}^{(m)} - \mu^{(m-1)}\right), & i = p+1 \end{cases} \tag{18}$$

Let's define a (p+1) × (p+1) matrix $\mathbf{B}$ as

$$\mathbf{B} = \mathbf{A}^T\mathbf{A}. \tag{19}$$

Eigenvalues $\lambda_i^{*(m)}$ and eigenvectors $\mathbf{u}_i^{*(m)}$ of $\mathbf{B}$ satisfy the equations:

$$\mathbf{B}\mathbf{u}_i^{*(m)} = \lambda_i^{*(m)}\mathbf{u}_i^{*(m)}, i = 1, \ldots, p+1. \tag{20}$$

If we multiply both sides of Eq. (11) by the matrix $\mathbf{A}$ from the left side, we obtain:

$$\mathbf{A}\mathbf{B}\mathbf{u}_i^{*(m)} = \mathbf{A}\mathbf{A}^T A\mathbf{u}_i^{*(m)} = \lambda_i^{*(m)} \mathbf{A}\mathbf{u}_i^{*(m)}, i = 1, \ldots, p+1. \tag{21}$$

Using Eq. (8), by comparing Eq. (12) with Eq. (16) we can easily see that:

$$\lambda_i^{(m)} \approx \lambda_i^{*(m)}$$
$$\mathbf{u}_i^{(m)} \approx \mathbf{A}\mathbf{u}_i^{*(m)} \tag{22}$$

Hence, the estimates of the new eigenvalues and eigenvectors of the covariance matrix can be obtained

by transforming results of the eigenvalue decomposition of the matrix **B**.

However, it is important to observe that eigenvectors $u_i^{*(m)}$ (as a result of eigenvalue decomposition of **B**) are orthonormal, while the vectors $u_i^{(m)}$ from Eq. (22) *are not*. To use these vectors in the next iteration of Eqs (16)–(18), we need to normalize such that

$$\left|\mathbf{u}_i^{(m)}\right| = 1. \tag{23}$$

To initialize algorithm for incremental estimation of PCA, we use Eq. (8) to estimate the covariance matrix $\Sigma^{(q)}$ and the mean vector $\mu^{(q)}$ from first q observation vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)}$. Then, we compute the initial estimates for $\Lambda_p^{(q)}$ and $\mathbf{U}_{np}^{(q)}$ by performing standard eigenvalue decomposition (analog to Eq. (16)) of $\Sigma^{(q)}$ and taking p most significant eigenvalues and eigenvectors. The complete algorithm is summarized in Fig. 6.

4.3. Moving objects detection based on local variation

The speed and characteristics of texture change over time indicates whether the corresponding object texture is stationary or moving. Recall that each sp vector represents texture of the corresponding block. Hence, by observing the characteristics of sp vectors change over time, we are able to detect whether a particular block belongs to a moving object or to a background. Consider a single block position in a video plane. We can observe the trajectory of its sp vectors, i.e., the loci of sp vectors in successive time frames. If during an observed time interval, there is no moving object in the block, the sp vectors will be close to each other. Hence the variance of sp vectors during the time interval will be small. In contrast, if there is a moving object passing through this block, the sp texture vectors will change fast, i.e., the sp vectors will be spread in the space of their values. Therefore, the variance of sp vectors within an observation time window will be fairly large. In Fig. 7, we show the trajectory of sp vectors corresponding to block location (24,28) in *Campus 1* video. To make this visualization possible, we use only first three PCA components for each sp vector. It can be observed that frames when only stationary objects are visible in this block correspond to regions where sp vectors are clustered into fairly spherical shapes (black dots) with small spread. In contrary, when moving objects are moving passing through this block location, the trajectory of sp vectors (blue-gray dots) is typically elongated and the variance is relatively large.

A simple way to determine the speed of sp vector change would be to compute the norms of their first derivatives. However, computing finite differences of consecutive sp vectors is unreliable. In order to determine whether the consecutive vectors belong to elongated trajectories, we need to observe whether they are making a consistent progress in a certain time interval. We propose to assess the sp vector spread in the direction of maximal variance. To measure the variance of sp vectors, we compute the covariance matrix of sp vectors corresponding to the same block location for a pre-specified number of consecutive frames. We use the maximal eigenvalue as the measure of trajectory elongation.

More formally, for each location (x,y), and temporal instant t, we consider vectors

$$b^*_{x,y,t-W}, b^*_{x,y,t-W+1}, \ldots, b^*_{x,y,t}, \ldots, b^*_{x,y,t+W}$$

corresponding to a symmetric window of size $2W+1$ around the instant t. For these vectors, we compute the covariance matrix $\mathbf{C}_{x,y,t}$. We assign the largest eigenvalue of $\mathbf{C}_{x,y,t}$, denoted as $\Lambda_{x,y,t}$, to a given spatiotemporal video position to define a local variance measure, which we will also refer to as *motion measure*

$$mm(x, y, t) = \Lambda_{x,y,t}.$$

The larger the motion measure *mm*(*x,y,t*), the more likely is the presence of a moving object at position (x, y, t). We can then label each video position as moving or stationary (background) depending whether the motion measure is larger or smaller than a suitably defined threshold. We use a dynamic thresholding algorithm (described in Section 2) to determine the threshold value at position (x, y, t) based on the history of *mm*(*x,y,s*) values over time ($s = 1, \ldots, t-1$).

5. Conclusions

In this paper, we proposed and evaluated two methods to monitor the reliability of features applied in video surveillance and motion detection. The methods have been evaluated on real-life surveillance videos. Both methods correctly identified time intervals when an observed feature becomes non useful for motion detection (e.g., due to flicker, artifacts introduced by compression algorithm, etc.). The proposed methodology is potentially applicable to other domains where unsupervised learning is performed under open-world assumption (where we cannot anticipate all the events which could occur during the operational life of an automated intelligent system).

Acknowledgements

D. Pokrajac has been partially supported by NIH-funded Delaware Biomedical Research Infrastructure Network (BRIN) Grant (P20 RR16472), and DoD HBCU/MI Infrastructure Support Program (45395-MA-ISP Department of Army).

References

[1] D. Buttler, S. Sridharan and V.M. Bove, *Real-time adaptive background segmentation*, in Proc. IEEE Int. Conf. on Multimedia and Expo (ICME), Baltimore 2003.

[2] R.T. Collins, A.J. Lipton and T. Kanade, Introduction to the Special Section on Video Surveillance, *IEEE PAMI* **22**(8) (2000), 745–746.

[3] J.L. Devore, *Probability and Statistics for Engineering and the Sciences*, 5th edn., Int. Thomson Publishing Company, Belmont, 2000.

[4] R. Duda, P. Hart and D. Stork, *Pattern Classification*, 2nd edn., John Wiley & Sons, 2001.

[5] B. Flury, *A First Course in Multivariate Statistics*, Springer Verlag, 1997.

[6] I. Haritaoglu, D. Harwood and L. Davis, W4: Real-Time Surveillance of People and Their Activities, *IEEE PAMI* **22**(8) (2000), 809–830.

[7] R. Jain, D. Militzer and H. Nagel, *Separating nonstationary from stationary scene components in a sequence of real world TV images*, In Proc. IJCAI, 612–618, Cambridge, MA, 1977.

[8] I.T. Jolliffe, *Principal Component Analysis*, 2nd edn., Springer Verlag, 2002.

[9] O. Javed, K. Shafique and M.A. Shah, *Hierarchical approach to robust background subtraction using color and gradient information*, in Proc. IEEE Workshop on Motion and Video Computing (MOTION), Orlando, 2002, 22–27.

[10] N.M. Oliver, B. Rosario and A.P. Pentland, A Bayesian Computer Vision System for Modeling Human Interactions, *IEEE PAMI* **22**(8) (2000), 831–843.

[11] D. Pokrajac and L.J. Latecki, Spatiotemporal Blocks-Based Moving Objects Identification and Tracking, *IEEE Visual Surveillance and Performance Evaluation of Tracking and Surveillance* (*VS-PETS*) (October, 2003).

[12] R. Miezianko, L.J. Latecki and D. Pokrajac, Link to test results. http://knight.cis.temple.edu/∼video/VA.

[13] P. Remagnino, G.A. Jones, N. Paragios and C.S. Regazzoni, eds, *Video-Based Surveillance Systems*, Kluwer Academic Publishers, 2002.

[14] C. Stauffer and W.E.L. Grimson, Learning patterns of activity using real-time tracking, *IEEE PAMI* **22**(8) (2000), 747–757.

[15] R. Westwater and B. Furht, *Real-Time Video Compression: Techniques and Algorithms*, Kluwer Academic Publishers, 1997.

[16] C. Wren, A. Azarbayejani, T. Darrell and A.P. Pentland, Pfinder: Real-time Tracking of the Human Body, *IEEE PAMI* **19**(7) (1997), 780–785.

[17] L.J. Latecki, R. Miezianko and D. Pokrajac, *Motion Detection Based on Local Variation of Spatiotemporal Texture*, CVPR Workshop on Object Tracking and Classification Beyond the Visible Spectrum (OTCBVS), Washington, July 2004.

[18] A. Hyvärinen, *New approximations of differential entropy for independent component analysis and projection pursuit*, in Advances in Neural Information Processing Systems, (Vol. 10), MIT Press, 1998, 273–279.

[19] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley& Sons, 1991.

[20] A. Hyvärinen, J. Karhunen and E. Oja, *Independent Component Analysis*, Wiley, 2001.

[21] D. Pokrajac and L.J. Latecki, *Entropy-Based Approach for Detecting Feature Reliability*, Invited Paper, 48th Conf. for Electronics, Telecommunications, Computers, Automation, and Nuclear Engineering (ETRAN). Cacak, Serbia, June, 2004.

[22] E. Oran Brigham, *The Fast Fourier Transform: An Introduction to Its Theory and Application*, Prentice Hall, 1973.

[23] Y. Li, On Incremental and Robust Subspace Learning, *Pattern Recognition* **37** (2004), 1509–1518.

[24] S. McKenna, Y. Raja and S. Gong, *Object Tracking Using Adaptive Color Mixture Models*, Proceedings of the Third Asian Conference on Computer Vision-Volume I, 1998, 615–622.

[25] L. Ljung and T. Söderström, *Theory and Practice of Recursive Idenfication*, MIT Press, 1983.

[26] G. Box, G. Jenkins and G. Reinsel, *Time Series Analysis*, Prentice-Hall Inc, Englewood Hills, New Jersey, 1994.

[27] Juyang Weng, Yilu Zhang and Wey-Shiuan Hwang, Candid covariance-free incremental principal component analysis, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **25**(8) (2003), 1034–1040.

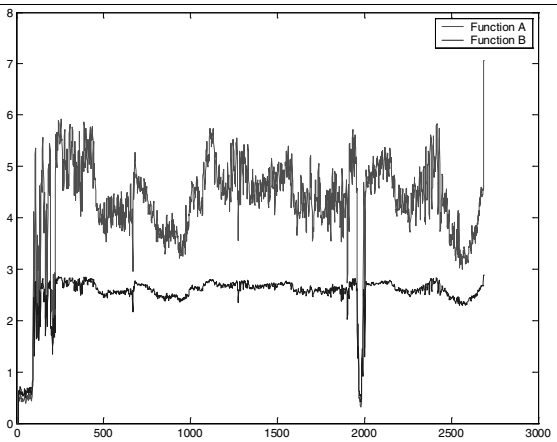
[28] P.M. Hall, D.R. Marshall and R.R. Martin, Incremental eigenanalysis for classification, *British Machine Vision Conference* **1** (1998), 286–295.

[29] R. Freitas, J. Santos-Victor, M. Sarcinelli-Filho and T. Bastos-Filho, *Performance Evaluation of Incremental Eigenspace Models for Mobile Robot Localization*, Proc. ICAR 2003 (11th Int. Conf. on Advanced Robotics), Coimbra, Portugal, 2003

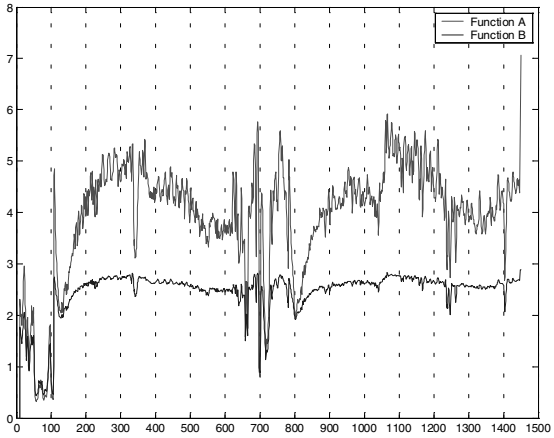
[30] Artac, Matej and Jogan, Matjaz and Leonardis, Ales, Incremental PCA for on-line visual learning and recognition, in: *Proceedings 16th International Conference on Pattern Recognition 3*, R. Kasturi, D. Laurendeau and C. Suen, eds, Quebec City, QC, Canada, 2002, pp. 781–784.

[31] D. Skocaj amd A. Leonardis, Incremental approach to robust learning of eigenspaces, in: *26th Workshop of the Austrian Association for Pattern Recognition* (*AGM/AAPR*), F. Leberl and F. Fraundorfer, eds, Graz (Austria), 2002.

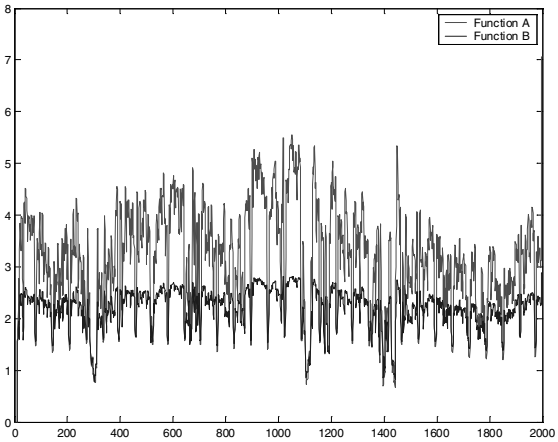
[32] Seiichi Ozawa, Shaoning Pang and Nikola Kasabov, *A Modified Incremental Principal Component Analysis for On-Line Learning of Feature Space and Classifier*, Proc. PRICAI 2004 (8th Pacific Rim Int. Conf. on Artificial Intelligence), Auckland, New Zealand, 2004, pp. 231–240.



(a)



(b)



(c)

Fig. 4. Estimated negentropy using Function A Eq. (5) and Function B Eq. (5) for (a) *Campus 1*; (b) *Campus 3*; (c) *Temple 2* videos. The negentropy is computed per frame of each video.