Pattern Recognition Letters 34 (2013) 2126–2134

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Performance comparison of feature extraction algorithms for target detection and classification ☆

Soheil Bahrampour [a], Asok Ray [b,*], Soumalya Sarkar [b], Thyagaraju Damarla [c], Nasser M. Nasrabadi [c]

[a] Department of Electrical Engineering, Pennsylvania State University, University Park, PA 16802-1412, USA
[b] Department of Mechanical Engineering, Pennsylvania State University, University Park, PA 16802-1412, USA
[c] Army Research Laboratory, Adelphi, MD 20783-1179, USA

ARTICLE INFO

Article history:
Received 19 April 2013
Available online 4 July 2013

Communicated by D. Dembele

Keywords:
Feature extraction
Pattern classification
Unattended ground sensors: border control

ABSTRACT

This paper addresses the problem of target detection and classification, where the performance is often limited due to high rates of false alarm and classification error, possibly because of inadequacies in the underlying algorithms of feature extraction from sensory data and subsequent pattern classification. In this paper, a recently reported feature extraction algorithm, symbolic dynamic filtering (SDF), is investigated for target detection and classification by using unmanned ground sensors (UGS). In SDF, sensor time series data are first symbolized to construct probabilistic finite state automata (PFSA) that, in turn, generate low-dimensional feature vectors. In this paper, the performance of SDF is compared with that of two commonly used feature extractors, namely Cepstrum and principal component analysis (PCA), for target detection and classification. Three different pattern classifiers have been employed to compare the performance of the three feature extractors for target detection and human/animal classification by UGS systems based on two sets of field data that consist of passive infrared (PIR) and seismic sensors. The results show consistently superior performance of SDF-based feature extraction over Cepstrum-based and PCA-based feature extraction in terms of successful detection, false alarm, and misclassification rates.

1. Introduction

Several feature extraction methods have been proposed to generate patterns from time series data for classification purposes. The well-known kurtosis method (Altmann, 2004) provides a statistical measure of the amplitude of the time series. In another method (Tian and Qi, 2002), a feature vector is constructed using the spectrum, where the power spectral density and the wavelet coefficients are used along with principal component analysis for feature extraction; similar time–frequency domain methods have been proposed by other researchers Li et al. (2002) and Succi et al. (2001). Recently, feature extraction from time series of robot motion behavior has been proposed by Mallapragada et al. (2012), based on symbolic dynamic filtering (SDF) (Ray, 2004; Rajagopalan and Ray, 2006), where the time series data were preprocessed by Hilbert transform that requires conversion of the real-valued signal into complex-valued analytic signal to extract the phase information (Subbu and Ray, 2008). In the SDF-based feature extraction, the time series data are first converted into symbol sequences, and then probabilistic finite-state automata (PFSA) are constructed from these symbol sequences to compress the pertinent information into low-dimensional statistical patterns. More recently, SDF-based feature extraction from (wavelet-transformed) time series has been proposed by Jin et al. (2012) for target detection and classification in border regions. The rationale for using wavelet-based methods is time–frequency localization and denoising of the underlying sensor time series. However, this method requires selection and tuning of several parameters (e.g., wavelet basis function and scales) for signal pre-processing in addition to the size of the symbol alphabet that is needed for SDF.

Feature extraction based on the concept of Cepstrum (Childers et al., 1977) has been reported as a conceptually simple and computationally efficient tool for use in pattern classification (Nguyen et al., 2011). The objective of the current paper is to make a comparative evaluation of Cepstrum-based, PCA-based, and SDF-based feature extraction for target detection and classification in the border regions using unattended ground sensor (UGS) systems. However, unlike the previous work of Jin et al. (2012) and Mallapragada

☆ This work has been supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under Grant No. W911NF-07-1-0376, and by the U.S. Air Force Office of Scientific Research (AFOSR) under Grant No. FA9550-12-1-0270. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

* Corresponding author.
E-mail addresses: soheil@psu.edu (S. Bahrampour), axr2@psu.edu (A. Ray), svs5464@psu.edu (S. Sarkar), thyagaraju.damarla.civ@mail.mil (T. Damarla), nasser.m.nasrabadi.civ@mail.mil (N.M. Nasrabadi).

0167-8655/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2013.06.021

et al. (2012), this paper has used direct partitioning and symbolization on time series data for feature extraction without performing the wavelet transform or Hilbert transform for signal pre-processing. The rationale is to make SDF comparable to Cepstrum and PCA for feature extraction, which do not usually preprocess the time series. In this case (i.e., without wavelet-transform or Hilbert-transform pre-processing of the signal), the SDF-based feature extraction method has only a single parameter to tune, which is the (finite) cardinality of the symbol alphabet; this is computationally more efficient than if wavelet or Hilbert transformation is used. For comparison of the Cepstrum-based, PCA-based, and SDF-based feature extraction, the performance of the respective feature extractors has been evaluated on the same sets of field data in conjunction with the sparse representation (Wright et al., 2008), support vector machines (SVM) (Bishop, 2006), and k-nearest neighbor (k-NN) (Bishop, 2006) as the pattern classifiers. These three pattern classifiers have been employed to compare the performance of the feature extractors for target detection and human/animal classification by unattended ground sensor (UGS) systems based on two sets of field data that consist of passive infrared (PIR) and seismic sensors as explained below.

Tools of target detection and classification by unattended ground sensor (UGS) systems have been extensively used to monitor human activities for border security; typical examples of UGS are seismic and Passive Infrared (PIR) sensors. Seismic sensors are suitable for long-range target detection, because they are relatively less dependent on exogenous disturbances (e.g., Doppler effects), as compared to acoustic sensors (Tian and Qi, 2002) that are prone to contamination by environmental noise. In contrast, PIR sensors are well-suited for motion detection and has the advantage of low power consumption (Zhang et al., 2007). Nevertheless discriminating human footstep signals from other target types and noise sources is a challenging problem, because of the rapidly decreasing trend of signal to noise ratio (SNR) as the distance between the sensor and the moving target increases. Furthermore, the footstep signals have dissimilar signatures for different environments and persons, which make the problem of target detection and classification even more challenging (Gramann et al., 1999).

In contrast to the Cepstrum-based and PCA-based feature extraction, the advantage of the SDF-based algorithm is that it takes into account the local information of the signal and it is capable of mitigating noise in the data even if the signal is not pre-processed for denoising. It is shown in a later section of the paper that the SDF-based feature extraction yields superior successful detection, false alarm, and overall correct classification rates compared to Cepstrum-based and PCA-based feature extraction.

This paper is organized into five main sections including the present section. Section 2 briefly describes the algorithms of Cepstrum-based, PCA-based, and SDF-based feature extraction. Section 3 succinctly discusses the three pattern classification algorithms that have been used in this paper. Section 4 presents the results of target detection and classification based on two sets of field data of passive infrared (PIR) and seismic sensors. Section 5 summarizes and concludes this paper with recommendations for future research.

2. Feature extractors

Feature extraction from sensor signals is an important step in target detection and classification which is accomplished in this paper by three alternative algorithms, namely Cepstrum (Nguyen et al., 2011), PCA (Bishop, 2006), and SDF (Ray, 2004). While the details of these algorithms have been reported in earlier publications, this section briefly reviews the underlying concepts of feature extraction from sensor time series for completeness of this paper and lists the algorithms that have been used in this paper for feature extraction.

2.1. Cepstrum for feature extraction

This subsection briefly describes the Cepstrum-based feature extraction that has been widely used in speech recognition and acoustic signal classification (Childers et al., 1977). Given a signal $f(t),\ t = 1, \ldots, N$, Cepstra are usually computed in the following form (Oppenheim and Shafer, 1975), which is used in the *rceps* Matlab function (http://www.mathworks.com/help/signal/ref/rceps.html):

$$f_c(t) = \Re(\mathcal{F}^{-1}(log \mid F(\omega) \mid)) \tag{1}$$

where $F(\omega)$ is the Fourier transform of the signal $f(t)$; the operator $\mathcal{F}^{-1}$ is the inverse Fourier transform; and $\Re(z)$ indicates the real part of a complex scalar z.

A slightly modified version of this algorithm is used here. After obtaining the Fourier transform of the signal, frequency components with small values are discarded before taking the inverse Fourier transform to prevent high unimportant components from gaining higher Cepstrum values. After finding the Cepstrum features, the first N_c components are used as Cepstrum features for classification. Algorithm 1 for Cepstrum-based feature extraction is presented below.

Algorithm 1. Cepstrum for feature extraction

Input: Time series data sets $\boldsymbol{x} \in \mathbb{R}^{1\times N}$; Cut-off sample N_f (where $N_f \leqslant N$); and dimension of the Cepstrum feature N_c (where $N_c \leqslant N_f$).

Output: Extracted Cepstrum-based feature $\boldsymbol{p} \in \mathbb{R}^{1\times N_c}$ of the time-series $\boldsymbol{x}$

1: Compute the magnitude of FFT $\mid F(\omega) \mid$ of the given time series where $\omega = 1, \ldots, N$

2: Store the first N_f frequency components and discard the rest

3: Compute $f_c(t) = \Re(\mathcal{F}^{-1}(log \mid F(\omega) \mid)), t = 1, \ldots, N_f$

4: Compute the feature $\boldsymbol{p} = [f_c(1)\ f_c(1) \ldots f_c(N_c)]$

2.2. Principal component analysis for feature extraction

This subsection briefly describes the principal component analysis (PCA) for feature extraction, which has been widely used in diverse applications (Bishop, 2006). While the Cepstrum-based feature extraction makes use of only the information imbedded in a time series, PCA takes advantage of the information of the ensemble of training data in addition to the local information to extract the features (Bishop, 2006). Let the training data sets (i.e., the ensemble of time series) be organized as an $(M \times N)$-dimensional data matrix, where M is the number of training data sets and N is the length of each data set (i.e., 1-D time series). For the detection problem, the training data consists of both "No Target" and "Target Present" classes while, in the human/animal classification problem, the training data consists only of samples from the "Target Present" class.

Let X be the centered version of the original $(M \times N)$ data matrix, where each row of X is an individual data sample x with zero mean. In the application of target detection and classification, M is often smaller than N, i.e., the number of training samples are fewer than the length of time-series. Therefore, it is numerically efficient to analyze the $(M \times M)$ matrix $(1/M)XX^T$ that has the same non-zero eigenvalues as the $(N \times N)$ computed covariance matrix $(1/M)X^TX$ (Bishop, 2006).

Let v_i be the normalized eigenvectors of the real symmetric matrix $(1/M)XX^T$ corresponding to the (real positive) eigenvalues λ_i that are arranged in the decreasing order of magnitude, i.e., $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_M$. Let m be the smallest integer such that $\sum_{i=1}^{m}\lambda_i > \eta\sum_{i=1}^{M}\lambda_i$ where $1 \leqslant m \leqslant M$, where the threshold parameter η is a real positive fraction close to one. The corresponding (normalized) eigenvectors u_i in the original data space are obtained in terms of v_i and λ_i as follows (Bishop, 2006; Mallapragada et al., 2012):

$$u_i = \frac{1}{\sqrt{M\lambda_i}} X^T v_i, \quad i = 1, 2, \ldots, m. \tag{2}$$

The PCA projection matrix $W \in \mathbb{R}^{N\times m}$ is then obtained by grouping the computed eigenvectors as follows:

$$W = [u_1 u_2 \ldots u_m] \tag{3}$$

The PCA-based feature vector $p \in \mathbb{R}^{1\times m}$ for a given (train or test) time-series $x \in \mathbb{R}^{1\times N}$, is then computed by projection as follows:

$$p = xW. \tag{4}$$

Algorithm 2 for PCA-based feature extraction is presented below.

Algorithm 2. Principal component analysis for feature extraction

Input: Training time series data sets $\boldsymbol{x}_j \in \mathbb{R}^{1\times N}, j = 1, \ldots, M$; Tolerance $\eta \in (0, 1)$; and test time series data set $\boldsymbol{x} \in \mathbb{R}^{1\times N}$

Output: Extracted feature vector $\boldsymbol{p} \in \mathbb{R}^{1\times m}$ for the time-series $\boldsymbol{x}$

1: Construct the "centered version" training data matrix $\mathbf{X} \in \mathbb{R}^{M\times N}$, where each row $\boldsymbol{x}_j$ has zero mean.
2: Compute the matrix $\mathbf{S} = (1/M)\mathbf{X}\mathbf{X}^T$
3: Compute the normalized eigenvectors $\{\mathbf{v}_i\}$ of $\mathbf{S}$ with their corresponding eigenvalues $\{\lambda_i\}$ in the decreasing order of magnitude
4: Compute the normalized eigenvectors $\mathbf{u}_i = \frac{1}{\sqrt{M\lambda_i}}(\mathbf{X})^T\mathbf{v}_i$
5: Find the smallest $m \leqslant M$ such that $\sum_{i=1}^{m}\lambda_i > \eta\sum_{i=1}^{M}\lambda_i$
6: Construct $(N \times m)$ projection matrix $\mathbf{W} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m]$
7: Generate $(1 \times m)$ reference patterns $\boldsymbol{p} = \boldsymbol{x}\,\mathbf{W}$

2.3. Symbolic dynamic filtering for feature extraction

This subsection succinctly presents the underlying concept and theory of symbolic dynamic filtering (SDF) (Ray, 2004) for extraction of (low-dimensional) features from time-series data. As stated earlier in Section 1, the SDF algorithm reported in Ray (2004) and Rajagopalan and Ray (2006) includes wavelet-transformed preprocessing of signals to facilitate time–frequency localization and denoising. However, inclusion of wavelet transform requires tuning of additional design parameters, such as the wavelet basis and a set of scales. Since multiplicity of selectable parameters decreases the ease of usage and may affect the performance of the algorithms, this paper has applied SDF directly on the time-series data (i.e., without wavelet preprocessing) to make a fair comparison of the three feature extraction methods based on the same sets of data.

The pertinent steps of the SDF procedure for feature extraction are delineated below.

2.3.1. Symbolization of time series

This step requires partitioning (also known as quantization) of the time series data. The signal space, approximately represented by the training data set, is partitioned into a finite number of cells that are labeled as symbols, i.e., the number of cells is identically equal to the cardinality $|\Sigma|$ of the (symbol) alphabet Σ. As an example for the one-dimensional sensor time series data in Fig. 1, the alphabet $\Sigma = \{\alpha, \beta, \gamma, \delta\}$, i.e., $|\Sigma| = 4$, and three partitioning lines divide the ordinate (i.e., y-axis) of the time series profile into four mutually exclusive and exhaustive regions. These disjoint regions form a partition, where each region is labeled with one symbol from the alphabet Σ. If the value of time series at a given instant is located in a particular cell, then it is coded with the symbol associated with that cell. As such, a symbol from the alphabet Σ is assigned to each (signal) value corresponding to the cell where it belongs. (Details are reported in Rajagopalan and Ray (2006).) Thus, a (finite) array of symbols, called a symbol string (or symbol block), is generated from the (finite-length) time series data.

The ensemble of time series data are partitioned by using a partitioning tool (e.g., maximum entropy partitioning (MEP) or uniform partitioning (UP) methods Rajagopalan and Ray (2006)). In UP, the partitioning lines are separated by equal-sized cells. On the other hand, MEP maximizes the entropy of the generated symbols and therefore, the information-rich cells of a data set are partitioned finer and those with sparse information are partitioned coarser, i.e., each cell contains (approximately) equal number of data points under MEP. In both UP and MEP, the choice of alphabet size $|\Sigma|$ largely depends on the specific data set and the allowable error of detection and classification.

In this paper, MEP has been adopted as the partitioning tool to accommodate sparsity of the time series data to be symbolized. For the purpose of pattern classification, the training data set is partitioned with a given alphabet size $|\Sigma|$ that is selected by trade-off between information loss and computational complexity (Rajagopalan and Ray, 2006) and the partitioning is subsequently kept constant for the test data.

2.3.2. Construction of probabilistic finite state automata (PFSA)

The core assumption for construction of probabilistic finite state automata (PFSA) is that the symbolic process for different classes of targets can be approximated as a Markov chain of order D, called the D-Markov machine, where D is a positive integer. While the details of the D-Markov machine are given in Ray (2004), the pertinent information on the construction of a D-Markov machine is presented below.

A D-Markov chain is a statistically (quasi-) stationary stochastic process $S = \cdots s_{-1}s_0 \cdots s_1 \cdots$, where the probability of occurrence of a new symbol depends only on the last D symbols, i.e.,

$$P[s_n|s_{n-1}\cdots s_{n-D}\cdots] = P[s_n|s_{n-1}\cdots s_{n-D}]$$

The construction of a D-Markov machine is based on: (i) *state splitting* that generates symbol blocks, also called words, of different lengths according to their relative importance; and (ii) *state merging* that assimilates histories from symbol blocks leading to the same symbolic behavior (P. Adenis et al., 2011). Words of length D on a symbol string are treated as the states of the D-Markov machine before any state-merging is executed. Thus, on a (finite) alphabet with cardinality $|\Sigma|$, the total number of possible states is less than or equal to $|\Sigma|^D$; and operations of state merging may significantly reduce the number of states. As the value of D is increased, more memory is imbedded in the Markov states of the PFSA. However, the benefits of having additional memory associated with a larger value of D could be offset by the increased computational load. So, one must make a trade-off between the two competing requirements of:

- Capturing information from the time series, and
- Reduction of computational complexity in terms of memory requirements and execution time in the construction of D-Markov machine algorithms.

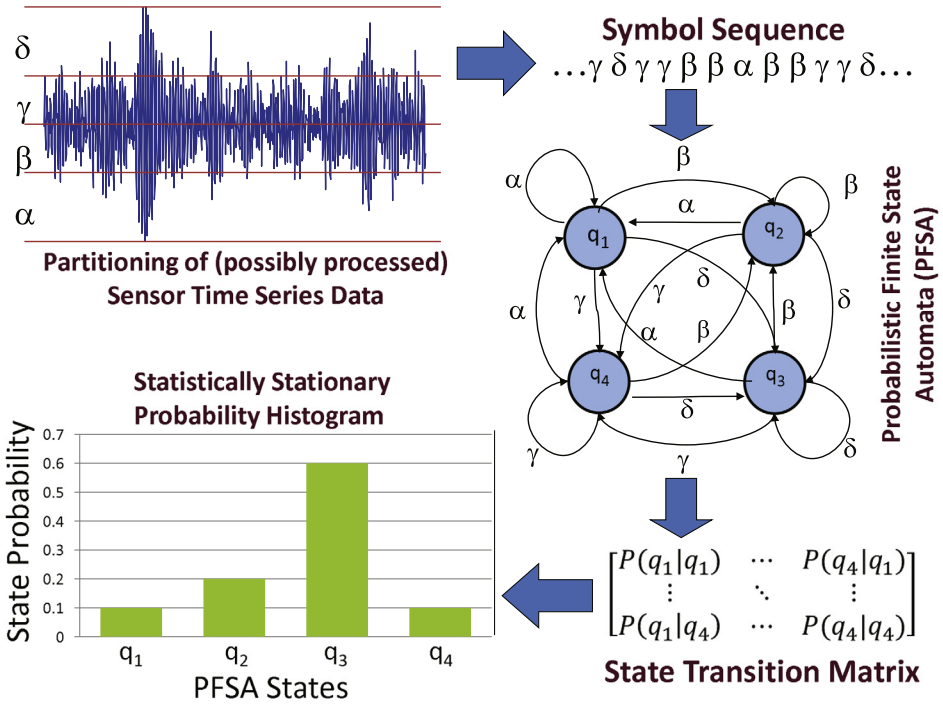


Fig. 1. Concept of symbolic dynamic filtering (SDF) as a feature extractor.

The PFSA states represent different combinations of words on the symbol sequence. In the graph of a PFSA, the directional edge that interconnects a state (i.e. a node) to another state represents the transition probability between these states. Therefore, the "states" denote all possible words within a window of certain length. In this paper, the word size D is taken to be 1 and hence the number of possible states is equal to the number of symbols; therefore, the results reported in this paper are limited to $D = 1$. However, in general, larger values of the integer parameter D (i.e., $D > 1$) might be necessary to capture the long time-scale information about the process. It is noted that the operations of state splitting and state merging might be necessary for PFSA construction with $D > 1$, while these operations are not required for $D = 1$ (P. Adenis et al., 2011).

As a consequence of having $D = 1$, the number of states is equal to the number of symbols, i.e., $|\mathcal{Q}| = |\Sigma|$, where the set of all possible states is denoted as $\mathcal{Q} = \{q_1, q_2, \ldots, q_{|\mathcal{Q}|}\}$ and $|\mathcal{Q}|$ is the number of (finitely many) states. The (estimated) state transition probabilities are defined as:

$$p(q_k|q_l) \triangleq \frac{N(q_l, q_k)}{\sum_{i=1,2,\ldots,|\mathcal{Q}|} N(q_l, q_i)} \quad \forall q_k, q_l \in \mathcal{Q} \tag{5}$$

where $N(q_l, q_k)$ is the total count of events when q_k occurs adjacent to q_l in the direction of motion. Having computed all these probabilities $p(q_k|q_l) \forall q_k, q_l \in \mathcal{Q}$, the (estimated) state transition probability matrix of the PFSA is given as

$$\Pi = \begin{bmatrix} p(q_1|q_1) & \cdots & p\left(q_{|\mathcal{Q}|}|q_l\right) \\ \vdots & \ddots & \vdots \\ p\left(q_1|q_{|\mathcal{Q}|}\right) & \cdots & p\left(q_{|\mathcal{Q}|}|q_{|\mathcal{Q}|}\right) \end{bmatrix}. \tag{6}$$

By appropriate choice of partitioning, it is ensured that the resulting Markov chain model satisfies the ergodicity conditions, i.e., the stochastic matrix Π is irreducible. The rationale is that, under statistically stationary conditions, the probability of every state being reachable from any other state within finitely many transitions must be strictly positive (Berman and Plemmons, 1994). For a given time series, after the matrix Π is constructed, its left eigenvector $\boldsymbol{p}$ corresponding to the (unique) unity eigenvalue is computed. Then, the vector $\boldsymbol{p}$, which is the stationary state probability vector, serves as the "feature" vector extracted from the time series as seen in Fig. 1. This feature vector $\boldsymbol{p}$ is used for pattern classification in the sequel.

2.3.3. Formulation of SDF-based feature extraction algorithms

The SDF-based feature extraction is executed as a combination of two algorithms:

- Maximum entropy partitioning (MEP) of time series data.
- Symbolic dynamic filtering (SDF) for feature extraction, which makes use of the MEP algorithm.

Algorithm 3 and Algorithm 4 for SDF-based feature extraction are presented below.

Algorithm 3. Maximum entropy partitioning

Input: Finite-length string y of time-series data; and number of symbols $|\Sigma|$

Output: Partition vector $\wp \in \mathbb{R}^{|\Sigma|+1}$

1: Sort the time series data string x in the ascending order
2: Let K = length (y)
3: Assign $\wp(1) = y(1)$, i.e., minimum element of y
4: **for** $i = 2$ to $|\Sigma|$ **do**
5: $\quad \wp(i) = y\left(ceil\left(\frac{(i-1)*K}{|\Sigma|}\right)\right)$
6: **end for**
7: Assign $\wp(|\Sigma|+1) = y(\mathrm{K})$, i.e., maximum element of y

Algorithm 4. Symbolic dynamic filtering for feature extraction

Input: Training time series data sets $x_j \in \mathbb{R}^{1\times N}, j = 1,\ldots,M$, a test time series data set $x \in \mathbb{R}^{1\times N}$, and number of symbols $|\Sigma|$

Output: Extracted SDF-based feature vector $p \in \mathbb{R}^{1\times|\Sigma|}$ for the time-series $\boldsymbol{x}$

1: Initialize $y = \emptyset$
2: **for** $j = 1$ to M **do**
3: $y = \boldsymbol{y} \bigcup x_j$
4: **end for**
5: Partition y using Algorithm 3 to obtain the (common) partition vector $\wp$
6: Use $\wp$ on the test data set $\boldsymbol{x}$ to obtain the symbol string $\boldsymbol{s}$
7: Construct the (irreducible) state transition probability matrix Π by using Eqs. (5) and (6)
8: Compute the (sum-normalized) left eigenvector $\boldsymbol{p}$ corresponding to the (unique) unity eigenvalue of Π

3. Target detection and classification

Two classification problems are addressed in this paper, both of which are of binary type. In the first problem of target detection, the output of the classifier is either "no target" or "target present"; and the second problem is about classifying the target as "human" or "animal" led by a human. In an earlier publication (Jin et al., 2012) on this topic, these two classifiers are correlated, i.e., the human\ animal classification is performed only after a target is detected, regardless of whether this detection is correct or false. However, the current paper treats these two classification problems separately, because the main objective here is to compare the performance of Cepstrum, PCA and SDF as feature extractors and thus treating the detection and classification problems separately would yield unambiguous comparison. In this context, three different classifiers that have been used in conjunction with each of Cepstrum, PCA and SDF feature extractors are: (i) support vector machines (SVM) (Bishop, 2006), (ii) k-nearest neighbor (k-NN) (Bishop, 2006), and (iii) sparse representation classifier (SRC) (Nguyen et al., 2011) that are briefly reviewed in the following paragraphs.

3.1. Support vector machine (SVM) and k-nearest neighbor (k-NN) algorithms

The SVM and k-NN algorithms are among the most frequently used tools of pattern classification (Bishop, 2006). For k-NN, the neighborhood size k is kept is not too small to avoid the overfitting problem. Similar to the earlier work of Jin et al. (2012), the SVM method has also been used in the current paper to regenerate the results and compare the performance with other classification algorithms. A Gaussian kernel has been used for SVM, similar to what was done in Nguyen et al. (2011) for target classification with UGS systems.

3.2. Sparse representation classification (SRC) algorithm

The SRC algorithm was first used for solving the problem of face recognition (Wright et al., 2008), where a large matrix $A \triangleq [A_1 A_2 \cdots A_C]$, consisting of the training data, is constructed where $A_i, i \in \{1,\cdots,C\}$ is a $(n \times N_i)$ training matrix consisting of the training samples belonging to the ith class, n is the dimension of the feature vector, and $N_i, i \in \{1,\cdots,C\}$ is the number of training samples in the class i, and C is the total number of class labels. It is assumed that a test sample from the i^{th} class lies approximately within the subspace formed by the training data of the ith class. For a given test vector y in the sparse classification algorithm, the following ℓ_1-optimization problem (Donoho, 2006) needs to be solved as:

$$\min \|x\|_{\ell_1} \text{ such that } \|y - Ax\|_{\ell_2} \leqslant \epsilon$$

In the above optimization problem, the user-selected parameter ϵ is a representative of the upper bound of the noise spectrum in the data (Candes and Tao, 2006), where the optimal solution x of the above ℓ_1 optimization is shown to be a sparse vector. For $x \in \mathbb{R}^M$, where M is the total number of training samples, let $\delta_i(x) \in \mathbb{R}^M$ be a new vector whose only non-zero elements are the entries in x that are associated with class i in A. In the noiseless scenario, if the test data y belongs to the ith class, then $\delta_j(x)$ should be a zero vector for all possible $j \neq i$. However, since noisy data are encountered in real-life applications, the residuals $\|y - A\delta_i(x)\|_2$ needs to be computed for classification of the test sample y and the label of the test vector y is predicted as the argument i^* that is the minimizer of the residuals.

One of the main advantages of the SRC algorithm is that a careful selection of the features is not necessary. What is critical, however, is whether the number of features is sufficiently large and whether the sparse representation is correctly computed. The SRC algorithm is also robust in dealing with noisy data (Candes and Tao, 2006).

4. Results and discussion

This section presents the results of target detection and classification, which were generated from two sets of field data, each consisting of time series generated from passive infrared (PIR) and seismic sensors in different sensor sites. Each of the test scenarios consists either of the two classes of targets: (i) human walking alone, and (ii) animal led by a walking human. These targets moved along an approximately 150 m long trail and returned along the same trail to the starting point; all targets passed by the sensor sites at a distance of approximately 5 m. Signals from both PIR and seismic sensors were acquired at a sampling frequency of 10 kHz and each test was conducted over a period of approximately 50 s.

The field data set #1 was collected on three different days, Day#1, Day#2 and Day#3, from test fields on a wash (i.e., the dry bed of an intermittent creek) and at a choke point (i.e., a place where the targets are forced to go due to terrain difficulties). In contrast, the field data set #2 was collected on four different days, Day#1, Day#2, Day#3 and Day#4, from test fields on different types of terrains such as wash, trail and watering stations. Table 1 lists the numbers of different labeled scenarios for each day for both data sets #1 and #2.

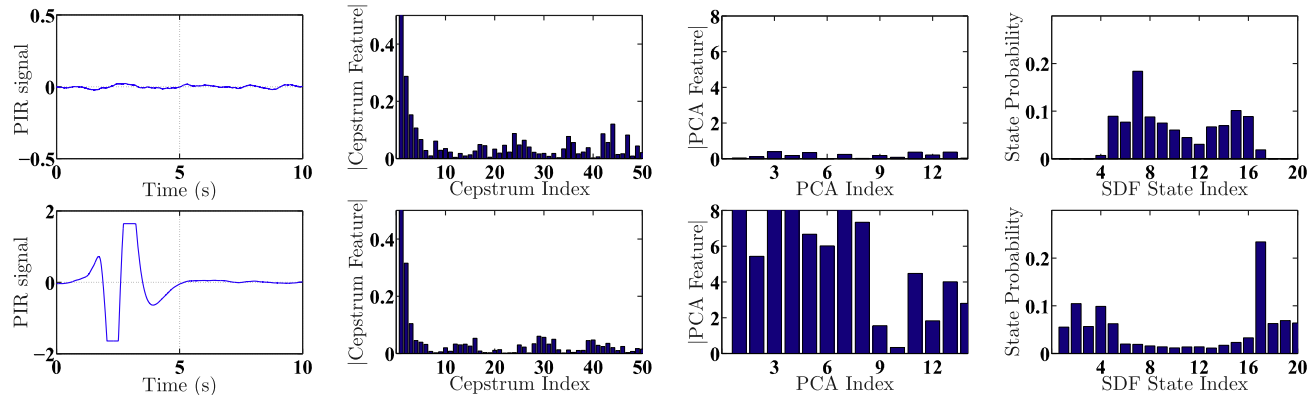
The data sets #1 and #2 have been analyzed for performance comparison of the feature extractors, Cepstrum, PCA, and SDF in conjunction with the (binary) classifiers, SVM, k-NN, and SRC that have the following respective design parameters:

- SVM: Variance of the Gaussian kernel and regularization parameter (Scholkpf and Smola, 2002; Bishop, 2006);
- k-NN: Neighborhood size k (Bishop, 2006);
- SRC: Error upper bound ϵ (Nguyen et al., 2011).

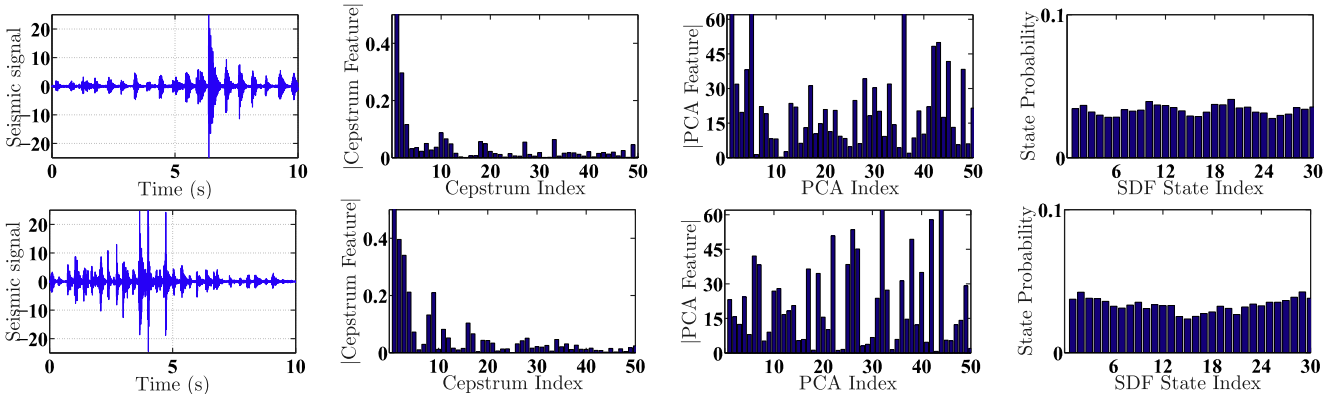
Fig. 2 presents typical time series and extracted features from PIR and seismic sensor signals, where individual rows represent different scenarios and the individual columns represent the following: sensor signals in Column 1 along with the extracted features generated by Cepstrum in Column 2, PCA in Column 3, and SDF in Column 4.

Table 1
Number of different scenarios in each day for the data sets #1 and #2.

Target	Data set #1				Data set #2				
	Day1	Day2	Day3	Total	Day1	Day2	Day3	Day4	Total
No target	32	28	50	110	6	56	82	66	210
Human	14	22	30	66	6	21	47	32	106
Animal	18	6	20	44	0	35	35	34	104



(a) Typical PIR sensor signals: "No target" (top row) and "Presence of a target: human and/or animal" (bottom row)



(b) Typical seismic sensor signals: "Human target" (top row) and "Animal target led by a human" (bottom row)

Fig. 2. Sensor signals (Column 1) along with the extracted features generated by Cepstrum (column 2), PCA (column 3) and SDF (column 4).

Let N_1 be the total number of test samples for the Class 1 target and let n_1 (where $0 \leqslant n_1 \leqslant N_1$) be the number of misclassifications of the Class 1 target as belonging to Class 2; similarly, let N_2 and n_2 (where $0 \leqslant n_2 \leqslant N_2$) be the corresponding parameters for the Class 2 target. The goal is to select the design parameter of a classifier by making a trade-off between two conflicting objectives of minimizing misclassifications of Class 1 and Class 2 targets, i.e., classifying a target to one class when it truly belongs to the other class. Let $P_1 \triangleq \frac{n_1}{N_1}$ and $P_2 \triangleq \frac{n_2}{N_2}$ be the estimated probabilities of misclassifying Class 1 and Class 2 targets, respectively. In this context, the composite objective function is selected to be a weighted linear combination of the misclassification rates.

$$J(\alpha) = \alpha P_1 + (1-\alpha)P_2 \tag{7}$$

where $\alpha \in (0,1)$ is the (user-selected) trade-off parameter; and $J(\alpha)$ is the cost functional to be minimized with respect to the classifier design parameter. Depending on how the parameter α is selected, minimization of $J(\alpha)$ may lead to different optimal solutions for P_1 and P_2. Given N_1 and N_2 and selecting $\alpha = \left(\frac{N_1}{N_1+N_2}\right)$, the cost functional J in Eq. (7) becomes $\left(\frac{n_1+n_2}{N_1+N_2}\right)$ that is the total misclassification rate of combined Class 1 and Class 2 targets.

For each field data set, parameters of the feature extraction algorithms are chosen by numerical experiments and are kept invariant for all tests. For Cepstrum method (see Algorithm 1 in Section 2), N_c and N_f are set to 100 and 150, respectively. Thus, the number of Cepstrum features is 100 for both detection and classification. For the PCA method (see Algorithm 2 in Section 2), 96% of the total variance is used as the cumulative proportion to select the number of principal components which, in turn, is the number of PCA features used for both target detection and classification. For SDF-based feature extraction (see Algorithm 3 and Algorithm 4 in Section 2), alphabet size $|\Sigma| = 20$ has been used for detection and $|\Sigma| = 30$ for classification. In the training phase, for a finite set of test samples in each class, the design parameter of each of the three (binary) classifiers, SVM, k-NN and SRC, is determined by minimizing the total misclassification rate as explained above.

A three-way cross-validation, based on the data collected on three days for the data set #1, has been used to assess the performance of the three feature extractors, Cepstrum, PCA and SDF, for target detection and classification by using time series data from PIR and seismic sensors. The data sets are divided into three sets

by date, i.e., first data from Day 1 and Day 2 are used for training and the classifiers are tested on Day 3. This process is repeated for two other combinations: (a) Day 3 and Day 1 data as training data and Day 2 as test data, and (b) Day 2 and Day 3 data as training data and Day 1 as test data. Thus, three different combinations of training and testing have been performed for the field data set #1. Similarly, for the data set #2, a four-way cross-validation has been used based on the data collected on four days to assess the performance of the three feature extractors, Cepstrum, PCA, and SDF, for target detection and classification from both PIR and seismic sensor data. Thus, four different combinations of training and testing have been performed for the field data set #2.

4.1. Target detection

Table 2 shows averaged confusion matrices for the target detection problem for each of the three feature extractors, where the upper part pertains to data set #1 and the lower part to data set #2. These confusion matrices are obtained by applying the SRC classifier individually on PIR and seismic sensor data when, in the cross validation process, the respective data on each of the single days are used for testing and the remaining data for training; the results of each test scenario are then summed up to form the respective confusion matrix in Table 2. Each column of the twelve confusion matrices (i.e., six confusion matrices for each data set) represents the instances in a predicted class of "No Target" or 'Target present," and each row represents the instances in an actual class (i.e., the ground truth). For data set #1, out of a total of 220 scenarios for PIR sensors, Cepstrum and PCA yield 196 and 179 correct decisions, respectively, while SDF makes 215 correct decisions; the corresponding number for data set #2 are: out of a total of 420 scenarios for PIR sensors, Cepstrum and PCA yield 340 and 328 correct decisions, respectively, while SDF makes 415 correct decisions. The results are similar for seismic sensors.

Table 3 summarizes the average rates of successful target detection, false alarm, and wrong detection by cross-validation for both data sets #1 and #2, when either Cepstrum, PCA or SDF is used as the feature extractor for target detection along with the three different classifiers: SVM, k-NN and SRC. In terms of wrong detection, the performance of SDF is significantly superior to that of both Cepstrum and PCA for all three classifiers. It is seen that, for data set #1, SDF has yielded higher successful target detection rates than Cepstrum and PCA with consistently small false alarm rates. It is also seen that, on the average, SDF yields better results than Cepstrum and PCA for data set #2.

4.2. Target classification

Table 4 shows averaged confusion matrices with the SRC classifier for classification of human versus animal led by human, using PIR and seismic sensors, respectively, for each of the three feature extractors, where the upper part pertains to data set #1 and the lower part to data set #2. The results are obtained by applying SRC individually on PIR and seismic sensor data when, in the cross validation process, the respective data on each of the single days are used for testing and the remaining data for training. Each column of the twelve confusion matrices (i.e., six confusion matrices for each data set) represents the instances in a predicted class of "Human" or "Animal," and each row represents the instances in an actual class (i.e., the ground truth). For the data set #1, out of a total of 110 scenarios for seismic sensors, Cepstrum and PCA yield 76 and 62 correct classifications, respectively, while SDF makes 84 correct classifications; the corresponding number for data set #2 are: out of a total of 220 scenarios for seismic sensors, Cepstrum and PCA yield 119 and 109 correct classifications, respectively, while SDF makes 158 correct classifications. The results are qualitatively similar for PIR sensors.

Table 5 summarizes the average rates of successful human classification, false alarm, and misclassification by cross-validation for both data sets #1 and #2, when either Cepstrum, PCA or SDF is used as the feature extractor for target detection along with the three different classifiers: SVM, k-NN and SRC. In terms of misclassification in each of the data sets #1 and #2, the performance of SDF is significantly superior to that of both Cepstrum and PCA for all three classifiers.

It is also observed that, in most of the scenarios, usage of PIR sensors resulted in better classification performance compared to that of seismic sensors for both detection and classification. Better results are expected if fusion techniques are employed to take advantage of the cross-correlated information of the sensors, which is a topic for future research.

4.3. Computational costs

This section presents a comparison of the computational costs (i.e., execution time) of the three feature extraction algorithms under consideration. To this end, typical simulation costs have been generated based on a data set consisting of 156 sets of time series as training samples and 64 different sets of time series as test samples. The results have been generated on a single core of a 2.13 Ghz CPU with 6 GB memory. It is noted that each of PCA and SDF has a

Table 2
Confusion matrices obtained by the SRC classifier using PIR and seismic sensors of data sets #1 and #2 for target detection. Each confusion matrix is obtained by summing up the confusion matrices that are generated by cross validation.

		PIR		Seismic	
		No Target	Target Present	No Target	Target Present
Data set #1					
Cepstrum	No Target	91	19	104	6
	Target Present	5	105	11	99
PCA	No Target	95	15	70	40
	Target Present	26	84	39	71
SDF	No Target	108	2	107	3
	Target Present	3	107	9	101
Data set #2					
Cepstrum	No Target	199	11	179	31
	Target Present	69	141	54	156
PCA	No Target	153	57	43	167
	Target Present	35	175	28	182
SDF	No Target	207	3	198	12
	Target Present	2	208	28	182

Table 3
Results of three-way and four-way cross-validation for target detection of data set #1 and #2, respectively.

		Successful target detection			False alarm			Wrong decision		
		Cepstrum	PCA	SDF	Cepstrum	PCA	SDF	Cepstrum (%)	PCA (%)	SDF (%)
Data set #1										
PIR	SVM	0.97	0.95	0.97	0.05	0.00	0.00	3.64	2.27	1.36
	k-NN	0.95	0.90	0.97	0.00	0.00	0.00	2.72	5.00	1.36
	SRC	0.95	0.76	0.97	0.17	0.14	0.02	10.91	18.64	2.27
Seismic	SVM	0.87	0.60	0.94	0.14	0.14	0.09	13.18	27.27	7.73
	k-NN	0.78	0.15	0.94	0.25	0.05	0.08	23.64	45.00	7.27
	SRC	0.90	0.65	0.92	0.05	0.36	0.03	7.72	35.91	5.45
Data set #2										
PIR	SVM	1.00	1.00	0.99	0.02	0.00	0.01	1.43	0.23	1.43
	k-NN	1.00	0.88	1.00	0.01	0.01	0.01	0.24	6.43	0.71
	SRC	0.67	0.83	0.99	0.05	0.27	0.01	19.05	21.90	1.19
Seismic	SVM	0.88	0.91	0.90	0.16	0.08	0.07	14.05	8.33	8.57
	k-NN	0.89	0.59	0.88	0.10	0.01	0.08	10.71	20.95	10.00
	SRC	0.74	0.83	0.87	0.15	0.73	0.05	20.24	45.00	9.29

Table 4
Confusion matrices obtained by SRC classifier for PIR and seismic sensors of data sets #1 and #2 for human/animal classification.

		PIR		Seismic	
		Human	Animal	Human	Animal
Data set #1					
Cepstrum	Human	51	15	56	10
	Animal	25	19	24	20
PCA	Human	61	5	49	17
	Animal	21	23	31	13
SDF	Human	57	9	55	11
	Animal	16	28	15	29
Data set #2					
Cepstrum	Human	64	42	71	35
	Animal	52	52	56	48
PCA	Human	59	47	16	90
	Animal	33	71	11	93
SDF	Human	83	23	91	15
	Animal	54	50	37	67

Table 5
Results of three-way and four-way cross-validation for human/animal classification of data set #1 and #2, respectively.

		Successful Human Classification			Human False Alarm			Wrong Decision		
		Cepstrum	PCA	SDF	Cepstrum	PCA	SDF	Cepstrum (%)	PCA (%)	SDF (%)
Data set #1										
PIR	SVM	0.73	0.92	0.88	0.45	0.43	0.20	34.55	21.82	15.45
	k-NN	0.88	0.91	0.89	0.41	0.52	0.34	23.64	26.36	20.00
	SRC	0.77	0.92	0.86	0.57	0.48	0.36	36.36	23.64	22.73
Seismic	SVM	0.68	0.24	0.85	0.59	0.45	0.41	41.82	63.64	25.45
	k-NN	0.70	0.55	0.86	0.86	0.68	0.50	52.73	54.55	28.18
	SRC	0.85	0.74	0.83	0.55	0.70	0.34	30.91	43.64	23.64
Data set #2										
PIR	SVM	0.76	0.61	0.89	0.42	0.58	0.38	32.86	48.09	24.76
	k-NN	0.73	0.59	0.76	0.72	0.46	0.52	49.05	43.33	37.62
	SRC	0.60	0.56	0.78	0.50	0.32	0.52	44.76	38.10	36.67
Seismic	SVM	0.63	0.92	0.91	0.38	0.53	0.38	37.62	30.48	23.81
	k-NN	0.56	0.97	0.77	0.36	0.69	0.41	40.00	35.71	31.90
	SRC	0.67	0.15	0.86	0.54	0.10	0.36	43.33	48.10	24.76

training phase which is usually performed off-line, while Cepstrum treats training and test samples uniformly.

In the experiments, the Cepstrum features were extracted in $\sim$ 16 ms for each time series. The training phase for PCA, which includes finding the projection matrix and extracting the features for the training samples, took $\sim$ 1.8 s while the feature for each test sample was generated in less than 10 microseconds. Finally, The training phase for SDF, which includes partitioning of the data set and extraction of features for the training samples, took $\sim$ 2.4 s while the feature of a test sample was generated in

$\sim$ 37 ms. Therefore, for real-time applications, the operations of both SDF and Cepstrum are performed in the order of milliseconds to extract features from test time series with a slight advantage toward Cepstrum, and PCA is the fastest because it only accounts for a linear projection for extracting the features.

5. Summary, conclusions, and future work

This paper presents a comparative evaluation of Cepstrum, principal component analysis (PCA) and symbolic dynamic filtering (SDF) as feature extractors for target detection and classification. All three methods have been tested on two data sets, consisting of passive infrared (PIR) and seismic sensors, collected from different fields. The underlying algorithms of feature extraction have been executed in conjunction with three different classification algorithms, namely, support vector machines (SVM), k-nearest neighbor (k-NN), and sparse representation classifier (SRC). Cross-validation has been used to assess the performance of Cepstrum, PCA and SDF as feature extractors for both PIR and seismic sensor data sets. The results show consistently superior performance of SDF-based feature extraction over both Cepstrum-based and PCA-based feature extraction in terms of successful detection, false alarm, and wrong detection and classification decisions. The rationale for superior performance of SDF over Cepstrum and PCA as a feature extractor is presented below.

Cepstrum features are extracted using only the local information of a given time series without making use of the training data. In contrast, PCA utilizes the training data to find the linear transformation and approximates the subspace in which most of the variance of the training data lies. Although this is an advantage of PCA over Cepstrum, a linear transformation is not necessarily a good method of feature extraction from the data; this might indeed be a possible reason of degraded classification performance of PCA. On the other hand, SDF is a nonlinear feature extraction algorithm that extracts the pertinent information as a feature vector and makes use of the full training data to find the appropriate partitioning of the test time series.

Topics for future research in this area include further theoretical study on refinement of the SDF algorithms and field testing under different environments. In this regard, a few key topics are delineated below.

1. Investigation of the impact of finite length of training and test data sets on classification performance (Wen et al., 2013; Wright et al., 2008).
2. Comparison with additional methods of feature extraction (i.e., besides Cepstrum and PCA) in conjunction with other classification algorithms (i.e., besides SVM, k-NN and SRC).
3. Development of a rigorous field test procedure for validating robustness of SDF-based feature extraction under different environmental conditions and data types.
4. Performance evaluation of SDF as a feature extractor for classification of different types of human motion, such as walking and running.
5. Fusion of different types of sensors to extract cross-correlated information between the informational sources to improve the classification performance.

References

Adenis, P., Mukherjee, K., Ray, A. 2011. State splitting and state merging in probabilistic finite state automata. In: Proceedings of 2011 American Control Conference, pp. 5145–5150.

Altmann, J., 2004. Acoustic and seismic signals of heavy military vehicles for cooperative verification. Proceedings of the IEEE 273 (4–5), 713740.

Berman, A., Plemmons, R., 1994. Nonnegative Matrices in the Mathematical Sciences. SIAM, Philadelphia, PA, USA.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. Springer.

Candes, E., Tao, T., 2006. Near-optimal signal recovery from random projections: universal encoding strategies? IEEE Transactions on Information Theory 52, 5406–5425.

Childers, D., Skinner, D., Kemerait, R., 1977. The cepstrum: a guide to processing. Proceedings of the IEEE 65 (10), 1428–1443.

Donoho, D., 2006. Compressed sensing. IEEE Transactions on Information Theory 52, 1289–1306.

Gramann, R.A., Bennett, M., Obrien, T.D., 1999. Vehicle and personnel detection using seismic sensors, Sensors, C3I. Information, and Training Technologies for Law Enforcement 3577 (1), 7485.

Jin, X., Sarkar, S., Ray, A., Gupta, S., Damarla, T., 2012. Target detection and classification using seismic and PIR sensors. IEEE Sensors Journal 12 (6), 1709–1718.

Li, D., Wong, K.D., Hu, Y.H., Sayeed, A.M., 2002. Detection, classification, and tracking of targets. IEEE Signal Processing Magazine 19 (2), 17–29.

Mallapragada, G., Ray, A., Jin, X., 2012. Symbolic dynamic filtering and language measure for behavior identification of mobile robots. IEEE Transactions on Systems, Man, and Cybernetics, Part B 42 (3), 647–659.

Nguyen, N., Nasrabadi, N., Tran, T. 2011. Robust multi-sensor classification via joint sparse representation. In: Proceedings of the 14th International Conference on Information Fusion (FUSION), pp. 1–8.

Oppenheim, A., Shafer, R., 1975. Digital Signal Processing. Prentice Hall, NJ, USA.

Rajagopalan, V., Ray, A., 2006. Symbolic time series analysis via wavelet-based partitioning. Signal Processing 86 (11), 3309–3320.

Ray, A., 2004. Symbolic dynamic analysis of complex systems for anomaly detection. Signal Processing 84, 1115–1130.

Scholkpf, B., Smola, A., 2002. Learning with Kernels Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA.

Subbu, A., Ray, A., 2008. Space partitioning via Hilbert transform for symbolic time series analysis. Applied Physics Letters 92, 084107-1.

Succi, G., Clapp, D., Gampert, R., Prado, G., 2001. Footstep detection and tracking. Unattended Ground Sensor Technologies and Applications III 4393 (1), 22–29.

Tian, Y., Qi, H. 2002. Target detection and classification using seismic signal processing in unattended ground sensor systems. In: International Conference on Acoustics Speech and, Signal Processing.

Wen, Y., Mukherjee, K., Ray, A., 2013. Adaptive pattern classification for symbolic dynamic systems. Signal Processing 93 (1), 252–260.

Wright, J., Yang, A., Ganesh, A., Sastry, S., Ma, Y., 2008. Robust face recognition via sparse representation. The IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2), 210–227.

Zhang, Z., Gao, X., Biswas, J., Wu, K. 2007. Moving targets detection and localization in passive infrared sensor networks. In: Proceedings of 10th International Conference on, Information Fusion.