Chapter 1

Symbolic Dynamic Filtering for Pattern Recognition in Distributed Sensor Networks

Xin Jin
Department of Mechanical and Nuclear Engineering, The Pennsylvania State University, University Park, PA 16802

Shalabh Gupta
Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269, USA

Kushal Mukherjee
Department of Mechanical and Nuclear Engineering, The Pennsylvania State University, University Park, PA 16802

Asok Ray
Department of Mechanical and Nuclear Engineering, The Pennsylvania State University, University Park, PA 16802

1 Introduction

Proliferation of modern sensors and sensing applications provides us with large volumes of data. By extracting useful information from these data sets in real time, we enhance the ability to better comprehend and analyze the environment around us. Tools for data-driven feature extraction and pattern classification facilitate performance monitoring of distributed dynamical systems over a sensor network, especially if the physics-based models are either inadequate or unavailable [14]. In this regard, a critical issue is real-time analysis of sensor time series for information compression into low-dimensional feature vectors that capture the relevant information of the underlying dynamics [22][10][9][3]. Time series analysis is a challenging task if the data set is voluminous (e.g., collected at a fast sampling rate), high-dimensional, and noise-contaminated. Moreover, in a distributed sensor network, data collection occurs simultaneously at multiple nodes. Consequently, there is a need for low-complexity algorithms that could be executed locally at the nodes to generate compressed features and therefore reduce the communication overhead. In general, the success of data-driven pattern classification tools depends on the quality of feature extraction from the observed time series. To this end, several feature extraction tools, such as principal component analysis (PCA) [10], independent component analysis (ICA) [21], kernel PCA [31], dynamic time warping [2], derivative time series segment approximation [12], artificial neural networks (ANN) [20], hidden Markov models (HMM) [36], and wavelet transforms [35, 26, 27] have been reported in technical literature. Wavelet packet decomposition (WPD) [26] and fast wavelet transform (FWT) [27] have been used for extracting rich problem-specific information from sensor signals. Feature extraction is followed by pattern classification (e.g., using support vector machines (SVM)) [9][3].

The concepts of Symbolic Dynamics [23] have been used for information extraction from time series in the form of symbol sequences [14][8]. Keller and Lauffer [19] used tools of symbolic dynamics for analysis of time series data to visualize qualitative changes of electroencephalography (EEG) signals related to epileptic activity. Along this line, a real-time data-driven statistical pattern analysis tool, called *symbolic dynamic filtering* (SDF) [30][13], has been built upon the concepts of symbolic dynamics and information theory. In the SDF method, time series data are converted to symbol sequences by appropriate partitioning [13]. Subsequently, probabilistic finite-state automata (PFSA) [30] are constructed from these symbol sequences that capture the underlying system's behavior by means of information compression into the corresponding matrices of state-transition probability. SDF-based pattern identification algorithms have been experimentally validated in the laboratory environment to yield superior performance over several existing pattern recognition tools (e.g., PCA, ANN, particle filtering, unscented Kalman filtering, and kernel regression analysis [9][3]) in terms of early detection of anomalies (i.e., deviations from the normal behavior) in the statistical characteristics of the observed time series [29][15].

Partitioning of time series is a crucial step for symbolic representation of sensor signals. To this end, several partitioning techniques have been reported in literature, such as

symbolic false nearest neighbor partitioning (SFNNP) [4], *wavelet-transformed space partitioning* (WTSP) [28], and *analytic signal space partitioning* (ASSP) [32]. In particular, the wavelet transform-based method is well-suited for time-frequency analysis of non-stationary signals, noise attenuation, and reduction of spurious disturbances from the raw time series data without any significant loss of pertinent information [24]. In essence, WTSP is suitable for analyzing the noisy signals, while SFNNP and ASSP may require additional preprocessing of the time series for denoising. However, the wavelet transform of time series introduces two new domain parameters (i.e., scale and shift), thereby generating an image of wavelet coefficients. Thus, the (one-dimensional) time series data is transformed into a (two-dimensional) image of wavelet coefficients. Jin et al. [17] have proposed a feature extraction algorithm from the wavelet coefficients by directly partitioning the wavelet images in the (two-dimensional) *scale-shift* space for SDF analysis.

This chapter focuses on feature extraction for pattern classification in distributed dynamical systems, possibly served by a sensor network. These features are extracted as statistical patterns using symbolic modeling of the wavelet images, generated from sensor time series. An appropriate selection of the wavelet basis function and the scale range allows the wavelet-transformed signal to be de-noised relative to the original (possibly) noise-contaminated signal before the resulting wavelet image is partitioned for symbol generation. In this way, the symbolic images generated from wavelet coefficients capture the signal characteristics with larger fidelity than those obtained directly from the original signal. These symbolic images are then modeled using probabilistic finite state automata (PFSA) that, in turn, generate the low-dimensional statistical patterns, also called feature vectors. In addition, the proposed method is potentially applicable for analysis of regular images for feature extraction and pattern classification. From these perspectives, the major contributions of the chapter are as follows:

1. Development of a SDF-based feature extraction method for analysis of two-dimensional data (e.g., wavelet images of time series in the scale-shift domain);

2. Validation of the feature extraction method in two different applications:
   (i) Behavior recognition in mobile robots by identification of their type and motion profiles, and
   (ii) Target detection and classification using unattended ground sensors (UGS) for border security.

The chapter is organized into seven sections including the present one. Section 2 briefly describes the concepts of symbolic dynamic filtering (SDF) and its application to wavelet-transformed data. Section 3 presents the procedure of feature extraction from the symbolized wavelet image by construction of a probabilistic finite state automaton (PFSA). Section 4 describes the pattern classification algorithms. Section 5 presents experimental validation for classification of mobile robot types and their motion profiles. The experimental facility incorporates distributed pressure sensors under the floor to track and classify the mobile robots. Section 6 validates the feature extraction and pattern classification algorithms based the field data of unattended ground sensors (UGS) for target detection and classification. The chapter is concluded in Section 7 along with recommendations for future research.

2 Symbolic Dynamics and Encoding

This section presents the underlying concepts of symbolic dynamic filtering (SDF) for feature extraction from sensor time series data. Details of SDF have been reported in previous publications for analysis of (one-dimensional) time series [30][13]. A Statistical Mechanics-based concept of time series analysis using symbolic dynamics has been presented in [14]. This section briefly reviews the concepts of SDF for analysis of (two-dimensional) wavelet images for feature extraction. The major steps of the SDF method for feature extraction are delineated as follows:

1. Encoding (possibly nonlinear) system dynamics from observed sensor data (e.g., time series and images) for generation of symbol sequences;
2. Information compression via construction of probabilistic finite state automata (PFSA) from the symbol sequences to generate feature vectors that are representatives of the underlying dynamical system's behavior.

2.1 Review of Symbolic Dynamics

In the symbolic dynamics literature [23], it is assumed that the observed sensor time series from a dynamical system are represented as a symbol sequence. Let Ω be a compact (i.e., closed and totally bounded) region in the phase space of the continuously-varying dynamical system, within which the observed time series is confined [30][13]. The region Ω is partitioned into $|\Sigma|$ cells $\{\Phi_0, \cdots, \Phi_{|\Sigma|-1}\}$ that are mutually exclusive (i.e., $\Phi_j \cap \Phi_k = \emptyset \ \ \forall j \neq k$) and exhaustive (i.e., $\bigcup_{j=0}^{|\Sigma|-1} \Phi_j = \Omega$), where Σ is the *symbol alphabet* that labels the partition cells. A trajectory of the dynamical system is described by the discrete time series data as: $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \cdots\}$, where each $\mathbf{x}_i \in \Omega$. The trajectory passes through or touches one of the cells of the partition; accordingly the corresponding symbol is assigned to each point $\mathbf{x}_i$ of the trajectory as defined by the mapping $\mathcal{M} : \Omega \to \Sigma$. Therefore, a sequence of symbols is generated from the trajectory starting from an initial state $\mathbf{x}_0 \in \Omega$, such that

$$\mathbf{x}_0 \rightarrowtail \sigma_0\sigma_1\sigma_2 \ldots \sigma_k \ldots \tag{1.1}$$

where $\sigma_k \triangleq \mathcal{M}(\mathbf{x}_k)$ is the symbol at instant k. (Note: The mapping in Eq. (1.1) is called *Symbolic Dynamics* if it attributes a legal (i.e., physically admissible) symbol sequence to the system dynamics starting from an initial state.) The next subsection describes how the time series are transformed into wavelet images in scale-shift domain for generation of symbolic dynamics.

2.2 Transformation of Time Series to Wavelet Domain

This section presents the procedure for generation of wavelet images from sensor time series for feature extraction. A crucial step in symbolic dynamic filtering [30][13] is partitioning of the data space for symbol sequence generation [8]. Various partitioning techniques

have been suggested in literature for symbol generation, which include variance-based [34], entropy-based [6], and hierarchial clustering-based [18] methods. A survey of clustering techniques is provided in [22]. Another partitioning scheme, based on *symbolic false nearest neighbors* (SFNN), was reported by Kennel and Buhl [4]. These techniques rely on partitioning the phase space and may become cumbersome and extremely computation-intensive if the dimension of the phase space is large. Moreover, if the data set is noise-corrupted, then the symbolic false neighbors would rapidly grow in number and require a large symbol alphabet to capture the pertinent information. Therefore, symbolic sequences as representations of the system dynamics should be generated by alternative methods because phase-space partitioning might prove to be a difficult task.

Technical literature has suggested appropriate transformation of the signal before employing the partitioning method for symbol generation [30]. One such technique is the *analytic-signal-space partitioning* (ASSP) [32] that is based on the analytic signal which provides the additional phase information in the sensor data. The wavelet-transformed space partitioning (WTSP) [28] is well-suited for time-frequency analysis of non-stationary signals, noise attenuation, and reduction of spurious disturbances from the raw time series data without any significant loss of pertinent information [24][13]. Since SFNNP and ASSP may require additional preprocessing of the time series for denoising, this chapter has used WTSP for construction of symbolic representations of sensor data as explained below.

In wavelet-based partitioning, time series are first transformed into the wavelet domain, where wavelet coefficients are generated at different shifts and scales. The choice of the wavelet basis function and wavelet scales depends on the time-frequency characteristics of individual signals [13]. The wavelet transform of a function $f(t) \in \mathbb{H}$ is given by

$$F_{s,\tau} = \frac{1}{\sqrt{\alpha}} \int_{-\infty}^{\infty} f(t)\psi^*_{s,\tau}(t)dt, \tag{1.2}$$

where $s > 0$ is the scale, τ is the time shift, $\mathbb{H}$ is a Hilbert space, $\psi_{s,\tau}(t) = \psi(\frac{t-\tau}{s})$ and $\psi \in \mathbf{L}_2(\mathbb{R})$ is such that $\int_{-\infty}^{\infty} \psi(t)dt = 0$ and $||\psi||_2 = 1$.

Wavelet preprocessing of sensor data for symbol sequence generation helps in noise mitigation. Let $\tilde{f}$ be a noise-corrupted version of the original signal f expressed as:

$$\tilde{f} = f + k\ w, \tag{1.3}$$

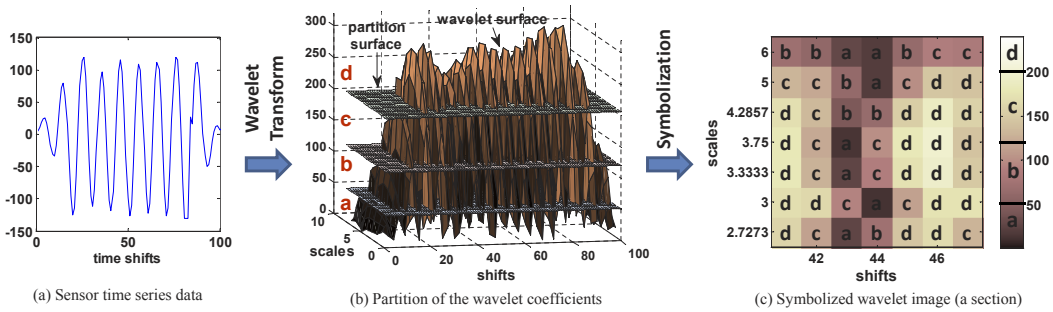
where w is additive white gaussian noise with zero mean and unit variance and k is the noise level. The noise part in Eq. (1.3) would be reduced if the scales over which coefficients are obtained are properly chosen.

For every wavelet, there exists a certain frequency called the center frequency F_c that has the maximum modulus in the Fourier transform of the wavelet. The pseudo-frequency f_p of the wavelet at a particular scale α is given by the following formula [1]:

$$f_p = \frac{F_c}{\alpha\ \Delta t}, \tag{1.4}$$

where Δt is the sampling interval. Then the scales can be calculated as follows:

$$\alpha^i = \frac{F_c}{f_p^i\ \Delta t} \tag{1.5}$$



(a) Sensor time series data (b) Partition of the wavelet coefficients (c) Symbolized wavelet image (a section)

FIGURE 1.1: Symbol image generation via wavelet transform of the sensor time series data and partition of the wavelet surface in ordinate direction

where $i = 1, 2, ...,$ and f_p^i are the frequencies that can be obtained by choosing the locally dominant frequencies in the Fourier transform. The maximum pseudo-frequency f_p^{max} should not exceed the Nyquist frequency [1]. Therefore, the sampling frequency f_s for acquisition of time series data should be selected at least twice the larger of the maximum pseudo-frequency f_p^{max} and the signal bandwidth B, i.e., $f_s \geq 2\max(f_p^{max}, B)$.

Figure 1.1 shows an illustrative example of transformation of the (one-dimensional) time series in Fig. 1.1(a) to a (two-dimensional) wavelet image in Fig. 1.1(b). The amplitudes of the wavelet coefficients over the scale-shift domain are plotted as a surface. Subsequently, symbolization of this wavelet surface leads to the formation of a symbolic image as shown in Fig. 1.1(c).

2.3 Symbolization of Wavelet Surface Profiles

This section presents partitioning of the wavelet surface profile in Fig. 1.1(b), which is generated by the coefficients over the two-dimensional scale-shift domain, for construction of the symbolic image in Fig. 1.1(c). The $x - y$ coordinates of the wavelet surface profiles denote the shifts and the scales respectively, and the z-coordinate (i.e., the surface height) denotes the pixel values of wavelet coefficients.

Definition 2.1 *(Wavelet Surface Profile) Let* $\mathcal{H} \triangleq \{(i, j) : i, j \in \mathbb{N},\ 1 \leq i \leq m,\ 1 \leq j \leq n\}$ *be the set of coordinates consisting of* $(m \times n)$ *pixels denoting the scale-shift data points. Let* $\mathcal{R}$ *denote the interval that spans the range of wavelet coefficient amplitudes. Then, a wavelet surface profile is defined as*

$$\mathcal{S} : \mathcal{H} \to \mathcal{R} \tag{1.6}$$

Definition 2.2 *(Symbolization) Given the symbol alphabet* Σ*, let the partitioning of the interval* $\mathcal{R}$ *be defined by a map* $P : \mathcal{R} \to \Sigma$*. Then, the symbolization of a wavelet surface profile is defined by a map* $\mathcal{S}_\Sigma \equiv P \circ \mathcal{S}$ *such that*

$$\mathcal{S}_\Sigma : \mathcal{H} \to \Sigma \tag{1.7}$$

that labels each pixel of the image to a symbol in Σ*.*

The wavelet surface profiles are partitioned such that the ordinates between the maximum and minimum of the coefficients along the z-axis are divided into regions by different planes parallel to the $x-y$ plane. For example, if the alphabet is chosen as $\Sigma = \{a, b, c, d\}$, i.e., $|\Sigma| = 4$, then three partitioning planes divide the ordinate (i.e., z-axis) of the surface profile into four mutually exclusive and exhaustive regions, as shown in Fig. 1.1 (b). These disjoint regions form a partition, where each region is labeled with one symbol from the alphabet Σ. If the intensity of a pixel is located in a particular region, then it is coded with the symbol associated with that region. As such, a symbol from the alphabet Σ is assigned to each pixel corresponding to the region where its intensity falls. Thus, the two-dimensional array of symbols, called *symbol image*, is generated from the wavelet surface profile, as shown in Fig. 1.1 (c).

The surface profiles are partitioned by using either the maximum entropy partitioning (MEP) or the uniform partitioning (UP) methods [28][13]. If the partitioning planes are separated by equal-sized intervals, then the partition is called the *uniform partitioning* (UP). Intuitively, it is more reasonable if the information-rich regions of a data set are partitioned finer and those with sparse information are partitioned coarser. To achieve this objective, the *maximum entropy partitioning* (MEP) method has been adopted in this chapter such that the entropy of the generated symbols is maximized. The procedure for selection of the alphabet size $|\Sigma|$, followed by generation of a MEP, has been reported in [13]. In general, the choice of alphabet size depends on specific data set and experiments. The partitioning of wavelet surface profiles to generate symbolic representations enables robust feature extraction, and symbolization also significantly reduces the memory requirements [13].

---

3 Construction of Probabilistic Finite-state Automata for Feature Extraction

This section presents the method for construction of a *probabilistic finite state automaton* (PFSA) for feature extraction from the symbol image generated from the wavelet surface profile.

3.1 Conversion from Symbol Image to State Image

For analysis of (one-dimensional) time series, a PFSA is constructed such that its states represent different combinations of blocks of symbols on the symbol sequence. The edges connecting these states represent the transition probabilities between these blocks [30][13]. Therefore, for analysis of (one dimensional) time series, the 'states' denote all possible symbol blocks (i.e., words) within a window of certain length. Let us now extend the notion of 'states' on a two-dimensional domain for analysis of wavelet surface profiles via construction of a '*state image*' from a '*symbol image*'.

Definition 3.1 *(State) Let $\mathcal{W} \subset \mathcal{H}$ be a two-dimensional window of size $(\ell \times \ell)$ that is*

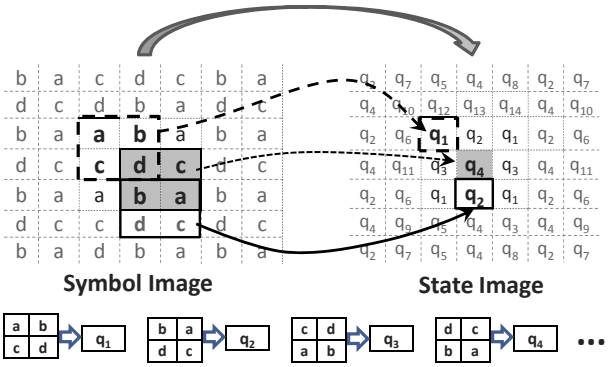


FIGURE 1.2: Conversion of the symbol image to the state image

denoted as $|\mathcal{W}| = \ell^2$. *Then, the state of a symbol block formed by the window* $\mathcal{W}$ *is defined as the configuration* $q{=}S_\Sigma(\mathcal{W})$.

Let the set of all possible states (i.e., two-dimensional words or blocks of symbols) in a window $\mathcal{W} \subset \mathcal{H}$ be denoted as $\mathcal{Q} \triangleq \{q_1, q_2, ..., q_{|\mathcal{Q}|}\}$, where $|\mathcal{Q}|$ is the number of (finitely many) states. Then, $|\mathcal{Q}|$ is bounded above as $|\mathcal{Q}| \leq |\Sigma|^{|\mathcal{W}|}$; the inequality is due to the fact that some of the states might have zero probability of occurrence. Let us denote $\mathcal{W}_{i,j} \subset \mathcal{H}$ to be the window where (i, j) represents the coordinates of the top-left corner pixel of the window. In this notation, $q_{i,j}{=}S_\Sigma(\mathcal{W}_{i,j})$ denotes the state at pixel $(i, j) \in \mathcal{H}$. Thus, every pixel $(i, j) \in \mathcal{H}$ corresponds to a particular state $q_{i,j} \in \mathcal{Q}$ on the image. Every pixel in the image $\mathcal{H}$ is mapped to a state , excluding the pixels that lie at the periphery depending on the window size. Figure 1.2 shows an illustrative example of the transformation of a *symbol image* to the *state image* based on a sliding window $\mathcal{W}$ of size (2×2). This concept of state formation facilitates capturing of long range dynamics (i.e., word to word interactions) on a symbol image.

In general, a large number of states would require a high computational capability and hence might not be feasible for real-time applications. The number of states, $|\mathcal{Q}|$, increases with the window size $|\mathcal{W}|$ and the alphabet size $|\Sigma|$. For example, if $\ell = 2$ and $|\Sigma| = 4$, then the total number of states are $|\mathcal{Q}| \leq |\Sigma|^{\ell^2} = 256$. Therefore, for computational efficiency, it is necessary to compress the state set $\mathcal{Q}$ to an effective reduced set $\mathcal{O} \triangleq \{o_1, o_2, ..., o_{|\mathcal{O}|}\}$ [13] that enables mapping of two or more different configurations in a window $\mathcal{W}$ to a single state. State compression must preserve sufficient information as needed for pattern classification, albeit possibly lossy coding of the wavelet surface profile.

In view of the above discussion, a probabilistic state compression method is employed, which chooses the m most probable symbols, from each state as a representation of that particular state. In this method, each state consisting of $\ell \times \ell$ symbols is compressed to a reduced state of length $m < \ell^2$ symbols by choosing the top m symbols that have the highest probability of occurrence arranged in descending order. If two symbols have the same probability of occurrence, then either symbol may be preferred with equal probability. This procedure reduces the state set $\mathcal{Q}$ to an effective set $\mathcal{O}$, where the total number of compressed states is given as: $|\mathcal{O}| = |\Sigma|^m$. For example, if $|\Sigma| = 4$, $|\mathcal{W}| = 4$ and $m = 2$,

then the state compression reduces the total number of states to $|\mathcal{O}| = |\Sigma|^m = 16$ instead of 256. This method of state compression is motivated from the renormalization methods in *Statistical Physics* that are useful in eliminating the irrelevant local information on lattice spin systems while still capturing the long range dynamics [14]. The choice of $|\Sigma|$, ℓ and m depends on specific applications and noise level as well as the available computational power, and is made by an appropriate tradeoff between robustness to noise and capability to detect small changes. For example, a large alphabet may be noise-sensitive while a small alphabet could miss the information of signal dynamics [13].

3.2 Construction of PFSA

A probabilistic finite state automaton (PFSA) is constructed such that the states of the PFSA are the elements of the compressed state set $\mathcal{O}$ and the edges are the transition probabilities between these states. Figure 1.3(a) shows an example of a typical PFSA with four states. The transition probabilities between states are defined as:

$$\wp(o_k|o_l) = \frac{N(o_l, o_k)}{\sum_{k'=1,2,\ldots,|\mathcal{O}|} N(o_l, o_{k'})} \quad \forall\ o_l, o_k \in \mathcal{O} \tag{1.8}$$

where $N(o_l, o_k)$ is the total count of events when o_k occurs adjacent to o_l in the direction of motion. The calculation of these transition probabilities follows the principle of sliding block code [23]. A transition from the state o_l to the state o_k occurs if o_k lies adjacent to o_l in the positive direction of motion. Subsequently, the counter moves to the right and to the bottom (row-wise) to cover the entire state image, and the transition probabilities $\wp(o_k|o_l)$, $\forall\ o_l, o_k \in \mathcal{O}$ are computed using Eqn. (1.8). Therefore, for every state on the state image, all state-to-state transitions are counted, as shown in Fig. 1.3(b). For example, the dotted box in the bottom-right corner contains three adjacent pairs, implying the transitions $o_1 \rightarrow o_2$, $o_1 \rightarrow o_3$, and $o_1 \rightarrow o_4$ and the corresponding counter of occurrences $N(o_1, o_2)$, $N(o_1, o_3)$ and $N(o_1, o_4)$, respectively, are increased by one. This procedure generates the stochastic state-transition probability matrix of the PFSA given as:

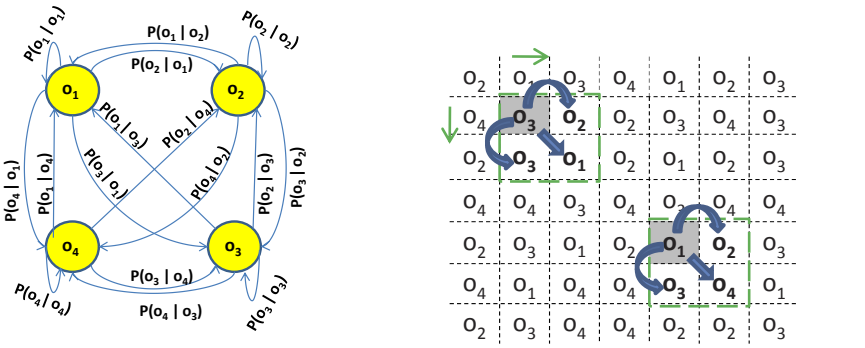
$$\Pi = \begin{bmatrix} \wp(o_1|o_1) & \ldots & \wp(o_{|\mathcal{O}|}|o_1) \\ \vdots & \ddots & \vdots \\ \wp(o_1|o_{|\mathcal{O}|}) & \ldots & \wp(o_{|\mathcal{O}|}|o_{|\mathcal{O}|}) \end{bmatrix} \tag{1.9}$$

where $\Pi \equiv [\pi_{jk}]$ with $\pi_{jk} = \wp(o_k|o_j)$. Note: $\pi_{jk} \geq 0\ \forall j, k \in \{1, 2, \ldots|\mathcal{O}|\}$ and $\sum_k \pi_{jk} = 1\ \forall j \in \{1, 2, \ldots|\mathcal{O}|\}$.

In order to extract a low-dimensional feature vector, the stationary state probability vector $\mathbf{p}$ is obtained as the left eigenvector corresponding to the (unique) unity eigenvalue of the (irreducible) stochastic transition matrix Π. The state probability vectors $\mathbf{p}$ serve as the '*feature vectors*' and are generated from different data sets from the corresponding state transition matrices. These feature vectors are also denoted as '*patterns*' in this chapter.

3.3 Summary of SDF for Feature Extraction

The major steps of SDF for feature extraction are summarized below:



(a) An example of a 4-state PFSA

(b) Feature extraction from the state image

FIGURE 1.3: An example of feature extraction from stage image by constructing a PFSA

- Acquisition of time series data from appropriate sensor(s) and signal conditioning as necessary;
- Wavelet transform of the time series data with appropriate scales to generate the wavelet surface profile;
- Partitioning of the wavelet surface profile and generation of the corresponding symbol image;
- Conversion from symbol image to state image via probabilistic state compression strategy;
- Construction of PFSA and computation of the state transition matrices that in turn generate the state probability vectors as the feature vectors (i.e., patterns).

The advantages of SDF for feature extraction and subsequent pattern classification are summarized below:

- Robustness to measurement noise and spurious signals;
- Adaptability to low-resolution sensing due to the coarse graining in space partitions [30];
- Capability for detection of small deviations because of sensitivity to signal distortion;
- Real-time execution on commercially available inexpensive platforms.

4 Pattern Classification Using SDF-based Features

Once the feature vectors are extracted in a low-dimensional space from the observed sensor time series, the next step is to classify these patterns into different categories based on the particular application. Technical literature abounds in diverse methods of pattern classification, such as divergence measure, k-nearest neighbor (k-NN) algorithm [7], support vector machine (SVM) [3], and artificial neural network (ANN) [16]. The main focus of this chapter is to develop and validate the tools of Symbolic Dynamic Filtering (SDF) for feature

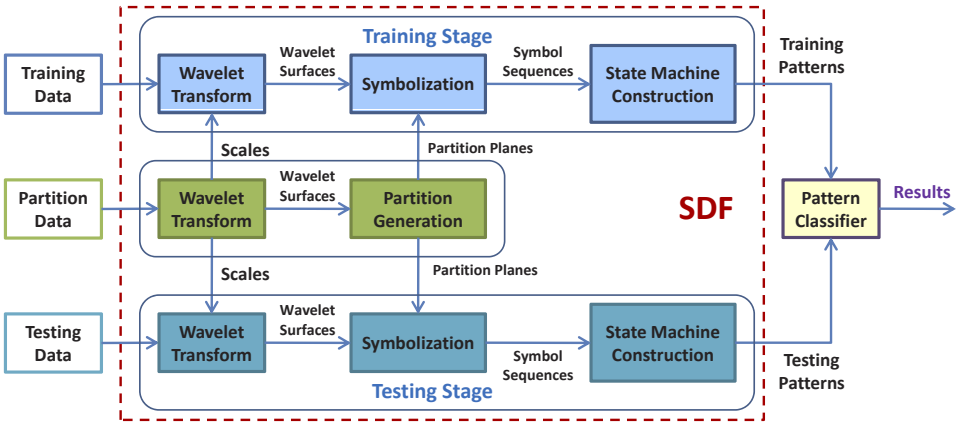


FIGURE 1.4: Flow chart of the proposed methodology

extraction from wavelet surface profiles generated from sensor time series data. Therefore, the SDF method for feature extraction is used in conjunction with the standard pattern classification algorithms, as described in the experimental validation sections.

Pattern classification using SDF-based features is posed as a two-stage problem, i.e., the training stage and the testing stage. The sensor time series data sets are divided into three groups: i) partition data, ii) training data, and iii) testing data. The partition data set is used to generate partition planes that are used in the training and the testing stages. The training data set is used to generate the training patterns of different classes for the pattern classifier. Multiple sets of training data are obtained from independent experiments for each class in order to provide a good statistical spread of patterns. Subsequently, the class labels of the testing patterns are generated from testing data in the testing stage. The partition data sets may be part of the training data sets, whereas the training data sets and the testing data sets must be mutually exclusive.

Figure 1.4 depicts the flow chart of the proposed algorithm that is constructed based on the theory of SDF. The partition data is wavelet-transformed with appropriate scales to convert the one-dimensional numeric time series data into the wavelet image. The corresponding wavelet surface is analyzed using the *maximum entropy principle* [28][13] to generate the partition planes that remain invariant for both the training and the testing stage. The scales used in the wavelet transform of the partitioning data also remain invariant during the wavelet transform of the training and the testing data. In the training stage, the wavelet surfaces are generated by transformation of the training data sets corresponding to different classes. These surfaces are symbolized using the partition planes to generate the symbol images. Subsequently, PFSAs are constructed based on the corresponding symbol images, and the training patterns (i.e., state probability vectors $\mathbf{p}$ or state transition matrices Π) are extracted from these PFSAs. Similar to the training stage, the PFSA and the associated pattern is generated for different data sets in the testing stage. These patterns are then classified into different classes using pattern classifier, such as SVM, k-NN and ANN.

Consider a classification problem of $|\mathcal{C}|$ classes, where $\mathcal{C}$ is the set of class labels. In the

training stage, feature vectors $\mathbf{p}_j^{\mathcal{C}_i},\ j = 1, 2, ...n_i$ are generated from the training data sets of class $\mathcal{C}_i$, where n_i is the number of samples in class $\mathcal{C}_i$. The same procedure is carried out for all other classes. In the testing stage, a testing feature vector $\mathbf{p}_{test}$ with unknown class labels is generated using SDF. Two examples of using the pattern classifiers with SDF are provided here. For k-NN algorithm, the estimated class label of a testing feature vector $\mathbf{p}_{test}$ is equal to the most frequent class among the k-nearest training features [7]. For SVM, a separating hyperplane/hypersurface is generated based on training feature vectors $(\mathbf{p}_j^{\mathcal{C}_i},\ j = 1, 2, ...n_i)$. The estimated class label of the testing feature vector $\mathbf{p}_{test}$ depends on which side of the hyperplane/hypersurface the testing feature vector falls [3].

---

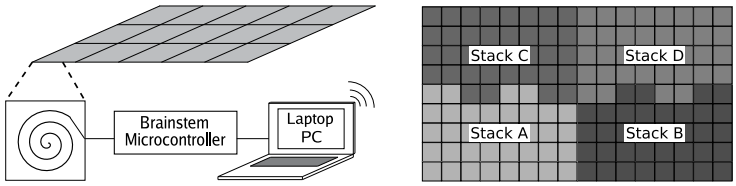
5 Validation I: Behavior Recognition of Mobile Robots in a Laboratory Environment

This section presents experimental validation of the proposed wavelet-based feature extraction method in a laboratory environment of networked robots. The objective here is to identify the robot type and the motion profile based on the sensor time series obtained from the pressure sensitive floor. These experiments are inspired from various real-life applications of pattern classification, such as (i) classification of enemy vehicles across the battlefield through analysis of seismic and acoustic time series data; and (ii) classification of human and animal movements through analysis of seismic time series.

5.1 Experimental Procedure for Behavior Identification of Mobile Robots

The experimental set up consists of a wireless network incorporating mobile robots, robot simulators, and distributed sensors as shown in Fig. 1.5 and Fig. 1.6. A major component of the experimental set up is the pressure sensitive floor that consists of distributed piezoelectric wires installed underneath the floor to serve as arrays of distributed pressure sensors. A coil of piezoelectric wire is placed under a $0.65m \times 0.65m$ square floor tile as shown in Fig. 1.6(a) such that the sensor generates an analog voltage due to pressure applied on it. This voltage is sensed by a *Brainstem*$^{\text{TM}}$ microcontroller using one of its 10-bit A/D channels thereby yielding sensor readings in the range of 0 to 1023. The sampling frequency of the pressure sensing device that captures the dynamics of robot motion is 10 Hz, while the maximum pseudo-frequency f_p^{max} is 4.44 Hz (see Subsection 2.2). A total of 144 sensors are placed in a 9×16 grid to cover the entire laboratory environment as shown in Fig. 1.6(b). The sensors are grouped into four quadrants, each being connected to a stack consisting of 8 networked *Brainstem* microcontrollers for data acquisition. The microcontrollers are, in turn, connected to two laptop computers running *Player* [11] server that collects the raw sensor data and distributes to any client over the wireless network for further processing.

FIGURE 1.5: The Robot Hardware: Pioneer 2AT (Left) and Segway RMP (Right)



(a) Sensor (b) Distribution of Sensors

FIGURE 1.6: Layout of the Distributed Pressure Sensors in the Laboratory Environment

TABLE 1.1: Parameters used for various types of motion

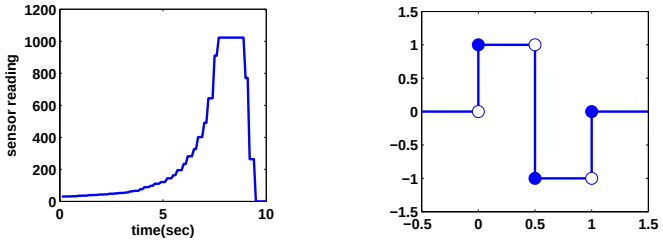
Motion Type	Parameter	Value
Circular	Diameter	4m
Square	Edge length	3m
Random	Uniform distribution	x-dir 1 to 7 m
		y-dir 1 to 4 m

Figure 1.5 shows a pair of Pioneer robots and a Segway RMP that have the following features:

- Pioneer 2AT is a four-wheeled robot that is equipped with a differential drive train system and has an approximate weight of 35 kg.
- Segway RMP is a two-wheeled robot (with inverted pendulum dynamics) that has a zero turn radius and has an approximate weight of 70 kg.

Since Pioneer is lighter than Segway and Pioneer's load on the floor is more evenly distributed, their statistics are dissimilar. Furthermore, since the kinematics and dynamics of the two types of robots are different, the texture of the respective pressure sensor signals are also different.

The objective is to identify the robot type and motion type from the time series data. The Segway RMP and Pioneer 2AT robots are commanded to execute three different motion trajectories, namely, *random motion*, *circular motion* and *square motion*. Table 1.1 lists the parameters for the three types of robot motion. In the presence of uncertainties (e.g., sensor noise and fluctuations in robot motion), a complete solution of the robot type and motion

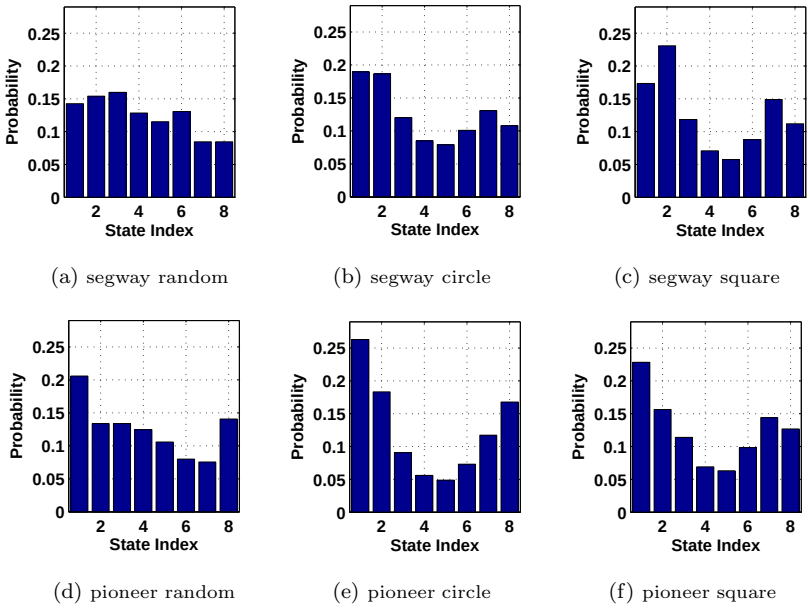


(a) Sensor Reading (b) Haar Wavelet

FIGURE 1.7: Example of Sensor Readings and Plot of Haar Wavelet

identification problem may not be possible in a deterministic setting because the patterns would not be identical for similar robots behaving similarly. Therefore, the problem is posed in the statistical setting, where a family of patterns is generated from multiple experiments conducted under identical operating conditions. The requirement is to generate a family of patterns for each class of robot behavior that needs to be recognized. Therefore, both Segway RMP and Pioneer 2AT robots were made to execute several cycles of each of the three different types of motion trajectories on the pressure sensitive floor of the laboratory environment. Each member of a family represents the pattern of a single experiment of one robot executing a particular motion profile. As a robot changes its type of motion from one (e.g., circular) to another (e.g., random), the pattern classification algorithm is capable of detecting this change after a (statistically quasi-stationary) steady state is reached. During the brief transient period, the analysis of pattern classification may not yield accurate results because the resulting time series may not be long enough to extract the features correctly.

Figure 1.7(a) shows an example of the sensor reading when the robot moves over it. The voltage generated by the piezoelectric pressure sensor gradually increases as the robot approaches the sensor, and discharge occurs in the sensor when the robot moves away from the sensor and hence the voltage resumes to be 0. The choice of mother wavelet depends on the shape of the sensor signal; the mother wavelet should match the shape of the sensor signal in order to capture the signature of the signal. Haar wavelet ($db1$), as shown in Fig. 1.7(b), is chosen to be the mother wavelet in this application. The sensor data collected by the 9×16 grid is stacked sequentially to generate a one-dimensional time series. For each motion trajectory consisting of several cycles, the time series data collected from the pressure sensors was divided into 40 to 50 data sets. The length of each data set is 3.0×10^5 data points, which corresponds to about three minutes of the experiment time. The data sets are randomly divided into half training and half testing. Among the training data, 10 sets are chosen to serve as the partitioning data sets as well.



(a) segway random (b) segway circle (c) segway square

(d) pioneer random (e) pioneer circle (f) pioneer square

FIGURE 1.8: Ensemble Mean of the State Probability Vectors (feature vectors)

5.2 Pattern Analysis for Behavior Identification of Mobile Robots

This subsection provides a description of the application of different pattern analysis methods to time series data of pressure sensors for classification of the robots and their motion types.

For feature extraction using SDF, each data set of a family (or class) is analyzed to generate the corresponding state probability vectors (i.e., patterns). Thus, the patterns $\mathbf{p}_j^{\mathcal{C}_i}$, $j = 1, 2, ..., n_i$, are generated for n_i samples in each class $\mathcal{C}_i$ corresponding to robot type and motion. Following the SDF procedure, each time-series data set is analyzed using $|\Sigma| = 8$, $\ell = 2$ and $m = 1$. Ensemble mean of pattern vectors for different motion profiles of Segway and Pioneer robots is shown in Fig. 1.8. It can be observed in Fig. 1.8 that the state probability vectors of Segway and Pioneer robots are quite distinct. Following Fig. 1.4, for each motion type, the state probability vectors $\mathbf{p}_j^{\mathcal{C}_i}$ were equally divided into training sets and testing sets.

In this application, the efficacy of SDF for feature extraction is evaluated by comparison with PCA. The time series data are transformed to the frequency domain for noise mitigation and then the standard PCA method is implemented to identify the eigen-directions of the transformed data and to obtain an orthogonal linear operator that projects the frequency-domain features onto a low-dimensional compressed-feature space. For the purpose of comparison, the dimension of this compressed feature space is chosen to be the same as that of the feature vectors obtained by SDF. In this application, the support vector machine (SVM), k-NN algorithm, radial basis Neural Network (rbfNN), and multilayer perceptron Neural Network (mlpNN) have been used as the pattern classifiers to identify

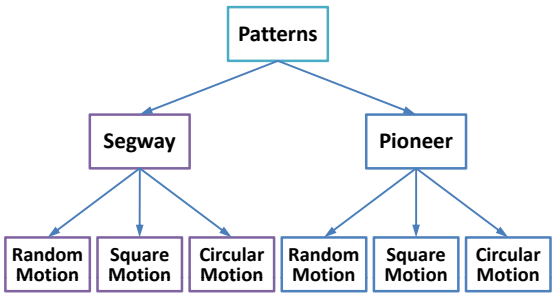


FIGURE 1.9: Tree Structure for Pattern Classification

different classes of feature vectors extracted by SDF and PCA. The pattern classifiers identify the type of the robot and its motion profile, based on the acquired statistical patterns. Since, in this pattern classification problem, there are two robots and each robot has three different types of motion profiles, it is natural to formulate this problem as a two-layer classification problem, where the robot type is identified in the first layer followed by identification of the motion type in the second layer. Thus, the above problem is formulated using a tree-structure classification as shown in Fig. 1.9.

5.3 Experimental Results for Behavior Identification of Mobile Robots

The performance comparison between SDF and PCA that are used in conjunction with different classifiers is presented in Table 1.2. The left part of Table 1.2 shows the results of robot type and robot motion classification using SDF for feature extraction, and the right part shows the corresponding results using PCA for feature extraction. As stated earlier, SVM, k-NN, rbfNN, and mlpNN have been used as pattern classifiers in both cases. The *polynomial* kernel is used in SVM [5], and a neighbor size of $k = 5$ is used in the k-NN classifier. The rbfNN uses one hidden layer and one output layer with a single neuron. Optimal training is obtained with 100 neurons in the hidden layer that uses a radial basis function, while the output layer uses a linear transfer function. The mlpNN utilizes a feed-forward back-propagation network that consists of one hidden layer with 50 neurons and an output layer with a single neuron; the tangent sigmoid function has been used in the hidden layers as a transfer function, while the output layer uses a linear function.

It is noted that since the tree structure is used for pattern classification, the motion recognition results are affected by the robot recognition results. For example, the samples that are incorrectly classified in the robot recognition stage will be incorrectly classified for motion recognition also. However, this particular aspect is application dependent and the tree structure for classification can be redesigned accordingly. The classification results are presented in Table 1.2 that show the accuracy percentage equal to $\left(\frac{\#\ \text{correct classifications}}{\#\ \text{total data sets}} \times 100\right)$. In the left part of Table 1.2, the combination of SDF with all four classifiers yield good accuracy in recognizing the robot type, namely, 100% for Segway and more than 94% for Pioneer. Although not explicitly shown in Table 1.2, but all four classifiers successfully identified the three types of motions of the Pioneer robot with 100% accuracy in the robot motion classification stage. The errors in the motion recognition, as seen in Table 1.2,

TABLE 1.2: Results of robot and motion classification

Feature Extraction using SDF						**Feature Extraction using PCA**					
Pattern	Robot Recognition		Motion Recognition			Pattern	Robot Recognition		Motion Recognition		
Classifier	Robot	Result	Motion	Result	Total	Classifier	Robot	Result	Motion	Result	Total
SVM	Segway	**100%** ($\frac{58}{58}$)	Random	92% ($\frac{23}{25}$)	**95%** ($\frac{55}{58}$)	SVM	Segway	**91%** ($\frac{53}{58}$)	Random	84% ($\frac{21}{25}$)	**81%** ($\frac{47}{58}$)
			Circular	92% ($\frac{12}{13}$)					Circular	77% ($\frac{10}{13}$)	
			Square	100% ($\frac{20}{20}$)					Square	80% ($\frac{16}{20}$)	
	Pioneer	**94%** ($\frac{61}{65}$)	Random	100% ($\frac{20}{20}$)	**94%** ($\frac{61}{65}$)		Pioneer	**100%** ($\frac{65}{65}$)	Random	20% ($\frac{4}{20}$)	**65%** ($\frac{42}{65}$)
			Circular	90% ($\frac{18}{20}$)					Circular	65% ($\frac{13}{20}$)	
			Square	92% ($\frac{23}{25}$)					Square	100% ($\frac{25}{25}$)	
k-NN	Segway	**100%** ($\frac{58}{58}$)	Random	88% ($\frac{22}{25}$)	**91%** ($\frac{53}{58}$)	k-NN	Segway	**100%** ($\frac{58}{58}$)	Random	84% ($\frac{21}{25}$)	**52%** ($\frac{30}{58}$)
			Circular	85% ($\frac{11}{13}$)					Circular	69% ($\frac{9}{13}$)	
			Square	100% ($\frac{20}{20}$)					Square	0% ($\frac{0}{20}$)	
	Pioneer	**94%** ($\frac{61}{65}$)	Random	95% ($\frac{19}{20}$)	**94%** ($\frac{61}{65}$)		Pioneer	**88%** ($\frac{57}{65}$)	Random	0% ($\frac{0}{20}$)	**51%** ($\frac{33}{65}$)
			Circular	100% ($\frac{20}{20}$)					Circular	55% ($\frac{11}{20}$)	
			Square	88% ($\frac{22}{25}$)					Square	88% ($\frac{22}{25}$)	
rbfNN	Segway	**100%** ($\frac{58}{58}$)	Random	84% ($\frac{21}{25}$)	**91%** ($\frac{53}{58}$)	rbfNN	Segway	**100%** ($\frac{58}{58}$)	Random	92% ($\frac{23}{25}$)	**72%** ($\frac{42}{58}$)
			Circular	92% ($\frac{12}{13}$)					Circular	31% ($\frac{4}{13}$)	
			Square	100% ($\frac{20}{20}$)					Square	75% ($\frac{15}{20}$)	
	Pioneer	**97%** ($\frac{63}{65}$)	Random	95% ($\frac{19}{20}$)	**97%** ($\frac{63}{65}$)		Pioneer	**95%** ($\frac{62}{65}$)	Random	0% ($\frac{0}{20}$)	**66%** ($\frac{43}{65}$)
			Circular	100% ($\frac{20}{20}$)					Circular	100% ($\frac{20}{20}$)	
			Square	96% ($\frac{24}{25}$)					Square	92% ($\frac{23}{25}$)	
mlpNN	Segway	**100%** ($\frac{58}{58}$)	Random	96% ($\frac{24}{25}$)	**98%** ($\frac{57}{58}$)	mlpNN	Segway	**100%** ($\frac{58}{58}$)	Random	96% ($\frac{24}{25}$)	**98%** ($\frac{57}{58}$)
			Circular	100% ($\frac{13}{13}$)					Circular	100% ($\frac{13}{13}$)	
			Square	100% ($\frac{20}{20}$)					Square	100% ($\frac{20}{20}$)	
	Pioneer	**100%** ($\frac{65}{65}$)	Random	100% ($\frac{20}{20}$)	**100%** ($\frac{65}{65}$)		Pioneer	**100%** ($\frac{65}{65}$)	Random	85% ($\frac{17}{20}$)	**92%** ($\frac{60}{65}$)
			Circular	100% ($\frac{20}{20}$)					Circular	95% ($\frac{19}{20}$)	
			Square	100% ($\frac{25}{25}$)					Square	96% ($\frac{24}{25}$)	

TABLE 1.3: Comparison of computational complexity of feature extraction methods

Method	Training Stage		Testing Stage	
	execution time	memory requirement	execution time	memory requirement
SDF	5.21 sec	65.2 MB	5.17 sec	64.9 MB
PCA	6.62 sec	233.85 MB	0.04 sec	37.5 MB

originate from the robot recognition stage. The success rate in recognizing Segway motion is slightly lower due to the following possible reasons: (i) complicated kinematics of the Segway robot, and (ii) the non-stationarity in the samples due to uncertainties in the laboratory environment (e.g., floor friction). It is expected that this accuracy would further improve if the number of stationary samples is increased. The right part of Table 1.2 shows that PCA yields slightly worse (but still comparable) results than SDF in robot recognition. However, SDF significantly outperforms PCA in motion recognition, because the feature vectors extracted by PCA lack class separability among different types of motions, which in turn yields poor motion recognition accuracy.

The proposed SDF-based method has a computational complexity of $O(N)$ for a given algebraic structure of the PFSA, with a leading constant that is proportional to the number of scales used in the wavelet transform [25]. A comparison of the computational complexity of SDF and PCA is presented in Table 1.3 in terms of execution time and memory requirements for processing each data set. For the data set consisting of 3.0×10^5 data points, which is about 3.5 minutes of the experimentation time, it takes an average of 5.21 seconds for SDF and 6.62 seconds for PCA to process each data set in the training stage, respectively. The memory requirement is 65.2 MB for SDF, and 233.9 MB for PCA. PCA takes longer execution time and consumes more memory in the training stage because it needs to calculate the covariance matrix using all training data sets. In the testing stage, the execution time and memory requirement for SDF are almost the same as those in the training stage, while the PCA requires less time and memory than those in the training stage. Both feature extraction methods have real-time implementation capability since the execution time in the testing stage is much less than the experiment time spent for collecting each data set. The rationale for SDF taking longer time than PCA in the testing stage is that the SDF-based method involves wavelet transformation and PFSA construction from the two-dimensional wavelet image in both training and testing stages, while the PCA-based method only involves Fourier transform and finding the projection of the testing data set using the projection matrix that is already constructed in the training stage; this is a price paid for the superior performance and robustness achieved in SDF-based feature extraction (see Table 1.2). It is anticipated that the PCA-based method will be relatively slower if the raw time-series is (more effectively) de-noised by wavelet transform instead of Fourier transform. In these experiments, the data analysis was performed on a 2.83 GHz Quad Core CPU desktop computer with 8.0 GB of RAM.

6 Validation II: Target Detection and Classification Using Seismic and PIR Sensors

The objective of this application is to detect and classify different targets (e.g., humans, vehicles, and animals led by human), where seismic and PIR sensors are used to capture the characteristic signatures. For example, in the movement of a human or an animal across the ground, oscillatory motions of the body appendages provide the respective characteristic signatures.

The seismic and PIR sensor data, used in this analysis, were collected on multiple days 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). During multiple field tests, sensor data were collected for several scenarios that consisted of targets walking along an approximately 150 meters long trail, and returning along the same trail to the starting point. Figure 1.10 illustrates a typical data collection scenario.

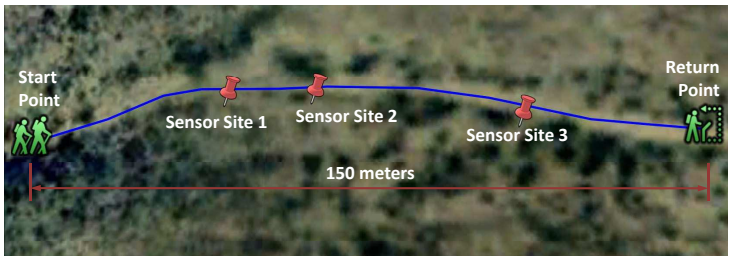


FIGURE 1.10: Illustration of the test scenario with three sensor sites

(a) Human (b) Vehicle (c) Animal led by human

FIGURE 1.11: Examples of test scenarios with different targets

The targets consisted of (male and female) humans, animals (e.g., donkeys, mules, and horses), and all-terrain vehicles (ATVs). The humans walked alone and in groups with and without backpacks; the animals were led by their human handlers (simply denoted as "animal" in the sequel) and they made runs with and without payloads; and ATVs moved at different speeds (e.g., 5 mph and 10 mph). Examples of the test scenarios with different targets are shown in Fig. 1.11. There were three sensor sites, each equipped with seismic and PIR sensors. The seismic sensors (geophones) were buried approximately 15 *cm* deep

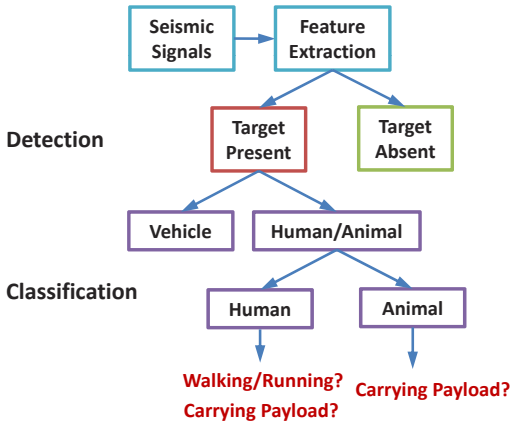


FIGURE 1.12: Tree structure formulation of the detection & classification problem

underneath the soil surface, and the PIR sensors were collocated with the respective seismic sensors. All targets passed by the sensor sites at a distance of approximately 5 m. Signals from both sensors were acquired at a sampling frequency of 10 kHz.

The tree structure in Fig. 1.12 shows how the detection and classification problem is formulated. In the detection stage, the pattern classifier detects the presence of a moving target against the null hypothesis of no target present; in the classification stage, the pattern classifiers discriminate among different targets, and subsequently identify the movement type and/or payload of the targets. While the detection system should be robust to reduce the false alarm rates, the classification system must be sufficiently sensitive to discriminate among different types of targets with high fidelity. In this context, feature extraction plays an important role in target detection and classification because the performance of classifiers largely depends on the quality of the extracted features.

In the classification stage, there are multiple classes (i.e., humans, animals, and vehicles); and the signature of the vehicles is distinct from those of the other two classes. Therefore, this problem is formulated into a two-layer classification procedure. A binary classification is performed to detect the presence of a target and then to identify whether the target is a vehicle or a human/animal. Upon recognizing the target as a human/animal, another binary classification is performed to determine its specific class. More information could be derived upon recognition of the target type. For example, if the target is recognized as a human, then further binary classifications are performed to identify if the human is running or walking, and if the human is carrying a payload or not.

Field data were collected in the scenario illustrated in Fig. 1.10. Multiple experiments were made to collect data sets of all three classes, i.e., human, vehicle and animal. The data were collected over three days at different sites. Table 1.4 shows the number of runs of each class.

Each data set, acquired at a sampling frequency of 10 kHz, has $100,000$ data points that correspond to 10 seconds of the experimentation time. In order to test the capability of the proposed algorithm for target detection, a similar data set was collected with no target present. The problem of target detection is then formulated as a binary pattern classification, where no target present corresponds to one class, and target present (i.e.,

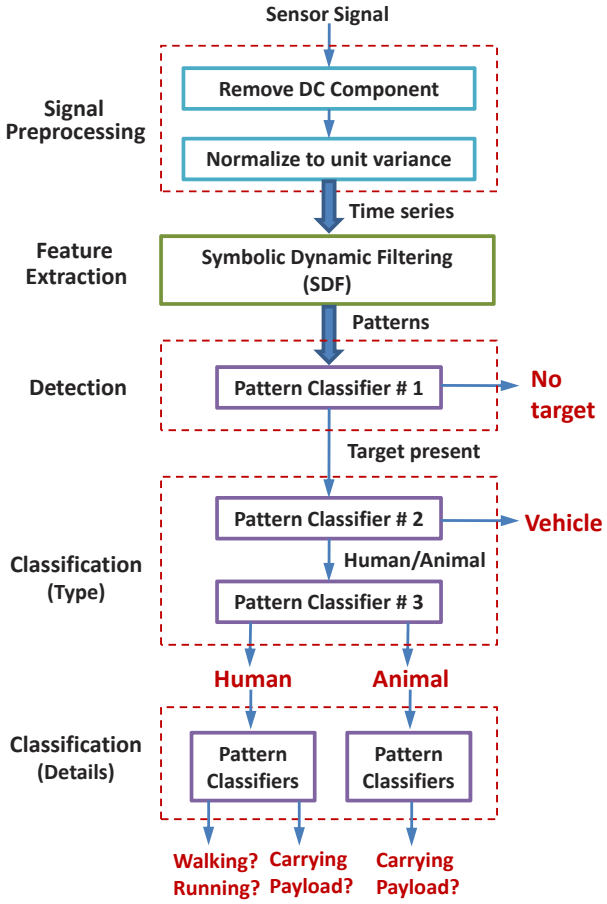


FIGURE 1.13: Flow chart of the problem of target detection and classification

TABLE 1.4: The number of feature vectors for each target class

	Day 1	Day 2	Day 3	Total
No target	50	36	32	118
Vehicle	0	8	0	8
Human	30	22	14	66
Animal	20	6	18	44

human, vehicle or animal) corresponds to the other class. The data sets, collected by the channel of seismic sensors that are orthogonal to the ground surface and the PIR sensors that are collocated with the seismic sensors, are used for target detection and classification. For computational efficiency, the data were downsampled by a factor of 10 with no apparent loss of information.

Figure 1.13 depicts the flow chart of the proposed detection and classification algorithm that is constructed based on the theories of symbolic dynamic filtering (SDF) and support

vector machines (SVM) [3]. The proposed algorithm consists of four main steps: signal preprocessing, feature extraction, target detection, and target classification, as shown in Fig. 1.13.

In the signal preprocessing step, the DC component of the seismic signal is eliminated, and the resulting zero-mean signal is normalized to unit variance. The amplitude of seismic signal of an animal with a heavy payload walking far away is possibly similar to that of a pedestrian passing by at a close distance due to the fact that the signal-to-noise ratio (SNR) decreases with the distance between the sensor and the target. The normalization of all signals to unit variance makes the pattern classifier independent of the signal amplitude and any discrimination should be solely texture-dependent. For PIR signals, only the DC component is removed and the normalization is not performed because the range of PIR signals do not change during the field test.

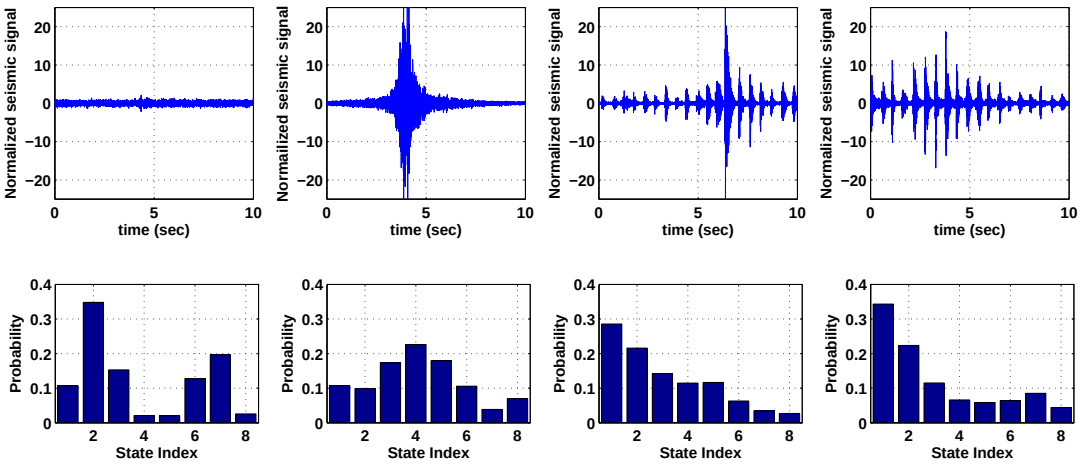
In the feature extraction step, SDF captures the signatures of the preprocessed sensor time-series for representation as low-dimensional feature vectors. Based on the spectral analysis of the ensemble of seismic data at hand, a series of pseudo-frequencies from the 1-20 Hz bands have been chosen to generate the scales for wavelet transform, because these bands contain a very large part of the footstep energy. Similarly, a series of pseudo-frequencies from the 0.2-2.0 Hz bands have been chosen for PIR signals to generate the scales. Upon generation of the scales, continuous wavelet transforms (CWT) are performed with an appropriate wavelet basis function on the seismic and PIR signals. The wavelet basis $db7$ is used for seismic signals because it matches the impulse shape of seismic signals, and $db1$ is used for the PIR case because PIR signals' shape is close to that of square waves. A maximum-entropy wavelet surface partitioning is then performed. Selection of the alphabet size $|\Sigma|$ depends on the characteristics of the signal; while a small alphabet is robust against noise and environmental variations, a large alphabet has more discriminant power for identifying different objects. The same alphabet is used for both target detection and classification. The issues of optimization of the alphabet size and data set partitioning are not addressed in this chapter. Subsequently, the extracted low-dimensional patterns are used for target detection and classification. One pattern is generated from each experiment, and the training patterns are used to generate the separating hyperplane in SVM.

6.1 Performance Assessment using Seismic Data

This subsection presents the classification results using the patterns extracted from seismic signals using SDF. The leave-one-out cross-validation method [3] has been used in the performance assessment of seismic data. Since the seismic sensors are not site-independent, they require partial information of the test site, which is obtained from the training set in the cross-validation. Results of target detection and classification, movement type and target payload identification are reported in this section.

6.1.1 Target Detection and Classification

Figure 1.14 shows the normalized seismic sensor signals and the corresponding feature vectors extracted by SDF of the three classes of targets and the no target case. It is observed



(a) No target (b) Vehicle (c) Human (d) Animal

FIGURE 1.14: Examples of seismic sensor measurements (top) and the corresponding feature vectors extracted by SDF of the four classes (bottom)

TABLE 1.5: Confusion matrices of leave-one-out cross-validation using SDF and kurtosis

SDF	No target	Vehicle	Human	Animal
No target	114	1	1	2
Vehicle	0	7	1	0
Human	3	0	61	2
Animal	0	0	1	43

Kurtosis	No target	Vehicle	Human	Animal
No target	102	0	5	11
Vehicle	0	0	7	1
Human	1	0	47	18
Animal	1	0	30	13

that the feature vectors are quite different among the no target, vehicle, and human/animal cases. The feature vectors of human and animal are somewhat similar and yet still distinguishable. In the feature vector plots in Fig. 1.14, the states with small index number corresponds to the wavelet coefficients with large values, and vice versa.

For the purpose of comparative evaluation, kurtosis analysis [33] has been used for target detection and classification as a benchmark tool of footstep detection. Kurtosis analysis is useful for footstep detection because the kurtosis value is much higher in the presence of

TABLE 1.6: Comparison of detection and classification accuracy using SDF and kurtosis

	Detection	Classification	
		Vehicle vs. Others	Human vs. Animal
SDF	97.0%	99.1%	97.2%
Kurtosis	92.4%	93.1%	55.6%

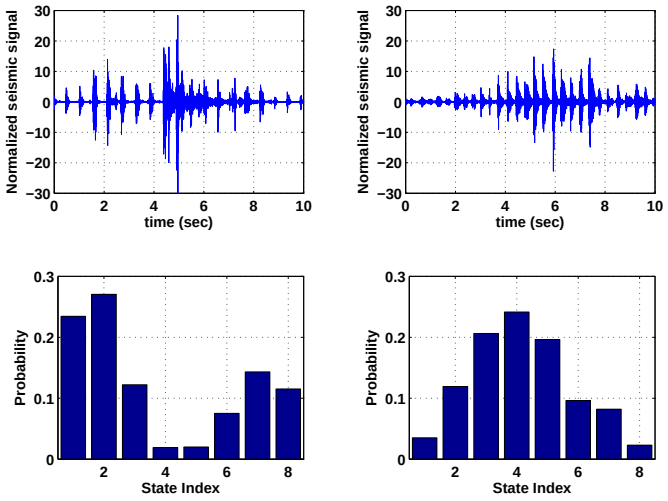
impulse events (i.e., target present) than in the absence of a target [33]. The results of SDF and kurtosis analysis are shown in Table 1.5. The shaded area in Table 1.5 represents the confusion matrices of target classification. The detection and classification accuracy is summarized in Table 1.6. Although the performance of kurtosis analysis is slightly inferior, it is comparable to that of SDF for target detection and vehicle classification; however, SDF significantly outperforms kurtosis analysis for distinguishing humans from animals.

The execution of the Matlab code takes 2.27 seconds and 43.73 MB of memory for SDF and SVM on a desktop computer to process a data set of $10,000$ points and perform pattern classification with the following parameters: alphabet size $|\Sigma| = 8$, number of scales $|\alpha| = 4$, window size $\ell \times \ell = 2 \times 2$, number of most probable symbol $m = 1$, and *quadratic* kernel for SVM. Pattern classification consumes about 80% of the total execution time because by using leave-one-out cross-validation, the pattern classifier need to be trained with all the remaining patterns (e.g., 235 in the detection stage). The choice of *quadratic* kernel in SVM improves the performance of the classifier; however, it also increases the computation time in training the classifier. It is expected that the execution time and memory requirements will be reduced significantly if fewer training patterns are used.

6.1.2 Movement Type Identification

Upon recognition of human, more information can be derived by performing another binary classification to identify whether the human is running or walking. The physical explanations are: i) the cadence (i.e., interval between events) of human walking is usually larger than the cadence of human running; ii) the impact of running on the ground is much stronger than that of walking, and it takes longer for the oscillation to decay. Figure 1.15 shows the seismic signal and corresponding feature vectors of human walking and running. The feature vectors of human walking and running are very different from each other, which is a clear indication that the SDF-based feature extraction method is able to capture these features (cadence and impact). It is noted that the feature vectors shown in Fig. 1.15 are different from those in Fig. 1.14 because different partitions are used in the target classification and movement type identification stages.

Ideally, the identification of movement type should be performed based on the results of human classification. However, in order to assess the performance of SDF in this particular application, a binary classification between human walking and human running is directly performed, and the result is shown in Table 1.7. It is seen in Table 1.7 the proposed feature extraction algorithm and SVM are able to identify the human movement type with an accuracy of 90.9%.



(a) Walking (b) Running

FIGURE 1.15: Examples of seismic sensor measurements (top) and the corresponding feature vectors extracted by SDF (bottom) for human walking and running

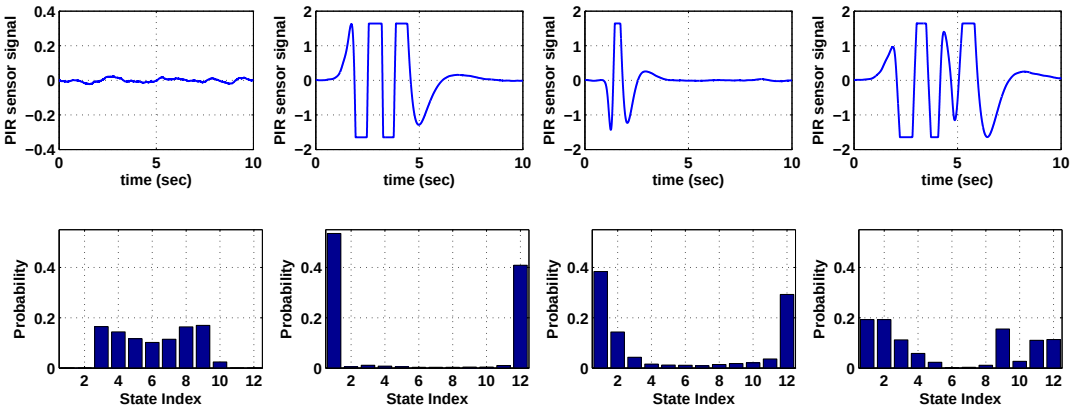
TABLE 1.7: Confusion matrix of leave-one-out cross-validation for movement type identification

	Human Walking	Human Running
Human Walking	47	1
Human Running	5	13

6.2 Performance Assessment using PIR Data

PIR sensors are widely used for motion detection. In most applications, the signals from PIR sensors are used as discrete variables (i.e., on or off). This may work for target detection, but will not work well for target classification because the time-frequency information is lost in the discretization. In this chapter, the PIR signals are considered to be continuous signals, and continuous wavelet transform (CWT) is used to reveal the distinction among different types of targets in the time-frequency domain. Since the PIR sensor does not emit an infrared beam but merely passively accepts incoming infrared radiation, it is less sensitive to environmental variations (i.e., variation in test sites) than the seismic sensor. A three-way cross-validation [3] is used for the performance assessment of PIR data. The data are divided into three sets by date (i.e., Day 1, Day 2 and Day 3) and three different sets of experiments are performed:

1. Training: Day 1 + Day 2; Testing: Day 3
2. Training: Day 1 + Day 3; Testing: Day 2
3. Training: Day 2 + Day 3; Testing: Day 1



(a) No Target (b) Human Walking (c) Human Running (d) Animal

FIGURE 1.16: Examples of PIR sensor measurements (top) and the corresponding feature vectors extracted by SDF (bottom) of the four classes

Training and testing on feature vectors from different days is very meaningful in practice. In each run of the cross-validation, no prior information is assumed for the testing site or the testing data. The classifiers' capability to generalize to an independent data set is thoroughly tested in the three-way cross-validation. In this section, four types of targets are considered, namely, no target, human walking, human running, animal led by human. Following Fig. 1.13, the following cases are tested:

1. Detection of target presence against target absence;
2. Classification of target type, i.e., Human vs. Animal;
3. Classification of target movement type (i.e., walking vs. running) upon recognition of the target as human.

Figure 1.16 shows the PIR sensor measurements (top) and the corresponding feature vectors extracted by SDF (bottom) of the four classes. For the no target case, the PIR signal fluctuates around zero and no information is embedded in the wavelet coefficients, thus the states in the middle (i.e., states 3-10) are occupied; whereas for the target present cases, the PIR sensors are excited by the presence of the targets, so states 1-2 and 11-12 that correspond to the crests and troughs in the PIR signals are more populated than other states.

The following parameters are used in SDF and SVM for processing the PIR signals: alphabet size $|\Sigma| = 12$, number of scales $|\alpha| = 3$, window size $\ell \times \ell = 2 \times 2$, number of most probable symbol $m = 1$, and *quadratic* kernel for SVM. The execution of SDF and SVM takes 1.13 seconds and 39.83 MB of memory on a desktop computer to process a data set of 1×10^4 points, which is a clear indication of the real-time implementation capability onboard UGS systems.

Table 1.8 shows the confusion matrix of the three-way cross-validation results using PIR sensors. The shaded area represents the target classification stage. It is seen in Table 1.8

TABLE 1.8: Confusion matrix of the three-way cross-validation

		No target	Human		Animal
			Walking	Running	
No target		110	0	0	0
Human	Walking	1	33	7	7
	Running	0	5	13	0
Animal		0	2	0	42

that the proposed feature extraction algorithm works very well with the PIR sensor; the target detection accuracy is 99.5%, the human/animal classification accuracy is 91.7%, and the human movement type classification accuracy is 79.3%. Leave-one-out cross-validation usually underestimates the error rate in generalization because more training samples are available; it is expected that the classification accuracy will further improve for the PIR signals if leave-one-out cross-validation is used.

7 Summary, Conclusions and Future work

This chapter addresses feature extraction from sensor time-series data for situation awareness in distributed sensor networks. While wavelet transformation of time series has been widely used for feature extraction owing to their time-frequency localization properties, the work reported here presents a symbolic dynamics-based method for feature extraction from wavelet images of sensor time series in the (two-dimensional) *scale-shift* space. In this regard, symbolic dynamics-based models (e.g., probabilistic finite state automata (PFSA)) are constructed from wavelet images as information-rich representations of the underlying dynamics, embedded in the sensor time series. Subsequently, low-dimensional feature vectors are generated from the associated Perron-Frobenius operators (i.e., the state-transition probability matrices) of the PFSA. These feature vectors facilitate *in-situ* pattern classification for decision-making in diverse applications. The proposed method has been experimentally validated for two different applications: (i) identification of mobile robots and their motion profiles in a laboratory environment, and (ii) target detection & classification from the field data of unattended ground sensors (UGS).

The proposed SDF-based feature extraction and pattern classification methodology is executable in real time on commercially available computational platforms. A distinct advantage of this method is that the low-dimensional feature vectors, generated from sensor time series in real time, can be communicated as short packets over a limited-bandwidth wireless sensor network with limited-memory nodes.

Further theoretical and experimental research is recommended in the following areas:

1. Exploration of other wavelet transform techniques for wavelet image generation;

2. Optimization of the partitioning scheme for symbolization of the wavelet images;

3. Experimental validation in other applications.

Acknowledgement

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. Office of Naval Research under Grant No. N00014-09-1-0688. 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.

Bibliography

[1] P. Abry. Ondelettes et turbulence, multiŕesolutions, algorithmes de décomposition, invariance déchelles. *Diderot Editeur, Paris, France*, 1997.

[2] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases*, pages 359–370, 1994.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, USA, 2006.

[4] M. Buhl and M.B. Kennel. Statistically relaxing to generating partitions for observed time-series data. *Physical Review E*, 71(4):046213, 2005.

[5] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. SVM and kernel methods Matlab toolbox. Perception Systèmes et Information, INSA de Rouen, Rouen, France, 2005.

[6] T. Chau and A. K.C. Wong. Pattern discovery by residual analysis and recursive partitioning. *IEEE Transactions on Knowledge and Data Engineering*, 11(6):833–852, 1999.

[7] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. on Information Theory*, 13(1):21–27, 1967.

[8] C. S. Daw, C. E. A. Finney, and E. R. Tracy. A review of symbolic analysis of experimental data. *Review of Scientific Instruments*, 74(2):915–930, 2003.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, New York, NY, USA, 2nd edition edition, 2001.

[10] K. Fukunaga. *Statistical Pattern Recognition, 2nd Edition*. Academic Press, Boston, MA, USA, 1990.

[11] B. Gerkey, R. Vaughan, and A. Howard. The Player/Stage project: Tools for multi-robot and distributed sensor systems. *Proceedings of the International Conference on Advanced Robotics, Coimbra, Portugal, June 30 - July 3, 2003*, pages 317–323, 2003.

[12] F. Gullo, G. Ponti, A. Tagarelli, and S. Greco. A time series representation model for accurate and fast similarity detection. *Pattern Recognition*, 42(7):2998–3014, 2009.

[13] S. Gupta and A. Ray. Symbolic dynamic filtering for data-driven pattern recognition. In E. A. Zoeller, editor, *Pattern Recognition: Theory and Application*, chapter 2, pages 17–71. Nova Science Publishers, Hauppage, NY, 2007.

[14] S. Gupta and A. Ray. Statistical mechanics of complex systems for pattern identification. *Journal of Statistical Physics*, 134(2):337–364, 2009.

[15] S. Gupta, A. Ray, and E. Keller. Symbolic time series analysis of ultrasonic data for early detection of fatigue damage. *Mechanical Systems and Signal Processing*, 21:866–884, 2007.

[16] S. S. Haykin. *Neural networks and learning machines*. Prentice Hall, New York, NY, USA, 3 edition, 2009.

[17] X. Jin, S. Gupta, K. Mukherjee, and A. Ray. Wavelet-based feature extraction using probabilistic finite state automata for pattern classification. *Pattern Recognition*, 44(7):1343–1356, July 2011.

[18] Y. Kakizawa, R.H. Shumway, and N. Taniguchi. Discrimination and clustering for multivariate time series. *J. Amer. Stat. Assoc.*, 93(441):328–340, 1999.

[19] K. Keller and H. Lauffer. Symbolic analysis of high-dimensional time series. *Int. J. Bifurcation Chaos*, 13(9):2657–2668, 2003.

[20] O. R. Lautour and P. Omenzetter. Damage classification and estimation in experimental structures using time series analysis and pattern recognition. *Mechanical Systems and Signal Processing*, 24:1556–1569, 2010.

[21] T.W. Lee. *Independent component analysis: Theory and applications*. Kluwer Academic Publishers, Boston, USA, 1998.

[22] T. W. Liao. Clustering of time series data–a survey. *Pattern Recognition*, 38:1857–1874, 2005.

[23] D. Lind and M. Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, United Kingdom, 1995.

[24] S. G. Mallat. *A wavelet tour of signal processing : the sparse way*. Academic Press, 3 edition, 2009.

[25] Arrate Muñoz, Raphaël Ertlé, and Michael Unser. Continuous wavelet transform with arbitrary scales and o(n) complexity. *Signal Processing*, 82(5):749 – 757, 2002.

[26] D. B. Percival and A. T. Walden. *Wavelet Methods for Time Series Analysis*. Cambridge University Press, Cambridge, UK, 2000.

[27] S. Pittner and S. V. Kamarthi. Feature extraction from wavelet coefficient for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(1):83–88, 1999.

[28] V. Rajagopalan and A. Ray. Symbolic time series analysis via wavelet-based partitioning. *Signal Processing*, 86(11):3309–3320, Nov 2006.

[29] C. Rao, A. Ray, S. Sarkar, and M. Yasar. Review and comparative evaluation of symbolic dynamic filtering for detection of anomaly patterns. *Signal, Image, Video Processing*, 3:101–114, 2009.

[30] A. Ray. Symbolic dynamic analysis of complex systems for anomaly detection. *Signal Processing*, 84(7):1115–1130, 2004.

[31] R. Rosipal, M. Girolami, and L. Trejo. Kernel PCA feature extraction of event-related potentials for human signal detection performance. *Proc. Int. Conf. Artificial Neural Networks Medicine Biol.*, pages 321–326, 2000.

[32] A. Subbu and A. Ray. Space partitioning via Hilbert transform for symbolic time series analysis. *Applied Physics Letters*, 92(8):084107–1 to 084107–3, 2008.

[33] G.P. Succi, D. Clapp, R. Gampert, and G. Prado. Footstep detection and tracking. In *Unattended Ground Sensor Technologies and Applications III*, volume 4393, pages 22–29. SPIE, 2001.

[34] C. J. Veenman, M. J. T. Reinders, E. M. Bolt, and E. Baker. A maximum variance cluster algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1273–1280, 2002.

[35] K. P. Zhu, Y. S. Wong, and G. S Hong. Wavelet analysis of sensor signals for tool condition monitoring: A review and some new results. *International Journal of Machine Tools and Manufacture*, 49:537–553, 2009.

[36] Walter Zucchini and Iain L. MacDonald. *Hidden Markov Models for Time Series: An Introduction*. CRC Press, Boca raton, FL, USA, 2009.