Pers Ubiquit Comput (2010) 14:645–662
DOI 10.1007/s00779-010-0293-9

ORIGINAL ARTICLE

Preprocessing techniques for context recognition from accelerometer data

Davide Figo · Pedro C. Diniz · Diogo R. Ferreira · João M. P. Cardoso

Received: 1 February 2010 / Accepted: 2 March 2010 / Published online: 30 March 2010

Abstract The ubiquity of communication devices such as smartphones has led to the emergence of context-aware services that are able to respond to specific user activities or contexts. These services allow communication providers to develop new, added-value services for a wide range of applications such as social networking, elderly care and near-emergency early warning systems. At the core of these services is the ability to detect specific physical settings or the context a user is in, using either internal or external sensors. For example, using built-in accelerometers, it is possible to determine whether a user is walking or running at a specific time of day. By correlating this knowledge with GPS data, it is possible to provide specific information services to users with similar daily routines. This article presents a survey of the techniques for extracting this activity information from raw accelerometer data. The techniques that can be implemented in mobile devices range from classical signal processing techniques such as FFT to contemporary string-based methods. We present experimental results to compare and evaluate the accuracy of the various techniques using real data sets collected from daily activities.

Keywords Activity detection · Context-aware applications · Mobile computing · Sensor data

D. Figo · P. C. Diniz · D. R. Ferreira (✉)
IST, Technical University of Lisbon, Avenida Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Portugal
e-mail: diogo.ferreira@ist.utl.pt

J. M. P. Cardoso
Faculty of Engineering, University of Porto (FEUP), Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

1 Introduction

Mobile communication devices as the ubiquitous cellular phones, and more recently smartphones, have exploded in number and computing capabilities in recent years. Rather than supporting only voice communications, contemporary devices have sophisticated internal hardware architectures and possibly also an extended range of functions such a GPS location, e-mail, organizer and synchronization with external, often centralized services. Advanced units can even be equipped with a wide range of internal sensors including three-dimensional accelerometers as well as the ability to interface with external web-based sensor services such as traffic information.

Using sensor data, mobile devices can provide users with an wide range of added-value services. For example, by analyzing accelerometer data, a device can understand that the user is performing some physical activity such as walking or running. This knowledge can be gathered over a period of time, say a week or even a month, to recognize trends or daily habits. Knowing that at a specific time of day a user might be jogging at a specific location, it is possible to send a message advertising a refreshment booth or advertising a specific brand of running shoes.[1] Potential services are not limited to individual end-users. By correlating daily activity patterns, communication providers can also offer services to communities of users with similar weekly habits, thus promoting and enhancing the use of the underlying communication infrastructure.

[1] For example, the Nike+ project (http://www.nikerunning.nike.com/nikeplus/) collects data captured by an accelerometer located on the user's running shoes. The user can then upload the data to a personal computer and use an application that analyzes the running habits and physical effort to recommend training regimes.

The services are not restricted to leisure-oriented activities. Understanding the physical situation of a user can also be used for early warning healthcare-related applications. Recognizing that an elderly person has fallen at his/her home and has not moved in the last 30 s indicates a potential emergency situation for which a relative or a local emergency unit should be alerted. In civil protection scenarios, knowing the location and the state of readiness of the elements of an emergency response team could dramatically reduce dispatching time and thus the response lag.

A key enabler for these context-aware services that providers can now offer lies the ability of mobile devices to acquire, manage, process, and obtain useful information from raw sensor data. From these data, devices must be able to accurately discover the characteristics or *features* of the signal coming from a given sensor. Sensors do generate a high volume of raw data possibly contaminated with environment noise that needs to be filtered out. In addition, the device must generate a very low number of incorrectly recognized features to improve the accuracy of subsequent information processing stages where features are analyzed and organized into user context patterns.

The inclusion of sensors for context discovery in mobile devices is commonly organized as part of a software stack with a general architecture similar to the one depicted in Fig. 1. At the lowest levels of the stack, we have preprocessing phases where the device attempts to extract a set of basic features from the sensor signal. These features include specific short-term contexts or *states* such *absence of light, quick movement* or more sophisticated contexts such as *running*. It is based on these states that the next layer—the base-level classifier—will determine higher-level user contexts such as *activities* like jogging or exercising. Finally, a layer of application code and scripting will use the context history to infer daily or weekly activities or routines.

In contrast with other sensors, which provide an instant value that can be used directly for context inference, the signal coming from an accelerometer may require the use of a fairly complex preprocessing stage in order to characterize the physical activity of the user within a certain time frame. Given the significance of this problem, a large number of techniques have been developed to address it. In this article, we survey the most representative domain approaches and techniques, including spectral analysis techniques such as Fast Fourier Transforms (FFT), statistics-based metrics and even string-matching approaches. These approaches vary widely in their context-recognition accuracy and often require specific input representations.

In Sect. 2, we survey a wide range of techniques used to recognize user activities; these techniques are organized into several broad domain approaches. Then in Sect. 3, we present the results on the application of these techniques to a set of experimental data to compare their benefits and computational cost. We conclude the article in Sect. 4.

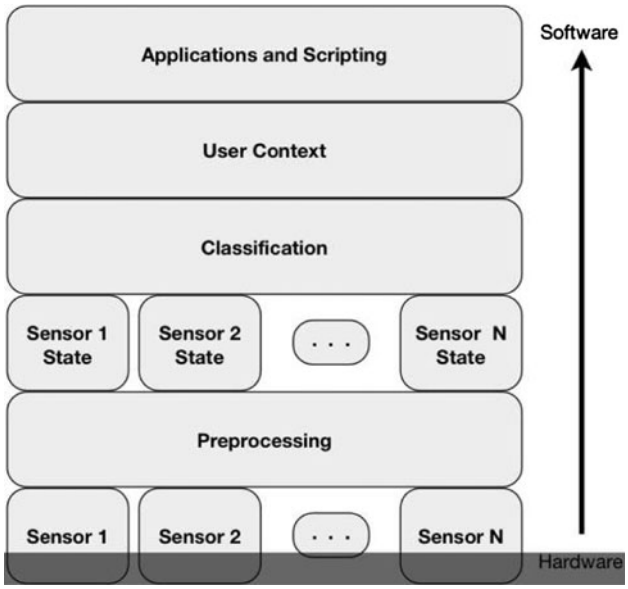

Fig. 1 Layered architecture for context inference applications

2 Preprocessing techniques: domains and approaches

The need to extract key signal features that enable advanced processing algorithms to discover useful context information has led to the development of a wide range of algorithmic approaches. These approaches rely on converting or transforming the input signals to and from different domains of representation. In each domain there are specific methods to abstract raw signal data and to provide, in addition to an early classification, some form of data compression that makes it possible in many cases to apply higher-level algorithms for context recognition.

As depicted in Fig. 2, it is possible to classify the available sensor signal processing techniques is three broad domains, namely: the *time* domain, the *frequency* domain and what we call *discrete representation* domains. The following subsections describe the most representative techniques in each of these domains in order to compare their implementation complexity and accuracy in extracting signal features and identifying user activities.

2.1 Time domain: mathematical and statistical techniques

Simple mathematical and statistical metrics can be used to extract basic signal information from raw sensor data. In addition, these metrics are often used as preprocessing

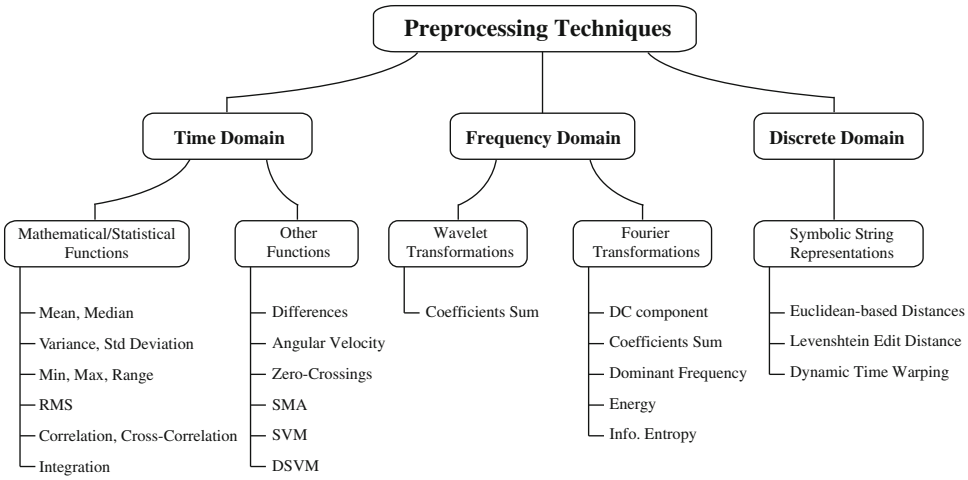

Fig. 2 Classification of techniques applied to sensor signals for feature extraction

steps for metrics in other domains as a way to select key signal characteristics or features. These techniques are often used in practical activity recognition algorithms.

2.1.1 *Statistical metrics: mean, variance and standard deviation*

The mean over a window of data samples is a meaningful metric for almost every kind of sensor. This metric can be calculated with small computational cost [48] and be done on the fly with minimal memory requirements. The mean is usually applied in order to preprocess raw data by removing random spikes and noise (both mechanical and electrical) from sensor signals, smoothing the overall dataset.

There have been various early uses of the mean metric in activity recognition. Several researchers have used the mean to either directly or indirectly identify user posture (sitting, standing or lying) [11, 19, 22, 23] and also to discriminate the *type* of activity as either dynamic or static [60]. Others have used the mean as input to classifiers like Neural Networks [51, 59], Naive Bayes [27], Kohonen Self-Organizing Maps [29], Decision Trees [5], and even Fuzzy Inference [20]. Other applications of the mean value include, for example, axial calibration by finding the average value for all the different orientations [7] and the recognition of complex human gesture using Hidden Markov Models [8].

Another important statistical metric is the variance (σ^2) defined as the average of the squared differences from the mean. The standard deviation (σ) is the square-root of the variance and represents both the variability of a data set and a probability distribution. The standard deviation can give an indication of the stability of a signal. The measure is less useful if it is known that the signal can include spurious values, as even a single value can distort the result.

These two statistical metrics are often used as a signal feature in many activity recognition approaches where they have been used as an input to a classifier or to threshold-based algorithms [12, 15, 30].

In other approaches [22, 23], the variance and standard deviation were used to infer user movement or have been used as the base metric for classifiers like the Naive Bayes [27], Dynamic Bayesian Networks [61], Neural Networks [59] and by [41] to identify the mode of transport. The variance has also been used in [58] in a combination with other metrics such as mean and maximum.

2.1.2 *Envelope metrics: median, maximum, minimum and range*

The median is the number that separates the higher half of data samples from the lower half. The value provided by the median is typically used to replace missing values from a sequence of discrete values (see e.g., [63]).

Despite their simplicity, these envelope metrics still offer some value in activity recognition (e.g., [2]) or as an input to Neural Networks for identifying different inclination angles while walking, as well as in to distinguish between types of postures with threshold-based techniques [3].

The range (the difference between maximum and minimum sample values) was used in [11] together with other indicators to discriminate between walking and running. Application of the maximum and minimum values in accelerometer-based systems was explored to detect steps with the Twiddler keyboard [4], to detect gestures as mnemonical body shortcuts [13], and in activity recognition as an input to a Neural Network classifier [59].

2.1.3 Root mean square (RMS) metric

The root mean square (RMS) of a signal x_i that represents a sequence of n discrete values $\{x_1, x_2, \ldots, x_n\}$ is obtained using Eq. (1) and can be associated with meaningful context information.

$$x_{\mathrm{RMS}} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}} \tag{1}$$

The RMS has been used to classify wavelet results by distinguishing walking patterns [53] and is present in works of activity recognition like [42] as an input to a classifier such as a Neural Network. It was also used as input to a multi-layer neural network in [8] for the recognition of a set of gestures and integration with video records.

2.1.4 Position and velocity using numeric integration

The integration metric measures the signal area under the data curve and is commonly applied to accelerometer signals to obtain estimates of speed and distance [40].

Several approaches have explored this integration technique. In gesture recognition (e.g., [13]), researchers have used a double integration technique to compute the distance covered by a gesture. Others have used this technique to determined velocity values and thus identify gestures using Nintendo Wiimote and a Neural Network classifier [63].

Using the integral of the RMS signal and a simple threshold technique, researchers have been able to distinguish between higher and lower states of activity intensity [15]. In other approaches, Lee et al. [30] used integration to compute the angular velocity of data supplied by a gyroscope.

2.1.5 Signal correlation and correlation coefficient

Signal correlation is used to measure the strength and direction of a linear relationship between two signals. In activity recognition, correlation is especially useful in differentiating between activities that involve translation in a single dimension [43].

In order to calculate the degree of correlation, it is necessary to calculate the correlation coefficients between the signals for the various axes. These coefficients can be obtained by several statistical and geometrical formulas, depending on whether the situation involves simple statistical values or mathematical vectors. The most commonly used is Pearson's product-moment coefficient ($\rho_{x,y}$) [45] also known as the sample correlation coefficient, calculated as the ratio of the covariance of the signals along the x-axis and y-axis to the product of their standard deviations:

$$\rho_{x,y} = \frac{cov(x,y)}{\sigma_x \sigma_y} \tag{2}$$

The sample correlation coefficient was applied in [43] in order to determine which are the best classifiers (or combination of them) for recognizing activities, and which among several features/attributes are the most important. Other researchers (e.g., [57]) also used this coefficient with accelerometer data to correlate the various axes in order to detect, with the aid of a decision tree, the surface under a robot as it walks.

Another way to calculate the correlation coefficient is by geometric interpretation. For normalized or centered data sets, with a mean of zero, the correlation coefficient can also be obtained as the cosine of the angle between two vectors of samples. To determine the correlation coefficient using the angle, the product of vectors representing each axis of the sensor has been used. In [28], the authors applied the normalized cross product between each axis of an accelerometer, together with mean, energy and entropy, to identify a set of daily activities. Other researchers [5, 16, 18] used the dot product divided by the window length as the correlation feature, computed between two different acceleration axes of a hoarder board.

2.1.6 Cross-correlation

The cross-correlation is a measure of the similarity between two waveforms and is commonly used to search for a known pattern in a long signal. The cross-correlation coefficients are calculated by computing a dot product between the signals, normalized over the window size of n samples as denoted by Eq. (3). The various coefficients are obtained by computing the correlation for the "time shifted" versions of one signal with respect to the other.

$$Cross\,Correlation_{(x,y)} = \max_{d=1}^{n-1}\left(\frac{1}{n}\sum_{i=1}^{n} x_i \cdot y_{i-d}\right) \tag{3}$$

The typical implementation of this metric computes the cross-correlation coefficients for the pairs of signals corresponding to the three axes in a pairwise fashion (i.e., (x,y), (x,z) and (y,z)). It then selects the pair of signals that exhibits the largest coefficients to distinguish between dynamic activities, as described in [60].

2.2 Other time-domain metrics

There are several other time-domain characteristics that can be obtained directly from raw accelerometer data and that are popular since they require less computing power than signal characteristics calculated in other domains.

2.2.1 Sample differences

Computing the difference (delta value) between signals in a pairwise arrangement of samples allows a basic comparison between the intensity of user activity. In general, every activity will be noticeable in one or more of the accelerometer axis, so different activities can in principle be distinguished by comparing the signal strength in all three axis.

This approach has been used in [55] to make a crude detection between classes of activities by inspecting of the maximum values (or peaks) of the differences between all axes to determine the type of movement.

2.2.2 Zero-crossings

Zero-crossing can be defined as the points where a signal passes through a specific value corresponding to half of the signal range. The delimiter value can be either the mean value of the sensor range or an extracted mean value. The number of times the signal crosses the reference value is the number of zero-crossings.

This metric has been used to make early recognition of stepping movements [30] and for the detection of the appropriate timing for the application of other techniques (e.g., [11]) to distinguish walking from running. Zero-crossings rate is the rate of zero-crossings along a signal and is commonly applied to audio signals to identify the surrounding environment [10] or the type of sound such as music, speech and noise [49]. Also, in gesture recognition the zero-crossings have been used together with other features and with Hidden Markov Models to detect complex human gestures [8].

2.2.3 Angle and angular velocity

The use of angular position and velocity together with the use of other sensor data and integration techniques allows a basic determination of the user orientation. The angle between the accelerometer axis and the gravity pull can be determined from the mean of the accelerometer signal in all three axis [21, 60].

This metric has been used as described in [30] to identify cyclic behavior using a simple fuzzy-logic reasoning method. Angle variation has been used in fall detection [7, 9] to check sudden variation and final orientation of the user. To detect the location of a device equipped with an accelerometer from a predetermined set of locations, it is possible to use the angle with gravity and its variation along time [23].

2.2.4 Signal magnitude area

In [6], the authors describe an approach that uses the sum of the area encompassed by the magnitude of each of the three-axis accelerometer signals to compute the *energy expenditure* in daily activities. This activity predictor is referred to as the Signal Magnitude Area (SMA) [37] as depicted in Eq. 4 where $x(t)$, $y(t)$ and $z(t)$ are the acceleration signals from each axis with respect to time t.

$$\mathrm{SMA} = \frac{1}{t}\left(\int_0^t |x(t)|\mathrm{d}t + \int_0^t |y(t)|\mathrm{d}t + \int_0^t |z(t)|\mathrm{d}t\right) \tag{4}$$

The SMA metric can be used to distinguish between a resting state and user activity [38] in a classification framework for the recognition of basic daily movements. It is also possible to use SMA as the basis for identifying periods of user activity [21].

2.2.5 Signal vector magnitude

Signal Vector Magnitude (SVM) [21] and Differential Signal Vector Magnitude (DSVM) [19] metrics are similar to the norm as defined below:

$$\mathrm{SVM} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{x_i^2 + y_i^2 + z_i^2}$$

$$\mathrm{DSVM} = \frac{1}{t}\left(\int_0^t \left(\left|\sum \mathrm{SVM}'\right|\right)\mathrm{d}t\right) \tag{5}$$

The SVM metric was used by [21] to identify possible falls and to monitor and classify behavior patterns in cattle using an accelerometer attached to the hind legs [44]. DSVM was developed to facilitate the classification of dynamic daily activities, including falls, using a single metric and several thresholds operations.

2.3 Frequency domain

Frequency-domain techniques have been extensively used to capture the repetitive nature of a sensor signal. This repetition often correlates to the periodic nature of a specific activity such as walking or running. A commonly used signal transformation technique is the Fourier transform, which allows one to represent in the frequency domain (or spectrum) important characteristics of a time-based signal such as its average (or DC component) and dominant frequency components. In this spectral representation, the main periods or repetition intervals of the signal are represented by non-zero values or coefficients at the corresponding frequency axis value. For instance, a time signal with periodic patterns centered around the 0.5-s repetition interval will exhibit a noticeable Fourier coefficient centered around the 2 Hz frequency axis. This frequency analysis is commonly computed for a time signal of a specific length or window using the discrete Fourier

transform with algorithms such as the Fast Fourier Transform (FFT) and the Fast Time Frequency Transform (FTFT) [36]. In addition to the FFT and its spectral representation, other frequency-based representation has been used. For example, the Wavelet Haar transforms [34] represent a time-domain signal as a decomposition of a set of weighted orthonormal vector basis or coefficients. These transforms, although less common, provide computational advantages over the more established FFT computation.

The following subsections highlight the common uses of frequency-domain analysis for the recognition of user activity from accelerometer data.

2.3.1 DC component

The DC component is the first coefficient in the spectral representation of a signal and its value is often much larger than the remaining spectral coefficients. As described earlier, the mean is used as signal characteristic in several activity recognition approaches along with correlation, energy and entropy [5, 16, 18, 28].

2.3.2 Spectral energy

The energy of the signal can be computed as the squared sum of its spectral coefficients normalized by the length of the sample window. The energy metric has been used [41] to identify the mode of transport of a user with a single accelerometer, respectively, walking, cycling, running and driving. In other contexts, researchers used a microphone sensor to obtain an audio context and thus identify when a user would be on the street, engaged in a conversation, or indoors in a loud (e.g., restaurant) or in a quiet place (e.g., lecture) [27].

2.3.3 Information entropy

The entropy metric can be computed using the normalized information entropy of the discrete FFT coefficient magnitudes excluding the DC component [16]. Entropy helps to differentiate between signals that have similar energy values but correspond to different activity patterns. Together with the mean, energy and correlation, entropy has been used in several activity recognition approaches. For example, Bao et al. [5] have used frequency-domain entropy to distinguish between activities with similar energy levels, as is the case of cycling and jogging.

2.3.4 Spectral analysis of key coefficients

Several authors have used the summation of a set of spectral coefficients as a key metric for the recognition of specific activities. For example, the coefficients from 0.5 Hz to 3 Hz can be used as the key discriminating coefficients for the running and walking activities [47, 62].

In [37], the author developed an algorithm to determine the average step rate from the signal spectrum. The algorithm looks for a frequency peak within the 0.7–3 Hz range. The magnitude of the largest signal peak is compared against the baseline noise value. If the signal-to-noise ratio (SNR) is greater than a fixed threshold value then the frequency at which this peak occurs is identified as the step rate. Other authors have used the frequency value corresponding to the maximal spectral coefficient to determine whether a person is either walking or running, and if running at what pace [22, 23, 24].

2.3.5 Wavelet analysis

The Wavelet transform can be used to examine the time-frequency characteristics of a signal as it can be computed more efficiently than the Fourier transform [35]. For the spectral representation, the Wavelet transform makes use of a set of orthonormal basis typically chosen from a family of possible base generating functions. Among the many possible transforms we have opted for the simpler Haar transforms (see e.g., [34]), and in this family of transforms we used the transform of order 2, i.e., H_2. This particular transform only requires additions and subtractions for the computation of the spectral coefficients and can be performed *in place*, thus not requiring any additional storage. Once the spectral coefficients are computed, a simple approach is to add all the coefficients thus generating a single metric value.

Because the wavelet transform can capture sudden changes in signals like the ones measured by an accelerometer, it is often chosen by several activity recognition approaches. However, the direct use of wavelets in the detection of user physical activities raises several issues as there exist many wavelet transforms and of various kinds. In addition, the resulting representation (the wavelet coefficients) do not capture any quantity with physical meaning. As a result, wavelets are commonly used in conjunction with other higher-level techniques (e.g., [1]).

In [52, 53, 54], the authors use the discrete wavelet transform to classify the acceleration signal for horizontal level and stairway walking. Other authors [1] have used the Daubechies wavelets over the preprocessed signals with the SVM transformation. They then feed the data to a Hidden Markov Model to detect human activity with special attention to falls of elderly people.

2.4 Symbolic strings domain

There has been a recent interest in transforming accelerometer and other sensor signals into strings of discrete

symbols. A key aspect in this transformation has been the discretization process, and while there is a potential for information loss, a limited symbol alphabet can lead to a substantial compression in the representation of a signal. Typically, the sequence of n input samples (possibly already normalized as in the case of a 3-axis accelerometer signal) is split into windows of w consecutive samples. An average value is computed for each of these windows followed by a discretization over a fixed size alphabet a.

A simple discretization process uses a domain-value function that defines the interval of values that correspond to a given symbol. A recently developed technique called symbolic aggregate approximation (SAX) uses a piecewise aggregate approximation (PAA) [31], which relies on a gaussian equiprobable distribution function to map range values into string symbols.

Once signals have been mapped to strings, exact or approximate matching and edit distances are key techniques used to evaluate string similarity and thus either find known patterns or classify the user activity.

2.4.1 Euclidian-related distances

For two strings S and T of length n we define the Euclidian distance as

$$EuclidianDist(S,T) = \sqrt{\sum_{i=1}^{n}(|s_i - t_i|)^2} \tag{6}$$

where the distance between symbols is defined by the corresponding numeric distance between the signal values that correspond to each symbol in the string representation.

A related metric often used for time-series classification that is a lower bound on the Euclidian distance is the minimum distance (*MinDist*) metric [31] defined as:

$$MinDist(S,T) = \sqrt{\frac{n}{w}} \cdot \sqrt{\sum_{i=1}^{n} dist(s_i - t_i)^2} \tag{7}$$

where n is the length of the two strings, and the symbols in both strings have been quantized using a Gaussian distribution. The *dist* function is a symmetric distance matrix based on the cut-off points of the same Gaussian. The fraction $\sqrt{\frac{n}{w}}$ is a normalization of the compression ratio achieved by the SAX transformation.

These distance metrics, albeit simple, allow for the quick discrimination of signals and thus for fast evaluation of similarity between strings.

2.4.2 Levenshtein edit distance

Having a representation based on strings of symbols allows the use of a plethora of string-approximation algorithms in order to determine which of a set of representative samples is "closest" to a given input sample.

One of such metrics is the Levenshtein edit distance. For two strings S and T, this metric determines the minimum number of symbol insertions, deletion and substitutions needed to transform one string into the other. A common implementation uses dynamic-programming techniques with the recurrent expression [14]:

$$\begin{aligned} d(i,j) = min\{&d(i-1,j) + insert, d(i,j-1) \\ &+ insert, d(i-1,j-1) + subs(i,j)\} \end{aligned} \tag{8}$$

where m and n are the length of the two strings and d is a $m \times n$ table whose first row and column are initialized with the costs of creating each of the input strings, i.e., by inserting all their symbols. The minimal cost of converting S into T is determined by the value of the table position $d(n, m)$.

In [56], the authors used this distance metric to identify a number of gestures with reported 83% accuracy using a particular training methodology. According to the trajectory of the movement, one of seven symbols is attributed to the signal acquired from a distributed set of inertial sensor modules mounted on the lower arms, the upper arms and the torso of the body. During training, some template samples of the gestures were collected and through a string-matching method based on the computation of weighted edit distances, each gesture is identified.

2.4.3 Dynamic time warping (DTW)

Dynamic time warping (DTW) [46] is a metric for measuring similarity between two sequences that may vary in length and can thus correspond to different time basis. DTW is used in automatic speech recognition based on a temporal alignment of the input signal with template models. It can capture similarities of strings with distinct sampling period, and thus speeds, but has a relatively high computational cost. The goal is to find a mapping W, where in some cases an element of one string can map to a sequence of consecutive elements in the other string.

The DTW algorithm implementation builds a matrix of mapping costs, along which the mapping between the two strings is laid out as a path between the two opposite corners. Of the many possible such paths, DTW finds the mapping W that minimizes:

$$\mathrm{DTW}(S,T) = min\left\{\frac{1}{K} \cdot \sqrt{\sum_{k=1}^{K} w_k}\right\} \tag{9}$$

where the cost of the path through the cost matrix is found using a dynamic-programming approach as the example of the metric in the previous subsection.

DTW has been applied to find similarity metrics between signals [26] in particular in the context of gesture recognition, given the potentially more limited number of required samples [33]. More recently, researchers have also developed the *derivative* DTW (DDTW) [25] and used it to detect activities such as walking, going up and down flights of stairs [39].

2.5 Analysis of suitability of implementation

The various techniques described in Sect. 2.1 through Sect. 2.4 have different computational costs and storage requirements making them more or less suitable for implementation on mobile devices such as smartphones with limited storage or computational resources. We now discuss these implementation costs first on an quantitative basis using abstract complexity measures, and then qualitatively by using these measures to provide an assessment of the suitability of the use of each metric in mobile devices.

2.5.1 Quantitative analysis: complexity of implementation

Rather than using device- or processor-dependent metrics that could be masked by platform-specific features or even by the capabilities of the corresponding compilers[2], we opted for an abstract complexity measure that is tied to the number of operations in the computation of each metric. As such, we divide the costs into several components, where each component becomes a function of the number of samples n in the input data, possibly separated for each of the three accelerometer axes. In the computational cost, we have included the number of additions, multiplications and other arithmetic and logic operations. For simplicity, all operations are assumed to use floating-point (32-bit single precision) representations. In some metrics, there is the need for an initial normalization step in order to transform three integer input vectors (one for each of the three accelerometer axis) into a single floating-point vector. So that this step does not mask the inherent complexity of each metric, we present the complexity results for each metric excluding the cost of this preliminary step. Some metrics, however, do not require this initialization step and are labeled with an asterisk (*) in Table 1. For completeness, we have also included the number of comparison operations, as invariably their implementation will include an arithmetic operation.

[2] Characteristics such as clock rates, caches, functional units and pipelines could influence the results of such comparison, as some techniques may be more amenable to specific architectural or compiler features.

Table 1 Summary of classification of time-domain metrics regarding computational costs, where n is the number of input samples

Time-domain metric	Computational complexity				
	Add	Mult	Div	Sqrt	Comparisons
Normalization	$3n$	$3n$	0	n	0
Mean	n	n	1	0	0
Std. deviation	$2n + 1$	$n + 1$	2	1	0
Median	0	0	0	0	$n(n + 1)/2$
Range	0	0	0	0	$2n$
Maximum	0	0	0	0	n
Minimum	0	0	0	0	n
RMS	n	n	1	1	0
Integration	$2n$	0	0	0	0
Correlation*	$4n + 4$	$3n + 1$	7	3	0
Cross-correlation*	$3n(n - 1)$	$3n(n - 2)$	$3(n - 1)$	3	$3(n - 1)$
Differences	n	0	1	0	$5n$
Zero-crossings	n	0	1	0	$2n$
SMA*	$3n + 6$	0	$3n - 2$	0	0
SVM*	$3n$	$6n$	1	0	0
DSVM*	$4n$	$6n$	$n + 1$	0	0

Table 2 Summary of classification of time-domain metrics regarding storage costs and memory operations, where n is the number of input samples

Time-domain metric	Storage	Memory	
		Read	Write
Normalization	$3n$	$3n$	0
Mean	0	0	0
Std. deviation	0	0	0
Median	0	$n(n + 1)/2$	$n(n + 1)/2$
Range	0	0	0
Maximum	0	0	0
Minimum	0	0	0
RMS	0	0	0
Integration	0	0	0
Correlation*	0	$3n$	$2n$
Cross-correlation*	0	$3n^2$	0
Differences	0	0	0
Zero-crossings	0	n	0
SMA*	$3n$	$3n$	0
SVM*	$3n$	$3n$	0
DSVM*	$3n$	$3n$	0

In separate tables, such as Table 2, we have included the memory requirements to hold temporary variables in addition to the space required to hold the input signal data. These tables also include the number of memory *read* and

write operations to the storage, excluding operations on scalar variables, which tend to be absorbed in register instructions that implement arithmetic operations. Notice that the vast majority of the metrics have 0 values for the *read* and *write* counts as the computation can be done *on-the-fly* since the normalization step is generating the values for each sample.

Overall, these complexity results show that time-domain metrics are dominated by the cost of normalization and none of them involve complex arithmetic operations such as trigonometric or logarithmic functions. Some of the simple metrics exhibit a very small number of multiplications, in some extreme cases (e.g., Differences) only additions and comparisons are required. In general, any of these metrics can be a good candidate for implementation on mobile devices where resources (computing, energy and storage) are at a premium.

As with the computational complexity costs, the storage and memory operations are dominated by the normalization step. In general, all metrics have memory requirements comparable to the normalization with the exception of the *median* metric. The *median* metric uses a *bubble sort* algorithm implementation to find the median value and thus the $O(n^2)$ complexity in terms of comparisons and read/write operations. For long signals with a large number of input samples, asymptotically more efficient algorithms such as *merge sort* with $O(n \log n)$ could reduce the number of such operations at the expense of a more complicated control-flow implementation of the sorting algorithm.

Tables 3 and 4 provide a similar account on the time and space complexity of frequency-domain techniques. This analysis shows that overall these frequency-domain metrics are computationally more expensive than the time-domain metrics, not being necessarily dominated by the normalization phase. When computing the FFT of the normalized input signal, the implementation uses a precomputed table with *sine* and *cosine* values for various discrete points between positions 0 and $n - 1$ thus avoiding expensive trigonometric computations. For the *entropy* metric, however, the use of the logarithmic operator is required.

Table 4 Summary of classification of frequency-domain metrics regarding storage costs and memory operations, where n is the number of input samples

Frequency-domain metric	Storage	Memory	
		Read	Write
Normalization	$3n$	$3n$	n
Energy	$4n$	$3n$	$4n$
Entropy	$4n$	$3n$	$4n$
Coeff. sum	n	0	$4n$
Dominant freq.	n	0	$4n$
Wavelet (H_2 and coeff. sum)	$2n$	$2n$	0

Regarding the storage requirements as depicted in Table 4, these frequency-domain metrics also require more space than the time-domain metrics but still only as a linear function with respect to the number of input samples.

Tables 5 and 6 provide an account on the time and space complexity of string-domain metrics. For these metrics, there is no normalization of the input samples as in the case of the time-domain and frequency-domain techniques. However, there is a transformation of representation from continuous values to discrete symbols. This transformation is accomplished by the SAX operation, which is used for all the string-domain metrics and is therefore not included in each specific metric.

As can be observed, string-domain metrics exhibit much lower costs for expensive operations such as *sqrt* or even multiplications when compared to metrics in the time and frequency domains. For additions, however, and in the case of metrics that rely on dynamic-programming algorithms (*Levenshtein* and *DTW*) the number of operations is quadratic with respect to the n/w ratio. For long input signals, with large values of n and small sample-to-symbol compression ratios (w) this value can become quite large. A similar observation holds for memory read and write operations. As to the requirements of additional storage, they are overall fairly low. The only potentially large factor of a^2 is due to the alphabet size and thus can be kept in check.

Table 3 Summary of classification of frequency-domain metrics regarding computational costs, where n is the number of input samples

Frequency-domain metric	Computational complexity					
	Add	Mult	Div	Sqrt	Log	Comparisons
Normalization	$2n$	$3n$	0	n	0	0
Energy	$8n$	$6n$	$3n + 2$	n	0	n
Entropy	$9n$	$6n$	$3n + 1$	n	n	n
Coeff. sum	$7n + 5$	$5n$	$2n + 1$	n	0	0
Dominant freq.	$7n$	$5n$	$2n + 1$	n	0	n
Wavelet (H_2 and coeff. sum)	$5n$	0	0	0	0	0

Table 5 Computational complexity of symbolic string-domain metrics where n is the number of samples, w the number of consecutive samples aggregated in a symbol, and a the alphabet size

String-domain metric	Computational complexity				
	Add	Mult	Div	Sqrt	Comparisons
SAX					
Normalization	$3n$	n	$n+2$	1	0
Piecewise approx. (PAA)	n	0	n/w	0	0
Gaussian discretization	0	0	0	0	$n/w \times a$
Minimum distance	n/w	1	0	1	0
Levenshtein	$3(n/w)^2$	0	0	0	0
DTW	$(n/w)^2$	0	0	0	0

Table 6 Summary of classification of symbolic string-domain techniques regarding storage costs and memory operations, where n is the number of samples, w the number of consecutive samples aggregated in a symbol, and a the alphabet size

String-domain metric	Storage	Memory	
		Read	Write
SAX			
Normalization	const.	n	0
Piecewise approx. (PAA)	const.	n	n/w
Gaussian discretization	$n/w + a$	$n/w \times a/2$	$n/2$
Minimum distance	a^2	$2(n/w)$	0
Levenshtein	$2(n/w) + a^2$	$6(n/w)^2$	$(n/w)^2$
DTW	$2(n/w) + a^2$	$8(n/w)^2$	$(n/w)^2$

2.5.2 *Qualitative analysis: suitability for mobile devices*

Table 7 presents the qualitative results regarding the implementation of each technique on a mobile device. In this qualitative analysis, we label each metric being suitable to be implemented on a mobile device (`Yes` label); not being suitable (`No` label) and as being moderately suitable (`Moderate` label) this last classification corresponding to an implementation that requires either medium or high computing or storage resources.

To aid and distinguish this classification we further label each metric with four cost-indication levels, namely:

- A metric exhibits a *very low* computational cost if it requires only a number of operations that have a linear relation to the number of input samples n and these operations are in its vast majority arithmetic additions and subtractions.
- A metric exhibits a *low* computational cost if it requires a number of operations that has a linear relation to the number of input samples n, which will include multiplications and division. A fixed number of operations can be advanced arithmetic operations such as square-root or logarithm.
- A *medium* computational cost will include metrics that are quadratic in terms of the number of input samples n of simple addition/subtraction or multiplication/division operations. A fixed number of operations can be advanced arithmetic operations such as square-root or logarithm.
- A *high* computational cost will include techniques that require a number of operations larger than an asymptotic quadratic bound, but where the individual operations are simple arithmetic additions/subtractions or multiplications/division. In this category, a linear number of advanced operations such as *sine* or *log* are required.

In addition to this implementation complexity, we also indicate the category of precision, and we include the base data types required in common implementations of these techniques. While in some cases it is possible to use basic data types with less precision (e.g., use integer values in place of *doubles*), typically the accuracy of the specific metric can degrade substantially. For this reason, in some cases, we include more than one base data type.

Table 8 summarizes the results of the qualitative analysis for the frequency-domain metrics. With the exception of metrics based on Wavelet transforms, all others rely on the computation of the signal spectrum. As such they are deemed expensive in terms of computational cost but still requiring a very modest amount of storage. Given the sophistication of the computations, the required precision is typically high, either using double or using single precision arithmetic. Only the metrics based on Wavelets and those using simple arithmetics or transformations are classified as requiring a low computational effort.[3]

Table 9 summarizes the results of the qualitative analysis for the symbolic string-domain metrics. With the exception of the DTW metric, these string-based metrics require a fairly low computational cost as they do not rely on sophisticated arithmetic and usually require only integer arithmetic. Regarding storage, however, dynamic-programming metrics such as the Levenshtein or the DTW may require storage that is quadratic on the length of their input strings.[4] This apparent high-storage requirement is balanced by the fact that symbol-string representations lend themselves to substantial compression rates. As a result,

[3] In the simplest Wavelet examined, the Haar wavelet of order 2 ($H_2 = [11;\ 1\ -\ 1]$) only additions and subtractions are used and divisions are always by constant, which is optimized in many Floating-Point Units (FPU) hardware designs.

[4] Although clever implementations can reduce this requirement to a linear relationship.

Table 7 Summary of classification of time-domain techniques regarding computational costs, storage requirements, and precision (double/single/int)

Time-domain metric	Ref(s)	Comp. cost	Storage req.	Precision	Mobile device
Mean	[5, 27, 50, 59]	Very low	Very low	Single/int	Yes
Std. deviation	[15, 22, 23, 30, 59]	Very low	Very low	Double/single	Yes
Median	[2, 3]	Medium	Very low	Single/int	Yes
Range	[11]	Very low	Very low	Single/int	Yes
Maximum	[4, 59]	Very low	Very low	Single/int	Yes
Minimum	[4, 59]	Very low	Very low	Single/int	Yes
RMS	[8, 42, 53]	Very low	Very low	Double/single	Yes
Integration	[15, 30, 42]	Very low	Very low	Double/single	Yes
Correlation	[43, 57]	Medium	Low	Double/single	Moderate
Cross-correlation	[5, 16, 18, 28]	Medium	Low	Double/single	Moderate
Differences	[55]	Very low	Very low	Single/int	Yes
Zero-crossings	[8, 11, 30, 49]	Very low	Very low	Single/int	Yes
SMA	[6, 21, 37, 38]	Low	Low	Single/int	Yes
SVM	[21, 44]	Low	Low	Double/single	Yes
DSVM	[19]	Low	Low	Double/single	Yes

Table 8 Summary of classification of frequency-domain techniques regarding computational costs, storage requirements and precision (double/single/int)

Frequency-domain metric	Ref(s)	Comp. cost	Storage req.	Precision	Mobile device
Energy	[5, 16, 18, 27, 28, 41]	Medium	Low	Double/single	Moderate
Entropy	[5, 16, 18, 28]	High	Low	Double/single	No
Coeff. sum	[62]	Medium	Low	Double/single	Moderate
Dominant freq.	[21, 22, 23, 24, 37]	Medium	Low	Double/single	Moderate
Wavelet (H_2 and coeff. sum)	[34]	Low	Low	Double/single	Yes

Table 9 Summary of classification of symbolic string-domain techniques regarding computational costs, storage requirements, and precision (double/single/int)

String-domain metric	Ref(s)	Comp. cost	Storage req.	Precision	Mobile device
Minimum distance	[31]	Low	Low	Int	Yes
Levenshtein	[14]	Medium	Medium	Int	Moderate
DTW	[46]	Medium	Medium	Int	Moderate

these metrics are very suitable for direct implementation on mobile devices.

3 Experimental evaluation

In this section, we discuss and evaluate the suitability of the techniques described in Sect. 2 in recognizing specific user activities based on preprocessing the raw data coming from a three-axis accelerometer. We begin by a description of the way we collected the input data and of the way the various techniques have been implemented. Afterward, we present the experimental results and discuss the accuracy achieved in recognizing user activity in two scenarios: one where the goal is to be able to distinguish between three activities (namely *walking, running* and *jumping*) and another scenario where only two activities are considered (*walking* and *running*). We will refer to these as the *three-activity* and the *two-activity* scenarios, respectively.

The three possible activities are depicted in Fig. 3a, b and c, where the accelerometer device is visible in the right-hand pocket. By attempting to distinguish between three activities, our experiments differ from other applications of the same methods, where typically only a binary classification such as *active* and *non-active* is attained.

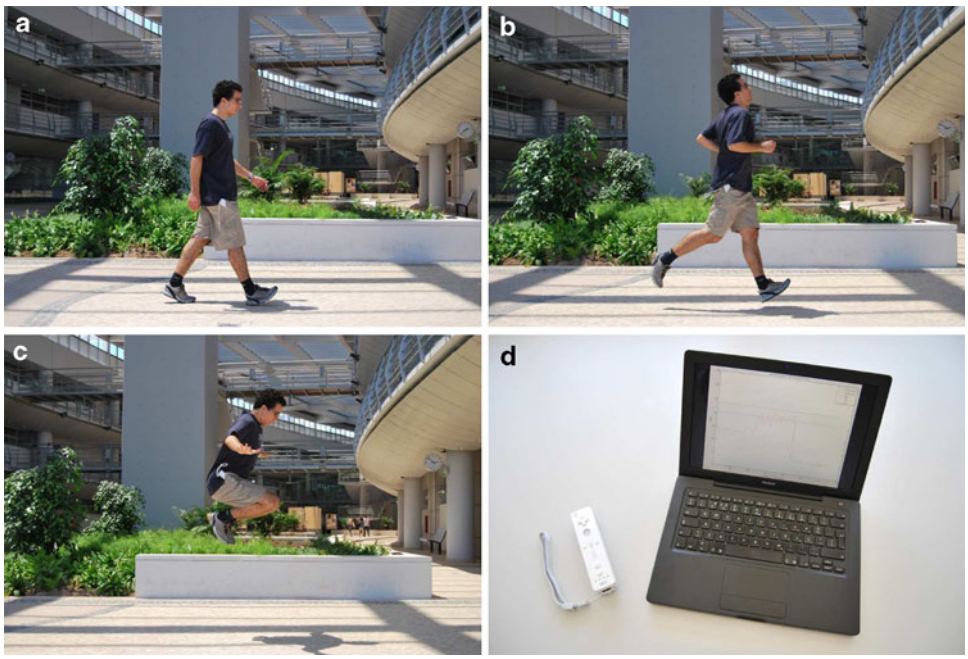

Fig. 3 Data collection illustration for the three activities, namely **a** *walking*, **b** *running* and **c** *jumping*, with a Wii Remote connected to a laptop for recording **d**

3.1 Evaluation methodology

For the evaluation of different preprocessing techniques, we collected raw sensor data using a Nintendo® Wii Remote (generally known as *Wiimote*) shown in Fig. 3d. The Wii Remote includes a three-axis accelerometer sensor (an ADXL330 chip from Analog Devices) that delivers up to 100 data samples per second, where each sample contains the values for the three accelerometer axes. It has built-in bluetooth communication capabilities, which were used to record the data directly to a laptop.

The accelerometer sensor measures acceleration with a minimum full-scale range of $3g$ and it senses both the gravity pull and the acceleration resulting from motion, shock or vibration. Raw numeric data values for each axis range from 0 ($-3g$) to 255 ($+3g$) with the value 127 corresponding to zero acceleration.

With this acquisition setup, we collected raw data for the three activities (*walking, running* and *jumping*) performed by volunteer students. Several minutes were collected for each activity and then split into separate files of 60 s each. As an illustration, Fig. 4 shows 60 s of *jumping*, 60 s of *running* and 60 s of *walking* concatenated together.

For these simple activities, each 60-s file has only one activity in its "pure" form, meaning that it is a full minute of either *walking, running* or *jumping*. There were a total of 15 files for *walking*, 9 files for *running* and 6 files for *jumping*. These files have been divided into a *training set* and a *test set*, as shown in Table 10. The files in the training set were used to empirically tune the parameters of each algorithm, while the files in the test set were used as independent samples to test the actual accuracy of the trained algorithm.

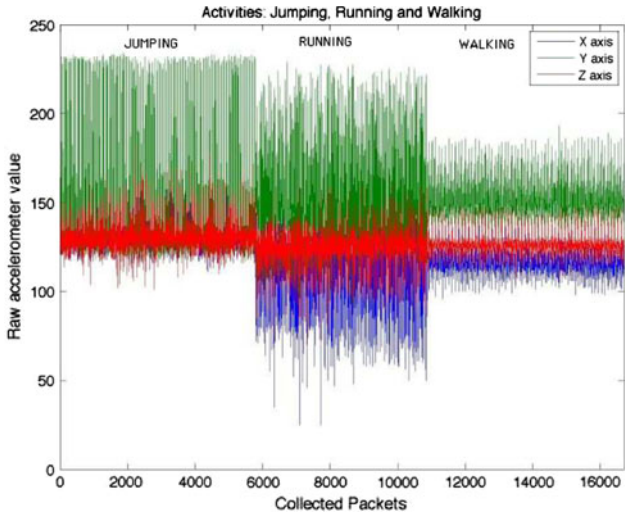

Fig. 4 Plot of a 60-s file for each activity, each file containing approximately 6000 samples

Table 10 Activities used in experimental evaluation

Activities	Training set		Test set		Total	
	Files	Windows	Files	Windows	Files	Windows
Walking	7	311	8	355	15	666
Running	4	175	5	214	9	389
Jumping	3	135	3	136	6	271

We report the results of activity recognition for both the two-activity and the three-activity scenario. Accuracy is computed as the percentage of correctly classified files from the test set, for all activities. The same measure of

accuracy has been computed separately for the files in the training set in order to provide an idea of the maximum accuracy that could be possibly achieved. For the training set, the accuracy is seldom 100% as each algorithm needs to distinguish the files of all activities, and the best choice of the algorithm parameters (as described below) may not allow for a complete discrimination.

3.1.1 Selection of threshold parameters

Given the sampling frequency of the accelerometer ($\leq$100 Hz) each 60-s file contains 6,000 samples each of which includes data from each of the three axis. The data in each of these files are further divided into windows of 256 samples, each window with a 50% overlap with the previous one. The 256-sample window is the basic unit to be classified by a given technique.

In order to carry out the evaluation experiments, we have implemented the techniques described in Sect. 2 in MATLAB® according either to the reference article or to the mathematical definition of each technique. For most metrics, the first step is to compute the norm n_i for each individual sample $(x_i;y_i;z_i)$, i.e., $n_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$ and then apply the chosen metric to the resulting normal signal. A second step consists in finding the two thresholds parameters—thr_1 and thr_2—that separate the activities based on the chosen metric. With three activities, there is always one threshold (thr_1) separating the lower-valued class from the intermediate one and another threshold (thr_2) separating the intermediate class to the higher-valued one. In the example of Fig. 5, thr_1 separates *walking* from *running* and thr_2 separates *running* from *jumping*.

The configuration of these thresholds may differ depending on the results of the chosen metric, so in the first step, it is necessary to pick up one file from each activity and identify the classes that the two thresholds will separate. Then, the second step consists in taking the whole training set and find the best values of thr_1 and thr_2, where $thr_1 < th_2$. The optimal values for these parameters are obtained when there is maximum accuracy in classifying the training set, i.e., when most (if not all) 256-sample windows are correctly identified as belonging to the activity that is specified in the training set.

In our implementation, we obtained the values of thr_1 and thr_2 for each metric by sweeping through the value range using a bisection method over the search space, until the accuracy can be improved no further. An example of such search on a space grid is shown in Fig. 6. In this case, we have a metric that varies between 0.0 and 1.0, and the best result is achieved when $thr_1 = 0.4$ and $thr_2 = 0.5$, respectively. In this work, however, we did not explore the impact of window size and of the degree of overlap, as has been studied by other authors [17].

3.1.2 Accuracy in activity recognition

After the threshold parameters thr_1 and thr_2 have been determined, training is complete, and it is time to apply the chosen metric to the test set in order to evaluate its real accuracy.

Testing for the accuracy of a given metric is straightforward: one has only to pick each window in the test set, compute its metric value, and compare this value to the trained thresholds. Since the two thresholds define three classes of activities, we have:

- if the metric value is below thr_1, the window is classified in the lower-class;

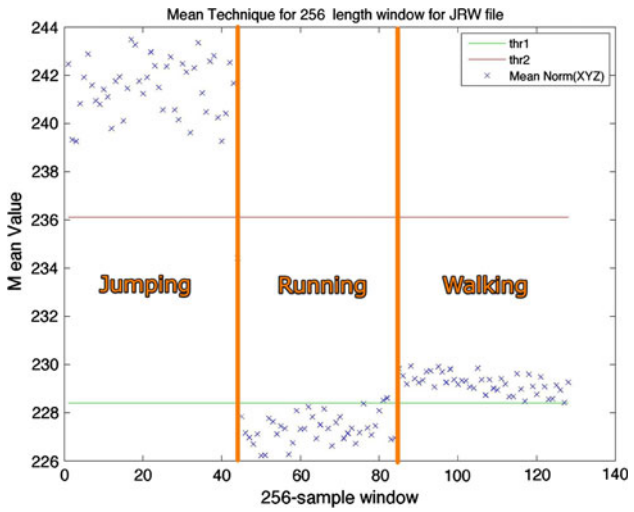

Fig. 5 For any given metric, two thresholds thr_1 and thr_2 can be used to separate the three classes of user activities

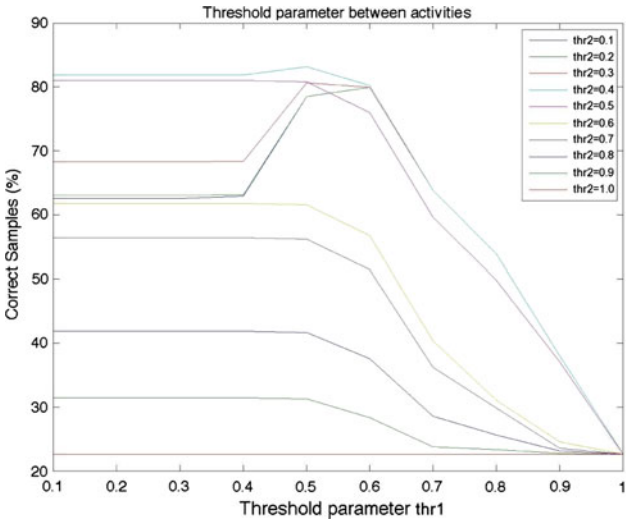

Fig. 6 Illustration of the selection of best threshold parameters for the correlation metric

- if the value is between thr_1 and thr_2, the window is classified in the middle-class;
- if the metric value is above thr_2, the window is classified in the higher-class.

The correspondence between classes and activities has been established previously during the training phase. The overall test accuracy is obtained by classifying all windows in the test set. This accuracy is defined as the percentage of windows that have been correctly assigned to their true activity.

3.2 Experimental results

We now present the evaluation results for the preprocessing techniques described in Sect. 2. The evaluation is carried out both for the three-activity scenario and for the two-activity scenario. For the two-activity scenario, the approach is basically the same but with a single threshold parameter.

3.2.1 Results for time-domain metrics

Table 11 presents the results for the time-domain metrics in the three-activity scenario. The threshold values thr_1 and thr_2 that have been derived from the training set are shown, together with the accuracy obtained both in the training set and in the test set.

Overall, the techniques exhibit a wide range of accuracy results with an average of about 70.4% for the training set and of 60.8% for the test set. Of all these metrics, the *differences* metric seems to be by far the most accurate one, followed by the *minimum* metric. All other metrics exhibit substantially less accuracy particularly for the test set. The case of the *differences* metric is somewhat misleading as the implementation relies on other metrics to increase its accuracy. Specifically, it makes use of an average over the window to eliminate spikes. The combination of these metrics (*differences* and *mean*) achieves an impressive 97.3% accuracy rate. Lastly, we note that metrics such as *SMA* and *DSVM* have a somewhat disappointing performance given the complexity of their implementation. The *RMS* and the *integration* metric also seem to be particularly ineffective in the test set.

Table 12 presents the results for the time-domain metrics in the two-activity scenario. As expected, given the simpler recognition task, the metrics exhibit in general better results than for the three-activity scenario. The *differences* and the *minimum* metrics are again the most accurate, with the *differences* metric achieving a perfect fit to the training set. For both metrics, the accuracy is quite high in the test set. When comparing against the results for the three-activity scenario, the metrics in general allow for a higher accuracy with an average of 79.7 and 70.1%, accuracy rate in the training set and in the test set, respectively.

3.2.2 Results for frequency-domain metrics

Table 13 presents the results for the frequency-domain metrics. Overall, the results for the three-activity scenario are unsatisfying, as the increased complexity does not

Table 11 Summary of the results obtained by applying a selected set of the time-domain techniques to collected data for three activities *walking, running* and *jumping*

Time-domain metric	Parameter values		Training accuracy (%)	Test accuracy (%)
	thr_1	thr_2		
Mean [5]	228.4	241.1	67.96	59.14
Std. deviation [23]	17.7	28	63.74	58.54
Median [2]	226.5	242.2	60.37	56.02
Range [11]	63	95.7	64.24	70.73
Maximum [59]	264.3	296	71.16	67.01
Minimum [59]	200.2	211.5	80.10	80.39
RMS [42]	228.1	237	65.76	46.81
Integration [42]	58,246	60,372	66.10	46.95
Correlation (x, y) [43]	−0.33	−0.78	70.32	68.05
Cross-correlation (x, y) [5]	449	511	70.82	57.06
Differences [55]	5.3	8.2	97.30	80.83
Zero-crossings [11]	16.8	36.6	76.50	74.44
SMA [21]	401	417	68.63	40.56
SVM [21]	228	241	67.45	60.33
DSVM [19]	58,280	60,290	66.27	46.36

Table 12 Summary of the results obtained by applying a selected set of the time-domain techniques to collected data for two activities of *walking* and *running*

Time-domain metric	Parameter value thr_1	Training accuracy (%)	Test accuracy (%)
Mean [5]	233.9	76.94	52.22
Std. deviation [23]	14.7	82.97	81.95
Median [2]	231.7	76.72	53.96
Range [11]	109.5	70.47	68.51
Maximum [59]	264.3	64.22	71.60
Minimum [59]	205.8	98.60	95.95
RMS [42]	233.7	75.43	53.41
Integration [42]	59,655	76.94	53.41
Correlation (x, y) [43]	−0.33	89.87	77.16
Cross-correlation (x, y) [5]	450	81.25	71.27
Differences [55]	6	100.00	99.63
Zero-crossings [11]	36.6	75.00	75.70
SMA [21]	402	79.09	53.78
SVM [21]	228.9	74.14	70.72
DSVM [19]	58,280	75.00	72.74

Table 13 Summary of the results obtained by applying the frequency-domain metrics to the collected data the three activities of *walking, running* and *jumping*

Frequency-domain metric	Parameter values		Training accuracy (%)	Test accuracy (%)
	thr_1	thr_2		
Energy [5]	511	987	81.28	76.97
Entropy [5]	6.94	6.93	67.45	65.23
Coeff. sum [62] ($F_i = 0.7$ Hz, $F_f = 3$ Hz)	2,420	6,070	77.07	79.90
Dominant freq. [23]	2.5 Hz	2.8 Hz	52.78	58.25
Wavelet ((H_2 and coeff. sum) [34]	22,950	30,730	68.47	41.60

translate on an improvement in accuracy. For both the *coefficient sum* metric and the *dominant frequency* metric, the accuracy for the test set is actually higher than the accuracy for the training set. This can be explained by the fact that the metric results on the training set are probably more dispersed than in the test set. During the training phase, when the thresholds are chosen, it may not be possible to accommodate some points of the training set; however, the points in the test set are closer together, which means that they can be classified with fewer errors.

On the other hand, techniques that display similar results in the training set and in the test set, even if their accuracy is not too high, are more easily adaptable to signal variations within the same activity. These variations may be due to different users, different kinds of clothing, or even different ways of walking, running and jumping. So it is important to look for those techniques that are more consistent across the training set and the test set, besides looking for the best accuracy.

Frequency-domain techniques seem to provide both fairly good accuracy (except perhaps for the *dominant frequency* metric) and also fairly good consistency. The average accuracy for the training set is 69.6% and for the test set is 70.1%, the best accuracy being achieved by the *coeff. sum* metric. This provides an overall idea of the typical accuracy that can be obtained with frequency-domain metrics in a three-activity scenario.

Similar to what had happened in the case of time-domain metrics, the results for the frequency-domain metrics are better in the two-activity scenario, as shown in Table 14. The average accuracy now increases to 79.5% for the training set and 78.0% for the test set. The most noticeable improvement is in the *entropy* metric, while the others provide smaller improvements to what had been obtained in the three-activity scenario.

3.2.3 Results for symbolic string-domain metrics

We now present results for the use of some metrics using a string representation of the raw accelerometer signal. In these experiments, we used the SAX representation [31] for windows with length of 512 samples. We varied the size of

Table 14 Summary of the results obtained by applying the frequency-domain metrics to the collected data for the two activities of *walking* and *running*

Frequency-domain metric	Parameter value thr_1	Training accuracy (%)	Test accuracy (%)
Energy [5]	589	83.19	81.79
Entropy [5]	6.92	85.56	81.58
Coeff. sum [62] ($F_i = 0.7$ Hz, $F_f = 3$ Hz)	2430	82.54	79.01
Dominant freq. [23]	2.6 Hz	66.81	69.61
Wavelet (coeff. sum) [34]	22,950	75.86	52.85

the alphabet and the number of consecutive raw samples that are represent by one string symbol. One file for each activity was selected as the reference sample for that activity. All other samples were then tested against that representative sample to evaluate the accuracy of the recognition.

The results for the three-activity scenario using this string representation and the set of metrics depicted in Table 15 are not very encouraging. The overall accuracy rate for the training set and the test set is 62.4 and 45.1% respectively. Even ignoring the *minimum distance* metric that achieves a level of accuracy that is comparable to a random selection of the activities, only the *Levenshtein* metric achieves a reasonable result, outperforming the more sophisticated *DTW* metric.

In contrast, the results for the recognition of two activities as depicted in Table 16 are significantly better with an overall accuracy rate for the training set and the test set of 73.4 and 77.5%, respectively. Not only the *Levenshtein* and the *DTW* metrics present much better accuracy than their use in the three-activity scenario, but the gap between training set and test set is very small, which is a good indicator of the adaptability of these techniques in scenarios where only a binary classification is needed.

3.3 Discussion

Overall, the results for the classification of the three physical activities are not outstanding. Frequency-domain metrics in particular were expected to perform better, given the computational cost of computing the frequency spectrum either using FFT-based methods or using Wavelets. Fortunately, the results are significantly better in a two-activity scenario. Somewhat surprising is the good performance of some of the simpler time-based metrics making them a method of choice for embedded mobile devices where energy and storage is at premium. String-based metrics exhibit good accuracy results only in the two-activity scenario. Given their simpler computational costs these methods also have the potential for better results if further preprocessing is applied, and they offer the added benefit of a natural compression feature and discrete representation.

Table 15 Summary of the results obtained by applying symbolic string-domain metrics to collected data for the three activities *walking, running* and *jumping*

String-domain metric	Parameter values		Training accuracy (%)	Test accuracy (%)
	Alphabet	Window		
Minimum distance [32]	3	6	50.00	32.03
Levenshtein [14]	9	9	76.80	52.50
DTW [46]	8	11	60.30	50.50

Table 16 Summary of the results obtained by applying symbolic string-domain metrics to collected data for the two activities of *walking* and *running*

String-Domain Metric	Parameter values		Training accuracy (%)	Test accuracy (%)
	Alphabet	Window		
Minimum distance [32]	6	6	58.47	67.63
Levenshtein [14]	3	6	81.36	80.58
DTW [46]	7	10	80.51	84.17

4 Conclusion

In this article, we have presented a survey of the techniques for recognizing daily physical activities such as *walking* from raw accelerometer data. We described techniques that operate on the time domain and on the frequency domain, and also on symbolic data representations that can be used to discriminate between user activities. We evaluated a representative set of techniques from these domains on a sample data obtained in a real-world environment for the activities of *walking, running* and *jumping*. The evaluation focused on the ability of the techniques to successfully distinguish those user activities in a three-activity and in a two-activity scenario. The results reveal that for the three-activity scenario some of the frequency-domain techniques are particularly robust and have comparable performance to selected time-domain techniques. For the two-activity scenario, however, the best time-domain techniques outperform the best frequency-domain techniques. Regarding the symbolic-domain techniques, their performance is only acceptable for the two-activity scenario, making them attractive if their simpler implementation and their potential for data compression is taken into account, Overall, we hope this assessment will contribute to identify the preprocessing techniques that are better suited for accelerometer-based and context inference applications.

References

1. Al-ani T, Ba QTL, Monacelli E (2006) On-line automatic detection of human activity in home using wavelet and hidden markov models scilab toolkits. In: Proceedings of the 16th international conference on control applications (ICCA)
2. Aminian K, Robert P, Jequier E, Schutz Y (1995) Estimation of speed and incline of walking using neural network. IEEE Trans Instrum Meas 44(3):743–746
3. Aminian K, Robert P, Buchser E, Rutschmann B, Hayoz D, Depairon M (1999) Physical activity monitoring based on accelerometry: validation and comparison with video observation. Med Biol Eng Comput 37(1):304–308
4. Ashbrook D (1999) Context sensing with the twiddler keyboard. In: Proceedings of the third international symposium on wearable computers (ISWC'99), pp 197–198
5. Bao L, Intille SS (2004) Activity recognition from user-annotated acceleration data. In: Proceedings of the interantional conference on pervasive computing (PERVASIVE'04). Springer, Lecture notes on computer sciences (LNCS), vol 3001, pp 1–17
6. Bouten C, Koekkoek K, Verduin M, Kodde R, Janssen J (1997) A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity. IEEE Trans Biomed Eng 44(3):136–147
7. Brown G, Bajcsy R, Eklund M (2005) An accelerometer based fall detector: development, experimentation, and analysis. Technical Report 12, University of California at Berkeley
8. Chambers G, Venkatesh S, West G, Bui H (2002) Hierarchical recognition of intentional human gestures for sports video annotation. In: Proceedings of the 16th international conference on pattern recognition, vol 2, pp 1082–1085
9. Chen J, Kwong K, Chang D, Luk J, Bajcs R (2005) Wearable sensors for reliable fall detection. In: Proceedings of the 27th annual IEEE conference on engineering in medicine and biology (EMB'05)
10. Dargie W (2006) A distributed architecture for computing context in mobile devices. Master's thesis, Dresden University of Technology, Department of Computer Science
11. Farringdon J, Moore AJ, Tilbury N, Church J, Biemond PD (1999) Wearable sensor badge and sensor jacket for context awareness. In: Proceedings of the IEEE international symposium on wearable computers (ISWC'99). IEEE Computer Society, pp 107–114
12. Golding A, Lesh N (1999) Indoor navigation using a diverse set of cheap, wearable sensors. In: Proceedings of the 3rd IEEE international symposium on wearable computers, pp 29–36
13. Guerreiro T, Gamboa R, Jorge J (2008) Mnemonical body shortcuts: improving mobile interaction. In: Proceedings of the 15th European conference on cognitive ergonomics (ECCE'08). ACM, pp 1–8
14. Gusfield D (1997) Algorithms on strings, trees, and sequences. Cambridge University Press, Cambridge
15. Healey J, Logan B (2005) Wearable wellness monitoring using ecg and accelerometer data. In: Proceedings of the ninth IEEE international symposium on wearable computers (ISWC'05). IEEE Computer Society Press, Washington, DC, pp 220–221
16. Ho J (2004) Interruptions: using activity transitions to trigger proactive messages. Master's thesis, Massachusetts Institute of Technology
17. Huynh T, Schiele B (2005) Analyzing features for activity recognition. In: sOc-EUSAI '05: Proceedings of the 2005 joint conference on smart objects and ambient intelligence, pp 159–163
18. Intille SS, Bao L, Tapia EM, Rondoni J (2004) Acquiring in situ training data for context-aware ubiquitous computing applications. In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI'04). ACM Press, pp 1–8
19. Jeong DU, Kim SJ, Chung WY (2007) Classification of posture and movement using a 3-axis accelerometer. In: Proceedings of the 2007 international conference on convergence information technology (ICCIT'07). IEEE Computer Society, Washington, DC, USA, pp 837–844
20. Jin G, Lee S, Lee T (2007) Context awareness of human motion states using accelerometer. J Med Syst 32(2):93–100
21. Karantonis D, Narayanan M, Mathie M, Lovell N, Celler B (2006) Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. IEEE Trans Inform Technol Biomed 10(1):156–167
22. Kawahara HSY, Hisashi Kurasawa HM, Aoyama T (2005) A context-aware collaborative filtering algorithm for real world oriented content delivery service. In: Proceedings of ubiPCMM
23. Kawahara Y, Kurasawa H, Morikawa H (2007) Recognizing user context using mobile handsets with acceleration sensors. In: (IEEE) international conference on portable information devices (PORTABLE'07), pp 1–5
24. Kawaharaq Y, Ryu N, Asami T (2009) Monitoring daily energy expenditure using a 3-axis accelerometer with a low-power microprocessor. e-Minds: Int J Hum Comput Interact 1(5):145–154
25. Keogh E, Pazzani M (2001) Derivative dynamic time warping. In: Proceedings of the first SIAM international conference on data mining (SDMÕ2001)
26. Keogh E, Lonardi S, Ratanamahatana CA, Wei L, Lee SH, Handley J (2007) Compression-based data mining of sequential data. Data Min Knowl Discov 14(1):99–129

27. Kern N, Schiele B, Schmidt A (2007) Recognizing context for annotating a live life recording. Pers Ubiquit Comput 11(4):251–263
28. Kim IJ, Im S, Hong E, Ahn SC, Kim HG (2007) ADL classification using triaxial accelerometers and RFID. In: International workshop on ubiquitous convergence technology (IWUCT'07)
29. Krause A, Siewiorek D, Smailagic A, Farringdon J (2003) Unsupervised, dynamic identification of physiological and activity context in wearable computing. In: Proceedings of seventh IEEE international symposium on wearable computers (ISWC'03), pp 88–97
30. Lee SW, Mase K (2002) Activity and location recognition using wearable sensors. IEEE Pervasive Comput 1(3):24–32
31. Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on research issues in data mining and knowledge discovery
32. Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing SAX: a novel symbolic representation of time series. Data Min Knowl Discover 15(2):107–144
33. Liu J, Wang Z, Zhong L, Wickramasuriya J, Vasudevan V (2008) uWave: accelerometer-based personalized gesture recognition. TR0630-08, Rice University and Motorola Labs, Houston, Texas
34. Mallat S (1999) A wavelet tour of signal processing. Academic Press, Berlin
35. Mäntyjärvi J (2003) Sensor-based context recognition for mobile applications. Ph.D. thesis, University of Oulu, Finland, Faculty of Technology, Department of Electrical and Information Engineering, Information Processing Laboratory
36. Martens W (1992) The Fast Time Frequency Transform (F.T.F.T.): a novel on-line approach to the instantaneous spectrum. In: Engineering in Medicine and Biology Society, vol 14. Proceedings of the annual international conference of the IEEE, vol 6, pp 2594–2595
37. Mathie M (2003) Monitoring and interpreting human movement patterns using a triaxial accelerometer. Ph.D. thesis, University of New South Wales
38. Mathie M, Celler B, Lovell N, Coster A (2004) Classification of basic daily movements using a triaxial accelerometer. Med Biol Eng Comput 42(5):679–687
39. Muscillo R, Conforto S, Schmid M, Caselli P, D'Alessio T (2007) Classification of motor activities through derivative dynamic time warping applied on accelerometer data. In: Proceedings of the 29th IEEE annual international conference on Engineering in Medicine and Biology Society (EMBS'07), pp 4930–4933
40. Nambu M (2007) Body surface mounted biomedical monitoring system using bluetooth. In: Proceedings of the 29th annual international conference of the IEEE on Engineering in Medicine and Biology Society, (EMBS'07), pp 1824–1825
41. Nham B, Siangliulue K, Yeung S (2008) Predicting mode of transport from iphone accelerometer data. CS 229: Machine Learning Final Projects, Stanford University, Stanford, California
42. Randell C, Muller H (2000) Context awareness by analyzing accelerometer data. In: Proceedings of the 4th IEEE international symposium on wearable computers (ISWC'00). IEEE Computer Society, Washington, DC, USA, pp 175–176
43. Ravi N, Dandekar N, Mysore P, Littman ML (2005) Activity recognition from accelerometer data. In: IAAI'05: Proceedings of the 17th conference on innovative applications of artificial intelligence. AAAI Press, pp 1541–1546
44. Robert B, White B, Renter D, Larson R (2009) Evaluation of three-dimensional accelerometers to monitor and classify behavior patterns in cattle. Comput Electron Agric 67(1–2):80–84
45. Rodgers J, Nicewander W (1988) Thirteen ways to look at the correlation coefficient. Am Stat 42(1):59–66
46. Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process 26(1):43– 49
47. Santos AC, Tarrataca L, Cardoso JMP, Ferreira DR, Diniz PC, Chainho P (2009) Context inference for mobile applications in the UPCASE project. In: Proceedings of the 2nd international conference on mobile wireless middleware, operating systems, and applications (MOBILWARE 2009). Springer, no. 7 in LNICST, pp 352–365
48. Schmidt A (2002) Ubiquitous computing—computing in context. PhD thesis, Lancaster University, England, UK
49. Schmidt A, van Laerhoven K (2001) How to build smart appliances? Pers Commun IEEE 8(4):66–71
50. Schmidt A, Beigl M, Gellersen HW (1999a) There is more to context than location. Comput Graph 23:893–901
51. Schmidt A, Gellersen HW, Beigl M (1999b) A wearable context-awareness component. finally a good reason to wear a tie. In: Proceedings of the 1999 international symposium on wearable computers, (ISWC'99), pp 176–177
52. Sekine M, Tamura T, Ogawa M, Togawa T, Fukui Y (1998) Classification of acceleration waveform in a continuous walking record. In: Proceedings of the 20th annual IEEE international conference of the engineering in medicine and biology society, vol 3, pp 1523–1526
53. Sekine M, Tamura T, Fujimoto T, Fukui Y (2000) Classification of walking pattern using acceleration waveform in elderly people. In: Proceedings of the 22nd annual IEEE international conference of the engineering in medicine and biology society, vol 2, pp 1356–1359
54. Sekine M, Tamura T, Akay M, Fujimoto T, Togawa T, Fukui Y (2002) Discrimination of walking patterns using wavelet-based fractal analysis. IEEE Trans Neural Syst Rehabil Eng 10:188–196
55. Siewiorek D, Smailagic A, Furukawa J, Krause A, Moraveji N, Reiger K, Shaffer J, Wong F (2003) Sensay: a context-aware mobile phone. In: Proceedings of the 7th international symposium on wearable computers (ISWC)
56. Stiefmeier T, Roggen D, Tröster G (2007) Gestures are strings: efficient online gesture spotting and classification using string matching. In: Proceedings of international conference BodyNets 07
57. Vail D, Veloso M (2004) Learning from accelerometer data on a legged robot. In: Proceedings of the 5th IFAC/EURON symposium on intelligent autonomous vehicles
58. Van Laerhoven K, Cakmakci O (2000) What shall we teach our pants. In: Proceedings of the fourth international symposium on wearable computers (ISWC'00)
59. Van Laerhoven K, Aidoo K, Lowette S (2001) Real-time analysis of data from many sensors with neural networks. In: Proceedings of the 5th international symposium on wearable computers (ISWC'01), pp 115–122
60. Veltink P, Bussmann H, de Vries W, Martens W, Van Lummel R (1996) Detection of static and dynamic activities using uniaxial accelerometers. IEEE Trans Rehabil Eng 4(4):375–385
61. Wang S, Pentney W, Popescu AM (2007) Common sense based joint training of human activity recognizers. In: Proceedings of the 20th international joint conference on artificial intelligence (IJCAI'07), pp 2237–2242
62. Welbourne E, Lester J, LaMarca A, Borriello G (2007) Mobile context inference using low-cost sensors. In: Proceedings of the first international workshop on Location- and Context-Awareness (LoCA 2005), Springer, LNCS, vol 3479, pp 254–263
63. Wiggins H (2008) Gesture recognition of Nintendo Wiimote input using an artificial neural network. Cognitive Systems, University of British Columbia, Vancouver, Canada