Pattern Recognition Letters 30 (2009) 1448–1456

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Embedded surface classification in digital sports

B. Eskofier [a,*], M. Oleson [b], C. DiBenedetto [b], J. Hornegger [a]

[a] *University of Erlangen-Nuremberg, Department of Computer Science, Institute of Pattern Recognition, Martensstrasse 3, 91058 Erlangen, Germany*
[b] *adidas AG, adidas innovation team, 5055 N Greeley Ave., Portland, OR 97217, USA*

A R T I C L E I N F O

Article history:
Received 30 January 2008
Received in revised form 22 July 2009
Available online 12 August 2009
Communicated by H. Sako

Keywords:
Microprocessor
Embedded classification
Digital sports
adidas_1
Surface classification

A B S T R A C T

In this presentation, we give a detailed analysis of the considerations needed for mapping the complete pattern classification chain to the restricted embedded system hardware environment. We describe the methodology of the design, realization and testing process that takes these hardware limitations into account. For this purpose, we consider a particular embedded application from the field of digital sports: a novel running shoe that is capable of sensing run-specific parameters and adapting the cushioning setting accordingly. Of utmost importance in this context is the classification of the current surface condition in order to enable optimal adaptation to the prevailing situation. Following our design approach, we provide a classification system with a runner-independent surface classification rate of more than 80%. This system is implemented in the current version of the aforementioned running shoe. The presented methodology is quite general as it makes no system-dependent assumptions and can thus be transferred to many other embedded classification applications.

1. Introduction

The ability to perform accurate classification in real-time is a key factor for many applications. This is not only true when computationally powerful hardware is used. It is most often crucial in the restricted hardware environment of the power-efficient, highly mobile microprocessors used in embedded systems. Consider, for example, portable devices performing image classification or speech recognition. As Hacker et al. (2006) showed, classification of the focus of attention of a user in interaction with a portable digital assistant (PDA) is possible with classification rates of up to 93%. The authors avail the signal of a built-in video camera and information from the speech signal to discern whether the user is trying to interact with the device or not.

The most important question is which of the complex algorithms known in pattern recognition can be used and implemented in the context of the restricted memory capacity and computational power of the employed microprocessors. Special considerations have to be made in order to adapt those algorithms to the specific hardware and classification task at hand. A lot of areas of engineering can benefit from the possibility of accurate classification in this restricted environment. Examples include, but are not limited to, automotive solutions, communications, industrial automation, speech recognition and medical care. In each of these fields, cheap and therefore mass producible systems that are highly portable can open up completely new ranges of applications.

We present an approach to accurate classification on a microcontroller that uses an example from the field of digital sports. It is quite common in this field to use rather straightforward statistical analysis and model building techniques even for large, multidimensional datasets. Lun et al. (2004) for example examined the relation between biomechanical variables of runners and the risk of running specific injuries. To facilitate this, they defined several injury classes and tried to identify significantly different parameters between those. While their work offers a lot of new insights, the underlying database is very complex so that important higher-dimensional coherences might not have been revealed. In the most recent years, pattern analysis concepts find their way into the field of digital sports, too. One application of pattern analysis methodology was shown in (von Tscharner and Goepfert, 2003). In this paper it is reported that electromyograph signals of muscle activity can be represented in pattern space using wavelet analysis. The authors furthermore demonstrate that the different activity patterns of males and females can be classified with a precision of more than 95%.

Our approach to guarantee accurate classification on the embedded system is to perform as much analysis as possible on computationally powerful PCs. This allows us to efficiently compare a lot of different approaches and select the one that is best suited for the classification task. Thereby, we keep the hardware restrictions in mind during every step of the pattern recognition chain. We identify the classifier that is best suited for the

* Corresponding author. Tel.: +49 9131 8527890; fax: +49 9131 303811.
E-mail addresses: bjoern.eskofier@informatik.uni-erlangen.de, bjoern@eskofier.eu (B. Eskofier), mark.oleson@adidas.com (M. Oleson), christian.dibenedetto@adidas.com (C. DiBenedetto), joachim.hornegger@informatik.uni-erlangen.de (J. Hornegger).

0167-8655/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2009.08.004

implementation on the specific microcontroller that is used, performing only the final verifications on the embedded hardware. For this presentation, we focus on the application of these concepts on the adidas_1 running shoe, which is the first shoe ever that features an embedded system. This shoe is built to adapt to various running conditions like the prevailing surface situation. A precise classification of these conditions is of course mandatory to guarantee this functionality. To facilitate this, the step signal of the runner is continually measured and processed by the embedded microcontroller. A detailed description of the adidas_1, its functionality and embedded system hardware can be found in Sections 2.1 and 2.2, in (DiBenedetto et al., 2004) and in (DiBenedetto et al., 2005). In the rest of Section 2 we describe the analysis methods that lead to accurate, real-time surface classification, including:

- the preprocessing steps that are a prerequisite to later obtain features that can be reliably computed (Section 2.4);
- the choice of discriminative features, which dependably represent the signal information while still being efficiently calculable (Section 2.6);
- a detailed description of the examined classifiers (Section 2.7).

The choice and parameterization of each of these factors of the classification chain is essential for ensuring optimal results. In Section 3 we present the conducted experiments and their results. In our summary in Section 4 we will show that, while the specific question of surface classification is solved, the employed methods are general in nature so that they can contribute to a wide area of applications featuring embedded systems. The presented example for a classification system has recently been implemented in the current version of the adidas_1 running shoe, which is commercially available. It is significantly contributing to the shoe's functionality and thereby offering runners an ideal adaptation during each phase of their run.

Signal classification techniques have long been successfully applied to radio signals (Schmidt, 1986), images (Haralick et al., 1973) and speech signals (Furui, 2004). Recently, examples of classification systems implemented on embedded systems have also been published, for example in (Englehart and Hudgins, 2003; Wolf et al., 2002). Pattern recognition algorithms are lately also applied in sports related problems (Assfalg et al., 2002; von Tscharner and Goepfert, 2003). However, to our knowledge, we are the first group to use these established techniques in order to classify a step signal on an embedded system in the context of sports.

2. Materials and methods

2.1. The adidas_1 running shoe

The adidas_1 is a running shoe which possesses a built-in 8-bit microcontroller, a sensor for heel compression measurement and a motor for cushioning adaptation. This shoe is designed for avid runners, and is constantly adjusting itself to the running situation. In this presentation, we will focus on the classification of the surface that the athlete is running on. While other parameters are also important, it was the first goal of the ongoing research to develop an algorithm that is well suited for surface classification from the sensor signal alone. The general demand to establish constant cushioning when a change of running surface takes place and all other running conditions remain constant is to have

- a soft shoe on hard surfaces (e.g. asphalt, concrete) and
- a hard shoe on soft surfaces (e.g. grass, trail).

Wearing a shoe that provides good cushioning on hard surfaces, for example, can reduce the risk of long-term wearout injuries in the knees. This effect was evidenced by Milgrom et al. (1992). For this reason, we aim at a good classification of any hard surface that the runner is on. Similarly, if we detect a change of surface, i.e. if the surface is not hard anymore, we adapt the shoe to this softer surface condition, making it harder and stiffer in order to prevent injuries that are attributed to a lack of control, e.g. an ankle sprain. We therefore decided not to use a continuous scale from the hardest possible surface to the softest, but rather to employ a hard decision threshold either for the hard or soft surface condition. This can thus also be regarded as a decision for either a 'control' or a 'cushioning' condition and a corresponding complete adaptation of the shoe. This automatic adaptation ideally takes into account the athlete's weight, speed, fatigue level and furthermore the current surface condition, elevation profile and shoe condition.

To facilitate this adaptation, the shoe features a cushioning element, whose ability to give way in vertical direction (hereafter defined as z-axis) can be regulated by a motor-driven cable system. The cushioning element is depicted in Fig. 1. The regulating cable is visible in the z-axis X-ray image of the adidas_1 in Fig. 2. It is running from the motor through the middle of the cushioning element to its opposite end and is fixated there. The motor shown in Fig. 1 can adjust the attenuation setting by turning a screw which lengthens or shortens the cable. When the cable is shortened, the cushioning element is tensed and compresses very little when external forces are applied. When the cable is longer, it allows the cushioning element to compress further by giving it more room to expand in the x-axis direction (forward–backward direction), effectively making the shoe softer. Changes to the softness setting are gradual. The attenuation setting from one extreme to the other is made in 15 increments. A decision for the current surface is made after every fourth step, taking the three preceding steps and the actual step into account by a majority vote. In the case of a tie, no adaptation is made. This is done to maintain the cushioning adaptation mechanism in the case that the runner takes only one or two single steps on a different surface and to save battery power. Thus, to go from the softest setting to the hardest and vice versa, 60 steps of the runner are required. We did not opt for a instantaneous change from one extreme to another once a definite surface change is detected, in order to once again save battery power. A complete change of the cushioning setting from one extreme to another is quite energy consuming. It is more economical to change the setting in small increments. This saves a lot of battery power if the runner only changes surface for a small number of steps, e.g. when running over a small stretch of grass while being mainly on a hard sidewalk surface. Using this approach, we can ensure that the battery (see also next Section 2.2) holds for the complete life-time of a running shoe, which is about 100 h. For more

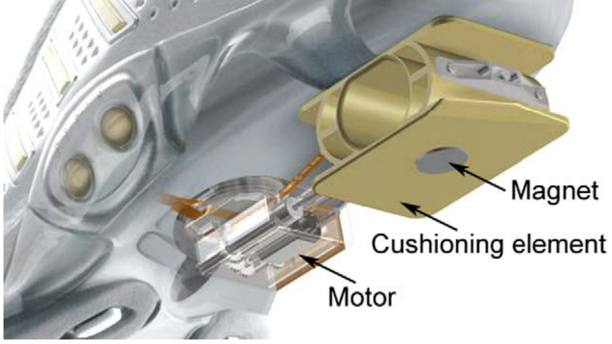


Fig. 1. A view of the adidas_1 shoe, depicting the cushioning element and motor unit. The indicated magnet induces a magnetic field for compression measurement.

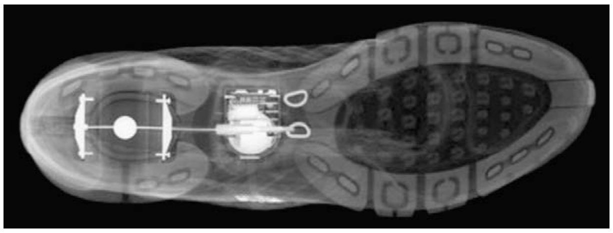


Fig. 2. X-ray image of the adidas_1 in *z*-axis direction. Motor unit, regulating cable and the magnet are clearly visible.

details on the shoe design the reader is referred to DiBenedetto et al. (2004) and DiBenedetto et al. (2005).

2.2. Embedded system hardware

The compression measurement of the adidas_1 shoe is made by a hall sensor that is mounted at the top of the cushioning element. It detects the magnetic field strength induced by a small magnet, see Fig. 1, and can be sampled with a rate f_s of up to 1 kHz. The sensor-magnet distance d_m can then be computed from the magnetic field strength with an accuracy of ± 0.1 mm. A decision whether the attenuation of the shoe has to be adapted is made based on the measured sensor data, see Section 2.3.

The sensor-magnet distance is sampled by the built-in microprocessor that is mounted on a flexible circuit board on the motor element. Currently, a Cypress Semiconductor Corporation controller CY8C21634 is used. However, the methodology that is presented below does not make any special requirements to the employed Microprocessor. The CY8C21634 possesses a clock speed f_{clock} of up to 24 MHz, 512 Bytes of SRAM and 8 kByte flash program store. Additional on-chip system resources include internal oscillators, control and communication interfaces and other highly configurable I/O options. The controller is designed with a standard Harvard architecture and focuses on low power consumption. The whole system is powered by a small 3 V coin cell, which is replaceable and lasts for the normal life-time of a shoe. The CY8C21634 and similar microprocessors are employed in a wide range of embedded applications, examples include automotive solutions and consumer products like handhelds and digital cameras.

2.3. Sensor data

In order to get the data needed for the analysis, there is a special prototype system equipped with a data collection interface. The data from the magnet sensor is stored with a 256 kByte EEPROM array and is evaluated offline in a later stage. An example running signal is depicted in Fig. 3 with the sensor-magnet distance d_m plotted against time t. During the time where the shoe is in the air, the measured signal consists mainly of noise. In contrast, the heel (de-)compression phases of four steps can be distinguished. This measured signal is the basis for the surface classification experiments in Section 3.

Because the signal from the hall sensor consists mostly of noise while the foot is in the air, no relevant information for the cushioning adaptation can be gained. Therefore the sensor system and microcontroller are powered down for 120 ms after registering a compression maximum. Energy consumption during this period is very low, thus the system is saving battery power again. This phase is short enough to ensure that no step is missed when running normally.

2.4. Preprocessing

First of all, we have to extract the specific events that need to be classified in a reliable way. In this context, the events correspond to individual steps which have to be found in the signal. All our features that we will present in Section 2.6 are based on exact identification of these single steps. To facilitate this identification, we first establish a baseline sensor-magnet distance value $d_{m,base}$. This value corresponds to the sensor-magnet distance when the shoe is in the air between steps. It can be reasonably assumed that it is the most frequently occurring value in the data. Next, all sample values that belong to a compressed state are detected. Initial experiments substantiated that compressed states occur when the sample values are below a sensor-magnet distance threshold $d_{m,thres} = d_{m,base} - 1.5 * \sigma_{data}$, where σ_{data} is the overall standard deviation of a dataset.

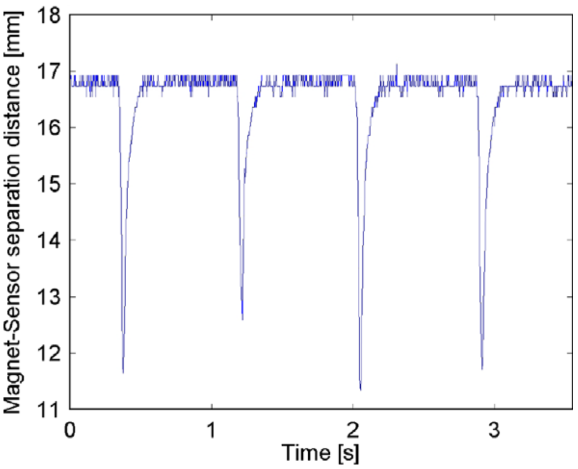


Fig. 3. Data example with 4 step maxima shown.

We define the start and end of the compression phase as those points in the compression states where the distance from $d_{m,base}$ drops below three sample units, which corresponds to 0.7 mm. By using this approach, all steps could be identified in the datasets. This was confirmed by manually extracting 449 steps in 6 datasets and comparing the manual and automatic approaches. The results were completely identical. This result proved that our step detection algorithm provides reliable input for feature computation.

The presented algorithm is rather straightforward, which is a main design criterion for all processing steps for the microcontroller implementation. However, in this case no trade-off had to be made between complexity and accurateness.

2.5. Labeling

In order to learn the necessary parameters for class separation, we needed information about the class membership of the samples. We therefore implemented a graphical user interface for data labeling. The interface is general and can be used for many different labeling tasks. Each event that has to be later classified is assigned to one of the classes manually because we believe this approach to be superior to an automatic labeling. Manual labeling was quite efficiently possible because we could batch label sequences of steps. This is due to the fact that a lot of consecutive steps are made on the same surface when running, i.e. the surface does not change at each step. We were therefore able to label from a start step to an end step of a sequence, assigning all intermediate steps to the same class. Due to the design of our data collection (see Section 3.1), these sequences were easily identifiable. This is because we knew which surface the runners were running on, in consequence we could store the different surface data in separate files. Thus, data labeling was consistently and efficiently possible. The user interface with a sequence of steps labeled as belonging to the soft surface class is depicted in Fig. 4. A labeling interface like this can easily be programmed for a wide range of classification tasks and does not decrease the generality of the approach.

2.6. Feature computation

A set of features that can be used for microprocessor classification has to fulfill two main design criteria. It has to represent the sensor input information consistently while being computationally cheap as it has to be computed on the controller. The choice of features is critical, and "has to be performed for each specific problem to decide which feature of which type one should use" (Ohanian and Dubes, 1992). We therefore manually selected a feature set that is very specific to our task of running surface classification. The selected features should contain the information of the step signal as good as possible. We accordingly computed them such that the important properties of steps are well represented. Our experiments indicated that these features were sufficient because we noticed no improvement using any other imaginable feature. The features are listed in Table 1.

Features 1–11 are calculated on one step alone, with the exception of feature 3, which is computed on two consecutive steps. Features 12–19 are computed on the N preceding steps. The standard deviations σ_{Nf}, where $f = 11, \ldots, 19$, are computed as an unbiased estimator (Fukunaga, 1990) according to

$$\sigma_N = \left(\frac{1}{N-1}\sum_{k=1}^{N}(x_k - \bar{x})^2\right)^{\frac{1}{2}}. \tag{1}$$

The unbiased estimator of the standard deviation has been chosen because σ_{Nf} is computed for a sample drawn from a larger population in our case. Fig. 5 additionally illustrates features 1–10.

The obvious redundancy contained in the extracted features is volitional. It was a goal from the start to use only a subset of the given features to reduce complexity further, thereby using only features with small or no mutual dependence. We will explain our choice for the feature subset selection algorithm in Section 2.8.

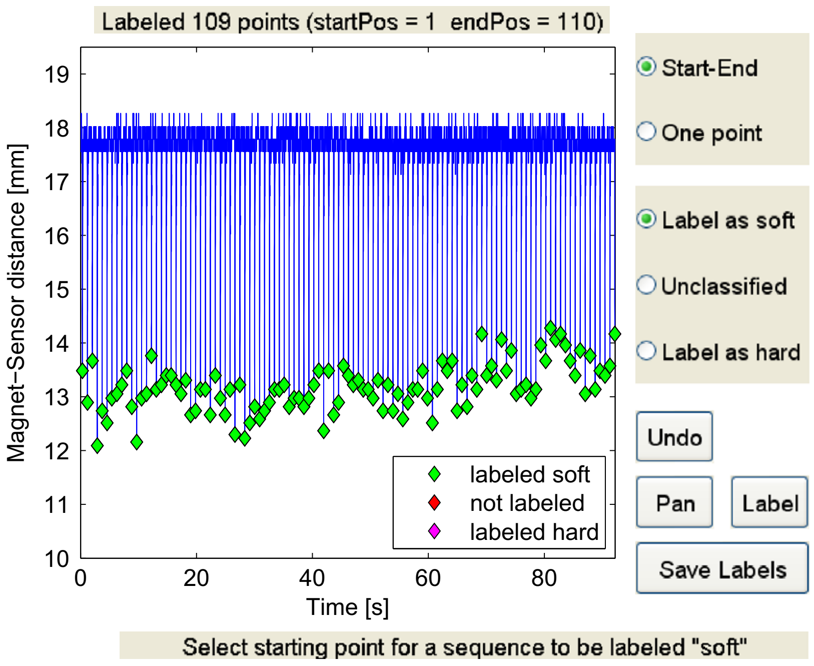
Every single feature can be computed in real-time on the employed microprocessor (Section 2.2). In order to substantiate our other claim that the features dependably represent the signal information, we used them in a different classification task. In (Eskofier et al., 2008), we report on work on running fatigue classification where we successfully applied the same step signal feature set. We also performed experiments that showed that our feature set even outperformed a computationally more demanding feature set successfully applied in biosignal classification. While our set could achieve classification rates of 73.9% in a comparative experiment, only 67.4% were achieved using the more complex feature set.

Table 1
Overview of the features used for classification. SD abbreviates standard deviation. Step compression and decompression refer to the respective phases where the shoe gets compressed and decompressed during heel strike.

Feature number	Feature description
1	Step compression first order least-squares fit
2	Step decompression first order least-squares fit
3	Time between step compression maxima points
4	Time from step compression maximum to step end
5	Time from step start to step compression maximum
6	Step curve area approximation by Trapezoid method
7	Time from step start to step end
8	Step mean value
9	Step median value
10	Step compression maximum value
11	SD of the values contained in one step
12	SD of the step minima (feature 10)
13	SD of the step means (feature 8)
14	SD of the step standard deviation (feature 11)
15	SD of the step duration (feature 7)
16	SD of the step area (feature 6)
17	SD of the time between steps (feature 3)
18	SD of the time to peak (feature 5)
19	SD of the time from peak (feature 4)

2.7. Classifiers

For our intended goal of embedded system classification we focused on classifiers that could be implemented in computationally efficient manner. Once again, this decision was motivated by the


Labeled 109 points (startPos = 1 endPos = 110)

[✓ Start-End] One point

[✓ Label as soft] Unclassified Label as hard

Select starting point for a sequence to be labeled "soft"

Fig. 4. The graphical user interface used for data labeling.

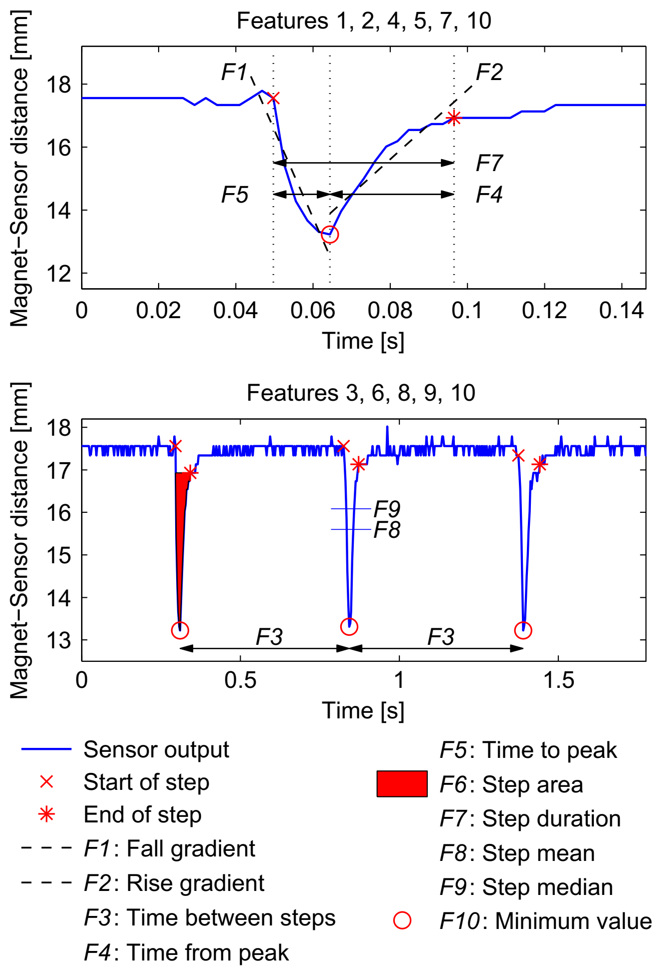


Fig. 5. Depiction of the step signal classification features 1–10.

hardware limitations presented by the embedded system described in Section 2.2. Our approach is to experimentally compare a set of classifiers. Duda et al. (2000) state that if one algorithm is outperforming another one in a particular situation, then this is a consequence of its fit to the particular pattern recognition problem, and not of the general superiority of the algorithm. We therefore want to find a fit to the problem by evaluating each particular classifier's performance. Of course there exists a large set of available classifiers which we can not all individually test, so we reasonably selected such classifiers that are quite well known and widely discussed in the literature. The existence of reference implementations to compare the classifier performance experimentally further motivated our selection. Our choices included:

- Naive Bayes (NB), which is described and experimentally evaluated for example in (Duda and Hart, 1973; Langley et al., 1992).
- Neural Networks (NNet) with one hidden layer and varying number of hidden nodes (Specht, 1990; Duda et al., 2000).
- Nearest Neighbor (NNeigh) classifiers with different number of neighbors, which has seen a lot of applications (e.g. Cover and Hart, 1967; Lee, 1991).
- Support Vector Machines (SVM) using kernels of low complexity (Vapnik, 1998; Duda et al., 2000).
- AdaBoost.M1 (Freund and Schapire, 1996) with decision stumps (Schapire et al., 1998) as weak classifiers.
- Rule-based (Duda et al., 2000) approaches like PART (Eibe and Witten, 1998).
- Linear Discriminant Analysis (LDA), see Fisher (1936) and Duda et al. (2000).

In order to test these classifiers, we could efficiently use the WEKA toolbox, see Witten and Eibe (2005). This toolbox allowed us to compare all different approaches on powerful PC hardware in order to identify the algorithm that is best suited for the microcontroller implementation. Our experiments (see Section 3.2) proved that in our case LDA classification yielded comparable classification rates to other, more complex approaches while being the only approach meeting the real-time criterion. Because of the restricted hardware environment, we therefore decided to train a computationally cheap linear polynomial classifier using LDA. While the theory for other approaches will be omitted here and can be found in the according references, we will give a brief overview of Linear Discriminant theory for the sake of self-sufficiency. LDA classification uses the statistical properties of features, and furthermore provides rather simple linear decision surfaces even in high-dimensional spaces. To accomplish this, LDA transforms the feature space in a way that

- *intra*class variation is minimized, i.e. the features of the same class are as densely packed as possible and
- *inter*class variation is maximized, i.e. distinct classes are as far apart from each other as possible.

For optimal classification, each point $\boldsymbol{x}$ in the d-dimensional feature space gets assigned to the class ω_i, where i denotes the class index, so that the posterior probability $P(\omega_i|\boldsymbol{x})$ is maximized. In our case, we only consider two classes ω_1 and ω_2; however, classification can easily be extended to multiple classes as well. According to LDA theory, the equation for the optimal decision boundary between two classes (see Duda et al., 2000, pp. 117–121) is

$$\boldsymbol{w}^t\boldsymbol{x} + w_0 = 0, \tag{2}$$

where

$$w_0 = -\frac{1}{2}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)^t\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \ln\frac{P(\omega_1)}{P(\omega_2)} \tag{3}$$

is an additive constant and

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_d)^t = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{4}$$

are coefficients for each of the d features used. Here, $\boldsymbol{\mu}_i$ are the d-component class specific mean vectors and $\boldsymbol{\Sigma}$ is the $d \times d$ covariance matrix that is identical for both classes but otherwise arbitrary. $P(\omega_i)$ denotes the prior probability of the class ω_i. The above equations hold only if the class specific densities $p(\boldsymbol{x}|\omega_i)$ can be assumed to be multivariate normal distributions $(p(\boldsymbol{x}|\omega_i) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}))$. In Section 3.1 we show that this assumption is justified.

In summary, the decision rule for two classes ω_1 and ω_2 becomes

$$\boldsymbol{w}^t\boldsymbol{x} \underset{\omega_2}{\overset{\omega_1}{\gtrless}} -w_0, \tag{5}$$

which can be straightforwardly implemented even on a microprocessor. In the case of equality, i.e. $\boldsymbol{w}^t\boldsymbol{x} = -w_0$, an engineering decision for one of the classes has to be made.

2.8. Feature selection methods

Feature selection means identifying a feature subset that delivers good classification rate while reducing the complexity of the overall process. The computation of the 19 features given in Table 1 on the microprocessor would be too time-consuming. Furthermore, the computation would require storing a lot of hall sensor sample values, which is not possible due to memory constraints. Thus, we implemented a method to select the best subset from the original features. This reduction not necessarily decreases the overall classification rate, by deselecting detrimental features the result can even improve. A widely used method for selection or reduction of features is the principal component analysis (PCA), see Duda et al. (2000). It identifies the major axes of variance within the feature space by a Karhunen–Loéve transform. Axes of low variance contribute less to the discriminative ability of the features and can thus be neglected, thereby reducing the feature space. In our case, however, the application of PCA has a major drawback. PCA needs all the input feature values in order to come up with a reduced set of features for classification. This means that the original set of 19 features would have to be computed on the embedded system before it is reweighed by the PCA coefficients. We cannot compute all these 19 features due to the real-time requirement. This means that we have to perform the feature reduction on the original set of 19 features, thus deciding for a subset of features that can directly be computed from our input signal and meets the real-time requirements. However, as we wanted to show whether our feature selection as described below is significantly inferior to feature reduction using PCA, we compared both methods in our experimental chapter (see Section 3.3).

For our proposed selection algorithm the obvious criterion for choosing the best subset is the overall classification rate for a given problem. This result is computed via leave-one out cross-validation to prevent any overfitting effects. One example is presented in the experiments Section 3. In this example, we collected data from 24 runners and performed leave-one-runner-out cross-validation and used LDA training and classification to compute the rates.

Our algorithm follows the principles of a beam search as proposed by Bisiani (1987). During initialization, we use all combinations of two features for training and evaluation of the LDA classifier. A total of $\binom{n_f}{2}$ combinations is tested where n_f is the number of features. The overall classification rates are stored for every feature pair. Subsequently, only a predefined (e.g. 20) number of feature pairs delivering the best results are promoted to the next algorithm step. In this step, the best pairs are combined with each of the remaining features, thus leading to feature triples. This process is iterated, in every iteration the overall classification results are computed, the best combinations are kept and then again combined with the remaining features. For each iteration, a subset is thus identified that delivers the best result amongst those combinations remaining in the pruned search space. While the beam search does not guarantee that the optimal solution is found, it is a very cost-effective search method and ensures a good trade-off between computational complexity and classification rate. Moreover, we could show that in our case the optimal solution and the one identified by the beam search are identical, see Section 3.3.

An important effect of the feature reduction approach is that it gives a very good overview of classification rates for different feature subset sizes. If the hardware framework is not completely specified, a system designer can easily decide what classification rate is necessary for the particular application and give an estimate on computational complexity and thus the required hardware.

3. Experiments

In the following, the experimental evaluation of the proposed surface classifier is presented. The important framework requirements were that the algorithm works for

- different runners (w.r.t. height, weight, running style, training level),
- different shoe sizes and
- all shoe settings (e.g. hard, medium, soft).

We will show that the system for surface classification that was developed works for these conditions. Additionally, the design considerations that specifically aim at meeting the hardware architecture restrictions are given below.

3.1. Collected data

In order to get a sufficient random sample for the subsequent classification experiments, a test course was selected where the desired surface conditions were present. The test course is located on the campus of the Faculty of Engineering of the University Erlangen-Nuremberg. It is depicted in Fig. 6. All runners were asked to run 12 sections of about 150 m each. The test was divided into 2 parts of 6 sections. The first 6 and the second 6 sections were each made with a manually chosen shoe setting. No automatic cushioning adjustment was made to make sure that signal changes only derive from surface or speed changes. The shoe setting was only changed after the first part in order to have data generated with varying cushioning setting, then the running procedure from that part was repeated. The 6 sections of one part were

- two runs on soft surface (grass) with constant speed;
- two runs on hard surface (asphalt) with constant speed;
- one run on changing surface, starting on grass, then switching to asphalt, and finally running on grass again, all with constant speed;
- one run on hard surface with a change in running speed, starting with the same constant speed from the previous sections and accelerating to a fast jog after the first half of the distance.

Each participant was asked to run normally with a comfortable but constant speed for the first 5 sections. Shoe setting, time information and an athlete profile (weight, height, training frequency) was noted for every runner. In addition to the shoe signal, a Polar RS800 system with foot pod was used to get speed and step frequency information.

Altogether, 24 test runners participated in this data collection. Shoes with sizes 7, 9 and 11 were used for those experiments. A total of 106 datasets with different shoe cushioning settings was collected for the subsequent experiments. Steps were extracted using the automatic procedure described in Section 2.4. Table 2 shows the number of valid steps for each of the 24 test runners that were used for the classification experiments. They amount to a total of 22,910 single steps with a fraction of 50.6% on soft surface. The data was labeled as belonging to soft or hard surface using the GUI described in Section 2.5. The distribution of each of the 19 features for the two classes has been tested for normal distribution

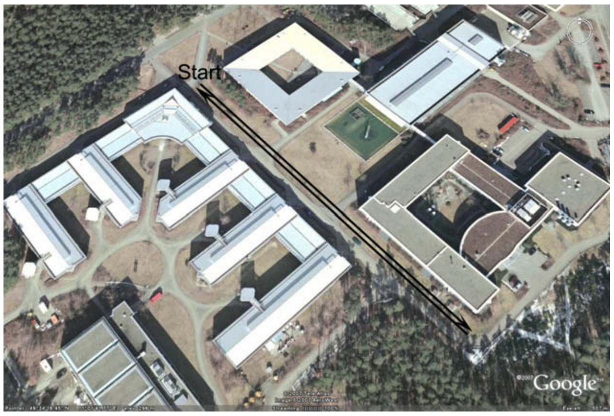


Fig. 6. Aerial view of the test course that was used for data collection.

with a χ^2-test (e.g. Fukunaga, 1990, Ch. 3) after labeling. As a result we could say that the null hypothesis of normal distribution for both classes and each feature is true at a 95% significance level.

3.2. Classifier selection

In order to substantiate our choice of classifier, we tested the performance of the algorithms given in Section 2.7 on our 19-feature set. We used leave-one-runner-out cross-validation to compute the results that are summarized in Table 3. Algorithm settings were as follows. For the NNets, we used one hidden layer and 12 hidden nodes in the layer. Nearest Neighbor classification was performed with $k = 1, 3, 5, 11$ nearest neighbors. For the SVM evaluation, we chose linear polynomial kernels, as more complex kernels could not be implemented on the embedded system anyway. AdaBoost.M1 was tested with decision stumps as weak classifiers. The number of training iterations, N_{it}, was set to 10, 30 and 50. PART was applied with at least 2 instances per rule (212 rules were trained in total). LDA and NB were tested as described in Section 2.7. As can be seen from Table 3, the results for LDA classification yielded comparable classification rates to most other algorithms that were tested. Nearest Neighbor classifiers were the only ones that performed significantly better. This indicates the existence of subclusters in the high dimensional feature space, which can not be correctly classified using linearly-based decision boundaries. Despite this improved performance, in the context of the very restricted memory capacity of only 512 Bytes of the currently employed CY8C21634 microcontroller, NNeigh methods would be impossible to implement. We would need to store and compare with too many single data points or cluster centroids than is feasible. As we will point out in Section 3.3, we already used 98% of the available microcontoller program memory with our current approach using LDA and three features. All the same, we will certainly reconsider the choice of microprocessor in future product generations in the light of these results.

Three other algorithms (AdaBoost.M1 with $N_{it} = 50$ iterations, NNet and PART) outperform LDA less significantly. Our decision not to implement them on the microcontroller was made for computational reasons, too. We will briefly discuss them here. The 50 decision stumps trained by AdaBoost require a more complex decision system and more memory than is currently available. Furthermore, the number of required comparisons is undetermined, leading to a variable decision time. The constant number of operations required for LDA classification is preferable in our case. Even for a simple Neural Net as tested in our case, we would have a lot more multiplications (240 for the described Neural Network versus 19 for LDA). More importantly, we have to evaluate the sigmoid function or a comparable non-linear function. These facts inhibit a NNet implementation on the employed embedded system. PART generates 212 rules with a total of 1305 possible comparisons, which is also impossible to implement on the embedded system that we utilize, given the fact that the available memory is already used with a much simpler solution. We did, however, compare all these results for the different classifiers in order to get good evidence of the performance of our proposed compromise of LDA classification. For other data and framework conditions, we expect that other solutions are more favorable. As we already stated in Section 2.7, is the experimental comparison of different solutions vital for a profound implementation decision.

Table 3
Cross-validated results for different classifiers on the complete 19-feature set.

Classifier	Classification rate (%)
NB	70.2
AdaBoost, $N_{it} = 10$	73.6
AdaBoost, $N_{it} = 30$	75.3
AdaBoost, $N_{it} = 50$	76.0
SVM	75.4
LDA	75.5
NNet	77.9
PART	78.4
NNeigh, $k = 1$	83.3
NNeigh, $k = 3$	84.9
NNeigh, $k = 5$	84.5
NNeigh, $k = 11$	83.6

3.3. Feature selection results

The results of the feature selection algorithm described in Section 2.8 are given in Table 4 (see Table 1 for details on the features). Only the combinations that perform best are shown. For this evaluation, we used the fact that the classification of single steps can be improved when additionally taking a context of preceding steps into account. In this case, a short context of three steps was used by casting a majority vote over the single decisions. In the implementation for the final product solution, a longer context can be used, which leads to even better classification results (see Section 3.6). We finally selected the feature triple 1, 12 and 17 for the implementation on the microcontroller for three reasons. First, it is the best three-feature combination and outperforms the two-feature classifier. Second, with the three-feature implementation we used 7816 Bytes program flash memory of the the embedded CY8C21634 microcontroller. This corresponds to 98% of the available program memory, see Section 2.2. Implementation of a fourth feature would not have been feasible with the selected processor. The third reason for the implementation decision was that we could show that even with calculating the features and classification decision, we could still sample with maximum sample rate and therefore meet the real-time computation criterion.

As we already stated in Section 2.8, the beam search does not guarantee that the identified subset performs optimal. We therefore computed the classification rates for all 1140 possible three-feature combinations. We could thereby show that the selected feature triple represents the optimal solution. We also compared the results of our feature selection with feature reduction using PCA as described in Section 2.8. A comparison of the classification rates of both methods for reduction to two to seven features is depicted in Fig. 7. For our data, it can be seen that our feature selection method outperforms PCA. In the comparable case of reduction to three features, a classification rate of 73.9% could be achieved with PCA and 76.3% using beam search.

Table 2
Individual identifiers for the 24 test runners with number of valid steps given for each of them. Shoe size of each participant is given in brackets.

ABo	(7)	1307	DE	(11)	1121	KR	(9)	1326	RS	(11)	384
ABr	(9)	1152	EK	(7)	961	MA	(11)	898	SK	(7)	1273
AM	(11)	1338	HH	(9)	936	MP	(9)	914	SW	(9)	1240
BD	(11)	781	JM	(11)	541	MS	(9)	627	TS	(11)	612
BE	(11)	1206	JP	(9)	1013	MW	(11)	1165	TT	(11)	791
CD	(11)	911	KH	(11)	903	RB	(11)	670	VD	(11)	840

Table 4
Results for the first 7 iterations of the feature selection algorithm.

Selected features	Classification rate (%)
1, 12	75.4
1, 12, 17	76.3
1, 2, 14, 17	76.9
1, 2, 5, 14, 17	77.0
1, 2, 7, 12, 13, 17	76.9
1, 2, 5, 14, 15, 16, 17	76.8

The confusion matrix for the selected feature combination is given in Table 5. Sensitivity is 77.7% and specificity is 73.6%. This result shows that no class is significantly favored over the other.

The characteristics of the selected feature subset become very clear when visualizing the three-dimensional feature space. Runners on soft surface generally have

- smaller compression gradient (feature 1);
- higher step minima deviation (feature 12) and
- higher interstep time deviation (feature 17)

compared to runners on hard surface.

3.4. Classifier implementation

With the three-feature subset described in Section 3.3 we additionally performed experiments using all classifiers presented in Section 2.7 to confirm our choice of the LDA classifier. For cross-validation, 24 subsets were used, each consisting of the samples of one individual runner. The results of these experiments are presented in Table 6. Algorithm settings are the same as the ones given in Section 3.2, with the exception that only 4 hidden nodes were used for the NNet and that PART produced only 32 rules on the reduced feature set. It can be seen in Table 6 that only the Neural Network slightly outperforms the Linear Discriminant Analysis. However, the gain in classification rate is not statistically significant. Moreover, the problem of the complexity of the NNet classifier as already described in Section 3.2 remains, an implementation on the CY8C21634 microcontroller is thus not possible. We therefore decided to use the LDA classifier in the final application.

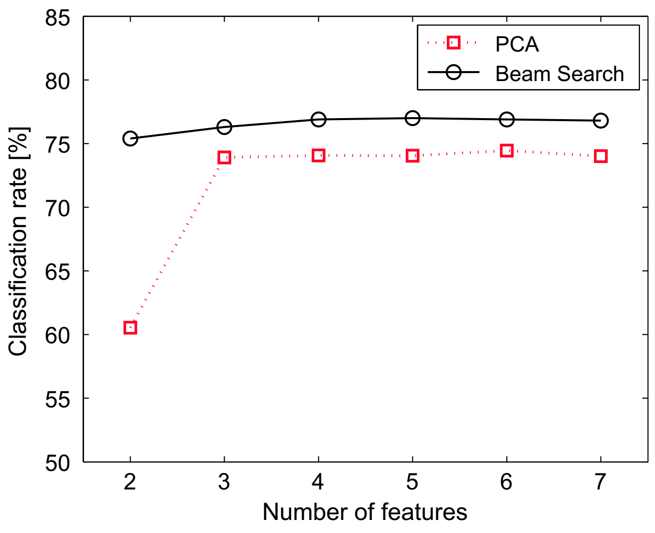


Fig. 7. Comparison of feature reduction using PCA and feature selection using beam search.

Table 5
Classification confusion matrix for the feature combination 1, 12 and 17.

	Class soft	Class hard
Classified as soft	9000	2993
Classified as hard	2583	8334

Table 6
Results of the experiments on the selected three-feature set using different classifiers. Classification rates are computed with a short context of three steps.

Classifier	Classification rate (%)
NNeigh, $k = 1$	71.2
NNeigh, $k = 3$	73.1
NNeigh, $k = 5$	75.3
NNeigh, $k = 11$	75.3
AdaBoost, $N_{it} = 10$	72.4
AdaBoost, $N_{it} = 30$	74.0
AdaBoost, $N_{it} = 50$	74.5
NB	74.6
PART	75.5
SVM	76.1
LDA	76.3
NNet	76.5

3.5. Effect of runner-dependent parameters

We already stated in Section 3 that our algorithm has to work independently of the runner, i.e. for different runner height, weight, shoe size and training level. The surface classification also has to work for different individual running speeds. We collected this information for all test participants. Table 7 shows the ranges for the parameters, as well as means μ and standard deviations σ. The training level was derived from the sports activity frequency of each individual on a range of one to four. On this scale, one indicates low or no regular sports activity whereas four stands for high running relevant activity, e.g. for marathon runners.

For the evaluation we computed the Spearman rank order correlation coefficient r_{spear} (Spearman, 1904) of the individual parameters and the classification rate of each runner. These correlation coefficients are ± 1 for perfect positive or negative correlation and 0 if the samples are uncorrelated. The results of the evaluation are given in Table 7. In our case of 24 value pairs the null hypothesis that the samples are uncorrelated has to be rejected for $|r_{spear}| > 0.359$ at the 95% significance level (Olds, 1938). The results for r_{spear} given in Table 7 are all below this value and show that the value pairs are uncorrelated. If there was a significant correlation, we would have to assume that the runner-dependent parameters have some kind of influence on the recognition. For instance, the decision threshold would have to be adjusted for lighter runners if there was a correlation. However, by showing that there is no correlation between the individual parameters and the recognition rate we could assure that classification with the proposed system works independently of the runner.

Table 7
Individual runner parameters. Ranges are given for each parameter, as well as mean and standard deviation. The Spearman correlation r_{spear} of the individual parameters and the classification rates of each runner is also given.

Parameter	Range; mean; standard deviation	r_{spear}
Height [cm]	[156;196]; $\mu = 181.6$; $\sigma = 10.5$	0.00
Weight [kg]	[46;125]; $\mu = 77.3$; $\sigma = 15.3$	−0.19
Shoe size [US]	{7,9,11}; $\mu = 9.9$; $\sigma = 1.4$	0.06
Training level	{1,2,3,4}; $\mu = 2.3$; $\sigma = 1.1$	−0.03
Runner mean speed [km/h]	[8.3;15.0]; $\mu = 12.0$; $\sigma = 1.4$	−0.15

Table 8
Description of the datasets that were used for the final evaluation on the microcontroller.

Dataset description	Number of recorded steps	Ratio (hard surface) (%)	Classification rate (%)
Park, grass and concrete	3480	61.5	82.8
Only asphalt surface	995	100	92.0
Forest soil, no inclination	4438	0	90.8
Forest soil and asphalt, running up/downhill	4448	65.9	80.3

3.6. Final evaluation on the microcontroller

It was important to implement our classification algorithm on the microcontroller that is employed in the product to verify our results. For these control experiments, we used the internal EEPROM (see Section 2.3) to store for each step only the classification decision derived with the described classifier. Longer contexts of 16 steps were used for the implementation. We let the test participants run totally freely, i.e. no requirements on running speed, step frequency or other parameters were made. An external observer counted and wrote down the number of steps on each of the surfaces that were tested during this evaluation. Thus, we could straightforwardly determine the classification rate. Table 8 shows the results of these experiments. Classification rates of more than 80% could be achieved.

4. Summary

For the realization of accurate surface classification using sensor output from the adidas_1, data was collected from 24 test runners on hard and soft surface. This data was labeled, and 19 features were extracted which were chosen because they consistently represent the step information. A classification system using Linear Discriminant Analysis was then proposed. Using the classification rate as a criterion, a subset of three features was found that is suited to be implemented on the embedded system that is integrated in the running shoe. The system was evaluated with regard to the parameters shoe cushioning setting, runner height, runner weight and runner training level, which were all found to have no important effect on the accuracy of the classifier. The described classifier has been implemented in the current version of the adidas_1 running shoe.

During the complete analysis procedure, no assumptions regarding sampling rate of the sensor, memory or clock frequency have been made. This makes the methodology applicable to many other problems which require accurate embedded classification. The first important point is that computationally cheap features can be identified that well represent the sensor information. Secondly, a classifier has to be determined that establishes a good trade-off between complexity and classification rate. Feature reduction is compulsory to provide a feature subset that is best suited for the problem at hand. Lastly, the result of the analysis that has been made on computationally powerful PC hardware has to be verified on the embedded system itself.

5. Future work

First results indicate that other important conditions can be classified using the shoe signal. One example includes the state of fatigue of a runner. An adaptation of the shoe hardness setting to a fatigued condition is definitely imaginable. Additionally, we will analyze the effect of elevation profile and speed changes in order to be able to classify these parameters, too.

We will furthermore investigate other application areas, for example accurate classification on microcontrollers in household appliances or for mobile phones. In the latter case, the computational framework conditions are not as critical as for microcontrollers. Still, a lot of effort similar to the one presented in this work has to be made to be able to implement pattern recognition algorithms on this kind of hardware.

References

Assfalg, J., Bertini, M., Colombo, C., Bimbo, A.D., 2002. Semantic annotation of sports videos. IEEE Multimedia 9 (2), 52–60.

Bisiani, R., 1987. Beam search. In: Shapiro, S.C. (Ed.), Encyclopedia of Artificial Intelligence. Wiley-Interscience, New York, NY, USA, pp. 55–58.

Cover, T., Hart, P., 1967. Nearest neighbor pattern classification. IEEE Trans. Inf. Theory 13 (1), 21–27.

DiBenedetto, C., Oleson, M.A., Roth, C., Thompson, M.C., 2004. Intelligent Footwear Systems. US Patent 20040177531.

DiBenedetto, C., Oleson, M.A., Roth, C., Thompson, M.C., 2005. Intelligent Footwear Systems. European Patent 1582108.

Duda, R.O., Hart, P.E., 1973. Pattern Classification and Scene Analysis. John Wiley and Sons Inc., New York, NY, USA.

Duda, R.O., Hart, P.E., Stork, D.G., 2000. Pattern Classification, second ed. Wiley-Interscience, New York, NY, USA.

Eibe, F., Witten, I.H., 1998. Generating accurate rule sets without global optimization. In: Shavlik, J. (Ed.), Machine Learning: Proc. 15th Internat. Conf.. Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 144–151.

Englehart, K., Hudgins, B., 2003. A robust, real-time control scheme for multifunction myoelectric control. IEEE Trans. Biomed. Eng. 50 (7), 848–854.

Eskofier, B., Hoenig, F., Kuehner, P., 2008. Classification of perceived running fatigue in digital sports. In: 19th Internat. Conference on Pattern Recognition, 2008 (ICPR 2008), Tampa, FL, USA, pp. 1–4.

Fisher, R., 1936. The use of multiple measurements in taxonomic problems. Ann. Eugenics 7, 179–188.

Freund, Y., Schapire, R.E., 1996. Experiments with a new boosting algorithm. In: Proc. 13th Internat. Conf. on Machine Learning. Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 148–156.

Fukunaga, K., 1990. Introduction to Statistical Pattern Recognition, second ed. Academic Press, San Diego, CA, USA.

Furui, S., 2004. Fifty years of progress in speech and speaker recognition. Acoust. Soc. Am. J. 116 (4), 2497–2498.

Hacker, C., Batliner, A., Nöth, E., 2006. Are you looking at me, are you talking with me – multimodal classification of the focus of attention. In: Sojka, P., Kopecek, I., Pala, K. (Eds.), Text, Speech and Dialogue: Proc. 9th Internat. Conf.. Springer Verlag, Berlin/Heidelberg, Germany, pp. 581–588.

Haralick, R.M., Shanmugam, K., Dinstein, I., 1973. Textural features for image classification. IEEE Trans. Syst. Man Cybernet. 3 (6), 610–621.

Langley, P., Iba, W., Thompson, K., 1992. An analysis of bayesian classifiers. In: Artificial Intelligence: Proc. 10th National Conf.. AAAI Press, pp. 223–228.

Lee, Y., 1991. Handwritten digit recognition using K nearest-neighbor, radial-basis function, and backpropagation neural networks. Neural Comput. 3 (3), 440–449.

Lun, V., Meeuwisse, W.H., Stergiou, P., Stefanyshyn, D., 2004. Relation between running injury and static lower limb alignment in recreational runners. Br. J. Sports Med. 38 (5), 576–580.

Milgrom, C., Finestone, A., Shlamkovitch, N., Wosk, J., Laor, A., Voloshin, A., Eldad, A., 1992. Prevention of overuse injuries of the foot by improved shoe shock attenuation. A randomized prospective study. Clin. Orthop. Relat. Res. 23 (281), 189–192.

Ohanian, P.P., Dubes, R.C., 1992. Performance evaluation for four classes of textural features. Pattern Recognit. 25 (8), 819–833.

Olds, E.G., 1938. Distribution of sums of squares of rank differences for small numbers of individuals. Ann. Math. Statist. 9 (2), 133–148.

Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S., 1998. Boosting the margin: A new explanation for the effectiveness of voting methods. Ann. Statist. 26 (5), 1651–1686.

Schmidt, R., 1986. Multiple emitter location and signal parameter estimation. IEEE Trans. Antennas Propag. 34 (3), 276–280.

Spearman, C., 1904. The proof and measurement of association between two things. Am. J. Psychol. 15 (1), 72–101.

Specht, D.F., 1990. Probabilistic neural networks. Neural Networks 3 (1), 109–118.

Vapnik, V.N., 1998. Statistical Learning Theory. Wiley-Interscience, New York, NY, USA.

von Tscharner, V., Goepfert, B., 2003. Gender dependent EMGs of runners resolved by time/frequency and principal pattern analysis. J. Electromyogr. Kinesiol. 13 (3), 253–272.

Witten, I.H., Eibe, F., 2005. Data Mining: Practical Machine Learning Tools and Techniques, second ed. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Wolf, W., Ozer, B., Lv, T., 2002. Smart cameras as embedded systems. Computer 35 (9), 48–53.