J Multimodal User Interfaces (2015) 9:211–221
DOI 10.1007/s12193-015-0194-3

ORIGINAL PAPER

LMNN metric learning and fuzzy nearest neighbour classifier for hand gesture recognition

Tea Marasovic[1] · **Vladan Papic**[1] · **Vlasta Zanchi**[1]

Received: 25 March 2014 / Accepted: 27 July 2015 / Published online: 27 August 2015

Abstract This paper presents a novel gesture recognition system using a single three-axis accelerometer, that is to serve as an alternative or supplementary interaction modality for controlling mobile devices. Capturing, training and classification of the detected hand gestures are expected to be executed in their entirety on the mobile device running the proposed system, instead of being passed to a nearby computer. As gesture recognition belongs to the group of pattern recognition problems where the underlying class probabilities are not a priori known, the classification is based on the distance between neighbouring examples. The distance metric is optimized by using large margin nearest neighbour (LMNN) method. To measure the amount of classification confidence, a fuzzy version of nearest neighbour algorithm is employed. Obtained results for recognition of nine hand gestures using proposed LMNN—fuzzy combination are presented and compared to that of other similar approaches. The system achieves near perfect recognition accuracy that is highly competitive with systems based on statistical methods and other accelerometer-based gesture recognition systems in the literature.

Keywords Accelerometer · Gesture recognition · Mobile devices · Human–computer interaction

✉ Tea Marasovic
tmarasov@fesb.hr

Vladan Papic
vpapic@fesb.hr

Vlasta Zanchi
vzanchi@fesb.hr

[1] Department of Electrical and Computer Engineering, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, Rudjera Boskovica 32, 21000 Split, Croatia

1 Introduction

With mobile computing having an increasingly important role in enhancing the quality of life, an increasing amount of research nowadays is directed towards more physical, intuitive and pervasive human–computer interaction. Most modern mobile devices share a common problem—they are attempting to provide users with powerful computing services and resources through very small interfaces. The device's shrunken form factors and physical size limitations in both keyboards and screens, make operating with these devices a rather cumbersome task and are frequently stressed as main sources of dissatisfaction and frustration for their users [2,3]. In this regard, gesture-based interaction can be seen as a viable alternative for controlling devices.

Up to date, gesture-based interaction has been implemented utilizing many different sensing technologies, including cameras and computer vision techniques, as well as devices such as magnetic field trackers, instrumented (data) gloves and body suits, attached to the user. Each of these technologies varies in accuracy, resolution, latency, range of motion, user comfort and cost [20]. Glove-based gestural interfaces [4] typically require the user to wear a cumbersome device and carry a bundle of cables connecting the device to the computer. This hinders the ease and naturalness of the user's interaction with the computer. While overcoming this problem, gesture recognition systems based on computer vision techniques need to contend with other problems related to occlusion of user's body parts and immobility. They tend to work well in controlled environments, but are not suitable for everyday use [13].

Identifying the movement (or "gesture") made by the user when holding the device from acceleration data is an emerging technique for gesture-based interaction, enabled by the rapid development of low-cost Micro Electro-Mechanical

System (MEMS) technology, which has resulted with a growing number of new generation consumer electronic devices that began to incorporate small and precise sensors, like accelerometers and gyroscopes. This allowed the development and exploration of new input modalities, in which the user interacts with the device not by traditional button pressing, but by rotating and moving the device, effectively turning it into a physical interface. Such modalities add new degrees of freedom to interaction and also make interaction calmer by reducing cognitive load.

1.1 Related work

Though majority of the available literature on gesture or action recognition combines data from a triaxial accelerometer with data from another sensing device, like a biaxial gyroscope [24], in order to improve the systems performance, there has also been a great deal of publications where solely accelerometer-based gesture recognition has been discussed, most prominent one being that by Hofmann et al. [11]. Most of the systems presented in these papers utilize continuous or discrete hidden Markov models based recognition algorithms [14,17,22,23], although other methods, such as conditional Gaussian models, support vector machines and Bayesian networks, have also been explored.

The design of a gesture recognition system can follow a user-independent or a user-dependent approach. The difference lies in whether the user has to train the system before being able to actually utilize it. User-dependent systems require the user to repeat the gesture movements several times to train the system; alternatively, user-independent systems are oriented to general users and do not need a training phase before being usable.

Liu et al. [16] developed μWave, a user-dependent accelerometer-based system that supports personalized gesture recognition, working on the premise that human gestures can be characterized by the time series of forces measured by the hand-held device. The core of the μWave is dynamic time warping (DTW), which measures the similarity between two sample sequences, varying in time or speed. The system's database undergoes two types of adaptation: positive and negative; however, since the database adaptation resembles a continuous training, in some cases removing an older template every other day might lead to replacing a very good representative of a gesture, which is best avoided. Despite this, μWave demonstrates both computational as well as recognition efficacy, and produces an accuracy of 98.6 % for a dictionary of 8 gestures.

An amalgamation of DTW and affinity propagation is the foundation of yet another novel accelerometer-based gesture recognizer [1], that utilizes the sparse nature of gesture sequences. After projecting all candidate gesture traces onto the same lower dimensional subspace, the recognition problem is formulated as an ℓ_1-minimization problem. The system can be implemented for user-dependent recognition, mixed-user recognition and user-independent recognition. In mixed-user recognition mode, the system yields a very respectable accuracy of 98.71 % for a dictionary of 18 gestures.

Xu et al. [28] presented a user-independent gesture recognition system, capable of recognizing seven hand gestures, based on the input signals from the Micro Inertial Measurement Unit (μIMU), which is essentially a MEMS triaxial acceleration sensing chip integrated with data management and Bluetooth wireless data chips. Their recognition system consisted of sensor data collection, segmentation and recognition. They have examined three different gesture recognition algorithms, including: (1) sign sequence and Hopfield based gesture recognition algorithm; (2) velocity increment based gesture recognition algorithm; and (3) sign sequence and template matching based gesture recognition algorithm; all of which were implemented and executed on PC. The best of the three algorithms, the one based on sign sequence and template matching, achieved an overall mean recognition accuracy of 95.6 %.

In this paper, we propose a concept of a novel gesture recognition system, based on the use of a single triaxial accelerometer sensor. Our system targets only user-dependent gesture recognition, due to the difficulties in user-independent gesture recognition originating from the great variation between different users, even for the same predefined gesture. What distinguishes our approach from most of the aforementioned existing systems is that, in the herewith presented system, all sensing and capturing of the data, as well as gesture recognition and training, should to be done on the mobile device itself. Most of the other approaches transmit the sensor data to a nearby computer, where the training and recognition software is running, or the off-line data analysis is performed. The proposed system uses distance metric learning—a recent technique that has not yet been exploited in this particular field—combined with fuzzy k-nearest neighbour rule to improve the results of multi-class gesture recognition and classification. The offline validation using an arbitrarily defined gesture vocabulary of 9 test gestures, has shown that the proposed system yields extremely competitive results in terms of both recognition accuracy and computational complexity.

This article presents the further extension of our work, published in [18], with significant improvements which reflect primarily in (1) the introduction of new feature descriptor and (2) the employment of fuzzy version of nearest neighbour classification algorithm, as a means of measuring the level of classification confidence. The remainder of it is organized as follows. Section 2 describes the proposed gesture recognition system set-up and gives an overview of its baseline functionality, whilst Sect. 3 provides a detailed

overview of the employed feature extraction scheme. Section 4 further introduces the general problem of distance metric learning for nearest neighbour classification and provides a short theoretical background to the fuzzy k-nearest neighbour rule and the corresponding membership assignment techniques. The results of simulations, conducted in order to evaluate the system's performance, are reported in Sect. 5. Finally, Sect. 6 ends the paper with concluding remarks.

2 Problem setup

Building a gesture communication interface for a mobile platform, is a daunting task, since there are several major general usability requirements that have to be met. Such a system should be available anywhere and any time. All the while, it should be unobtrusive (or inconspicuous), meaning that the end-user should barely notice that he/she is using it. Given that hand gestures suffer from temporal variations, i.e. they differ from one person to another and even the same person cannot perfectly replicate the same gestures, within the system gestures should be freely trainable, with minimum training effort. Data samples that do not match one of the predefined gestures well enough should not be classified as one of them, but instead stated separately as unclassifiable, unknown gesture.

Another key system characteristic that has to be carefully considered is its response time. Whenever the user inputs a gesture in the system, the actions tied to that gesture have to be executed almost instantly. Usability engineering books [21] suggest that response time should be as low as 100 ms. System response time greatly depends on the available resources, such as storage space and CPU performance, of the mobile device platform it is running on. The latest generations of mobile devices boast powerful processors and other advanced features that up to a few years ago belonged to the realm of desktop computers, which makes them capable of running complex and computationally demanding applications. However, due to their size and battery requirements, even today's most evolved models have constraints that limit their response.

Figure 1 visualizes the flow of the proposed gesture recognition system. Users freely move their mobile device (e.g. a smartphone) with a built-in accelerometer in space, to signal a particular gesture; thus, temporarily variable, 3-dimensional acceleration data signals are obtained. The system consists of a knowledge database, that stores several sets of gestures and the corresponding execution actions, and a gesture recognizer algorithm, that picks up the traced data and uses it to identify the gesture intended by the user. During the detection stage, the data is being processed to filter out any noise which might have accumulated in it, upon which a feature extraction process, that translates the original acceleration signals into a corresponding vectors of features, is started.

The system's operation is carried out in two distinct modes. In teaching mode, gesture example feature vectors are simply saved to the knowledge database with an appropriate label attached, so that they could, at some point, be used for metric learning. To ensure that the device's primary functionality remains unhindered, distance metric learning is realized as a background process, which triggers on only when the device is in stand-by mode. In the recognition mode, the gesture example feature vectors are passed on to the classifier component that re-evaluates the assigned fuzzy class memberships of the known gesture examples, stored in the knowledge database, and identifies the most probable gesture. Finally, the result of the gesture recognition process is displayed to the user and the corresponding actions are executed.

2.1 Device and sensing

The gesture traces are acquired using Android-operated Samsung i9000 *Galaxy S* smartphone, equipped with Bosch BMA-023 MEMS accelerometer. At $3 \times 3 \times 0.9$ mm, it is a

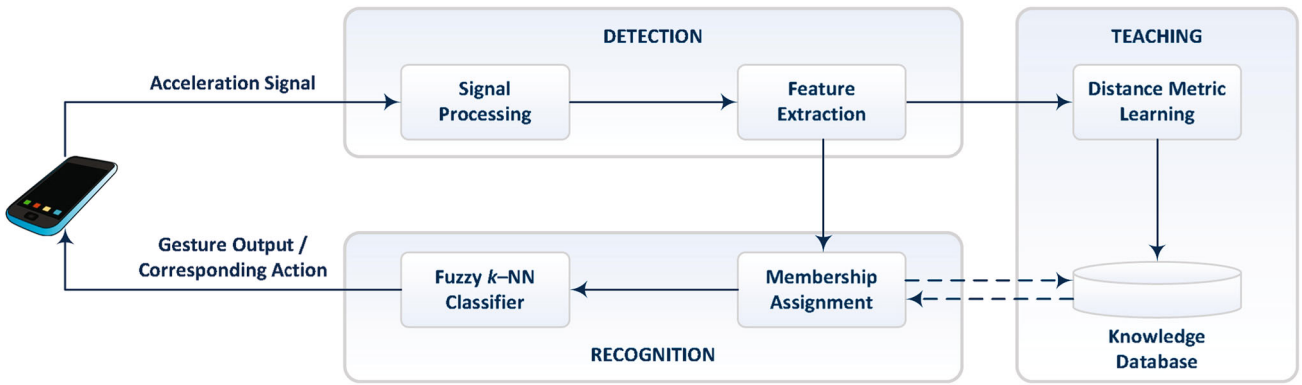

Fig. 1 Gesture recognition system functionality flowchart

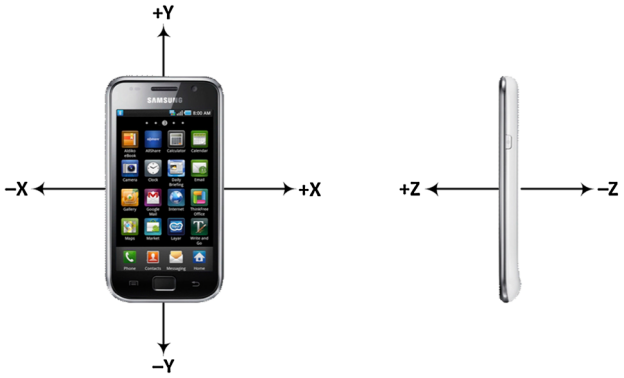

Fig. 2 Orientation of the Samsung i9000 accelerometer axes

tiny, low power, high performance, triaxial digital accelerometer with a nominal range of $\pm 2g$. The alignment of the sensor's axes with the device casing is such that—with the phone lying flat on a perfect horizontal surface in front of the user, oriented so that the screen is readable by the user in normal fashion—the x axis goes from left to right, the y axis goes from the user toward the device and the z axis is perpendicular to the surface (Fig. 2). The acceleration values, recorded in meters per second squared, are delivered in a form of a 3-valued vectors of floating point numbers.

2.2 Data processing

Threshold filtering Considering the relatively high accelerometer sensitivity and the average data sampling rate of approximately 30 Hz, which is well above the Nyquist sampling frequency, consecutive data vectors are almost identical when the data is produced by the natural hand movement. Therefore, the first processing step consists of removing all similar vectors from the input data set to speed up the consequent calculations. Vectors are eliminated if their components at time instant t are very close to the vector's component at instant $t-1$, i.e. if $|x_t - x_{t-1})| < \epsilon$. The threshold value ϵ is determined empirically and for this system it is set at the value $\epsilon = 0.055$.

Noise reduction There are two primary sources of noise in the received signal. The first are irregular sampling rates and the second is the noise inherent in discrete physical sampling of a continuous function.

The accelerometer data is likely irregularly sampled because of the Android's implementation of the sampling mechanism. On an Android platform, the accelerometer sensor data can be obtained by using the Android sensor framework, which include various classes and methods for accessing sensors, registering sensor event listeners and retrieving raw sensor data. The Android API offers four abstract sampling frequencies for its accelerometer sensor (listed from fastest to slowest): *Fastest*, *Game*, *Normal* and *UI*. However, the actual physical sampling capabilities of the accelerometers vary from device to device, so these sampling frequencies are used more as guidelines than as actual physical sampling frequencies. Another reason for irregular sampling lies in the manner in which applications on Android framework receive raw acceleration readings. The Android API allows the acquisition of acceleration samples on either the *onAccuracyChanged()* or *onSensorChanged()* event, which fire whenever the Android OS determines that the sensor accuracy or its values have changed. There is no public method for determining the rate at which the sensor framework is sending sensor events to an acceleration sampling application. The delay that is specified is only a guideline and it can be altered depending on the varying loads of activity of the Android OS, which in turn results with irregularly sampled values.

To handle irregularities in the accelerometer sampling frequency, a data linearisation process is used. The process involves choosing a desired sampling frequency and interpolating all the holes in the signal via linear interpolation. The linearised values are calculated by finding the closest sampled data points before and after the desired sampling times, and then interpolating what the values of the desired sample times would have been. One significant problem in the linearisation process is determining the optimal sampling frequency.

To ensure that not too much data is calculated via interpolation—which could result in a false recreation of the original signal—a program that determines the mean and minimum differences in time (in milliseconds) of subsequent readings was created. The arithmetic average of these two values was then calculated and analysed for the entire dataset in order to provide a general guideline of what might be an ideal sampling frequency. Based on this analysis, it was concluded that one sample every 21 ms or approximately 48 samples per second was a good enough desired sampling frequency for our data. It should be underlined that, though this particular choice of sampling frequency will cause the signal to almost double in its size, all the key features of the signal will remain unchanged. This is expected since data linearization is simply meant to fill the holes in the sampled signal and should not in any way significantly alter the signal itself. The signal length (i.e., number of samples) will be doubled because we decided to use the arithmetic average of the minimum and mean sampling time differences, which in this scenario corresponds to about half the mean difference.

Outside of irregular sampling frequencies, additional noise that can come from any number of sources is introduced in the signal. For example, a slight change in the phone orientation or a screen tap can result in anomalous peak in the corresponding signal. A 5-point smoothing algorithm was applied in order to reduce this type of noise in the obtained, linearised signal. This algorithm calculates each point to be

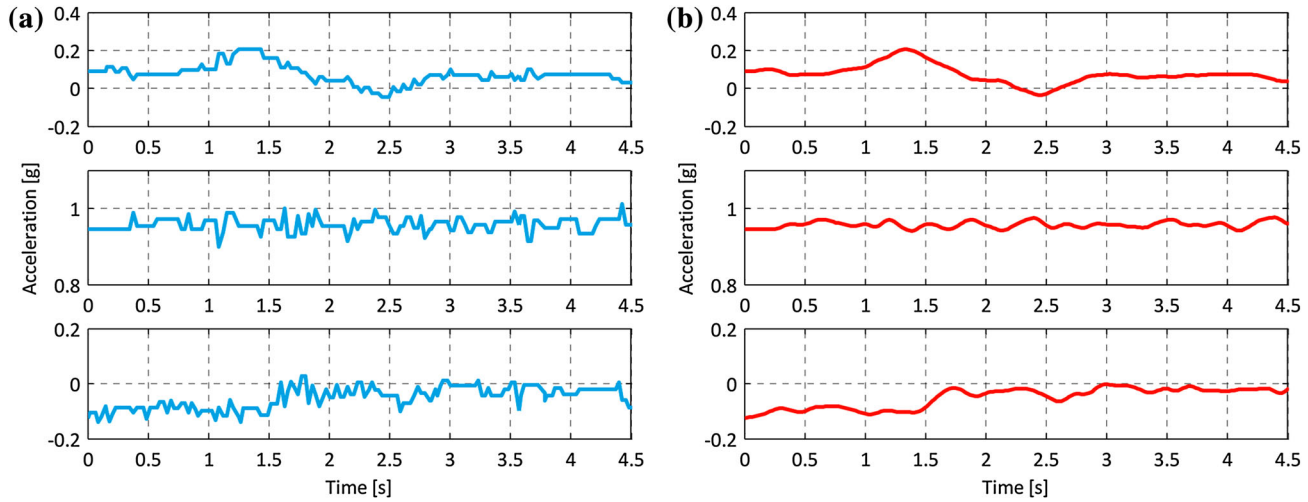

Fig. 3 **a** Raw vs. **b** Processed acceleration waveforms, along *x*- (*top*), *y*- (*middle*) and *z*- (*bottom*) axis, defining a left-to-right straight line gesture

the average of its four nearest neighbouring points, the two nearest before and the two nearest after. The 5-point smooth was chosen so that spikes with an observable, steady progression would be preserved, while anomalous, sudden spikes would be eliminated.

Figure 3 shows the acceleration waveforms along all three physical axis, before and after processing, for an upright-held device performing a left-to-right straight line gesture.

3 Feature extraction

Finding a compact, yet effective set of predictive features is a very important step in pattern recognition, since it can greatly reduce the data sensitiveness to intra-class variation and noise, thus improving the recognition performance. In our paper [19], we have shown that very good results in terms of gesture recognition accuracy can be obtained by using a mixed feature scheme, combining "blind", data-driven, statistical signal features—namely, gesture duration, Pearson's correlation coefficient, mean, standard deviation, energy and entropy of acceleration along individual axis—with our newly introduced, special-purpose feature descriptor, named histogram of directions (HoD), which is particularly suited for the task of accelerometer-based gesture recognition.

3.1 Histogram of directions

The HoD feature descriptor is based on a notion that the performed gesture shape and appearance can be clearly distinguished by the distribution of input device's movement directions. The direction in which the device has been moved is determined by the angle that its resultant velocity vector forms with the horizontal axes. Owning to the random noise associated with accelerometers, error-reduction techniques need to be employed, prior to actual direction detection, to compensate for velocity-drift.

When creating the movement direction histogram, each time sample casts a weighted vote for the direction histogram channel based on the orientation of the resultant velocity vector at the particular time instant k. The votes are accumulated in direction bins spread evenly over 0°–360°. As for the vote weight, sample contribution can be expressed in a form of some function of acceleration magnitude, $|A(k)|$; however, in actual tests, the magnitude itself generally produced the best results. In order to account for the noise and variation of gesture data, the resulting descriptor representation is further normalized by using the L2-normalization scheme.

The HoD descriptor has several key advantages. It captures the stroke structure that is very characteristic of gesture shape and produces a relatively simple gesture representation with an easily controllable degree of invariance to geometric transformations: tilts of the input device make little difference as long as they are smaller then the orientation bin size. It should be noted though that, in its current implementation, HoD feature descriptor is time-independent and, for that reason, for example, gestures up-left-down-right and up-down-left-right would yield roughly the same direction histograms.

4 Distance metric learning and classification algorithm

The k-nearest neighbour is one of the oldest and simplest machine learning algorithms for pattern classification. Following the k-nearest neighbour decision rule, unlabelled examples are classified by the majority vote of their k nearest neighbours in the training set. Needless to say, the accuracy of this algorithm depends strongly on the manner in which

distances between different examples are computed. With no prior knowledge available, most k-nearest neighbour implementations use simple Euclidean distances as a measure of similarity between different examples.

Seeing the Euclidean distance metric as overly simplistic, many researchers have begun to ask how to improve on ad-hoc or default choices of distance metric [6,9,27]. Distance metric learning is an emerging sub-area of machine learning in which the goal is to adapt the underlying metric in order to achieve better results of classification and pattern recognition. In many applications, a simple yet effective strategy is to replace the Euclidean metric with so-called Mahalanobis distance metric, which computes the squared distance between vectors x and y as:

$$d^2_{\mathbf{M}}(x, y) = (x - y)^T \mathbf{M}(x - y) \tag{1}$$

where the matrix $M \geq 0$, that directly represents the Mahalanobis distance metric itself, is required to be positive semidefinite.

4.1 Large margin nearest neighbour

Large margin paradigm, as described in [26], is based on two simple intuitions that ensure robust nearest neighbour classification: (1) each training input x_i should share the same label y_i as its k nearest neighbours; and (2) training inputs with different labels should be widely separated. These objectives are neatly balanced through two competing terms in the resulting loss function, which is given by:

$$F(\mathbf{M}) = (1 - \mu) \sum_{j \rightsquigarrow i} d^2_{\mathbf{M}}(x_i, x_j) + \mu \sum_{i,j \rightsquigarrow i} \sum_{l} (1 - y_{il})[1 + d^2_{\mathbf{M}}(x_i, x_j) - d^2_{\mathbf{M}}(x_i, x_l)]_+ \tag{2}$$

The first term in (2) pulls closer together neighbouring inputs sharing the same label, while the second term acts to alienate each input x_i from differently labelled inputs by an amount equal to one plus the distance from x_i to any of its k similarly labelled closest neighbours. The weighting parameter $\mu \in [0, 1]$ is used to balance the trade-off between these two terms. Generally, its value can be tuned via cross validation.

The optimization problem in (2) can be cast as an instance of convex programming [5]. Convex programming is a generalization of linear programming in which linear costs and constraints are replaced by convex costs and constraints. Due to recent advances in numerical optimization, convex programs can now be solved efficiently on modern computers.

4.2 Fuzzy k-nearest neighbour classifier

One of the main issues encountered in using the k-nearest neighbour classifier is that normally each of the examples in the training set is considered equally important in the assignment of class label to the input vector. This can cause problems in those places where the example sets overlap, since atypical vectors are given as much weight as those that are truly representative of the certain class. Another difficulty is that once an input vector is assigned to a class, there is no indication of the "strength" of its membership in that class. Both of these problems can be addressed by incorporating fuzzy set theory in k-nearest neighbour rule.

The basis of fuzzy k-nearest neighbour algorithm [15] is the assignment of membership as a function of the inverse of input's Mahalanobis distance from its k-nearest neighbours and their memberships in the potential classes. The labelled examples can be assigned membership in two ways. The first method, called crisp labelling, is to assign each labelled example complete membership in its known class and non-membership in all other classes. The second technique assigns memberships based on the k-nearest neighbour rule. The k (not the k of the classifier) nearest neighbours to each input x_i (say in class i) are found, and then membership in each class is assigned according the following equation:

$$u_j(x_i) = \begin{cases} 0.51 + n_j/k \cdot 0.49 & \text{if } j = i, \\ n_j/k \cdot 0.49 & \text{if } j \neq i. \end{cases} \tag{3}$$

where n_j denotes the number of found neighbours, that belong to the jth class. This technique attempts to "fuzzify" the memberships of labelled examples that are situated in the intersecting class regions in the input space, leaving the examples that are well away from this area with complete membership in the known class. As a result, an unknown example lying in this intersecting region is influenced to a lesser extent by the labelled examples that are in the "fuzzy" area of class boundaries.

5 Experimental results

For experimental purposes, a dataset containing 9 different test gestures, ranging from simple left-to-right and up-to-down gestures, to more complex gestures resembling geometrical shapes and letters, has been created. This definition of gesture dataset is to a certain degree inspired by the gesture vocabulary identified in the Nokia research study [14] as preferred by users for interaction with home appliances and is aimed at increasing the robustness of the gesture recognition system. All the defined gestures, depicted in Fig. 4, are limited to a vertical $(x - y)$ plane in front of the user as the

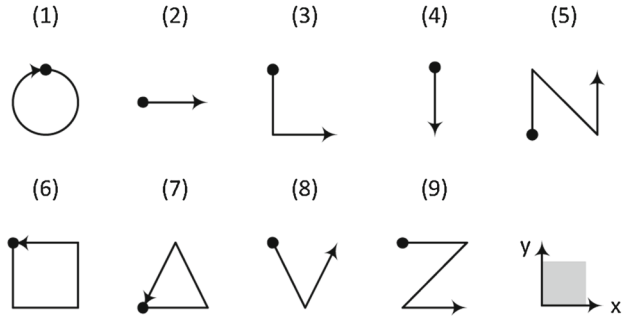

Fig. 4 The gesture dataset used in experiments

same previously mentioned Nokia study has indicated that users are more likely to use spatial 2-dimensional gestures and that utilizing all three dimensions in one gesture is fairly rare. Given the nature of the performed gestures and their stroke directions, an eight channel histograms were used in the experiments, with ith channel being denoted as:

$$ch(i) = \frac{i\pi}{4} \pm \frac{\pi}{8}, \quad i = 0, 1, \ldots, 7 \tag{4}$$

The gesture database, used in the experiments, consisted of 1890 examples, gathered from seven (2 male and 5 female) participants, aged between 24 and 58, all but one of whom were right handed. Each of them was asked to repeat each gesture for 30 times within the course of several weeks, which provided an overall total of 210 examples for all participants per gesture and 270 examples for all gestures per participant. During data collection, the participants were requested to hold the input device (smartphone) in a natural way and to try their best to avoid excessive tilting of the device as much as possible. As gesture samples acquired on the same day do not exhibit regular high variability over time, which may in turn lead to overly optimistic recognition results, participants were strictly forbidden to perform more than five repetitions of each gesture per day.

The herewith presented results were obtained by employing tenfold cross validation. For each participant, we divided the 30 examples of single gesture into 10 partitions, namely 3 examples per partition. At each turn, 7 out of 10 randomly chosen partitions were used as training (T) and the remaining 3 as testing (T') subset. This procedure was repeated ten times and then the average recognition rate was taken. Finally, the total average recognition accuracies were produced by averaging over all participants.

The performances of fuzzy classifiers on a raw dataset (without any metric learning), summed up in a Table 1, are reported in terms of the simplest crisp partition, where a sample vector is assigned to the class of maximum membership. Upon the comparison of the results of crisp classifier, fuzzy classifier with crisp initialization (1) and fuzzy classifier with nearest neighbour initialization (2), it can be seen that

Table 1 Summary of k-nearest neighbour classifiers recognition accuracies, depending on the varying training set size

	Crisp		Fuzzy-(1)		Fuzzy-(2)	
N	T	T'	T	T'	T	T'
9	0.8734	0.8379	0.9204	0.8596	0.9202	0.8558
12	0.9450	0.8818	0.9740	0.8981	0.9746	0.8946
15	0.9784	0.9128	0.9978	0.9222	0.9983	0.9200
18	0.9822	0.9215	0.9966	0.9284	0.9969	0.9274
21	0.9850	0.9339	0.9928	0.9374	0.9930	0.9366

both fuzzy classifiers produce somewhat higher recognition accuracies. But, more importantly, they provide membership assignments that give an indication of classification correctness degree. Although not shown in the tables, the results of fuzzy classifier using nearest neighbour initialization technique show that correctly classified samples were generally attributed relatively higher membership in their known class than in other classes. Furthermore, the majority of them had membership in the correct class greater than 0.7, thus providing an assurance, based on the membership assignment, that they are correctly classified. On top of that, only 4 % of misclassified samples were attributed high memberships (greater than 0.8) in the wrong class.

To assess the usability of LMNN metric learning scheme for accelerometer-based gesture recognition, its performance was compared to those of other, competing distance metric learning methods, namely the principal component analysis (PCA) [12], the linear discriminant analysis (LDA) [8] and the neighbourhood components analysis (NCA) [10]. In all the experiments reported in this paper, the classification was done by looking at $k = 3$ nearest neighbours. The initial target neighbours for LMNN classification were chosen based on Euclidean distances in input space. The LMNN training was done by utilising the Weinberger—Saul special purpose solver [25], which iteratively re-estimates the Mahalanobis distance metric as it attempts to minimize the objective function for LMNN classification. Multi-class support vector machines (SVM) [7] were also evaluated as providing a fair representation of state-of-the-art. They were trained using linear, polynomial and RBF kernels and the best kernel was chosen with cross validation.

Table 2 sums up the recognition results of the aforementioned distance metric learning algorithms, averaged across all participants, depending on the number of training samples. As can be seen, whilst all five tested approaches improve on nearest neighbour classification using a Euclidean distance metric and yield respectable recognition accuracies greater than 90 %, the LMNN classification constantly outperforms all other methods—even the benchmark multi-class SVM that do not perform very effectively with small training set sizes—and achieves the highest recognition rate of

Table 2 The performance analysis for different variants of nearest neighbour classification, using Euclidean vs. Mahalanobis distances, in view of the number of training samples

	Method					
N	Eucl.	PCA	LDA	NCA	LMNN	SVM
9	0.8596	0.9669	0.9514	0.9231	0.9904	0.9170
12	0.8981	0.9658	0.9531	0.9415	0.9943	0.9230
15	0.9222	0.9671	0.9603	0.9565	0.9958	0.9323
18	0.9284	0.9661	0.9679	0.9528	0.9968	0.9368
21	0.9374	0.9656	0.9675	0.9665	0.9989	0.9405

Note that SVM uses linear classifier and the results are presented here for comparison

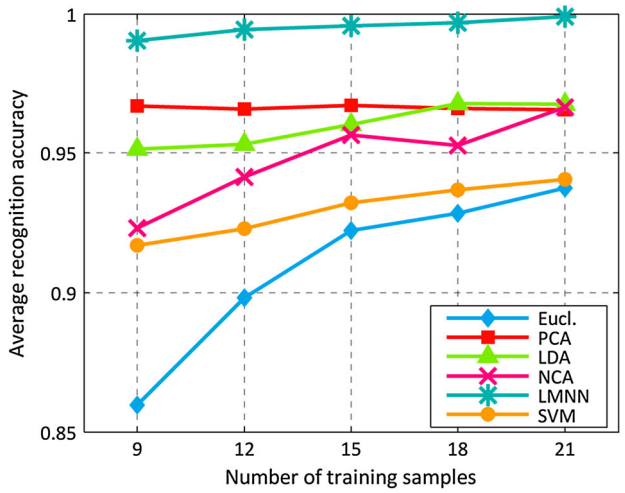

Fig. 5 Average recognition accuracies against number of training samples for a fuzzy 3-nearest neighbour classifier on the testing subset

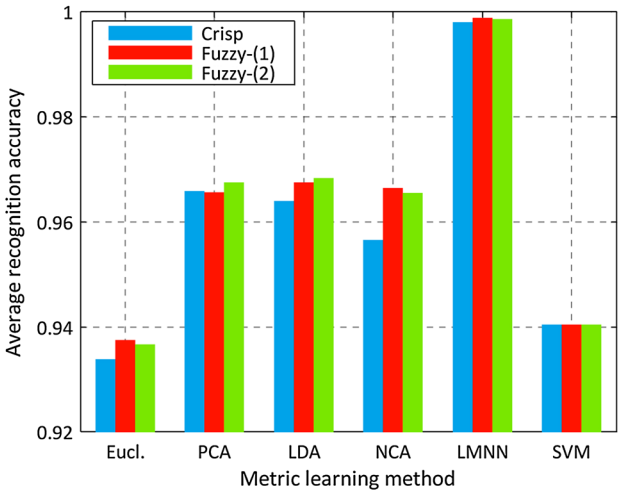

Fig. 6 Graphical representation of different nearest neighbour classifiers performances for testing subset. The reported accuracies refer to the largest training set size of 21 training samples per gesture

99.89 % with $N = 21$ training samples. A one-way analysis of variance (ANOVA) was conducted to further compare presented performance results. The analysis was significant at $p < 0.0001$ level ($F = 30.1408$). Post hoc comparison using Fisher's Least Significant Difference test indicated that the mean score for LMNN ($M = 99.89$, $V = 5.53 \times 10^{-5}$) is significantly different from scores of all other examined metric learning algorithms. On the other hand, the mean scores of remaining algorithms, except SVM, did not differ significantly amongst themselves. Figure 5 illustrates these improvements more graphically, while Fig. 6 provides further comparison of performance between two fuzzy classifiers and the baseline k-nearest neighbour algorithm. In addition, Fig. 7 shows the final, global confusion matrices for the fuzzy nearest neighbour classifier that record the percentages of times of how samples are recognized.

It is a well known fact that computational complexity is one of the main drawbacks regarding the use of convex optimization in practical applications. Accordingly, the LMNN algorithm faces the same challenges. In order to evaluate the computational efficacy of the tested methods, Table 3 provides the comparison of CPU times for a single run of different distance metric learning algorithms. These simulation results were generated on a 1.4 GHz notebook computer, that has 1 GB memory and runs a Windows XP operating system. As shown, the computational burden of LMNN is significantly larger than those of conventional nearest neighbour based metric learning methods, such as PCA, LDA, NCA and SVM. In practice, however, training (i.e. learning distance metric) is usually an off-line process and only recognition needs to be performed in real time. The recognition time of all algorithms is roughly the same and well under one-hundredth of a second, which is a standard set by other systems in literature.

The presented results of the performance analysis clearly prove that the system exhibits a very competitive performance, both computationally and in terms of recognition accuracy, by running the code on a computer. However, it is more reasonable to validate its novelty and performance by testing how it performs in real-life scenario. To test the real-world functionality of the proposed gesture recognition algorithm, the algorithm was ported to a mobile device, by implementing and executing a prototype user application on Google's Android platform. The user application was developed by making use of MIT App Inventor for Android IDE and Android SDK. An example of user application running on the smartphone is shown in Fig. 8.

To teach a system a new gesture, the user can choose the `Learning` tab from the main menu. The gesture start and finish are signalled by tapping on the designated button in the application screen. The recorded gesture trace is then processed and stored in the device's memory as training data. To recognize a gesture, the user selects the `Recognition`

(a)

	99.7	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0
	0.0	100	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	100	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	100	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	99.7	0.0	0.0	0.3	0.0
	0.0	0.0	0.0	0.0	0.0	99.7	0.0	0.0	0.3
	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0
	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.0	99.7

(b)

	87.4	0.0	0.0	0.0	0.0	1.1	11.5	0.0	0.0
	0.0	100	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	99.7	0.0	0.0	0.0	0.0	0.0	0.3
	0.0	0.0	0.0	100	0.0	0.0	0.0	0.0	0.0
	1.6	0.0	0.0	0.0	94.0	0.0	0.0	4.4	0.0
	0.0	0.0	0.0	0.0	0.0	87.7	4.6	0.0	7.7
	8.5	0.0	0.0	0.0	0.0	3.0	81.1	0.0	7.4
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0
	1.6	0.0	1.1	0.0	0.0	1.9	1.6	0.0	93.7

Fig. 7 Confusion matrices for the gesture dictionary (**a**) with and (**b**) without distance metric learning (%). *Columns* are recognized gestures and *rows* are the actual identities of input gestures

Table 3 Average CPU times (in seconds) spent on the training and recognition by different methods

		Recognition		
Method	Training	*Crisp*	*Fuzzy-(1)*	*Fuzzy-(2)*
Eucl.	0.0000	0.0009	0.0021	0.0024
PCA	0.0023	0.0013	0.0016	0.0021
LDA	0.0063	0.0017	0.0021	0.0027
NCA	3.4494	0.0011	0.0016	0.0021
LMNN	12.0994	0.0013	0.0013	0.0021
SVM	1.1654	0.0016	0.0016	0.0016

tab from the main menu. Same as before, data acquisition process is initiated by tapping on the button displayed in the application screen. When the same button is pressed again, the gesture recording is terminated and the application executes the gesture recognition algorithm. After the most probable gesture type has been identified, the confirmation message is displayed asking the user if he wishes to proceed with running the corresponding action.

The algorithm's execution time on mobile device was determined by computing the delay (time interval) that the user experiences between the time instant in which the gesture has finished and the instant in which it has been recognized. To measure this delay, the events shown in Table 4 were defined. Whenever a specific event occurred, the application invoked a system call in order to obtain system time. This experiment was repeated 20 times, whilst making sure no other applications were running on the phone simultaneously.

Fig. 8 Prototype user application **a** main menu and **b** data acquisition screenshots

Table 4 Two event types

Time	Event description
T_1	The user presses the *Stop* button
T_2	The confirmation message is displayed to the user

T_i denotes the time the i-th event is recorded

The resulting average execution time of the gesture recognition was 7.85 ms. This time is negligible in comparison to the acquisition time (9.33 s per measurement on average). Therefore, we can conclude that the computation of gesture recognition algorithm does not induce any human perceptible lag and that it can be done in real time.

6 Conclusion

Motivated by the proliferation of MEMS accelerometer sensors embedded in personal electronic devices, such as music players, smartphones and game consoles, this paper presents a novel gesture recognition system based solely on a data from a single triaxial accelerometer. The system utilizes LMNN framework for distance metric learning and a fuzzy version of k-nearest neighbour algorithm for efficient training and classification. The metric is learnt with the goal of separating the inputs from different classes by large local margins and pulling closer together k-nearest neighbours from the same class.

During recognition stage, the unknown gesture example feature vector is allocated fuzzy class memberships, based on its distance from its k closest training examples (nearest neighbours) and their memberships in the possible classes. Two methods of assigning class memberships are examined and the experimental results for "crisp" and fuzzy k-nearest neighbour rule are presented and compared. Regardless of the initialization technique being used, the fuzzy algorithm is shown to dominate its crisp counterpart not only by having a higher average classification accuracies, but also by producing membership values that provide a confidence measure to accompany the resultant classification.

The performance of the system is evaluated off-line, by comparing it to the series of other, commonly used, competing distance metric learning methods. Support vector machines were also included due to the parallels that can be drawn between them and the LMNN metric learning framework and their proven effectiveness in pattern recognition problems. Experiments, conducted on a user-defined vocabulary of 9 arbitrary gestures, have demonstrated that LMNN consistently outperforms all other distance metric learning methods that it was compared to for all training set sizes and yields near perfect results in terms of recognition accuracy.

As future work, we are planning to run a series of empirical measurements with real users using system metrics, such as time delay, battery consumption and CPU load, to further assess the proposed gesture recognition system's usability in real situations. The assessment will also include an elaborate subjective user study, which will provide more in-depth insight about the quality of user experience while running the system. With respect to the gesture recognition algorithm, a natural enhancement would be extending it to incorporate gesture spotting. In our proposed system, the starting and ending point of a gesture trace was marked by pressing the designated button on the application screen. This way of acquiring gesture traces—by assuming known starting and end points—is not realistic. A more realistic scenario would be to segment meaningful gesture traces from a stream of hand movements. The beginning and the end of a gesture can be detected by sliding a window over n consecutive accelerometer readings and checking if the variance inside the window is greater than set threshold. The window size and threshold value may be determined empirically based on the sensor noise floor and the average vibration caused when the device is held stationary.

References

1. Akl A, Feng C, Valaee S (2011) A novel accelerometer-based gesture recognition system. IEEE Trans Signal Process 59(12):6197–6205. doi:10.1109/TSP.2011.2165707
2. Balakrishnan V, Yeow PHP (2007) SMS usage satisfaction: influences of hand anthropometry and gender. Hum IT 9(2):52–75
3. Balakrishnan V, Yeow PHP (2008) A study of the effect of thumb sizes on mobile phone texting satisfaction. J Usability Stud 3(3):118–128
4. Bergamasco M, Frisoli A, Avizzano CA (2007) Exoskeletons as man-machine interface systems for teleoperation and interaction in virtual environments. Adv Telerobotics 31:61–76. doi:10.1007/978-3-540-71364-7_5
5. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge
6. Chopra S, Hadsell R, LeCun Y (2005) Learning a similarity metric dicriminatively, with application to face verification. In: Proceedings of IEEE conference on computer vision and pattern recognition—CVPR'05, pp 349–356. doi:10.1109/CVPR.2005.202
7. Crammer K, Singer Y (2001) On the algorithmic implementation of multiclass kernel-based vector machines. J Mach Learn Res 2:265–292. doi:10.1162/15324430260185628
8. Fisher RA (1936) The use of multiple measures in taxonomic problems. Ann Hum Genet 7:179–188. doi:10.1111/j.1469-1809.1936.tb02137.x
9. Frome A, Singer J, Sha F, Malik J (2007) Learning globally-consistent local distance functions for shape-based image retrival and classification. In: Proceedings of 11th IEEE international conference on comput vision—ICCV'07, pp 1–8. doi:10.1109/ICCV.2007.4408839
10. Goldberger J, Roweis S, Hinton G, Salakhutdnov R (2004) Neighbourhood components analysis. Adv Neural Inf Process Syst 17:513–520
11. Hofmann FG, Heyer P, Hommenl G (1998) Velocity profile based recognition of dynamic gestures with discrete hidden Markov models. In: Gesture and sign language in human–computer interaction. Lect Notes in Computer Science, vol 1371, pp 81–95. doi:10.1007/BFb0052991
12. Jolliffe IT (1989) Principal component analysis. Springer-Verlag, New York
13. Keir P, Payne J, Elgoyhen J, Horner M, Naef M, Anderson P (2006) Gesture recognition with non-referenced tracking. In: Proceedings of the IEEE conference on virtual reality—VR'06, pp 137. doi:10.1109/VR.2006.64
14. Kela J, Korpipää P, Mäntyjärvi J, Kallio S, Savino G, Jozzo L, Marca D (2006) Accelerometer based gesture control for a design environment. Pers Ubiquitous Comput 10(5):285–299. doi:10.1007/s00779-005-0033-8
15. Keller JM, Gray MR, Givens JA (1985) A fuzzy k-nearest neighbour algorithm. IEEE Trans Syst Man Cybern 15(4):580–585. doi:10.1109/TSMC.1985.6313426
16. Liu J, Zhong L, Wickramasuriya J, Vasudevan V (2009) μWave: accelerometer-based personalized gesture recognition and its applications. Pervasive Mob Comput 5(6):657–675. doi:10.1016/j.pmcj.2009.07.007

17. Mäntylä VM (2001) Discrete hidden Markov models with application to isolated user-dependent hand gesture recognition. VTT Publications 449
18. Marasović T, Papić V (2012) Accelerometer-based gesture recognition system using distance metric learning for nearest neighbour classification. In: Proceedings of 2012 IEEE international workshop on machine learning for signal processing—MLSP'12, pp 1–6. doi:10.1109/MLSP.2012.6349717
19. Marasović T, Papić V (2013) A novel feature descriptor for gesture classification using smartphone accelerometers. In: Proceedings of 18th IEEE symposium on computers and communications, pp 951–955. doi:10.1109/ISCC.2013.6755072
20. Mitra S, Archary T (2007) Gesture recognition: a survey. IEEE Trans Syst Man Cybern Part C (Appl Rev) 37(3):311–324. doi:10.1109/TSMCC.2007.893280
21. Nielsen J (1994) Usability engineering. Morgan Kaufmann, San Francisco, CA
22. Pylvänäinen T (2005) Accelerometer based gesture recognition using continuous hidden Markov models. In: Pattern recognition and image analysis. Lect Notes in Computer Science, vol 3522, pp 639–646. doi:10.1007/11492429_77
23. Schlömer T, Poppinga B, Henze N, Boll S (2008) Gesture recognition with a Wii controller. In: Proceedings of the 2nd international conference on tangible and embedded interaction—TEI'08, pp 11–14. doi:10.1145/1347390.1347395
24. Wang X, Tarrío P, Metola E, Bernardos AM, Casar JR (2012) Gesture recognition using mobile phone's inertial sensors. Distrib Comput Artif Intell 151:173–184
25. Weinberger KQ, Saul LK (2008) Fast solvers and efficient implementations for distance metric learning. In: Proceedings of 25th international conference on machine learning—ICML'08, pp 1160–1167. doi:10.1145/1390156.1390302
26. Weinberger KQ, Blitzer J, Saul LK (2009) Distance metric learning for large margin nearest neighbour classification. J Mach Learn Res 10:207–244
27. Xing EP, Ng AY, Jordan MI, Russell S (2003) Distance metric learning, with application to clustering with side-information. Adv Neural Inf Process Syst 15:505–512
28. Xu R, Zhou S, Li WJ (2012) MEMS accelerometer based nonspecific-user hand gesture recognition. IEEE Sens J 12(5):1166–1173