Computers in Biology and Medicine 95 (2018) 248–260

Contents lists available at ScienceDirect

Computers in Biology and Medicine

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

A general framework for sensor-based human activity recognition

Lukas Köping [a,*], Kimiaki Shirahama [a], Marcin Grzegorzek [a,b]

[a] Pattern Recognition Group, University of Siegen, Siegen, Germany
[b] University of Economics in Katowice, Faculty of Informatics and Communication, Katowice, Poland

ARTICLE INFO

Keywords:
Sensor-based activity recognition
Feature learning
Sensor data collection

ABSTRACT

Today's wearable devices like smartphones, smartwatches and intelligent glasses collect a large amount of data from their built-in sensors like accelerometers and gyroscopes. These data can be used to identify a person's current activity and in turn can be utilised for applications in the field of personal fitness assistants or elderly care. However, developing such systems is subject to certain restrictions: (i) since more and more new sensors will be available in the future, activity recognition systems should be able to integrate these new sensors with a small amount of manual effort and (ii) such systems should avoid high acquisition costs for computational power.

We propose a general framework that achieves an effective data integration based on the following two characteristics: Firstly, a smartphone is used to gather and temporally store data from different sensors and transfer these data to a central server. Thus, various sensors can be integrated into the system as long as they have programming interfaces to communicate with the smartphone. The second characteristic is a codebook-based feature learning approach that can encode data from each sensor into an effective feature vector only by tuning a few intuitive parameters. In the experiments, the framework is realised as a real-time activity recognition system that integrates eight sensors from a smartphone, smartwatch and smartglasses, and its effectiveness is validated from different perspectives such as accuracies, sensor combinations and sampling rates.

1. Introduction

Activity recognition has been identified as a key component to build personal fitness advisors or assist elderly people in living in their own homes for longer [1]. Hence, the proliferation of wearable devices like smartphones, smartwatches and smartglasses is paving the way to performing activity recognition with the help of technology that people are already familiar with (see Fig. 1). This is in stark contrast to systems that are monitoring people with the help of cameras. Not only are people concerned about their privacy when cameras are in use, but the installation thereof also comes with non negligible costs that are necessary to operate the system.

With the fast development of new wearable devices, we can also observe the increase in the number of built-in sensors. While several years ago wearable devices were mostly equipped with an accelerometer and a gyroscope, nowadays additional sensors like heart-rate, barometer, magnetometer, galvanic skin response or electrooculography are introduced. This opens new possibilities to track more details of a person's daily routine, but it also introduces the necessity to research each new sensor modality. The solution for that is to use data of a sensor in a machine learning set-up, where sensor data must firstly be represented by meaningful features. However, defining which parts of the data are in fact meaningful or which representation might be appropriate for the activity recognition classifier, is often a difficult task that requires specific domain knowledge or a lot of experimentation.

For this reason we focus on the evaluation of a feature learning approach [4] that extracts effective features from the underlying sensor data. We utilise the codebook approach to obtain a feature vector summarising local characteristics of the data only by tuning a few intuitive parameters [5]. Due to this strategy, we can easily integrate new sensors into our overall system.

The proposed system architecture can be summarised as follows: The framework uses the sensors of wearable devices like smartphones, smartwatches and smartglasses to determine the current activity of a person.

First of all, a sensor needs to send its data to a central server on which the activity recognition is performed. To make this data transfer flexible, we use a smartphone as temporal storage device. The data from the different sensors are firstly gathered on a smartphone and then sent to the central server via the AMQP protocol. This enables us to easily integrate various sensors into our system, since many sensors equipped in today's wearable devices have a programming interfaces to communicate with a

* Corresponding author.
E-mail addresses: lukas.koeping@uni-siegen.de (L. Köping), kimiaki.shirahama@uni-siegen.de (K. Shirahama), marcin.grzegorzek@uni-siegen.de (M. Grzegorzek).

https://doi.org/10.1016/j.compbiomed.2017.12.025
Received 12 April 2017; Received in revised form 29 December 2017; Accepted 29 December 2017

Fig. 1. Wearable devices are becoming more important for activity recognition ever since new sensor modalities are being integrated into them. A smartphone typically is equipped with an accelerometer, gyroscope and magnetometer, smartwatches have pulse-sensors and sensors for electrodermal activity [2] and some smartglasses offer electrooculography [3].

smartphone.

On the central server, the data from each sensor are encoded into the above-mentioned feature vector. Especially, as we will show in the experiments, the codebook approach analyses large amounts of data to capture statistically distinctive local characteristics without any preprocessing. In addition, it is applicable to sensor data with different sampling rates. Finally, the obtained feature vectors for each individual sensor are simply concatenated into a single high-dimensional vector, on which a Support Vector Machine (SVM) is constructed to perform activity recognition.

This paper contributes in the following ways to the field of activity recognition. (i) We present a general framework for sensor-based human activity recognition using a feature learning technique. This eliminates the necessity of assumptions about the specific sensors that are used. The various algorithmic parts of our framework are highly customisable so that it can be easily extended. (ii) Our system manages the data collection, transfer of the data from the devices to a small and cheap server, and the final recognition on it. In addition, the system offers high flexibility since many of the implemented internal algorithms and approaches can be interchanged with appropriate alternatives.

The paper is structured as follows. In Section 2 we present important works in the field of smart home architectures and activity recognition. Section 3 gives an overview of our system architecture, showing in detail how we collect, transfer and analyse data. In Section 4 we show results of various experiments regarding the detection and classification of activities. Finally, Section 5 gives a conclusion and shows path for future investigations.

2. Related work

Traditionally, the fields of smart home technology and activity recognition are highly overlapping. Smart homes provide the hardware to collect and process relevant data, while activity recognition approaches extract semantic meanings from the collected data by adopting machine learning methods. Our system is not an exception to this overlap. For this reason, we want to point out highly relevant work from the smart home area and research approaches from the field of activity recognition.

2.1. Smart home technology

For more than two decades the research community has developed different approaches that focus on the recognition of activities for in-home settings. The need for in-home experiments is motivated by the fact that people behave in a different manner when they feel observed and/or are not used to the experimental environment. This makes it hard to transfer results obtained in laboratory settings to more realistic scenarios or even commercial products. Our work intersects with the research of smart homes as well as with activity recognition based on sensor data of wearable devices. However, due to the intensive work in both areas we can only focus on the most important and relevant papers. For more detailed information we refer the interested reader to the various surveys that were published in the last years [6–9].

One common theme of many studies is to utilise motion sensors that are fixed at some location in the house/apartment or at specific objects like microwaves or toilets. The biggest advantage of motion sensors is their cheap price and the simplicity of the data. The output is often a binary value that, e.g., represents the presence of a resident at a specific location [10] or indicates if a certain object was used [11–16]. Participants of these studies also report that, even if motion sensors are placed visibly, they forget or ignore their existence over time [10]. The PlaceLab living lab [17] is a 1000 sq. ft. apartment that is fully equipped with various sensors and computers. PlaceLab also offers the possibility to use cameras and microphones to annotate activities. Furthermore, additional sensors like accelerometers and gyroscopes can be attached to participants. The output data are transferred over a wireless network and synchronised with all other sensors. This laboratory was used in Ref. [12], in which they monitor a married couple over a timespan of ten weeks. They compare the activity recognition accuracy for different sensor modalities and conclude that many activities can be detected only by knowing the location of the user, which is a strong argument for motion sensors.

In addition to the pure hardware set-up, researchers have also realised that an expandable architecture is the basis to build long-term usable smart home systems. An example is the Gator Tech Smart House [18]. While equipped with an impressive amount of smart devices, it also offers various services to the developer of such systems. Raw sensor data are abstracted into a physical layer so that developers can thus define services without having to understand the physical world" [18]. The CARDEA-MuSA [19,20] is an example for another framework with the ability to integrate and connect various sensors and use these for an advanced health-monitoring.

An approach that targets especially user-friendliness is described by the CASAS system [21]. The main components of the system are various motion sensors that the user can install on his own. Due to its architecture, the system can be extended by additional sensors and only a small, low-cost server is needed to perform activity recognition. At the time of publication the estimated costs for the whole system amount to $2765.

One common disadvantage among the different approaches is that additional sensors must be equipped within the house, even if the amount of work-load differs among the presented works. This leads to an obstacle that many users are not willing to overcome.

2.2. Activity recognition from continuous data

Architectures for smart homes mostly rely on binary data obtained from motion sensors or reed switches. However, the constantly increasing number of wearable devices has also brought a great deal of effort in working with real-valued sensor output, which we refer to as *continuous data* [22–26]. This is especially true since the variety of sensor modalities increases with every new wearable device generation. Sensors like accelerometers, gyroscopes or magnetometers are common for

almost all devices. However, new devices also offer sensors like barometer, heart rate sensor, electrooculography (EOG), electromyography (EMG) or sensors for electrodermal activity (EDA). The main advancement in the last years is that new sensors have a strong focus on the tracking of biometrical health data [27–29].

Since our framework relies only on a few assumptions about the underlying sensor data, we can easily deal with new sensor modalities, which is an important point for future research. This is often in contrast to related work that has to know many details about the given sensor data [23]. For example, template matching methods measure the similarity of time sequences on the basis of Euclidean distance or Dynamic Time Warping [30–32]. If two time sequences are alike, they are considered as the same activity. However, it was found that these methods are easily corrupted by noise in the signal [33]. Generative activity recognition methods estimate the probability that a given time sequence is generated by assuming that the user performed a specific activity. Hidden Markov Models and its extensions [34–36] belong to the predominantly used techniques of this group. The advantage of these methods is that they take temporal dependencies among data points into account. However, a substantial amount of training data is essential to capture all the different variations of an activity [22]. The overwhelming amount of research focuses on feature-based approaches. These "features" should represent characteristics of the underlying time sequence and be representative for each activity. For example, statistical features summarise time sequences in terms of their mean value or variance, while parameters from the frequency domain can represent repetitive patterns of the signal [37–44]. More sophisticated features describe characteristic events of an activity like the occurrence of specific eye-movements [45] or biomedical events, e.g., duration of inhalation/exhalation and heart beat intervals [46]. While such features might significantly improve the classification result, it often requires domain knowledge that only experts can provide.

Avoiding the creation of the above-mentioned hand-crafted features has become one of the main research issues in machine learning. "Feature learning", the task of automatically extracting meaningful features from data, is brought to the fore. A simple but effective version of feature learning is to count the occurrences of relevant pieces in the time signal. This so-called codebook approach has its origin in image classification [47,48], where it analyses a large number of patches (small regions) in images and constructs a codebook consisting of characteristic patches called *codewords*. An image is then represented by a feature signifying the distribution of codewords.

Codebook-based methods have also been shown to be successful candidates for various areas of activity and emotion recognition [5, 49–51]. More advanced feature-learning methods stem from the field of deep learning and have also already been applied to human activity recognition [52–56]. However, one big disadvantage of deep learning architectures is a large number of hyper-parameters that require an extensive optimization phase.

For this reason, we use the codebook approach in our general framework. It is applicable to any type of sensor data and requires only a few intuitively tunable parameters, namely the window size, sliding size, and number of codewords. With this, we can integrate any type of new streaming sensor into our framework without any specific adaptation.

3. System architecture

This section provides a general description of our activity recognition framework. One specific implementation of it will be presented in Section 4.1. Our framework consists of the following three main components: The first one includes all data sources, namely the sensors of all employed wearable devices. This encompasses, but is not restricted to, sensors like accelerometers, gyroscopes and magnetometers. These three sensors in particular are often included in all wearable devices like smartphones, smartwatches and smartglasses. The second component is responsible for transferring the collected data from the smartphone to the central server. In our version of the activity recognition framework we suggest using an Advanced Message Queuing Protocol (AMQP) [57] based transfer service. Using AMQP it is possible to send data from the sensors to another component in form of messages. Above all the protocol ensures that data is not lost or queued if the receiver is not available. The last component, the central server, receives the sensor data and runs the activity recognition algorithm. Since the codebook approach is a lightweight algorithm it is easily possible to achieve real-time performance, even if the server has only a limited amount of computational resources. The results of the activity recognition can be stored on the server for later use or send back directly to the user's phone again using the AMQP protocol.

3.1. Data acquisition

We expect that multiple devices are connected to the system and every device contains one or more different type of sensors. In addition, our proposed system can only process data under the assumption that the sensors' output are time series which, however, fits to almost all sensors that can be found in wearable devices. Especially three specific sensors, namely accelerometer, gyroscopes and magnetometer, can be found in almost all wearable devices. Accelerometers measure the change of velocity $a = \frac{\Delta v}{\Delta t}$ over time in a three-dimensional space [58]. If the sensor is carried in a person's pants pocket, it helps to identify basic activities like walking, running or standing. More definite activities can be discovered whenever an accelerometer is attached to a specific limb like the arm. In this case, detailed activities like "reaching out for something" or "making a phone call" might be identified. Gyroscopes supplement accelerometers by measuring the three possible rotations around the x-, y- and z-axis of the device. Magnetometers are similar to electric compasses and are utilised to infer the orientation of a device. Since this is done by taking measurements of the magnetic field around the device, magnetometers are very sensitive to other surrounding electrical devices that might influence the signal. Besides these three, various new sensors are now also built into commodity devices. The gravity sensor that is provided by the Android OS measures the earth's gravity force on the device and can be used to gain information about the phone's orientation, while linear acceleration measures acceleration of the device subtracted by the gravity force. The last two sensors, gravity and linear accelerometer, are not hardware sensors but are usually the result of the fusion of various other sensors and are sometimes called "virtual sensors" [59]. In our specific case we rely on the abilities of the Android SDK that is providing an implementation for all of the aforementioned sensors. For details of this implementation the reader is referred to the openly available source code and documentation [60].

Finally, there are two more points that can have an impact on the activity recognition results during the data acquisition. One is the sampling frequency of the sensors. Sensors with high sampling frequencies produce more data per second/minute, which might be necessary to detect short-term events. Typical ranges lie in the area of 20 Hz–100 Hz. Secondly, activities can only be detected if a sensor is attached at the part of the body where the movement is taking place. For example, an activity that involves the arm can only be measured if a sensor is also attached to the arm. The influence of different sensor positions has been explored in Refs. [61–65]. As we will show in our experiment, our system is not an exception to this limitation, as the best results are achieved once all available devices are used.

3.2. Data transfer

A device has to send its data to a central server where the activity recognition takes place. While today's computational resources of most mobile devices would be sufficient to process the data on the device itself, this would also result in a much higher battery consumption. However, since the data transfer only plays a minor role in our system we restrict ourselves to a short description of an exemplary implementation

using the AMQP protocol.

In the concrete implementation of our proposed framework three situations arise where data must be transferred. Fig. 2 illustrates an overview of the implemented system. Firstly, data from the smartwatch and smartglasses are sent to the smartphone. This is because the devices that we are currently using are not able to directly send their data to the central server. Instead, they are offering programming interfaces to transfer the data with a Low Energy Bluetooth 4.0 protocol to the phone. However, while the current state requires this additional step, we can already anticipate more powerful devices on the market that are using sophisticated operating systems, and therefore will also offer additional programmable ways to directly send their data to any other component. Thus, the phone is treated as a kind of temporal data storage. Secondly, our framework sends sensor data to the central server using an AMQP based middleware. We use RabbitMQ [66] as one popular implementation of AMQP since it offers interfaces to several programming languages like Python, Java or C++. AMQP enables the possibility to implement a middleware that receives and routes messages. Each sensor acts as a single producer, while components that are analysing the data are treated as receivers. In this way it is possible for each component to access its necessary data. For example, activity recognition needs data from the accelerometer and gyroscope, but probably not from the heart-beat sensor. On the other hand, recognition of emotional stress requires EDA data but not that of magnetometers. The only necessary thing is to configure which data the new component should receive. Also, if a new sensor is integrated into the system that might be useful for a specific component, e.g., a sensor to measure a person's respiration rate to detect emotional stress, the component just has to register for the new data source. After the data have been analysed, if necessary, the transfer of analysis results from the central server to the smartphone again uses the AMQP framework.

The last point concerns the transmission of the data. We have experienced computational performance issues on the smartphone when trying to send data one sample at a time. A significant improvement is made by packing data into chunks of, for example, 500 samples and sending this package to the central server. One major disadvantage, however, is the time delay. If we sample a sensor with 100 Hz and build chunks of 500 samples, this means that the recognition component incurs a delay of 5 s. It depends on the application whether this time delay is acceptable or not.

3.3. Data processing

First of all, we clarify some important terms. When we refer to time series data, we mean the whole dataset which contains all the data that are available for training/testing a classifier. For example, this could be one or more persons' data recorded over a time span of 24 h. A sequence, however, is only one part of the dataset, for example a window of 10 s. Our goal is to predict the label (i.e., underlying activity) of this sequence based on a machine learning approach. For this, we need a good representation by means of a feature vector for every possible sequence within the dataset. Finally, a subsequence is again one window within the sequence (see Fig. 4). The codebook approach constructs the feature of the sequence by considering the distributions of such subsequences.

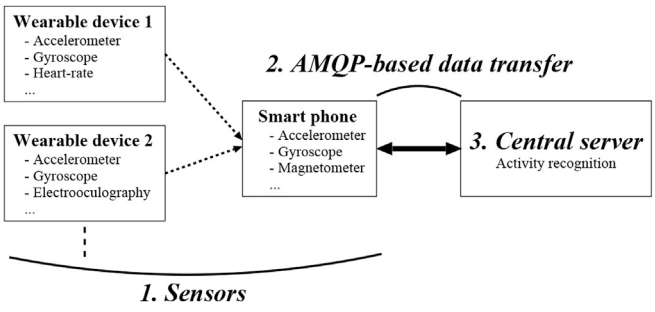

Fig. 2. An overview of our implementation of the activity recognition framework.

Fig. 3 shows an overview of our method consisting of the following three steps: Fig. 3 (a) illustrates the codebook construction step that groups subsequences collected from the dataset into clusters of similar ones. A codebook is constructed as a set of codewords, where each codeword is the centre of a cluster. Fig. 3 (b) outlines the second step, the codeword assignment, where a feature of a sequence is extracted by assigning each subsequence to the most similar codeword and counting the number of occurrences for each codeword. In other words, this feature is a histogram representing the frequency of each codeword within one sequence. The last step is classifier training/testing based on the fact that sequences represented by such features can be considered as points in the multi-dimensional space, as shown in Fig. 3 (c). As depicted by the dashed line in Fig. 3 (c), a classifier draws a classification boundary to discriminate between training sequences annotated with a certain activity class and those annotated with the others. Based on this boundary, the classifier determines the activity class of a test sequence.

Codebook construction: Since it is not possible to know beforehand which codewords are characteristic for a specific activity, it is necessary to learn them initially from the dataset and build a codebook. Constructing the codebook is a part of the training phase and is performed on unlabelled data. This is especially important since obtaining labelled data in activity recognition is still a task that requires a lot of effort.

The first step in the codebook construction is to find suitable codewords. For this, subsequences of equal length are extracted from the training dataset. A subsequence is a sliding window of size w that is shifted from the beginning until the end of the dataset. In more detail, the window is placed at every l-th time point, whereby a small l causes a large overlap between neighbouring subsequences. While this "dense sampling" of subsequences seems redundant, it has been shown to lead to better results than "sparse sampling" with a small overlap [67]. Note that the window size w and the sliding size l are hyper-parameters and must be chosen appropriately. However, w can be chosen independently from the expected duration of the activities since the final classification is based on the histogram of codewords within a given time interval but not on the codewords itself.

The second step groups all extracted subsequences into N different clusters. In our implementation k-means clustering [68] is performed to find N clusters consisting of similar subsequence. The distance among two subsequences is measured as the Euclidean distance. Since k-means clustering depends on initial cluster centres that are randomly determined, it is conducted 10 times to select the best result that yields the minimum sum of Euclidean distances between subsequences and their assigned cluster centres. Based on the best clustering result, we obtain a codebook consisting of N codewords whereby each cluster centre equals one codeword. The optimal number of codewords N must be determined experimentally.

We again want to point out the generality of our framework by mentioning that also different clustering techniques can be applied. In addition, although our results show that Euclidean distance performs well under our testing conditions, more time series specific distance measurements like Dynamic Time Warping (DTW) can be used. However, DTW requires significantly more computational resources.

Codeword assignment: To convert a sequence into a feature representation, we extract a histogram that represents the distribution of the N codewords within a sequence. Subsequences, which are extracted in the same way to the codebook construction step, are assigned to the most similar codeword and its frequency is incremented. Finally, this histogram of codewords is normalised so that the sum of frequencies of all the N codewords is one. Note that while this "hard assignment" approach assigns a subsequence to a single codeword, researchers have developed a "soft assignment" approach that implements its flexible assignment to multiple codewords [69]. However, our preliminary tests did not show a significant performance improvement of the soft assignment approach,

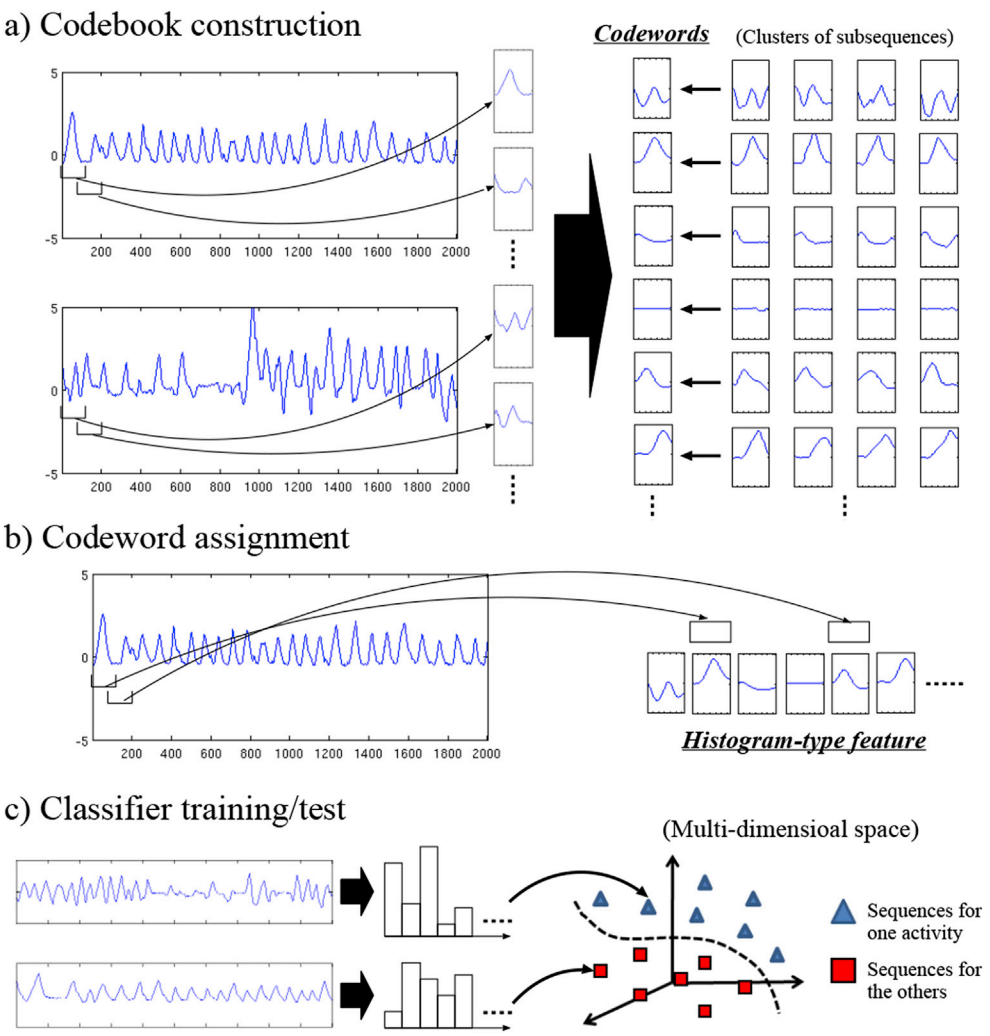

Fig. 3. An overview of our codebook-based human activity recognition method.

despite its expensive computational cost. Thus, the hard assignment approach is used throughout this paper.

Classifier training/test: A binary classifier is trained that distinguishes training sequences labelled with a target activity from other training sequences. The former and latter are called 'positive sequences' and 'negative sequences', respectively. To gain high discrimination power, a variety of characteristic subsequences need to be considered using hundreds of codewords (i.e., N is large). Because each sequence is represented with a high-dimensional feature, a Support Vector Machine (SVM) is used due to its effectiveness for high-dimensional data [70]. The SVM draws a classification boundary based on the 'margin maximization' principle so that the boundary is placed in the middle between positive and negative sequences, which makes the generalisation error independent of the number of dimensions [70]. Actually, the combination of the codebook approach and an SVM has been justified in many classification tasks of images/videos, which are represented by features with thousands of dimensions [48]. For a test sequence, the trained SVM outputs a score between 0 and 1 based on its distance to the classification boundary [71]. Finally, the recognition of C activities is conducted using C SVMs, each of which is built as a binary classifier for one activity. Then, the activity of a test sequence is determined as the one characterised by the highest SVM score.

Considering the generality of our method, we use a Radial Basis Function (RBF) kernel that has only a single parameter γ that controls the complexity of the classification boundary. It is known from the past [72] that setting γ as the mean of squared Euclidean distances among all training sequences offers a stable and reasonable performance without conducting computationally expensive cross validation. In addition, the SVM parameter C, which controls the penalty for misclassification, is empirically set to $C = 2$. This SVM parameter setting has been proven to be generally applicable to different activity recognition tasks [5], and is used throughout all the experiments.

Fusion of multiple features: Since different sensors capture different characteristics of an activity, the recognition performance can be improved by fusing features extracted from those sensor data. We especially consider the situation where a user may want to dynamically change sensors used for activity recognition. To realise this dynamic sensor selection, we explore three fusion approaches illustrated in Fig. 5: Fig. 5 (a) shows the early fusion approach [73] where codebook-based features extracted from different types of sensor data are combined into a single high-dimensional feature, based on which an SVM is constructed. Hence, a codebook is constructed for each single sensor and consequently, each sensor provides its own histogram-type feature.

Fig. 5 (b) depicts the late fusion approach [73] where each type of feature is used to construct a separate SVM. For a test example x, the final recognition score $f(x)$ is obtained by linearly combining the score $f_i(x)$ produced by the SVM on the ith sensor data, that is, $f(x) = \sum_i w_i f_i(x)$ ($\sum_i w_i = 1,\ w_i \geq 0$). Here, the weight w_i for the ith sensor data is computed in the following way: We firstly divide the training dataset into two subsets with the same size and use one of them for training SVMs on different types of sensor data, and the other one for testing these SVMs

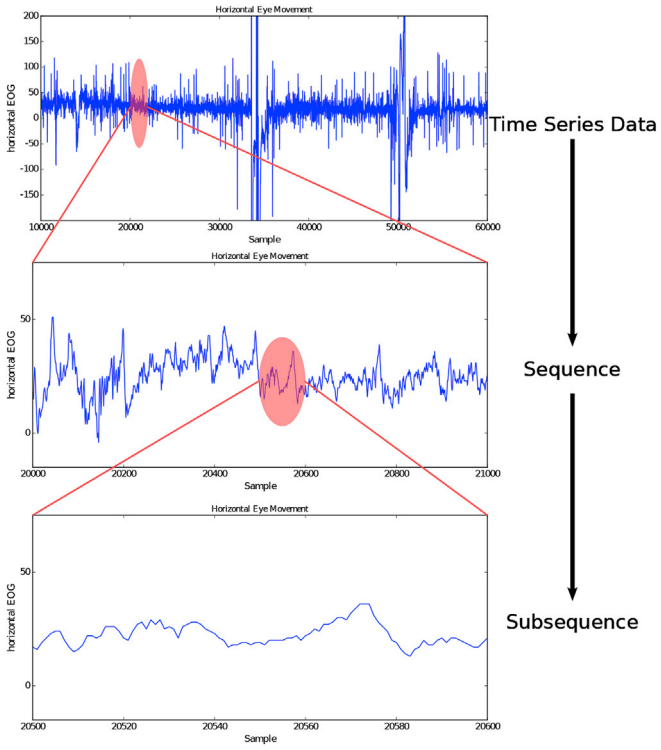

Fig. 4. An illustration of the decomposition of a dataset into subsequences.

and computing their optimal weighted combination. In particular, a gradient-ascend approach is used to obtain the weights that maximise the recognition performance (precisely, average precision explained in Section 4.1) on the second subset. Afterwards, these weights are used to fuse SVMs built using the whole training dataset. In other words, we assume that the weights obtained for SVMs trained on a half of the training dataset are applicable to more accurate SVMs trained on the whole of the dataset.

However, the early fusion approach needs to build an SVM for each of all possible sensor combinations. And every time the user activates or deactivates sensors, a different SVM has to be loaded. In addition, the late fusion approach requires an expensive computational cost, because weights have to be computed for all possible sensor combinations. To overcome this, we develop a 'dynamic late fusion' approach shown in Fig. 5 (c). Here, weights are computed only for the combination of all sensors, and adaptively re-calculated depending on sensor selection. Specifically, weights for not-selected sensors are set to zero, and weights for the selected sensors are normalised so that their summation is one.

4. Experiments

In this section, we firstly present implementation details of our activity recognition system, and datasets that are used to train and test recognition models. Then, the overall performance of our system is evaluated in terms of accuracies and computational times. Afterwards, we further elaborate on our system by analysing the effectiveness of individual sensors and devices and the capabilities of different fusion approaches.

4.1. Implementation details of our activity recognition system

Fig. 6 shows the hardware configuration of our activity recognition system. First, as depicted in Fig. 6 (a), a user takes three mobile devices, Google NEXUS 5X (smartphone), Microsoft Band 2 [74] (smartwatch) and JINS MEME [3] (smartglasses), which are used to capture the body, hand and head movements, respectively. From these devices, we obtain the following eight types of sensor data for which their abbreviations are shown in brackets:

1. Smart-phone's accelerometer (*sp-acc*): This sensor produces a three-dimensional sequence that indicates acceleration forces (including gravity forces) acting on the smartphone's x, y and z axes.
2. Smart-phone's gyroscope (*sp-gyro*): This sensor produces a three-dimensional sequence that presents angular velocities on the x, y and z axes.
3. Smart-phone's gravity (*sp-grav*): This sensor produces a three-dimensional sequence that represents gravity forces on the x, y and z axes. This is useful for capturing transitions of the smartphone's orientations over time.
4. Smart-phone's linear accelerometer (*sp-linacc*): This sensor produces a three-dimensional sequence that indicates acceleration forces on the x, y and z axes, where gravity forces are excluded.
5. Smart-phone's magnetometer (*sp-mag*): This sensor produces a three-dimensional sequence that describes intensities of the earth's magnetic field along the x, y and z axes. Such intensities are useful for determining the smartphone's orientation.
6. Smart-watch's accelerometer (*sw-acc*): Similar to *sp-acc*, this sensor produces a three-dimensional sequence of acceleration forces applied to the smartwatch's x, y and z axes.

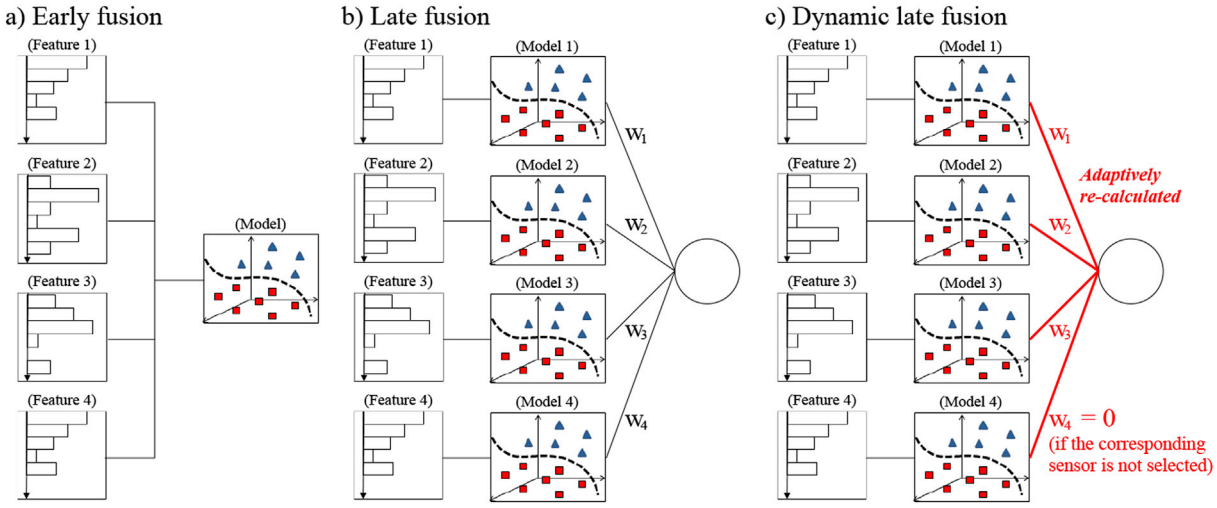

Fig. 5. An illustration of the early, late and dynamic late fusion approaches.

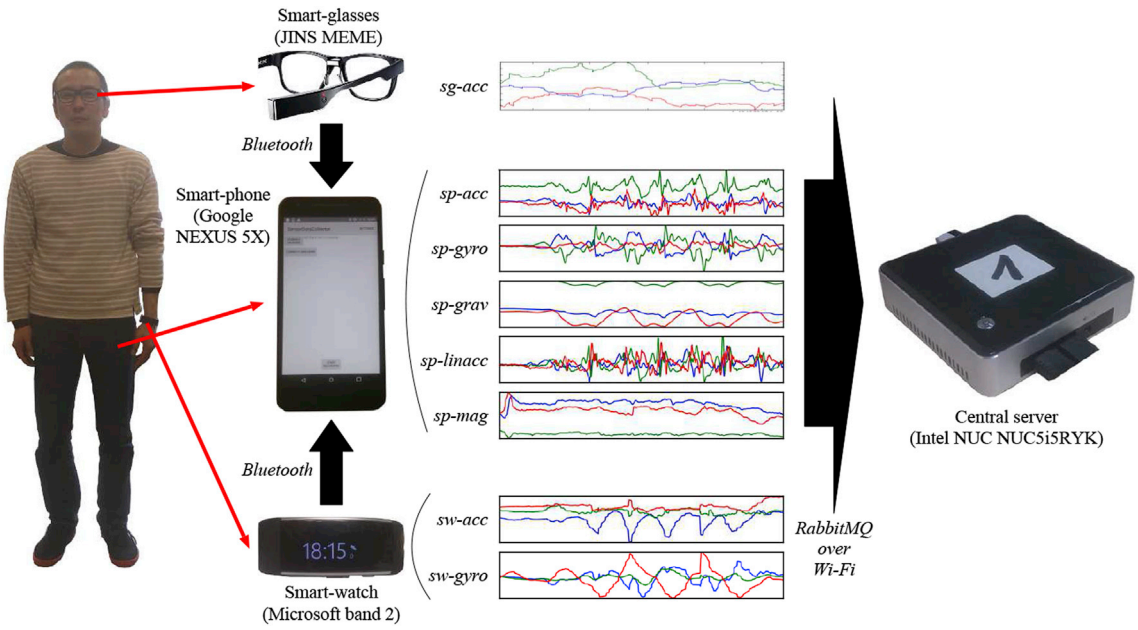

Fig. 6. An overview of our developed activity recognition system.

7. Smart-watch's gyroscope (*sw-gyro*): Similar to *sp-gyro*, this sensor produces a three-dimensional sequence of angular velocities on the x, y and z axes.
8. Smart-glasses' accelerometer (*sg-acc*): This sensor produces a three-dimensional sequence of acceleration forces on the smartglasses' x, y and z axes.

Sampling rates of *sp-acc*, *sp-gyro*, *sp-grav* and *sp-linacc* are 200 Hz, the sampling rate of *sp-mag* is 50 Hz, those of *sw-acc* and *sw-gyro* are 67 Hz, and that of *sg-acc* is 20 Hz. By following the general framework shown in Fig. 2, *sw-acc*, *sw-gyro* and *sg-acc* are firstly sent to the smartphone via Bluetooth connection, and then all the sensor data are transferred to the central server through RabbitMQ [66] over a Wi-Fi connection. Here, the server is established on Intel NUC NUC5i5RYK (CPU: Core i5-5250U 1.6 GHz, RAM: 16 GB, HDD: 450 GB, OS: Debian 4.8.4–1) where our activity recognition method is executed.

One codebook is constructed from each type of sensor data, that is, a set of three-dimensional sequences. In order to capture correlations among three dimensions, codewords are extracted by performing clustering on "three-dimensional subsequences". Specifically, a subsequence collected by a window of size w is represented as a $3w$-dimensional vector, where the first w, the subsequent w and the last w dimensions represent values on the x, y and z axes, respectively. It should be noted that this subsequence representation can be used in the subsequent codebook construction, codeword assignment and classifier training/test processes with no modification. Regarding the hyper-parameters, codebooks for *sp-acc*, *sp-gyro*, *sp-grav* and *sp-linacc* are constructed using the window size $w = 128$ and the sliding size $l = 8$, codebooks for *sp-mag*, *sw-acc* and *sw-gyro* are constructed with $w = 64$ and $l = 4$, and the codebook for *sg-acc* is built with $w = 32$ and $l = 1$. Although these values are chosen based on preliminary experiments, one criteria is that a sufficient number of subsequences can be collected from a sequence of 5 s, which is the unit to build recognition models (see below). More concretely, more than 100 subsequences are collected from *sp-acc*, *sp-gyro*, *sp-grav* and *sp-linacc* sequences, more than 45 subsequences are collected from a *sp-mag* sequences, and more than 60 subsequences are located in *sw-acc*, *sw-gyro* and *sg-acc* sequences. With this, the resulting feature of a sequence appropriately represents the distribution of subsequences. The number of codewords N is experimentally set to 1024 for all types of sensor data. Except the above-mentioned tuning of hyper-parameters w, l and N, no extra tuning or pre-/post-processing has been done. An overview over all sensor parameters can be found in Table 1.

Our current system targets the following 11 activities: 1. lying, 2. sitting, 3. standing, 4. walking, 5. bending, 6. getting up, 7. lying down, 8. putting a hand back, 9. sitting down, 10. standing up, and 11. stretching a hand. We prepare a training dataset which contains the eight types of sensor data for 145 activity executions of one user. For each of the 11 activities, the dataset includes more than 10 executions each of which lasts 5 s and signifies a different style (e.g., the user lies on his stomach, on his back, on his side, changes lying styles, etc.). This enables us to cover a variation of the same activity and carry out semantically meaningful recognition. In addition, our codebook-based recognition method does not require the exact boundaries of an activity. Instead, our method only requires the activity to be "included" in 5 s, so that the resulting feature encompasses subsequences corresponding to the moment of this activity.

For each of the 11 activities, an SVM is constructed using the training dataset. We regard the four activities (lying, sitting, standing and walking) and the remaining seven activities as *static* and *dynamic*, respectively. This is based on the fact that static activities indicate states of the user while dynamic activities represent short-time movements occurring in different states. In other words, dynamic activities occur while doing different static activities. For instance, the user can stretch his hand while standing or sitting. Thus, it is not reasonable that an SVM for a static activity is constructed by regarding executions for all the other activities as negative examples, because these executions may include moments of the static activity. Hence, negative examples for the static activity are collected as executions for the other 'static' activities, since each of static activity execution only includes one state of the user. For a dynamic activity, negative examples are collected as executions for all the other static and dynamic activities, as they do not contain any moment of the dynamic activity. Positive examples are obtained as

Table 1
Summary of all sensor parameters.

	sp-acc, sp-gyro sp-grav, sp-linacc	sp-mag	sw-acc, sw-gyro	sg-acc
Sampling rate (Hz)	200	50	67	20
Window size w	128	64	64	32
Sliding size l	8	4	4	1

a) Static activities

(Ground truth activities)	Lying	Sitting	Standing	Walking
Lying	4	3	2	2
Sitting	1	11	0	0
Standing	1	1	10	1
Walking	0	0	0	11

(Predicted activities)

b) Dynamic activities

Bending	10	0	0	0	0	0	0
Getting up	0	10	0	0	0	0	0
Lying down	0	0	9	0	2	0	0
Putting a hand back	1	0	0	9	0	0	2
Sitting down	0	0	0	0	11	0	0
Standing up	0	0	0	0	0	12	0
Stretching a hand	0	0	0	0	0	0	11

Fig. 7. The confusion matrix of our system for static activities, and the one for dynamic activities.

executions of an activity regardless of whether it is static or dynamic. Using such positive and negative examples, SVMs are built for 11 activities.

The performance of these SVMs is examined on a test dataset consisting of 124 executions (test examples), which are performed by the same user to the training dataset in a different day. We use two evaluation measures. The first is an accuracy representing the rate of correct predictions over 124 test examples. Considering concurrent occurrences of static and dynamic activities, the prediction for a test example of a static (or dynamic) activity is determined as the one for which the highest SVM score is observed among four static (or seven dynamic) activities. This is because the accuracy computation mixing static and dynamic activities underestimates the recognition performance. Let us assume that a test example, where the user stretches his hand while standing, is labelled as stretching a hand. For this test example, the SVM for stretching a hand should output a high score, but a higher score may be produced by the model for standing because the user is actually standing. Thus, the separation between static and dynamic activities is necessary for meaningful evaluation.

However, the deterministic evaluation based on accuracies may be too rigid. Thus, we use an Average Precision (AP) [75] as the second evaluation measure. The AP does not require any deterministic decision on a test example, but considers the difference between SVM scores assigned to test examples for a target activity and the ones for the other test examples. Specifically, test examples are ranked based on SVM scores for the target activity. Then, the AP is calculated as the average of precisions each of which is computed at the position of a test example for the target activity. In other words, the AP approximates the area under the recall-precision curve created based on SVM scores. A larger AP means a better result where more test examples for the target activity are ranked at higher positions. Because of this ranking-based statistical computation, we do not consider the separation between static and dynamic activities in AP computation. For evaluation of a static activity, some test examples for dynamic activities may have higher SVM scores than the ones for the static activity. But a good method assigns high SVM scores to test examples for the static activity, and ranks them at higher positions compared to the ranking produced by another method. Like this, APs are useful for relative performance comparison among methods. Finally, the Mean of APs (MAP) over 11 activities is used as an overall evaluation measure.

4.2. Overall performance evaluation

Overall, our system using the early fusion approach achieves an accuracy of 87.1% where evaluations for static and dynamic activities are separated as described before. Fig. 7 (a) and (b) show the confusion matrix for static activities and the one for dynamic activities. In each matrix, rows and columns correspond to ground truth and predicted activities, respectively. As shown in Fig. 7 (a), the most problematic activity is lying, because it is easily confused with the other activities depending on the user's postures. One possible solution is to capture the orientation of the smartglasses by equipping a gravity or magnetometer sensor in them. Except lying, the other activities are recognised quite accurately. Especially in several mis-recognition cases, our system's outputs are reasonable or even correct. For example, as depicted in Fig. 7 (b), our system predicts the activity “putting hand back” for two executions annotated with the ground truth activity “stretching hand”, and these predictions are evaluated as incorrect. However, after the user stretched his hand, he actually put it back during the 5-s executions and forgot the annotation. Like this, our system can perform detailed recognition which identifies activities that the user was not aware of. Fig. 8 displays APs and MAP of our system over 11 activities. Although APs vary depending on activities, our system accomplishes a very high MAP value 88.8%.

After validating the performance of our system in the aforementioned ‘off-line’ experiment, we let it run in ‘real-time’ on the central server (see Fig. 6). To this end, a buffer is defined for each type of sensor data in order to enqueue the latest data and dequeue the oldest ones. In particular, buffers with 1500 time points (7.5 s) are used for *sp-acc*, *sp-gyro*, *sp-grav* and *sp-linacc* obtained with the sampling rate 200 Hz. Buffers with 450 time points are prepared for *sp-mag*, *sw-acc* and *sw-gyro*. These retain 9-s of 50 Hz *sp-mag* data and 6.7-s of 67 Hz *sw-acc* and *sw-gyro* data. A buffer with 150 time points (7.5 s) is used for 20 Hz *sg-acc* data. Note that time lengths of these buffers are longer than those of training examples (5 s). This is due to delays of data transfer from the smartphone to the central server via RabbitMQ. Hence, time lengths of buffers are set to be longer than those of training examples, so that a sufficient information for activity recognition is included. As described in the previous section, the recognition criteria of our system is the ‘inclusion’ of subsequences specific to an activity. Thus, it is enough to roughly define the time length of a buffer, unless we use a very long one where the effect of such specific subsequences is significantly weakened by other subsequences.

Real-time activity recognition is performed at every 2.5 s by encoding sensor data stored in buffers into codebook-based features. But our system actually works much faster. Specifically, the total computational time required for feature extraction from eight types of sensor data and recognition of 11 activities is about 0.005 s (only 58 MB of RAM is consumed). A demonstration video of our activity recognition system can be found online on.[1] Note that this video was created using a previous version where the smartglasses were not incorporated and the computational speed is slightly slower than the current version. Even so, it can be seen that our system appropriately produces high SVM scores depending on the user's activities, and is robust to changes of his postures.

[1] https://youtu.be/sIL08IE_QLE.

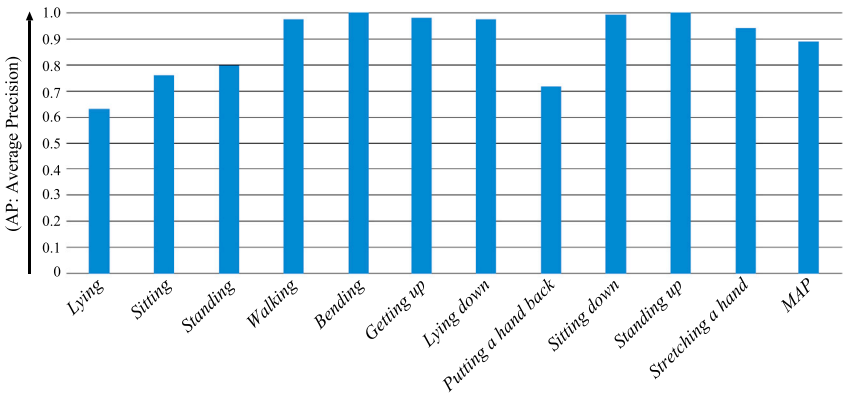

Fig. 8. Average Precisions (APs) and Mean of APs (MAP) of our system over 11 activities.

4.3. Effectiveness of different sensors and devices

Now we investigate the effectiveness of each sensor for activity recognition. The eight bars from the left in Fig. 9 show MAPs obtained using single sensors, while the rightmost one presents the MAP using all the eight sensors (i.e., 88.8% in Fig. 8). As can be seen from Fig. 9, the most effective sensor is *sp-grav* which leads to the MAP 80.0%. From the viewpoint of human perception, it is not intuitive that this smartphone's sensor delivers a high recognition performance over 11 activities, because hand or head movements are important for some of them. For example, although hand movements are a clear clue for stretching a hand, the AP 77.3% is achieved only using *sp-grav*. This implies that non-perceivable body movements characterising this activity exist, and are captured by features extracted with our codebook approach. In addition, Fig. 9 indicates that, compared to performances of single sensors, their (early) fusion yields the performance improvement.

It can happen that some devices are not worn by a user depending on his/her preferences or situations (e.g., he/she may forget to wear the smartglasses). Thus, we examine the effectiveness of each device. Fig. 10 presents MAPs obtained by early fusion of sensors equipped in single devices, and the ones in multiple devices. The rightmost MAP is accomplished using eight sensors in all the three devices (i.e., the rightmost MAPs in Figs. 8 and 9). As shown in Fig. 10, the MAP 86.7% obtained only using the smartphone is very close to the MAP 88.8% with all devices. Thus, one may think that only the smartphone is necessary for activity recognition. However, we think that this high performance of the smartphone is due to the current experimental setting where only 11 activities are targeted. In the future, we will address many more activities in order to exhaustively monitor the daily life of a user (elderly). In such a situation, we believe that the importance of the smartwatch and smartglasses will increase. Also, Fig. 10 indicates that the use of multiple devices causes no performance degradation, and always yields a performance improvement (although it may be small). Thus, using multiple devices is considered as useful for maintaining the generality of our system over different activities.

4.4. Comparison among different fusion approaches

Fig. 11 presents the performance comparison among the early, late and dynamic late fusion approaches. In order to obtain a general and statistically reliable observation, the comparison is made on all possible selections of devices or sensors. Specifically, Fig. 11 (a) displays their MAPs over device combinations adopted in Fig. 10. Fig. 11 (b) shows box plots of MAPs over all the 255 selections resulting from eight sensors. Here, the top, band and bottom of a box represent the third, second (median) and first quartiles of 255 MAPs, respectively. The top and bottom end of the whisker denote the maximum and minimum MAPs. As can be seen from Fig. 11 (a) and (b), the performances of the three fusion approaches are similar. This verifies the effectiveness of the dynamic late fusion approach where weights computed by targeting all the sensors are adaptively re-calculated depending of the user's selection of sensors.

4.5. Performance examination on different sampling rates

We examine whether our activity recognition system is generally applicable to sensor data with different sampling rates. In addition, if a similar performance is obtained for a lower sampling rate for the same sensor, we can reduce the electricity consumption of our system (smartphone) and make its battery life longer. We compare five approaches that perform early fusion of the eight sensors with the following sampling rates:

1. Original: This uses the eight sensors with their original sampling rates. (i.e., the MAP is 88.8% as shown in Figs. 8–11).
2. Half: Sampling rates of the eight sensors are reduced to half. Correspondingly, codebooks are constructed using *w*s and *l*s that are also reduced to half (i.e., $w = 64$ and $l = 4$ for *sp-acc*, *sp-gyro*, *sp-grav* and *sp-linacc*, $w = 32$ and $l = 2$ for *sp-mag*, *sw-acc* and *sw-gyro*, and $w = 16$ and $l = 1$ for *sg-acc*). As with *Original*, all the codebooks consist of $N = 1024$ codewords.
3. Half-codebook: Sampling rates of the eight sensors are reduced to half, and the settings of *w*, *l* and *N* are the same to *Half*. But codebooks are not newly constructed, instead they are created by subsampling values every two time points from codewords used in *Original*.
4. Quarter: Sampling rates of the eight sensors are quartered, correspondingly codebooks with $N = 1024$ codewords are constructed using *w*s and *l*s that are also quartered (i.e., $w = 32$ and $l = 2$ for *sp-acc*, *sp-gyro*, *sp-grav* and *sp-linacc*, $w = 16$ and $l = 1$ for *sp-mag*, *sw-acc* and *sw-gyro*, and $w = 8$ and $l = 1$ for *sg-acc*).
5. Quarter-codebook: Similar to *Hand-codebook*, for quartered sampling rates, codebooks are created by subsampling values every four time points from codewords used in *Original*.

Since the quartered sampling rate of *sg-acc* is very low 5 Hz, we do not test our system on unreasonably low rates. Moreover, it would be possible to finely tune *w*, *l* and *N* on each sampling rate and get a better recognition result. However, this is out of our scope because the main purpose of this paper is to show the generality of our activity recognition architecture including the codebook approach.

Fig. 12 shows the performance comparison among five approaches described above. As seen from this figure, their MAPs are very similar, specifically those of *Original*, *Half*, *Half-codebook*, *Quarter* and *Quarter-codebook* are 88.8%, 88.8%, 88.4%, 88.5% and 86.8%, respectively. This result indicates the generality of our system that can consistently work on different sampling rates. In addition, apart from the eight sensors used in this paper, our system is expected to work on other sensors with varied sampling rates. Also, since codebooks created by downsampling the original ones work on lower sampling rates, it would be possible to deal

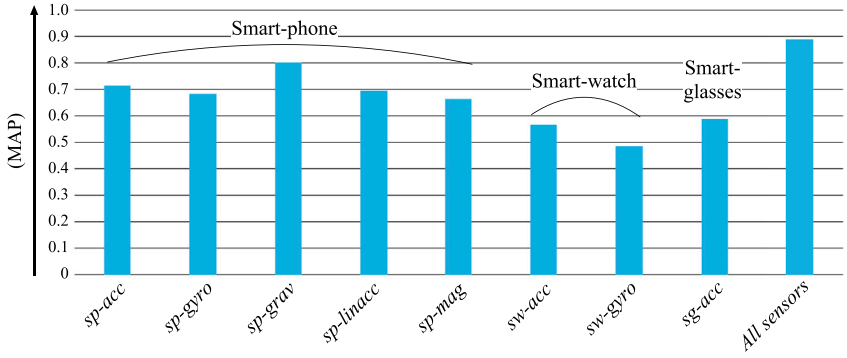

Fig. 9. Performance comparison among individual sensors.

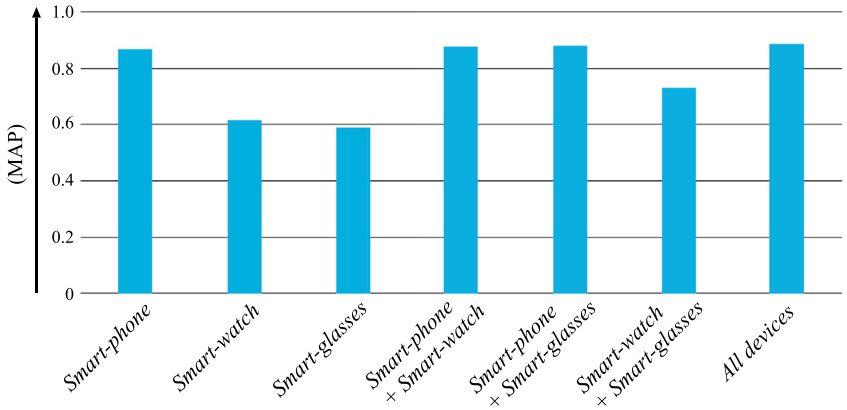

Fig. 10. Performance comparison in terms of device combinations.

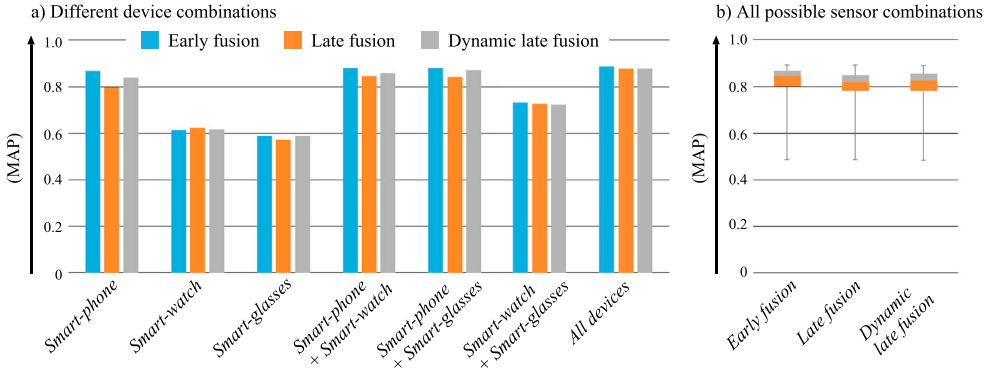

Fig. 11. Performance comparison among the early, late, and dynamic late fusion approaches.

with a situation where sampling rates of sensors dynamically change.

4.6. Cross-user performance examination

We examine the cross-user performance of our system by checking whether a recognition model trained on sensor data from one user is applicable to data from another. To this end, the following dataset is collected from three users, *S1*, *S2* and *S3*, using *sp-acc*, *sp-gyro*, *sp-grav*, *sp-linacc*, *sp-mag*, *sw-acc* and *sw-gyro*.[2] Every user executed each of the 11 activities at least 20 times. In total, sensor data for *S1*, *S2* and *S3* consist of 264, 240 and 220 activity executions, respectively. The sensor setting and parameters of the codebook-based method are exactly the same as those described in Section 4.1 (early fusion is used).

Fig. 13 shows MAPs in the following cross-user setting: For each bar, the left side of the symbol "$>$" indicates one user (or two users) whose sensor data are used to train a recognition model, which is applied to the data obtained from the user on the right side. It is natural that the MAPs in Fig. 13 are lower than those in the single-user case discussed before. But, as depicted by each set of 3 bars where the same test data are used, using sensor data from two users always leads to the performance improvement over using data from one user. Thus, as sensor data are collected from more users, a recognition model can cover a larger variety of activity executions and achieve a better performance. This generality of our system on multiple users can be also seen in our recent paper, where the codebook-based recognition method accomplished an excellent recognition performance on sensor data obtained for 12 subjects and 11 activities [76]. Moreover, this kind of generality has been experimentally validated in some existing works like [77,78].

[2] The dataset has been established by re-using past data that were collected before incorporating sg-acc into our system.

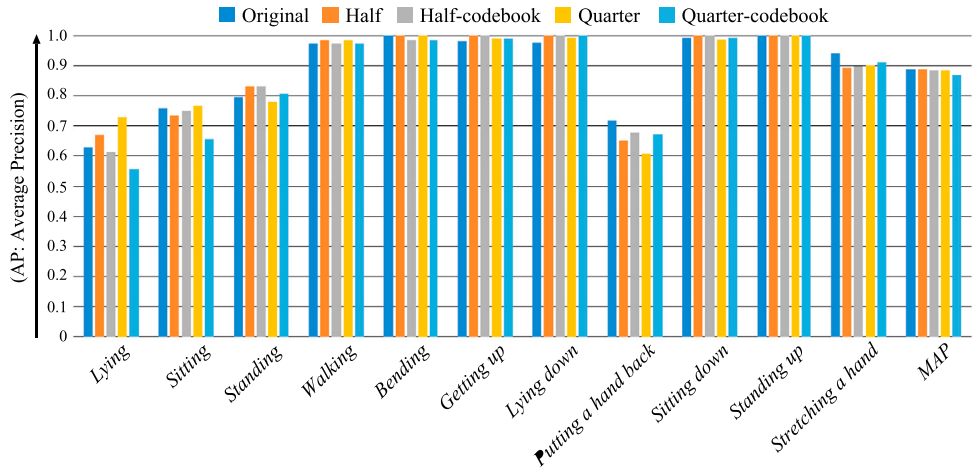

Fig. 12. Performance comparison on different sampling rates.

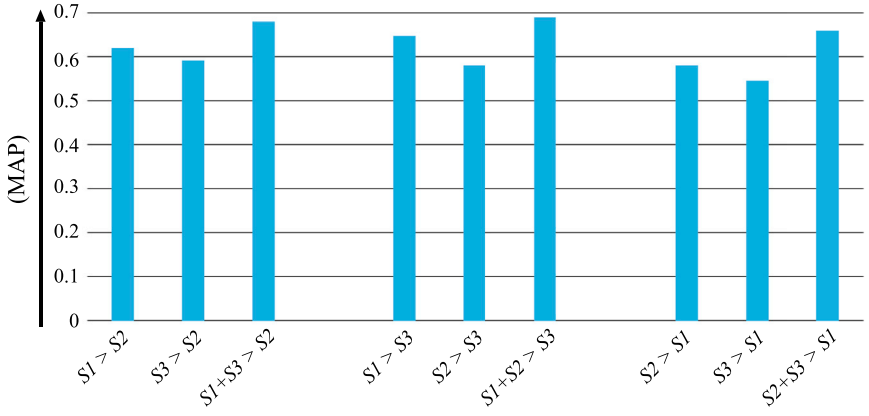

Fig. 13. Performance comparison in the cross-user settings.

5. Conclusion and future work

In the paper, we introduced a general framework for sensor-based human activity recognition. It can be easily adapted to various sensors equipped in mobile devices. For the former, a smartphone is used as a temporal data storage where sensor data from different devices are gathered with their own programming interfaces. Those data are then sent to the central server via AMQP-based data transfer. For data analysis, the codebook approach is used to analyse each type of sensor data and extract a useful representation (feature) for activity recognition without relying on prior knowledge. The effectiveness of our framework has been validated on a real-world dataset consisting of eight types of sensor data obtained from a smartphone, smartwatch and smartglasses.

For readers who are interested in the codebook approach, we refer to [5,76] for more details. There we have proven the effectiveness of the codebook approach for additional sensors that are not included in this paper, such as Blood Volume Pressure (BVP), Galvanic Skin Response (GSR), RESpiration (RES), ElectroMyoGram (EMG) and ElectroOculoGram (EOG). This is an additional evidence for the generality of the proposed framework. The source code for the codebook-based feature extraction is available on http://www.pr.informatik.uni-siegen.de/en/human-activity-recognition. Thus, one can implement an activity recognition system by combing this code with his/her own AMQP-based data transfer.[3]

In the future, we will collect a much larger training dataset in order to not only improve the recognition performance, but also increase the number of activities to be recognised. However, when such large-scale training data are used, non-linear SVMs that are currently used have problems in terms of computational costs and memory consumptions. This is because a non-linear SVM needs to compute kernel values (similarities) of a test example to training examples selected as support vectors (in the worst case all training examples become support vectors). Thus, we plan to use a linear SVM, since its classification can be done just by taking the product between the feature vector of a test example and the optimised weight vector. Furthermore, the linear SVM will run on an extended version of codebook-based feature, like Fisher vector representation or Vector of Locally Aggregated Descriptors (VLAD) [79]. This represents an example with a very high-dimensional feature, with which the linear SVM can attain a high discrimination power. Finally, we will

[3] Since on-line processing, including the data transfer, is application-dependent and many implementations for k-means clustering and SVM training/test are available nowadays, the source code only includes an off-line version of the codebook-based feature extraction. We assume that our code is customisable for one's own on-line processing.

also explore how to identify high-level activities (e.g., cooking, cleaning a room and dressing) by combining recognition results of low-level activities that our current system targets. To further improve our system, we will also add context information like location and the time of day, since these often are important features when trying to distinguish among various similar activities [80]. However, the integration of location and the time of day into the system requires an additional mechanism since their representation differs from the typical time series that are produced by sensors.

Conflict of interest statement

None Declared.

Acknowledgement

Research and development activities leading to this article have been supported by the German Federal Ministry of Education and Research within the project “Cognitive Village: Adaptively Learning Technical Support System for Elderly” (Grant Number: 16SV7223K) and the German Research Foundation (DFG) as part of the research training group GRK 1564 “Imaging New Modalities”.

References

[1] G.D. Abowd, A.F. Bobick, I.A. Essa, E.D. Mynatt, W.A. Rogers, The aware home: a living laboratory for technologies for successful aging, in: Proceedings of the AAAI-02 Workshop Automation as Caregiver, 2002, pp. 1–7.
[2] Monitor stress, seizures, activity, sleep. https://www.empatica.com/. (Accessed 15 March 2017).
[3] Jins meme: the world's first wearable eyewear that lets you see yourself. https://jins-meme.com/en/. (Accessed 15 March 2017).
[4] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[5] K. Shirahama, L. Köping, M. Grzegorzek, Codebook approach for sensor-based human activity recognition, in: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct, UbiComp '16, 2016, pp. 197–200.
[6] M. Amiribesheli, A. Benmansour, A. Bouchachia, A review of smart homes in healthcare, J. Ambient Intell. Humanized Comput. 6 (4) (2015) 495–517.
[7] S. Solaimani, W. Keijzer-Broers, H. Bouwman, What we do - and don't - know about the smart home: an analysis of the smart home literature, Indoor Built Environ. 24 (3) (2015) 370–383.
[8] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: a survey on enabling technologies, protocols, and applications, IEEE Commun. Surveys Tutorials 17 (4) (2015) 2347–2376.
[9] R. Li, B. Lu, K.D. McDonald-Maier, Cognitive assisted living ambient system: a survey, Digital Commun. Networks 1 (4) (2015) 229–252.
[10] E.M. Tapia, S.S. Intille, K. Larson, Activity recognition in the home using simple and ubiquitous sensors, in: A. Ferscha, F. Mattern (Eds.), Proceedings of the Second International Conference on Pervasive Computing, PERVASIVE'04, 2004, pp. 158–175.
[11] T. van Kasteren, A. Noulas, G. Englebienne, B. Kröse, Accurate activity recognition in a home setting, in: Proceedings of the 10th International Conference on Ubiquitous Computing, UbiComp '08, 2008.
[12] B. Logan, J. Healey, M. Philipose, E.M. Tapia, S. Intille, A long-term evaluation of sensing modalities for activity recognition, in: Proceedings of the 9th International Conference on Ubiquitous Computing, UbiComp '07, 2007.
[13] M. Philipose, K.P. Fishkin, M. Perkowitz, D.J. Patterson, D. Fox, H. Kautz, D. Hahnel, Inferring activities from interactions with objects, IEEE Pervasive Comput. 3 (4) (2004) 50–57.
[14] N.C. Krishnan, D.J. Cook, Activity recognition on streaming sensor data, Pervasive Mob. Comput. 10 (Part B) (2014) 138–154.
[15] S. Hagler, D. Austin, T.L. Hayes, J. Kaye, M. Pavel, Unobtrusive and ubiquitous in-home monitoring: a methodology for continuous assessment of gait velocity in elders, IEEE Trans. Biomed. Eng. 57 (4) (2010) 813–820.
[16] P. Rashidi, D.J. Cook, L.B. Holder, M. Schmitter-Edgecombe, Discovering activities to recognize and track in a smart environment, IEEE Trans. Knowl. Data Eng. 23 (4) (2011) 527–539.
[17] S.S. Intille, K. Larson, E.M. Tapia, J.S. Beaudin, P. Kaushik, J. Nawyn, R. Rockinson, Using a live-in laboratory for ubiquitous computing research, in: Proceedings of the Forth International Conference on Pervasive Computing, PERVASIVE'06, 2006, pp. 349–365.
[18] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, E. Jansen, The gator tech smart house: a programmable pervasive space, Computer 38 (3) (2005) 50–60.
[19] F. Grossi, V. Bianchi, G. Matrella, I.D. Munari, P. Ciampolini, An assistive home automation and monitoring system, in: 2008 Digest of Technical Papers - International Conference on Consumer Electronics, 2008, pp. 1–2.
[20] V. Bianchi, F. Grossi, I.D. Munari, P. Ciampolini, Musa: a multisensor wearable device for aal, in: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS), 2011, pp. 375–380.
[21] D.J. Cook, A.S. Crandall, B.L. Thomas, N.C. Krishnan, Casas: a smart home in a box, Computer 46 (7) (2013) 62–69.
[22] L. Chen, J. Hoey, C.D. Nugent, D.J. Cook, Z. Yu, Sensor-based activity recognition, IEEE Trans. Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (6) (2012) 790–808.
[23] A. Bulling, U. Blanke, B. Schiele, A tutorial on human activity recognition using body-worn inertial sensors, ACM Comput. Surv. 46 (3) (2014), 33:1–33:33.
[24] O.D. Lara, M.A. Labrador, A survey on human activity recognition using wearable sensors, IEEE Commun. Surveys Tutorials 15 (3) (2013) 1192–1209.
[25] L. Bao, S.S. Intille, Activity Recognition from User-annotated Acceleration Data, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 1–17.
[26] T.G. Stavropoulos, G. Meditskos, S. Andreadis, I. Kompatsiaris, Real-time health monitoring and contextualised alerts using wearables, in: 2015 International Conference on Interactive Mobile Communication Technologies and Learning (IMCL), 2015, pp. 358–363.
[27] N.D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, A.T. Campbell, A survey of mobile phone sensing, IEEE Commun. Mag. 48 (9) (2010) 140–150.
[28] M. Berchtold, M. Budde, D. Gordon, H.R. Schmidtke, M. Beigl, Actiserv: activity recognition service for mobile phones, in: Proceedings of the 2010 ACM International Symposium on Wearable Computers, ISWC '10, 2010, pp. 1–8.
[29] A. Mannini, A.M. Sabatini, Machine learning methods for classifying human physical activity from on-body accelerometers, Sensors 10 (2) (2010) 1154–1175.
[30] D. McGlynn, M.G. Madden, An ensemble dynamic time warping classifier with application to activity recognition, in: M. Bramer, M. Petridis, A. Hopgood (Eds.), Research and Development in Intelligent Systems XXVII, Springer London, London, 2011, pp. 339–352.
[31] J. Liu, L. Zhong, J. Wickramasuriya, V. Vasudevan, uwave: Accelerometer-based personalized gesture recognition and its applications, Pervasive Mob. Comput. 5 (6) (2009) 657–675.
[32] F. Zhou, F. Torre, Canonical time warping for alignment of human behavior, in: Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, A. Culotta (Eds.), Advances in Neural Information Processing Systems 22, 2009, pp. 2286–2294.
[33] J. Lin, Y. Li, Finding structural similarity in time series data using bag-of-patterns representation, in: Proceedings of the 21st International Conference on Scientific and Statistical Database Management, SSDBM 2009, 2009, pp. 461–477.
[34] D. Trabelsi, S. Mohammed, F. Chamroukhi, L. Oukhellou, Y. Amirat, An unsupervised approach for automatic activity recognition based on hidden markov model regression, IEEE Trans. Autom. Sci. Eng. 10 (3) (2013) 829–835.
[35] C. Zhu, W. Sheng, Human daily activity recognition in robot-assisted living using multi-sensor fusion, in: 2009 IEEE International Conference on Robotics and Automation, 2009, pp. 2154–2159.
[36] A. Bulling, J.A. Ward, H. Gellersen, G. Tröster, Robust recognition of reading activity in transit using wearable electrooculography, in: Proceedings of the Sixth International Conference on Pervasive Computing, PERVASIVE'08, 2008, pp. 19–37.
[37] J.-H. Hong, J. Ramos, A.K. Dey, Understanding physiological responses to stressors during physical activity, in: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, UbiComp '12, 2012, pp. 270–279.
[38] J. Kim, E. Andr, Emotion recognition based on physiological changes in music listening, IEEE Trans. Pattern Anal. Mach. Intell. 30 (12) (2008) 2067–2083.
[39] K. Plarre, A. Raij, S.M. Hossain, A.A. Ali, M. Nakajima, M. Al'absi, E. Ertin, T. Kamarck, S. Kumar, M. Scott, D. Siewiorek, A. Smailagic, L.E. Wittmers, Continuous inference of psychological stress from sensory measurements collected in the natural environment, in: Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks, 2011, pp. 97–108.
[40] T. Gu, L. Wang, Z. Wu, X. Tao, J. Lu, A pattern mining approach to sensor-based human activity recognition, IEEE Trans. Knowl. Data Eng. 23 (9) (2011) 1359–1372.
[41] S. Koelstra, C. Muhl, M. Soleymani, J.S. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt, I. Patras, Deap: a database for emotion analysis ;using physiological signals, IEEE Trans. Affective Comput. 3 (1) (2012) 18–31.
[42] M. Soleymani, J. Lichtenauer, T. Pun, M. Pantic, A multimodal database for affect recognition and implicit tagging, IEEE Trans. Affective Comput. 3 (1) (2012) 42–55.
[43] L.C. Jatoba, U. Grossmann, C. Kunze, J. Ottenbacher, W. Stork, Context-aware mobile health monitoring: evaluation of different pattern recognition methods for classification of physical activity, in: Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS '08, 2008, pp. 5250–5253.
[44] A. Phinyomark, P. Phukpattaranont, C. Limsakul, Feature reduction and selection for emg signal classification, Expert Syst. Appl. 39 (8) (2012) 7420–7431.
[45] A. Bulling, J.A. Ward, H. Gellersen, G. Troster, Eye movement analysis for activity recognition using electrooculography, IEEE Trans. Pattern Anal. Mach. Intell. 33 (4) (2011) 741–753.
[46] R.W. Picard, E. Vyzas, J. Healey, Toward machine emotional intelligence: analysis of affective physiological state, IEEE Trans. Pattern Anal. Mach. Intell. 23 (10) (2001) 1175–1191.
[47] G. Csurka, C. Dance, L. Fan, J. Willamowski, C. Bray, Visual categorization with bags of keypoints, in: Proceedings of the Workshop on Statistical Learning in Computer Vision, No. 1-22, 2004, pp. 1–2.
[48] Y.G. Jiang, J. Yang, C.W. Ngo, A.G. Hauptmann, Representations of keypoint-based semantic concept detection: a comprehensive study, IEEE Trans. Multimed. 12 (1) (2010) 42–53.

[49] J. Wang, P. Liu, M.F. She, S. Nahavandi, A. Kouzani, Bag-of-words representation for biomedical time series classification, Biomed. Signal Process Contr. 8 (6) (2013) 634–644.
[50] P. Ordonez, T. Armstrong, T. Oates, J. Fackler, Using modified multivariate bag-of-words models to classify physiological data, in: Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops, 2011, pp. 534–539.
[51] M.G. Baydogan, G. Runger, E. Tuv, A bag-of-features framework to classify time series, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2796–2802.
[52] T. Plötz, N.Y. Hammerla, P. Olivier, Feature learning for activity recognition in ubiquitous computing, in: Proceedings of the 22th International Joint Conference on Artificial Intelligence, IJCAI'11, 2011, pp. 1729–1734.
[53] H.P. Martinez, Y. Bengio, G.N. Yannakakis, Learning deep physiological models of affect, IEEE Comput. Intell. Mag. 8 (2) (2013) 20–33.
[54] F.J.O. Morales, D. Roggen, Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations, in: Proceedings of the 2016 ACM International Symposium on Wearable Computers, ISWC '16, 2016, pp. 92–99.
[55] V. Radu, N.D. Lane, S. Bhattacharya, C. Mascolo, M.K. Marina, F. Kawsar, Towards multimodal deep learning for activity recognition on mobile devices, in: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct, UbiComp '16, 2016, pp. 185–188.
[56] N.Y. Hammerla, S. Halloran, T. Plötz, Deep, convolutional, and recurrent models for human activity recognition using wearables, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, IJCAI '16, 2016, pp. 1533–1540.
[57] S. Vinoski, Advanced message queuing protocol, IEEE Internet Comput. 10 (6) (2006) 87–89.
[58] D. Cook, N. Krishnan, Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data, Wiley Series on Parallel and Distributed Computing, John Wiley & Sons, New Jersey, 2015.
[59] Sensors overview. https://developer.android.com/guide/topics/sensors/sensors_overview.html. (Accessed 25 March 2017).
[60] Sensors event overview. https://developer.android.com/reference/android/hardware/SensorEvent.html#values. (Accessed 18 July 2017).
[61] K. Kunze, P. Lukowicz, Sensor placement variations in wearable activity recognition, IEEE Pervasive Comput. 13 (4) (2014) 32–41.
[62] O. Banos, M.A. Toth, M. Damas, H. Pomares, I. Rojas, Dealing with the effects of sensor displacement in wearable activity recognition, Sensors 14 (6) (2014) 9995–10023.
[63] L. Atallah, B. Lo, R. King, G.Z. Yang, Sensor positioning for activity recognition using wearable accelerometers, IEEE Trans. Biomedical Circuits and Systems 5 (4) (2011) 320–329.
[64] I. Cleland, B. Kikhia, C. Nugent, A. Boytsov, J. Hallberg, K. Synnes, S. McClean, D. Finlay, Optimal placement of accelerometers for the detection of everyday activities, Sensors 13 (7) (2013) 9183–9200.
[65] W. Ugulino, D. Cardador, K. Vega, E. Velloso, R. Milidiú, H. Fuks, Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 52–61.
[66] RabbitMQ - messaging that just works. https://www.rabbitmq.com/. (Accessed 13 March 2017).
[67] E. Nowak, F. Jurie, B. Triggs, Sampling strategies for bag-of-features image classification, in: A. Leonardis, H. Bischof, A. Pinz (Eds.), Proceedings of the 9th European Conference on Computer Vision (ECCV 2006), Part IV, 2006.
[68] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, third ed., Morgan Kaufmann, 2011.
[69] J.C. van Gemert, C.J. Veenman, A.W.M. Smeulders, J.-M. Geusebroek, Visual word ambiguity, IEEE Trans. Pattern Anal. Mach. Intell. 32 (7) (2010) 1271–1283.
[70] V. Vapnik, Statistical Learning Theory, Wiley-Interscience, 1998.
[71] C.-C. Chang, C.-J. Lin, Libsvm: a library for support vector machines, ACM Trans. Intelligent System. Technol. 2 (3) (2011), 27:1–27:27.
[72] J. Zhang, M. Marszalek, S. Lazebnik, C. Schmid, Local features and kernels for classification of texture and object categories: a comprehensive study, Int. J. Comput. Vis. 73 (2) (2007) 213–238.
[73] C.G.M. Snoek, M. Worring, A.W.M. Smeulders, Early versus late fusion in semantic video analysis, in: Proceedings of the 13th Annual ACM International Conference on Multimedia, MM '05, 2005, pp. 399–402.
[74] Microsoft band. https://www.microsoft.com/microsoft-band. (Accessed 15 March 2017).
[75] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, New York, NY, USA, 2008.
[76] K. Shirahama, M. Grzegorzek, On the generality of codebook approach for sensor-based human activity recognition, Electronics 6 (2) (2017), 44.
[77] J. Lester, T. Choudhury, G. Borriello, A practical approach to recognizing physical activities, in: Proceedings of the 4th International Conference on Pervasive Computing (PERVASIVE 2006), 2006, pp. 1–16.
[78] H.-T. Cheng, M. Griss, P. Davis, J. Li, D. You, Towards zero-shot learning for human activity recognition using semantic attribute sequence model, in: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013), 2013, pp. 355–358.
[79] X. Peng, L. Wang, X. Wang, Y. Qiao, Bag of visual words and fusion methods for action recognition, Comput. Vis. Image Understand. 150 (C) (2016) 109–125.
[80] K. Koile, K. Tollmar, D. Demirdjian, H. Shrobe, T. Darrell, Activity zones for context-aware computing, in: Proceedings of the Fifth International Conference on Ubiquitous Computing (UbiComp 2003), 2003, pp. 90–106.