Real-Time Gesture Recognition using a Particle Filtering Approach

Frédéric Li[1], Lukas Köping[1], Sebastian Schmitz[2] and Marcin Grzegorzek[1,3]
[1]*Research Group for Pattern Recognition, University of Siegen, Siegen, Germany*
[2]*Fraunhofer SCAI, St. Augustin, Germany*
[3]*University of Economics in Katowice, Faculty of Informatics and Communication, Katowice, Poland*
{*lukas.koeping, frederic.li*}*@uni-siegen.de, sebastian.schmitz@scai.fraunhofer.de, marcin.grzegorzek@uni-siegen.de*

Keywords: Gesture Recognition, Particle Filter, Gesture Spotting, Dynamic Time Warping, DTW Barycenter Averaging.

Abstract: In this paper we present an approach for real-time gesture recognition using exclusively 1D sensor data, based on the use of Particle Filters and Dynamic Time Warping Barycenter Averaging (DBA). In a training phase, sensor records of users performing different gestures are acquired. For each gesture, the associated sensor records are then processed by the DBA method to produce one average record called template gesture. Once trained, our system classifies one gesture performed in real-time, by computing -using particle filters- an estimation of its probability of belonging to each class, based on the comparison of the sensor values acquired in real-time to those of the template gestures. Our method is tested on the accelerometer data of the Multimodal Human Activities Dataset (MHAD) using the Leave-One-Out cross validation, and compared with state-of-the-art approaches (SVM, Neural Networks) adapted for real-time gesture recognition. It manages to achieve a 85.30% average accuracy and outperform the others, without the need to define hyper-parameters whose choice could be restrained by real-time implementation considerations.

1 INTRODUCTION

Gesture recognition is the problem of finding meaning in the movement of humans' hands, arms, face, head and/or body in order to enable an interaction between humans and machines (Mitra and Acharya, 2007). The main purpose of gesture recognition systems (GRS) is to offer a natural interaction between humans and machines. An example for this are today's Virtual Reality applications where gesture recognition plays a major role in order to produce an immersive feeling. Systems that are based on additional input devices often destroy this immersive feeling since they require the user to act in an unnatural way. For example, grabbing an object should not be performed by pushing a button on some external device. Instead, the natural way would be to put forth one's hand and perform a grabbing gesture. However, the problem of gesture recognition is a difficult one since it requires to satisfy several different constraints:

- **User Independent Performance:** GRS must work for people who are not included in the initial training set. The alternative is to introduce some sort of calibration phase to fit the system's algorithm to the user. However, such a calibration phase might discourage people from using the system if it takes too much effort.
- **Naturalness of Gestures:** Even the same individual user does not always perform every gesture in the same way. Instead, gestures have some free parameters to them like the speed or the intensity in which they are performed. Gesture recognition systems should be able to acknowledge these variations and maybe even react to them.
- **Separation from Relevant and Irrelevant Data:** Not all of a user's movements are particular gestures. The GRS should be able to distinguish between gestures and random user movements (gesture spotting). Forcing the user to notify the system of the beginning and end of a gesture would again provide an unnatural feeling.
- **Realtime Performance:** The delay between the performance of a gesture and its recognition should be minimal. This is especially needed in applications that are using Virtual or Augmented Reality since big delays disturb the interactiveness of these systems. An early detection of gestures is only possible if the algorithm is able to process the data in realtime.

In this work we propose a method that accounts for

Li, F., Köping, L., Schmitz, S. and Grzegorzek, M.
Real-Time Gesture Recognition using a Particle Filtering Approach.
DOI: 10.5220/0006189603940401
In *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2017)*, pages 394-401
ISBN: 978-989-758-222-6

all the aforementioned difficulties. For this, we are formulating gesture recognition as a location tracking problem that can be solved using recursive state estimation methods. Our approach uses particle filtering in particular, and takes multi-modal data input in form of time series gathered by accelerometer signals. It can recognize gestures of different lengths, without needing input from the user at the start or end of a gesture.

The paper is structured as follows. In section 2 we give an overview of related work. Section 3 introduces our method which is evaluated in section 4. Finally, section 5 summarises our results and gives an outlook to future work.

2 RELATED WORK

The literature offers a variety of approaches to gesture recognition (Mitra and Acharya, 2007). To narrow down the existing work we exclude papers that focus on video data as input and instead focus on gesture recognition algorithms that are based on time series data, e.g. accelerometer. uWave (Liu et al., 2009) is a gesture recognition system that uses a three-axis accelerometer as input data. Templates of the gestures that should be recognised by the system are stored in a database, where for each template gesture one or two examples exist. uWave determines which gesture is performed by the user by using the best match between the input data and the template gestures based on Dynamic Time Warping (DTW) as similarity measure. However, this procedure delivers wrong results if the input gesture is totally different from any of the template gestures. To avoid these kinds of misclassifications the authors introduce a threshold for the similarity. If the similarity between the input gesture and all other template gestures is below this threshold, the gesture is recognised as unknown. uWave achieves strong results in the classification of user dependent gestures. However, the system requires the user to press and hold a button during the execution of a gesture.

In (Akl et al., 2011) the authors use DTW as similarity measure for clustering with the Affinity Propagation algorithm. This offline training phase returns a set of exemplars for the gestures of the system. In the recognition phase firstly those exemplars are chosen that are closest to the input. In a second step the data is projected into a lower dimensional space where the final classification takes place. For the user-dependent case a recognition rate of 96.84% on the same dataset as uWave is reported.

The authors of (Chudgar et al., 2014) focus on the problems of gesture spotting and the identification of the intensity of a gesture. Gesture spotting describes the problem of finding the beginning and the end of a gesture without explicitly marking it, e.g. by pressing a button. On the other hand, identifying the intensity of a gesture can help making gesture recognition systems more convenient, e.g. a fast gesture increases the TV volume more than a slow gesture. To incorporate gesture spotting (Chudgar et al., 2014) use a threshold that is based on the variance of the accelerometer signal. For recognition they match the input with the available training gestures by DTW similarity and identify the intensity of a gesture with the help of signal variance.

The topic of finding the intensity of a gesture can also be found in (Caramiaux et al., 2014). The proposed Gesture Variation Follower (GVF) does not only classify gestures but also has the ability to estimate its speed, scale and rotation. For this, they extend the work of (Black and Jepson, 1998), where drawings on a whiteboard are classified and simultaneously the speed and scaling of the gesture is estimated. Both approaches are based on the CONDENSATION algorithm which is also known as particle filtering or Sequential Monte Carlo (see (Isard and Blake, 1998), (Gordon et al., 1993), (Arulampalam et al., 2002) for a detailed explanation). Particle filters are a method to approximate over time the probability density of a hidden state by integrating a series of observations. Each particle represents one particular state and has an associated weighting. The hidden state of the GFV consists of a discrete variable for the class label and some real-valued variables for the speed, scale and rotation of a gesture. Observations are simply the incoming accelerometer data values. When the user performs a gesture with an arbitrary speed and rotation, the weightings of the particles increase that represent exactly this particular gesture class label and that also match with the speed, scale and rotation. The weighting of all the other particles will decrease over time. Our work is mostly motivated by this approach and follows its main principles. We formulate gesture recognition in the same way as (Caramiaux et al., 2014) using particle filtering. However, we change the way that our system evolves over time and the way observations are integrated into the probability estimation. In addition, we calculate average gestures in the training phase that serve as templates for the testing phase.

One point to be noted is the lack of recent similar approaches in the literature. To some extend related to our study, a survey presenting an overview of existing online and offline activity recognition methods by mobile phones (Shoaib et al., 2015) highlights the

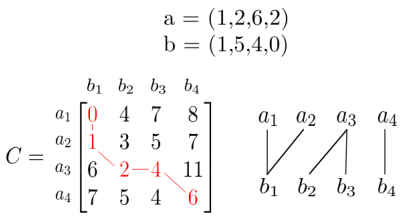

Figure 1: Example of DTW applied to two 1D time sequences. The *warping path* (in red) can be found after the computation of the cost matrix C. The DTW associations between the elements of the two sequences (to the right) can be deduced from its coordinates.

dominance of approaches like decision trees, SVM, KNN and naive Bayes, and the rarity of particle filtering for real-time gesture recognition problems.

3 METHOD DESCRIPTION

3.1 DTW Barycentric Averaging (DBA)

DBA is a method of time sequences averaging, which relies on the use of the *Dynamic Time Warping* algorithm (*DTW*) to compute similarities between time sequences while taking the time dimension into account. It also provides information about the matching between the elements of two different times sequences.

Given two sequences **a** and **b** (of lengths m and $n \in \mathbb{N}^*$ not necessarily equal, with values in $\mathbb{R}$), and a distance function $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, the DTW method provides a cost matrix $\mathbf{C} \in \mathbb{M}_{m \times n}(\mathbb{R})$ containing the DTW distances between all pairs of elements (a_i, b_j), recursively computed by the relations given in (Petitjean and Gançarski, 2012). A few interesting properties of this cost matrix **C** have been highlighted as well in (Petitjean and Gançarski, 2012). In particular, the cost matrix provides the associations between the respective elements of the two time series related to it, by considering the coordinates of the *warping path* (Petitjean and Gançarski, 2012) (i.e. path of minimal cost linking $\mathbf{C}(1,1)$ and $\mathbf{C}(m,n)$). The value in $\mathbf{C}(m,n)$ also provides the DTW distance between the two sequences compared.

Finding those time series element associations forms the basis of the DBA method: for a set of time series $S = (\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_M)$ (with $M \in \mathbb{N}^*$) of not necessarily equal lengths, the DBA algorithm computes $\mathbf{s}_{\text{avg}}$, an estimation of the average time sequence of all the $\mathbf{s}_i$, by the following process:

1. Initialize at random $\mathbf{s}_{\text{avg}}$, of length "well chosen".
2. For every sequence $\mathbf{s}_i$ in S, compute the DTW between $\mathbf{s}_i$ and $\mathbf{s}_{\text{avg}}$ to find the coordinates associations.
3. For every element $\mathbf{s}_{\text{avg}}(i)$ in $\mathbf{s}_{\text{avg}}$, replace $\mathbf{s}_{\text{avg}}(i)$ by the mean value of all coordinates across all $\mathbf{s}_i$ in S, associated with $\mathbf{s}_{\text{avg}}(i)$ at step 2.
4. Repeat steps 2 and 3 until $\mathbf{s}_{\text{avg}}$ converges.

In the scope of our project, we decided to apply the DBA algorithm M times during the training phase, by initializing $\mathbf{s}_{\text{avg}}$ to each sequence of the set S, and in the end select the resulting average sequence minimizing the average DTW distance between itself and the other sequences of the set S.

3.2 Velocity Factor and Template Gestures

A dataset of temporal sensor readings from A different sensors, and for G gestures, acquired from U different users, who each performed R different repetitions of each gesture, is available for the training phase of our system (with $(A,G,U,R) \in (\mathbb{N}^*)^4$). The timestamps (real time values in seconds, corresponding to each sampling moment) are also available for all sensor readings.

Since the number of readings in the training dataset could vary and result in large datasets, a way to guarantee the scalability of our system by compressing the information contained in those time sequences has to be employed. Furthermore, this compression of the information has to take into account the multi-sensor aspect of the dataset. To achieve both goals, we compute for each gesture $g \in \{1, ..., G\}$ a *template gesture* using DBA on the sensor readings of the dataset. However, we decide to use a variation of the DBA method described previously, by applying it on the set of multi-sensor time sequences related to the gesture g (vector of A vectors of sensor values). The distance function d, necessary to compute the DTW cost matrix, is chosen in this case as the Euclidean distance in $\mathbb{R}^{\mathbb{A}}$. The averaging function of elements of sequences, needed to update the elements of the average sequence during the iterations of the DBA process, is also extended to the multidimensional case: for $\mathbf{s}_1$, $\mathbf{s}_2$, ..., $\mathbf{s}_n$ $n \in \mathbb{N}^*$ multi-sensor coordinates (in $\mathbb{R}^{\mathbb{A}}$), we define the average of the $(\mathbf{s}_i)_i$, avg_A, by

$$\text{avg}_A(\mathbf{s}_1, ..., \mathbf{s}_n) = \begin{pmatrix} \text{avg}(\mathbf{s}_1(1), \mathbf{s}_2(1), ..., \mathbf{s}_n(1)) \\ \text{avg}(\mathbf{s}_1(2), \mathbf{s}_2(2), ..., \mathbf{s}_n(2)) \\ ... \\ \text{avg}(\mathbf{s}_1(A), \mathbf{s}_2(A), ..., \mathbf{s}_n(A)) \end{pmatrix} \in \mathbb{R}^{\mathbb{A}}$$

We denote $S_g = \{\mathbf{M}^g_{u,r} \mid u \in \{1,...,U\},\ r \in \{1,...,R\}\}$ the set of multi-sensor time series related to gesture g. Each element $\mathbf{M}^g_{u,r}$ of this set is the concatenation of sensor readings related to the r^{th} performance of gesture g by user u. $\mathbf{M}^g_{u,r}$ is therefore a matrix of size $l_{u,r} \times A$, with $l^g_{u,r} \in \mathbb{N}^*$ number of data points acquired for this recording of the gesture. DBA is applied on all sets S_g, by considering each $\mathbf{M}^g_{u,r}$ as a vector of *multi-sensor coordinates* (i.e. vectors of size A), to obtain an average multi-sensor time sequence $\mathbf{T}_g \in \mathbb{M}_{l_g \times A}(\mathbb{R})$ (with $l_g \in \mathbb{N}^*$ length of the average sequence determined by the DBA algorithm), called *template gesture*.

DBA provides an average sequence based on the values of the elements of the sequences used for the averaging process. However, it does not perform any matching on the timestamps associated to those elements, which is needed in our model. To address this issue, we introduce a new value called *velocity factor*, giving a partial information on the relative speeds of how the average and training gestures are performed, and using the fact that multi-sensor sequences and template gestures can be seen as vectors of multi-sensor coordinates (vectors of $\mathbb{R}^{\mathbb{A}}$). Given a set of multi-sensor time sequences related to gesture g $S_g = \{\mathbf{S}_{g,1}, \mathbf{S}_{g,2}, ..., \mathbf{S}_{g,M}\}$ and their associated template obtained with DBA $\mathbf{T}_g$, of size $l_g \times A$, we define $\mathbf{M}_{\text{assoc}} \in \mathbb{M}_{l_g \times M}(\mathbb{R})$ as the *association matrix between* $\mathbf{T}_g$ *and* S_g by:

$$\begin{gathered}\text{for all } 1 \leqslant i \leqslant l_g \text{ and } 1 \leqslant j \leqslant M,\\ \mathbf{M}_{\text{assoc}}(i,j) = \text{ number of multi-sensor coordinates}\\ \text{of sequence } \mathbf{S}_{g,j} \text{ associated to the } i^{th} \text{ multi-sensor}\\ \text{element of } \mathbf{T}_g \text{ by DTW}\end{gathered}$$

and the *velocity factor vector* $\mathbf{v}_g \in \mathbb{R}^{l_g}$ associated to $\mathbf{T}_g$ as:

$$\text{for } 1 \leqslant i \leqslant l_g,\ \mathbf{v}_g(i) = \frac{1}{N}\sum_{j=1}^{M} \mathbf{M}_{\text{assoc}}(i,j) \qquad (1)$$

The i^{th} element of the vector $\mathbf{v}_g$ is the average number of coordinates of the sequences of S_g associated to the i^{th} element of its average sequence $\mathbf{T}_g$. Intuitively, each element of $\mathbf{v}_g$ indicates how fast the average time sequence $\mathbf{T}_g$ is compared to the sequences of S_g. In particular, having $\mathbf{v}_g(i) < 1$ indicates that the sequence $\mathbf{T}_g$ is slower on average than the ones in S_g, at time i, whereas $\mathbf{v}_g(i) > 1$ denotes the contrary.

Denoting $\Delta t \in \mathbb{R}^*_+$ the time gap between two consecutive sampling moments (in seconds), we are then able to define the timestamps $(t^g_k)_{1 \leqslant k \leqslant l_g}$, associated to the average sequence $\mathbf{T}_g$ by:

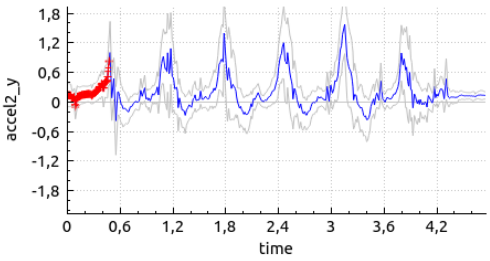

Figure 2: Example of one template (in blue) obtained for one gesture and one sensor, and its standard deviation at all timestamps (in gray). In red : the "temporal position" of the particles within the template gesture. The abscissa of each particle is its estimated timestamp.

$$t^g_k = \begin{cases} 0 & \text{if } k = 0 \\ t^g_{k-1} + \mathbf{v}_g(k)\,\Delta t & \forall 1 \leqslant k \leqslant l_g \end{cases} \qquad (2)$$

Note that for convenience purposes, we define an indexation of template gestures by timestamps (instead of regular indices), by $\mathbf{T}_g(t,a) = \mathbf{T}_g(k,a)$, with $1 \leqslant k \leqslant l_g$ so that $t = t^g_k$, for $t \in \mathbb{R}_+$ timestamp computed at the previous step, $g \in \{1,...,G\}$ and $a \in \{1,...,A\}$

Finally, given one gesture g, we keep in memory for all sensors $a \in \{1,...,A\}$ the standard deviation of sensor values at all timestamps of the template gesture $\mathbf{T}_g$, computed -for each timestamp- on the set of values related to sensor a from the sequences of the training set, linked by the DBA algorithm to the coordinate of $\mathbf{T}_g$ associated to this timestamp. We denote the $G \times A$ vectors that we obtain $\sigma_{g,a}$ $(\in \mathbb{R}^{l_g})$:

$$\forall g \in \{1,...,G\},\ \forall a \in \{1,...,A\},\ \forall 1 \leqslant k \leqslant l_g,$$
$$\sigma_{g,a}(k) = \sqrt{\frac{1}{|S_k|}\sum_{\mathbf{c} \in S_k} (\mathbf{c}(a) - \mathbf{T}_g(k,a))^2} \qquad (3)$$

where S_k is the set of multi-sensor coordinates (elements of $\mathbb{R}^A$) of the sequences of the training set, associated with the k^{th} multi-sensor coordinate of $\mathbf{T}_g$, and $|S_k|$ is the cardinality of S_k.

3.3 Transition Model

Once the template gestures $(\mathbf{T}_g)_{1 \leqslant g \leqslant G}$ and associated velocity vectors $(\mathbf{v}_g)_{1 \leqslant g \leqslant G}$ are obtained after the training phase, we can define the model of our particle filter for temporal tracking. Given an input gesture performed in real-time providing real-time sensor data, we propose to classify it by trying to estimate its "temporal position" within each template gesture, or in other words, to find to which of the G templates the input gesture is the closest, and to which part of it it corresponds to.

Given a set of $N \in \mathbb{N}^*$ particles, we define $\mathbf{x}_k^i$ the i^{th} particle ($i \in \{1,2,...,N\}$) at sampling time $k \in \mathbb{N}$ by:

$$\mathbf{x}_k^i = \begin{pmatrix} g^i \\ t_k^i \end{pmatrix} \tag{4}$$

where $g^i \in \{1,2,...,G\}$ is the index linking the particle to the gesture g^i, and $t_k^i \in \mathbb{R}$, called *timestamp*, is an estimation of the temporal position of the particle within the template gesture $\mathbf{T}_g$ (in seconds)

In our definition of the model, all particles, from birth to death, are assigned with one of the G different gestures, i.e. $\forall k \in \mathbb{N}, \forall i \in \{1,2,...,N\}, \mathbf{x}_k^i(1) = \mathbf{x}_0^i(1)$. The timestamp $\mathbf{x}_k^i(2)$ evolves according to the following relation:

$$\forall k \in \mathbb{N}^*,\ t_k^i = t_{k-1}^i + \alpha_k^i\, \Delta t \tag{5}$$

with Δt duration (in seconds) between two consecutive sampling steps, and $\alpha_k^i \in \mathbb{R}$ value drawn following a stationary process, given by the Gaussian distribution $\mathcal{N}(\mu, \sigma)$ of parameters μ and σ set manually.

3.4 Observation Model

The observation model of a particle filter is used to compute the weight associated to each particle, at all sampling times. This weight should be proportional to the likelihood of the particle state regarding the sensor observations obtained at the current sampling time. Given $\mathbf{e}_k = (e_{k,1}, e_{k,2}, ..., e_{k,A})$ the A sensor observations obtained in real time from the input gesture at sampling step $k \in \mathbb{N}$, and a particle $\mathbf{x}_k^i = (g^i, t_k^i)$, we therefore define the model as follows:

$$w_k^i = \prod_{a=1}^{A} \alpha_a(e_{k,a}) \tag{6}$$

$$\text{with} \quad \alpha_a(e_{k,a}) = \frac{1}{\sqrt{2\pi}\,\sigma_a(t_k^i)}\, e^{-\frac{(e_{k,a} - \mathbf{T}_{g^i}(t_k^i,a))^2}{\sigma_a(t_k^i)^2}} \tag{7}$$

$$\text{and} \quad \sigma_a(t) = \frac{1}{G}\sum_{g=1}^{G} \sigma_{g,a}(t) \tag{8}$$

Considering that the timestamp t_k^i of one particle $\mathbf{x}_k^i = (g^i, t_k^i)$ can take any value in R_+ and might not be one of the values computed using the velocity vector during the training phase, the sensor value $\mathbf{T}_{g^i}(t_k^i, a)$ and sensor standard deviation $\sigma_{g^i,a}(t_k^i)$ might not be defined. We therefore define those values by computing a simple linear interpolation using the two closest timestamps stored in memory after the training, and their associated sensor values: given t_1, t_2 timestamps in memory so that $t_1 < t_k^i < t_2$:

$$\mathbf{T}_{g^i}(t_k^i, a) = \mathbf{T}_{g^i}(t_1, a) + \frac{t_k^i - t_1}{t_2 - t_1}\left(\mathbf{T}_{g^i}(t_2, a) - \mathbf{T}_{g^i}(t_1, a)\right) \tag{9}$$

$$\sigma_{g^i,a}(t_k^i, a) = \sigma_{g^i,a}(t_1, a) + \frac{t_k^i - t_1}{t_2 - t_1}\left(\sigma_{g^i,a}(t_2, a) - \sigma_{g^i,a}(t_1, a)\right) \tag{10}$$

One important point to understand the way we built our model: we could observe very different orders of magnitude in the values of $\sigma_{g,a}(t)$ depending on the gesture $g \in \{1,...,G\}$ considered in our dataset (for all sensor a and timestamp t). The use of an averaged standard deviation across all gestures **(9)**, instead of the standard deviation of sensor values related to the gesture $\mathbf{x}_k^i(1) = g^i$ for the Gaussian distribution, prevents our model from giving more influence to gestures with low standard deviations on the computation of w_k.

All weights obtained at sampling step $k \in \mathbb{N}$ are then normalized in $[0,1]$, as follows:

$$\forall i \in \{1,2,...,N\},\ w_{\text{norm},k}^i = \frac{w_k^i}{\sum_{j=1}^{N} w_k^j} \tag{11}$$

3.5 Resampling

Particle filtering approaches are subject to degeneracy problems, highlighted in (Doucet et al., 2000): the particles of the system can evolve in a way that drives them further from sensor observation acquired in real time, therefore making the values of the associated weights drop to a point where no particle can be considered as significant anymore. In order to address this issue, particles are resampled when this phenomenon occurs, i.e. the N particles of the system are re-initialized with equal weights, after being drawn with a probability proportional to their former weights.

In order to determine when to resample, we define the *effective number of particles* $N_{\text{eff},k}$ at time $k \in \mathbb{N}$, introduced in (Liu and Chen, 1998), by:

$$N_{\text{eff},k} = \frac{1}{\sum_{i=1}^{N} (w_{\text{norm},k}^i)^2} \tag{12}$$

The resampling occurs when N_{eff} falls below a threshold $N_{\text{threshold}} \in \mathbb{R} : N_{\text{eff}} \leqslant N_{\text{threshold}}$, whose value is empirically determined.

One particular case where this system degeneracy can be observed with the model we defined occurs when the particles keep evolving, whereas the performance of the input gesture has not actually begun yet, or when the user started to perform a gesture some time after the particles started to evolve. In those situations, resampling cannot address this issue alone, considering that all current particles provide wrong estimations of the timestamp. For this reason, we perform an additional step, called *interspersing*, which consists of adding $N_{\text{intersperse}} \in \mathbb{N}^*$ new particles, spread across all G gestures, with timestamps taken randomly between 0 and the maximum duration of each gesture (i.e. timestamp associated with the last coordinate of the corresponding template), before the actual resampling. Associated weights to these new particles are computed using **(6)** and **(11)**. Resampling on the $N + N_{\text{intersperse}}$ is then performed to draw N new particles.

3.6 Gesture Classification

The classification of an input gesture performed in real time is done at any sampling step $k \in \mathbb{N}$ by computing its probability p_k^g to belong to the class of gesture $g \in \{1,...,G\}$, which can be obtained by summing the weights of all particles associated to this gesture:

$$\forall k \in \mathbb{N},\ \forall g \in \{1,...,G\},\ p_k^g = \sum_{\substack{i \in \{1,...,N\} \\ \text{with } \mathbf{x}_k^i(1)=g}} w_{\text{norm},k}^i \tag{13}$$

The class of the input gesture at time $k \in \mathbb{N}$, $g_k^{\text{estimated}} \in \{1,...,G\}$, is determined by checking the highest probability of belonging to one class:

$$\forall k \in \mathbb{N},\ g_k^{\text{estimated}} = \underset{g \in \{1,...,G\}}{\text{argmax}}\ (p_k^g) \tag{14}$$

In order to consider only classification results which would be meaningful (e.g. by not starting to classify until the input gesture is actually performed), we also decide to compute $g_k^{\text{estimated}}$ for $k \in \mathbb{N}$ only if the estimated probability of the most likely gesture is "high enough", i.e. $p_k^{g_k^{\text{estimated}}} \geqslant p_{\text{threshold}}$ with $p_{\text{threshold}} \in [0,1]$.

To summarize, the following steps are performed to achieve gesture recognition in our system:

At each sampling step $k \geqslant 0$:

1. At $k = 0$, assign N particles to all gestures ($\frac{N}{G}$ per gesture)
2. Move particles following the transition model **(5)**
3. Acquire real-time data $\mathbf{e}_k$ of the input gesture, and compute the weights w_k using **(6)**
4. Normalize weights using **(11)**
5. Predict the class using **(14)** if the certainty on the result is high enough
6. If $N_{\text{eff},k} \leqslant N_{\text{threshold}}$, intersperse new particles and resample
7. $k \leftarrow k+1$, and loop to step 2

4 EXPERIMENTAL RESULTS

4.1 MHAD Dataset

The performances of our system have been tested on the Multimodal Human Action Database (MHAD) from Berkeley University (Ofli et al., 2013), which features a wide range of different sensor data acquisitions (RGB and Kinect cameras, 3D accelerometers and audio sensors), acquired from 12 different subjects (7 male, 5 female), who were asked to perform $G = 11$ different actions basic enough to be considered as gestures (1 : jumping in place, 2 : jumping packs, 3 : bending, 4 : punching, 5 : waving with two hands, 6 : waving with the right hand, 7 : clapping hands, 8 : throwing a ball with the right hand, 9 : sitting down then standing up, 10 : sitting down, 12 : standing up), $R = 5$ times each.

Considering that we are limiting the scope of our project to the problem of real time recognition of gestures, using continuous time series data, we decided to use only the data from the 3D accelerometers (2 on the subjects' wrists, 2 on ankles and 2 on hips), resulting in $A = 6 \times 3 = 18$ different sensors. The sampling frequency of those sensors is $f = 30\,Hz$.

4.2 Results

After some manual tuning of our model, we set the parameters of our transition model to $\mu = 1$, $\sigma = 0.2$, the total number of particles to $N = 120$, the number of interspersed particles after resampling to $N_i = 60$, the resampling threshold to $N_{\text{threshold}} = 0.2N = 30$, and the probability threshold for classifying to $p_{\text{threshold}} = 0.7$. We achieve an average accuracy of 85.30% across all gestures, over the 12 different folds of the Leave-One-Out cross validation, with a maximum accuracy of 91.90%, a minimum of 72.73% and a standard-deviation of 7.29%.

After a look at the averaged confusion matrix over the 12 folds of the cross-validation (c.f. figure 3), it can be noted that most of the misclassifications

Input \ Estimation	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{11}
g_1	**100**	0	0	0	0	0	0	0	0	0	0
g_2	0	**100**	0	0	0	0	0	0	0	0	0
g_3	0	0	**100**	0	0	0	0	0	0	0	0
g_4	0	0	6.7	**85**	0	0	6.7	0	0	1.6	0
g_5	0	0	1.7	3.3	**85**	0	10	0	0	0	0
g_6	0	0	0	0	0	**100**	0	0	0	0	0
g_7	0	0	0	0	1.7	0	**98.3**	0	0	0	0
g_8	0	0	0	0	0	11.7	0	**86.7**	1.6	0	0
g_9	0	0	0	0	0	0	0	0	**71.7**	28.3	0
g_{10}	0	0	1.7	0	0	0	0	1.7	35	**61.6**	0
g_{11}	0	0	0	1.7	0	0	0	0	43.3	5	**50**

Figure 3: Confusion matrix for the classification of the 11 gestures of the MHAD dataset, using our particle filtering approach. The results (in percentages) are the accuracies averaged over the 12 subjects of the dataset.

concern some gestures which share some common submovements, and therefore are similar in terms of sensor values and variations. Our method is especially bad at making a distinction between g_9, g_{10} and g_{11} (respectively sitting down then standing up, sitting down, standing up). On the opposite, our model achieves very high classification rates for some of the other gestures of the dataset, regardless of the subject tested (g_1, g_2, g_3, g_6 and g_7).

4.3 Comparison to the State-of-the-Art

In order to compare our method's performances to the performances of the state-of-the-art ones, we also tested the following methods for real time gesture classification:

- **Soft-Margin SVM (C-SVM):** the sensor data from all A sensors, comprised within a sliding time window of $T \in \mathbb{N}^*$ sampling times, is directly used as input features of the model (dimension of $T \times A$). The sliding step is set to 1. The kernel used is the RBF.
- **Multi-Layer Perceptron (MLP):** classic feed-forward neural network with one input, output and hidden layers, trained using mini-batch gradient descent. Similarly to the previous C-SVM solution, the sensor data contained within a time window $T \in \mathbb{N}^*$ is taken as input of the network (vectors of size $T \times A$), with the same sliding step of 1.
- **Convolution Neural Network (CNN):** deep neural network adapted to time series processing, with two pairs of convolutional + pooling layers, and a MLP performing classification on the features computed at the end of the CNN. The sensor data comprised within a time window $T \in \mathbb{N}^*$ (with a sliding window of 1) is taken as input (therefore making input "images" of dimension $T \times A$). The training is performed using mini-batch gradient descent.

All models also have been tested using Leave-One-Out cross-validation on the 12 subjects of the dataset. The hyper-parameters of each model have been determined by grid-search for the C-SVM approach, and trial and error experiments for both neural networks approaches (MLP and CNN). The different models have been coded using Python, and the LibSVM and Theano libraries for C-SVM and neural networks respectively. The following table summarizes the best results obtained for each of them :

Accuracy (%)	**Average**	**Max**	**Min**	**Std**
Particle filters	85.30	91.83	72.73	7.29
C-SVM	75.52	88.95	14.58	19.79
MLP	79.22	90.57	61.93	7.41
CNN	79.56	90.84	61.94	7.99

Figure 4: Comparative accuracies of the different classification models tested, using Leave-One-Out cross validation.

Our method achieves a better accuracy averaged over the 12 folds of the MHAD dataset than both neural network approaches and C-SVM without feature extraction. It also has the advantage of not having the need to set an input time window hyper-parameter, which could be constrained by the requirements on the performances of an actual implementation in real-time of the system, or prevent the processing of short input sequences with length lower than the time window. The lower average accuracy obtained with Neural Network approaches and C-SVM can be explained by the low performances obtained by those models for some of the subjects of the dataset. We suppose that while the poor performances of C-SVM can be attributed to the fact that the raw input data comprised within a time window is not relevant enough to be used as input features of the model, the accelerometer training data used from the MHAD set might not be large and diverse enough to obtain optimal results after the training of both Neural Network models.

5 CONCLUSION AND FUTURE WORKS

In this paper, we presented a method for real time gesture recognition using 1D temporal sensor data. In a preliminary phase (training), the sensor readings from the training set are averaged to obtain one *template gesture*. The process is repeated for all of the different gestures of the set. In a secondary phase (recognition), a particle filter model is defined for the gesture classification. The sensor data of an input gesture performed by a user is acquired in real time, and compared to the gesture templates obtained after the training phase to determine an estimation of the "temporal position" of the input gesture. The classification

of the latter is performed by determining the closest template gesture.

Our method has been tested with the accelerometer data of the MHAD dataset (Ofli et al., 2013) using the Leave-One-Out cross validation on the 12 subjects of the set, and compared to other state-of-the-art classification approaches (SVM, MLP and CNN). It achieves a 85.30% average accuracy, and outperforms other real-time gesture classification models, without having any need -unlike the other approaches- to set a time-window parameter which could be constrained by the constraints of an actual real-time implementation, or not be suitable for the recognition of very short gestures. The results obtained show that most misclassifications of our method concern gestures which share common submovements (e.g. sitting-down, standing-up). The high accuracies obtained for the other gestures of the dataset, no matter the subject performing them, or the way they are performed, indicate that our method is robust to variations in execution of the gestures.

There are still some points on which our method could be improved or developed further though. Preliminary experiments carried out on a dataset of hand gestures using accelerometers (uWave, (Liu et al., 2009)) seem to confirm the fact that our method struggles to differenciate gestures similar in terms of sensor values and variations, as it provides accuracies much lower than other tested state-of-the-art approaches (SVM and Neural Networks) there. Future works will focus on finding axis of improvements to address this issue: in particular finding more relevant state features to be tracked by the particle filter, testing the classification problem with additional 1D temporal sensor data, and analyze the possibility to attribute an importance weight to each sensor in our observation model to favour the recognition of some gestures.

ACKNOWLEDGEMENTS

Research and development activities leading to this article have been supported by the German Research Foundation (DFG) as part of the research training group GRK 1564 ”Imaging New Modalities”, and the German Federal Ministry of Education and Research (BMBF) within the project ELISE (grant number: 16SV7512, www.elise-lernen.de).

REFERENCES

Akl, A., Feng, C., and Valaee, S. (2011). A novel accelerometer-based gesture recognition system. *IEEE Transactions on Signal Processing*, 59(12):6197–6205.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

Black, M. J. and Jepson, A. D. (1998). A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *Computer Vision — ECCV'98: 5th European Conference on Computer Vision Freiburg, Germany, June, 2–6, 1998 Proceedings, Volume I"*, pages 909–924.

Caramiaux, B., Montecchio, N., Tanaka, A., and Bevilacqua, F. (2014). Adaptive gesture recognition with variation estimation for interactive systems. *ACM Trans. Interact. Intell. Syst.*, 4(4):18:1–18:34.

Chudgar, H. S., Mukherjee, S., and Sharma, K. (2014). S control: Accelerometer-based gesture recognition for media control. In *Advances in Electronics, Computers and Communications (ICAECC), 2014 International Conference on*, pages 1–6.

Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. volume 140, pages 107–113.

Isard, M. and Blake, A. (1998). Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28.

Liu, J., Zhong, L., Wickramasuriya, J., and Vasudevan, V. (2009). uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657 – 675. PerCom 2009.

Liu, J. S. and Chen, R. (1998). Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044.

Mitra, S. and Acharya, T. (2007). Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324.

Ofli, F., Chaudhry, R., Kurillo, G., Vidal, R., and Bajcsy, R. (2013). Berkeley mhad: A comprehensive multimodal human action database. In *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, pages 53–60.

Petitjean, F. and Gançarski, P. (2012). Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment. *Theoretical Computer Science*, 414(1):76–91.

Shoaib, M., Bosch, S., Incel, O., H.Scholten, and Havinga, P. (2015). A survey of online activity recognition using mobile phones. *Sensors*, 15:2059–2085.