Unsupervised Gesture Segmentation by Motion Detection of a Real-Time Data Stream

Miguel A. Simão, Pedro Neto, and Olivier Gibaru

***Abstract*—Continuous and real-time gesture spotting is a key factor in the development of novel human-machine interaction (HMI) modalities. Gesture recognition can be greatly improved with previous reliable segmentation. This paper introduces a new unsupervised threshold-based hand/arm gesture segmentation method to accurately divide continuous data streams into dynamic and static segments from unsegmented and unbounded input data. This segmentation may reduce the number of wrongly classified gestures in real world conditions. The proposed approach identifies sudden inversions of movement direction which are a cause of oversegmentation (excessive segmentation). This is achieved by the analysis of velocities and accelerations numerically derived from positional data. A genetic algorithm is used to compute feasible thresholds from calibration data. Experimental tests with three different subjects demonstrated an average oversegmentation error of 2.70% in a benchmark for motion segmentation with a feasible sliding window size.**

***Index Terms*—segmentation, unsupervised, motion, gestures, human-machine interaction, robotics.**

I. Introduction

FLEXIBLE industrial machines in general and robots in particular are traditionally instructed either by text-based programming or by direct control, using a teach pendant. The ability to interact with a machine in a natural and intuitive way, e.g. using hand/arm gestures, has brought important advances to modern industry. The paradigm for robot usage has changed in the last few years, from an idea in which robots work with complete autonomy to a scenario in which robots cognitively collaborate with human beings. This brings together the best of each partner, robot and human, by combining the coordination and cognitive capabilities of humans with the robots' accuracy and ability to perform monotonous tasks. This will allow a greater presence of robots in our society, with consequent positive impact on life standards. The problem is that the existing interaction modalities are neither intuitive nor reliable. Instructing and programming an industrial robot by the traditional teaching method is a tedious and time-consuming task that requires technical expertise. The increasing demand by industry for robot-based solutions makes the need for intuitive human-machine interaction (HMI) more visible, especially in small and medium sized enterprises (SMEs).

Miguel Simão and Pedro Neto are with the Department of Mechanical Engineering, University of Coimbra, 3030-788 Coimbra, Portugal (e-mail: miguel.simao@uc.pt; pedro.neto@dem.uc.pt). Olivier Gibaru is with the Ecole Nationale Supérieure d'Arts et Métiers, ParisTech, Lille, 59800, France (e-mail: olivier.gibaru@ensam.eu).

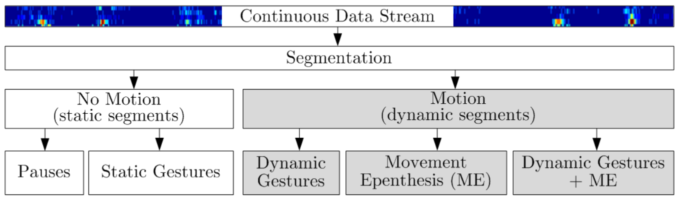

Fig. 1. The role of segmentation by motion in gesture recognition.

Multimodal interfaces combining gestures, speech and tactile based-actions are expected to be in a near future the standard for HMI. Nonverbal communication cues in the form of gestures are considered to be an effective way to approach intuitive HMI. For instance, a person can point to indicate a position to a robot, use a dynamic gesture to instruct a robot to move and a static gesture to stop the robot [1]. In this scenario the user has little or nothing to learn about the interface, focusing on the task and not on the interaction modality.

Temporal gesture segmentation from a real-time data stream is the problem of identifying data segments that are more likely to contain meaningful interactions. This may increase the reliability of subsequent pattern recognition. Most of the classification algorithms only present reliable results if the input data roughly represent a specific gesture on its database (previously trained – supervised method). A major challenge in continuous gesture recognition has to do with the fact that there are movement segments between gestures that have no meaning. Such inter-gesture transition periods are known as movement epenthesis (ME). Most studies treat ME as a classification problem [2], [3].

An effective segmentation is achieved with no previous knowledge of gesture data, no training and no information about the next gesture in the sequence. In general, there are two major difficulties in the segmentation process of a real-time stream of data:

1) Stream unboundedness: no information about when a gesture starts and ends in a continuous sequence;
2) Spatio-temporal variability: a gesture may vary in shape, duration and trajectory, even when it is performed by the same person.

Fig. 1 highlights the role of segmentation by motion as a preprocessing step before the classification of static/dynamic gestures and ME. Static segmentswill serve as input only to the static gesture pattern classifier. On the contrary, when there is motion, the data are not used to feed the static gesture classifier. In the presence of dynamic segments, the recorded data can have different meanings, i.e., the segment can be classified either as a dynamic gesture or as ME. This is a classification problem in which the dynamic gestures and ME can be correctly recognized if they were previously trained. ME classification based on its training is non-natural so that most authors exclude ME from trained gesture patterns.

A. Problem Specification and Challenges

A major problem in gesture-based HMI is related with the reliable recognition of gestures continuously from real-time streams. Continuous gesture recognition is the natural way used by humans to communicate, in which communicative gestures (with an explicit meaning) appear intermittently with pauses and ME, without a specific order.

It is difficult to accurately segment continuous data streams to feed the classifiers. It depends on several factors: (1) interaction technologies, (2) classification method (supervised or unsupervised), (3) if gestures are static, dynamic or both, (4) if ME was previously trained or not, among other factors. Another problem is related with the difficulty to eliminate the appearance of false positives and false negatives (leading to oversegmentation).

In the context of gesture segmentation, it can be stated that false negatives are more costly than false positives since they divide the data representing a dynamic gesture into two sections, completely corrupting the meaning of that gesture. False positives are more easily accommodated by the classifier, which just reports that the pattern is not a trained gesture. Several challenges in gesture segmentation can be pointed out:

1) Creating an unsupervised segmentation technique that is robust enough to accurately divide a data stream into dynamic and static segments without false negatives and false positives, avoiding over/undersegmentation;
2) Avoiding false negatives in the segmentation of dynamic gestures by anticipating inversions of movement;
3) Achieving real-time performance and being able to segment gestures in continuous streams;
4) Being user independent.

B. Related Work

Off-line analysis of gesture segmentation by motion detection has been applied with relative success, for example by computing the variance of motion data and applying a threshold that defines if motion exists or not. Many existing segmentation techniques use the backward spotting scheme to first detect the end point of a gesture and then traces back to the starting point. The problem is that in this methodology the real-time is lost, making it unsuitable for continuous gesture recognition [4]. This problem can be solved by implementing a forward spotting scheme for simultaneous gesture segmentation and recognition in which the start and end points of a gesture are determined by zero crossing from negative to positive (and vice versa) of a competitive differential observation probability [4]. This is a supervised automatic threshold method based on the comparison of the probability of a given frame being a gesture or non-gesture.

A reference study in the field reports the application of an adaptive threshold [5]. Such adaptive threshold is based on the addition of an additional label to the Conditional Random Field (CRF) model to overcome the weakness of the fixed threshold method. Initial training is necessary for the CRF.

Interesting studies in the field propose a method for spotting gestures in continuous data by using Hidden Markov Models (HMMs) [6]. They apply a threshold model that calculates the likelihood threshold of an input pattern. The start and end points of a gesture are defined by comparing the threshold model with predefined gesture models.

A fuzzy machine vision-based framework for humans' behavior recognition that relies on the analysis of feature cues reporting the human silhouette was studied in [7]. Meaningful movements can be recognized while concurrently separating unintentional movements from a given image sequence [8]. The importance of the selection and adaptation of the window data length and its dependency on the complexity, duration and granularity of the human activities to recognize is demonstrated in [9]. An analysis of motion segments using Principal Component Analysis (PCAs) to represent hand motion is in [10]. This method allows to reduce the data dimensionality but according to our experience and real-time requirements this can be a computationally expensive solution, depending on the size of input data, and in which important information for the segmentation process may be lost. An automated process of segmenting gesture trajectories based on a simple set of threshold values is proposed in [11]. The gesture segmentation process is also highlighted in a study on vision-based action recognition for the human entire body considering a large vocabulary of gestures [12]. A different approach is to study the perception of human postures and gestures recurring to a marker-less vision-based solution for upper body tracking with multiple cameras [13]. An unsegmented/unbounded vision-based continuous gesture recognition system is presented in [14]. For a library of 10 body-and-hand gestures, the achieved recognition accuracy was 94% for isolated gestures and 88% for continuous gestures. This clearly shows the importance of segmentation for continuous gesture recognition.

An approach to gesture spotting in a continuous data stream from body-worn inertial sensors is presented in [15]. A recent study reports the separation of acceleration and surface electromyography signals as a mean to segment gesture data [16].

Regarding the role of ME in gesture segmentation, some authors try to solve this problem by developing temporal gesture models which address the problem of ME detection without the need for explicit epenthesis training [3], while other authors train ME [1], [5], [6]. In both cases ME is analyzed in the context of a classification problem requiring previous training. Previous studies demonstrated that approaches that explicitly model ME yield better results than those ignoring ME [1]. However, modeling ME is difficult and its complexity increases exponentially with the number of distinct gestures.

Complex human activity recognition in sensor rich environments has been studied in the last few years [17], [18]. A multimodal segmentation method that merges information from inertial sensors (IMUs), muscle activity and location (RFID) is presented in [17]. A string-matching-based segmentation and classification method is also presented to recognize activity using wearable devices with limited computational power [17].

Online gesture recognition for wearable computing purposes based on crowdsourced annotations is in [18]. In this study segmentation and warping longest common subsequence algorithm (LCSS) are applied as template matching methods. Wearable accelerometers have been successfully applied for human activity recognition. A framework for the recognition of motion primitives relying on Gaussian mixture modeling (GMM) and Gaussian mixture regression (GMR) is presented in [19]. A recognition procedure based on dynamic time warping (DTW) and Mahalanobis distance is proposed to ensure run-time classification.

Analyzing existing studies allows us to conclude that most studies approach the segmentation problem together with classification, requiring previous training – supervised methods. This paper approaches segmentation by motion in an unsupervised and computationally inexpensive fashion.

C. Proposed Approach and Overview

Real-time segmentation relies on the comparison of the current state (frame) $\mathbf{f}_i$ with the previous states, $\{\mathbf{f}_{i-1}, \dots, \mathbf{f}_{i-n}\}$. A fixed threshold is a very limitative solution for motion segmentation. Existing studies present excellent results using automatic threshold methods or adaptive thresholds for gesture segmentation [4], [5]. However, such approaches are supervised, requiring a significant amount of training data from specific users.

This paper proposes a novel method to segment a continuous data stream into dynamic and static segments in an unsupervised fashion, i.e. without previous training or knowledge of gestures and the sequence, unsegmented and unbounded. We propose establishing a feasible (it can be optimal or not) single threshold for each motion feature using a genetic algorithm (GA) – because the performance function is non linear and non smooth – fed by a set of calibration data. Gesture patterns with sudden inversions of movement direction are analyzed recurring to the analysis of velocities and accelerations numerically derived from positional data. The proposed method deals with hand/arm gesture motion patterns varying in scale, rate of occurrence and different kinematic constraints. A sliding window addresses the problem of spatio-temporal variability.

The proposed system was evaluated by conducting experiments using a set of continuous dynamic and static gestures (Section III). Experimental tests were carried out using wearable/body-worn sensing (a data glove and a magnetic tracking device) and demonstrated that:

1) Motion is always detected for any gesture in a continuous sequence of unsegmented and unbounded data;
2) It is an unsupervised method, no prior training;
3) A quantitative analysis with samples from three different subjects indicates an oversegmentation error of 2.70% with a feasible sliding window size. For an optimal sliding window size (subject A) the error is 0%;
4) Segmented data present in most cases a shift-right and extend behavior in relation to the ground truth;
5) The GA to search for feasible thresholds demonstrated reliability for various subjects;
6) Velocity and acceleration features have to be combined to deal with sudden inversions of movement direction;
7) The proposed fine segmentation method reduces the noise in data for subsequent classification;
8) The system accepts any sequential positional data as input from different sensors: inertial, vision, etc.

II. Gesture Segmentation

A. Sliding Window Threshold Decision Method

The absence of movement can be defined by the lack of change in every degree of freedom (DOF) of a given system. At rest, natural human body shaking generates noisy DOF derivatives. To separate their noise from an actual static stance, there was a need to set a threshold for each of the features. We consider that there is motion if there are motion features above the defined thresholds. Section II-B shows how the feasible thresholds are achieved. The threshold is a vector, $\mathbf{t}_0$, with a length equal to the number of motion features chosen, n_t. The features obtained from a frame are represented by the vector $\mathbf{t}$. The sliding window $\mathbf{T}$ is composed of w consecutive frames of $\mathbf{t}$. At an instant i, the real-time sliding window $\mathbf{T}(i)$ is given by:

$$\mathbf{T}(i) = \begin{bmatrix} \mathbf{t}(i-w+1) & \cdots & \mathbf{t}(i-1) & \mathbf{t}(i) \end{bmatrix} \tag{1}$$

At each instant i, the w sized window slides forward one frame and $\mathbf{T}(i)$ is updated and evaluated. A static frame is only acknowledged as such if none of the motion features exceed the threshold within the sliding window. This way, we guarantee that a motion start is acknowledged with minimal delay (real-time). On the other hand, this also causes a fixed delay on the detection of a gesture end, equal to the size of the window w.

The proposed method to achieve the motion function $m(i)$ relies in the computation of the infinite norm of a vector $\mathbf{a}$ that contains feature-wise binary motion functions:

$$m(i) = \begin{cases} 1, & \text{if } \|\mathbf{a}\|_\infty \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where vector $\mathbf{a}$, for each instant of time i, is calculated by comparing the sliding window with the threshold vector:

$$\mathbf{a}_m = (\max_n |\mathbf{T}_{mn}| \geq k_s \cdot \mathbf{t}_{0_m}), \quad \begin{matrix} m = 1, \dots, n_t, \\ n = 1, \dots, w \end{matrix} \tag{3}$$

in which k_s represents a user-defined threshold sensitivity factor. A possible result for a single feature $\mathbf{t}_m$ is shown in Fig. 2. A motion segment starts as soon the feature rises above the threshold . On the other hand, even after the feature drops below the threshold, the segment is not finished until a full window of frames is below the threshold.

B. Threshold Definition

Calibrating the threshold $\mathbf{t}_0$ in an unsupervised fashion is not a straightforward task because it depends on several factors, namely the input data/devices, selected features, sliding window size and the device wearer himself. If the parameters are set too low, it can become oversensitive and generate too

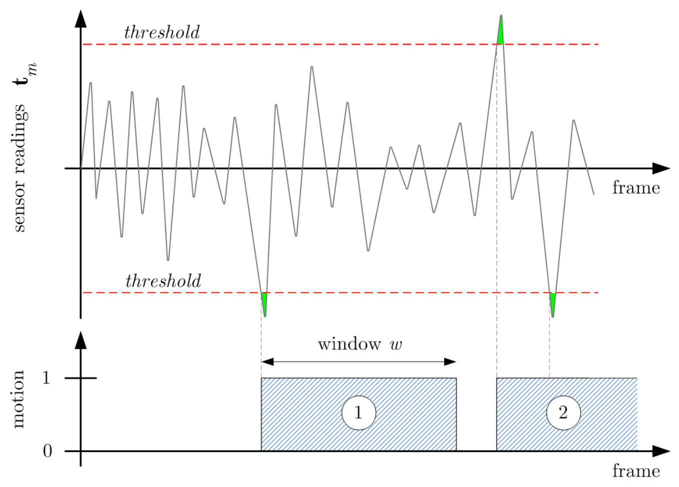

Fig. 2. Sliding window threshold decision method for motion detection. Two motion segments are represented.

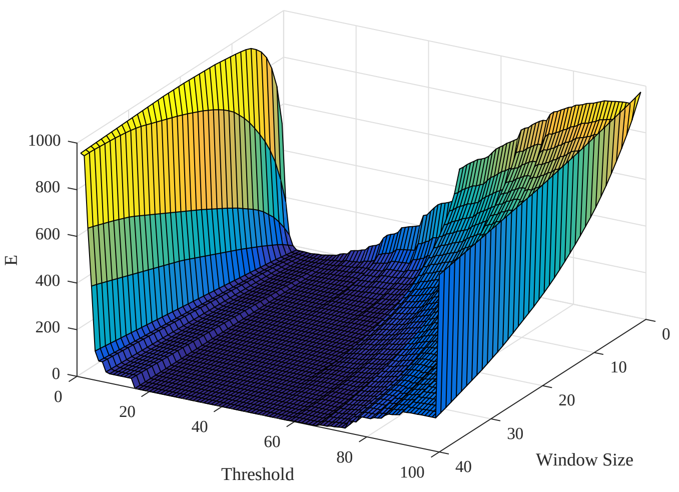

Fig. 3. Error dependence on threshold value and sliding window size.

many false positives, declaring motion when there is noise. If they are set too high, it will not be capable of detecting slow movements, generating false negatives and oversplitting the stream. Thus, a balance must be found.

The authors recommend obtaining two distinct sets of data (which we call calibration data) with equal length/time t_s:

1) A static sample $\mathbf{C}_S$, with dimension $n_t \times t_s$, recorded with the user at rest. The target motion function is equal to 0, with dimension $1 \times (t_s - w + 1)$;
2) A motion sample $\mathbf{C}_M$, with dimension $n_t \times t_s$, recorded with the user performing slow movements that activate the selected motion features to establish minimum thresholds. These movements are done randomly and should trigger every motion feature. The target motion function is constant and equal to 1, with dimension $1 \times (t_s - w + 1)$.

The matrices $\mathbf{C}_S$ and $\mathbf{C}_M$ establish what we call the ground truth for the process of calibrating the features' thresholds $\mathbf{t}_0$. This is not considered training as it is performed by the user in a short span of time and the samples do not need to have meaningful gestures.

The calibration process seeks to minimize the error of the sliding window applied to the calibration data. The window size is kept fixed and the variables are the motion features' thresholds. The error E is the objective function to minimize (4). It has two terms, one corresponding to error for the static sample, e_S, and the error for the motion sample, e_M.

$$E = \sum_{i=w}^{t_s} e_S(i) + e_M(i) \tag{4}$$

where e_S and e_M are binary functions computed from the samples $\mathbf{C}_S$ and $\mathbf{C}_M$ and m is a binary motion function:

$$e_S(i) = \begin{cases} 0, & \text{if } m\left(C_S(i)\right) = 0 \\ 1, & \text{if } m\left(C_S(i)\right) = 1 \end{cases} \tag{5a}$$

$$e_M(i) = \begin{cases} 0, & \text{if } m\left(C_M(i)\right) = 1 \\ 1, & \text{if } m\left(C_M(i)\right) = 0 \end{cases} \tag{5b}$$

It is recommended to calibrate each threshold individually, implementing the motion function $m(i)$ with a single motion feature. Each threshold is updated to approximate the segmentation output to the ground truth.

Since the objective function is nonlinear and non-smooth, the chosen algorithm was a genetic algorithm (GA). The GA benefits from having its variables constrained. The lower limit should be zero, and the upper limit can be the maximum values of the ground truth motion features. Fig. 3 shows that for this use case there are different values for the thresholds and sliding window size that when combined conduct to zero error. The proposed algorithm for the motion function in pseudocode is as follows.

Algorithm SlidingWindow($\boldsymbol{O}$,n,$\boldsymbol{t}$,k_s,w)

inputs: $\boldsymbol{O}$ observations matrix
n number of observations to evaluate
$\boldsymbol{t}$ threshold vector
k_s threshold sensitivity factor
w window size
output: m motion function

```
1:  for $i \in [1, \text{LENGTH}(\boldsymbol{t})]$ do          ▷ Apply sensitivity factor.
2:      $\boldsymbol{t}^{(i)} \leftarrow k_s \cdot \boldsymbol{t}^{(i)}$
3:  end for
4:  $l \leftarrow n + w - 1$
5:  $\boldsymbol{F} \leftarrow \text{GETFEATURES}(\boldsymbol{O})$   ▷ Calculate the features from
    the obtained observations, equations (7) and (8).
6:  for $i \in [1, l]$ do        ▷ Obtain the motion binary function.
7:      $\boldsymbol{m}(i) \leftarrow 0$
8:      for $j \in [1, \text{LENGTH}(\text{t})]$ do
9:          $m(i) \leftarrow (\boldsymbol{F}^{(i,j)} \geq \boldsymbol{t}^{(i)}) \vee m(i)$
10:     end for
11: end for
12: for $i \in [1, n]$ do  ▷ Search for motion in sliding window.
13:     for $j \in [1, w-1]$ do
14:         $m(i) \leftarrow m(i) \vee m(i+j)$
15:     end for
16: end for
```

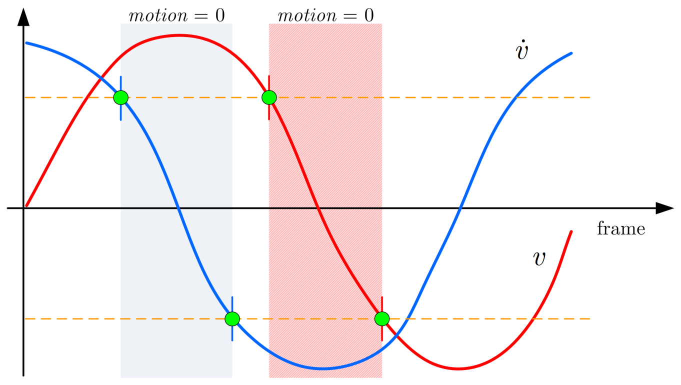

Fig. 4. Example of kinematic quantities in a dynamic gesture with sudden direction change. The flat shade represents acceleration below the threshold andthe stripe shade the velocity below the threshold. If they overlap during a large enough number of frames, a false negative will be triggered.

C. Motion Features

In an ideal system, the absence of movement would be defined by null differences of the system variables between frames. Therefore, the simplest set of features that can be used for this method is the frame differences, $\triangle \mathbf{f}$, that at an instant i is given by:

$$\Delta \mathbf{f}\,(i) = \mathbf{f}\,(i) - \mathbf{f}\,(i-1) \tag{6}$$

However, these features do not yield consistently reliable results. For example, if we consider as input a position in Cartesian coordinates, this approach performs poorly, since the differences would be relative to the coordinated axis. A motion pattern with a direction oblique to an axis would have lower coordinate differences compared to a pattern parallel to an axis with similar speed, thus producing different results. This issue can be solved by replacing the three coordinate differences with the respective Euclidian length, $d(i)$, which is a value proportional to the average speed between frames, here denominated by $v(i)$.

$$v\,(i) = \sqrt{\Delta x(i)^2 + \Delta y(i)^2 + \Delta z(i)^2}, \quad i \in \mathbb{R}^+ \tag{7}$$

where $\Delta x(i) = |x\,(i-1) - x\,(i)|$. The same reasoning for by $\Delta y(i)$ and $\Delta z(i)$.

In the presence of gesture patterns with sudden inversions of direction false negatives are very detrimental to the classifier accuracy. The proposed solution is adding an extra motion feature, the acceleration, $\dot{v}(i)$. The acceleration is at its highest when an inversion of direction occurs, which solves the low velocity problem. This feature does not cause false positives in a static gesture and deals successfully with the inversions of movement on dynamic gestures. The average acceleration between frames is proportional to the difference of velocities:

$$\dot{v}\,(i) = v\,(i) - v\,(i-1) \tag{8}$$

In Fig. 4 the shaded areas imply that a variable is below the threshold, meaning an absence of motion in the variable.

III. Testing Methodology

The performance of the proposed segmentation by motion methodology was evaluated in different tests performed by different subjects. Experimental tests were conducted with participants wearing a data glove and a magnetic tracker device. The participants performed a number of gestures (static, dynamic and ME) while the segmentation system was detecting motion in real-time from the gesture dataset. The results are compared and discussed, even with a supervised method. The algorithm has order $O(n)$, in which n is the number of motion features.

A. Interaction Technologies and Data Acquisition

Two different sensors are used to acquire human behavior during the interaction process: a magnetic tracker device (hand and arm motion) and a data glove (finger motion).

The electromagnetic tracker (Polhemus Liberty) provides the position and orientation of the sensor in relation to a magnetic source in a total of 6 DOF, $\mathbf{l}(i) = (l_1, l_2, \ldots, l_6)$, in which i represents a frame at a certain instant of time. The first three indexes describe the position along the three coordinated axes, $(x, y, z)_i = (l_1, l_2, l_3)_i$. The last three indicate the angles yaw, pitch and roll, respectively, $(\Psi, \theta, \phi)_i = (l_4, l_5, l_6)_i$.

The data glove (CyberGlove II) has 22 resistive bend sensors integrated: three flexion sensors per finger, four abduction sensors, a palm-arch sensor, and sensors to measure wrist flexion and abduction $\mathbf{g}(i) = (g_1, g_2, \ldots, g_{22})_i$. The system provides a low-accuracy timestamp for each frame based on software running time, t_g.

Data from the glove and the tracker are acquired and saved on device-specific buffers. These buffers are then sampled for the newest sampels and processed at 20 Hz. Since the sensors have different rates, the remaining (older) frames of the faster device are dropped. The frames from both devices are then concatenated into one single vector $\mathbf{f}$ (9) and put in a circular buffer.

$$\mathbf{f}(i) = (t_l, l_1, l_2, \ldots, l_6, t_g, g_1, g_2, \ldots, g_{22})_i \tag{9}$$

During sampling, to synchronize the system's output with the actual gesture (ground truth), the sampling sessions were recorded using a consumer grade camera at 30 Hz. The actual gestures were recorded as well as the system's graphical user interface (GUI), showing the segmented data stream.

B. Gesture Dataset

According to the functionalities to be achieved and based in [1], a dataset of continuous static/dynamic gestures, including ME, was created to test the developed method. It is our aim to have gestures containing fingers and arm motion from wearable sensing data, with different lengths (0.5 to 2 seconds), including the transition data between gestures. There is an infinite number of possible combinations of individual gestures to create a gesture sequence. A representative sequence of 8 gestures that triggers most of the variable's thresholds was selected, Fig. 5. The sequence was performed 20 times by each participant (subject A, B and C), summing up to 480 gesture samples. The sequence samples have an average of 16 seconds of data recorded at a rate of 100 Hz.

C. Motion Features and Sliding Window

The proposed motion features are: (1) wrist velocity (1 DOF), (2) wrist acceleration (1 DOF), (3) wrist angular velocity (1 DOF), and (4-25) finger joints angular velocities (21 DOF). These features are organized in a feature vector $\mathbf{t}$:

$$\mathbf{t}(i) = \begin{bmatrix} v(i) & \dot{v(i)} & \omega(i) & \dot{g}_1(i) & \cdots & \dot{g}_{22}(i) \end{bmatrix}^T \quad (10)$$

The wrist velocity $v(i)$ is defined by:

$$v(i) = \frac{\sqrt{\triangle l_1^2 + \triangle l_2^2 + \triangle l_3^2}}{t_l(i) - t_l(i-1)} \quad (11)$$

where $\Delta l_h = l_h(i) - l_h(i-1)$, $1 \leq h \leq 3$. The wrist acceleration:

$$\dot{v}(i) = \frac{|v(i) - v(i-1)|}{t_l(i) - t_l(i-1)} \quad (12)$$

The wrist angular velocity is:

$$\omega(i) = \frac{\sqrt{\triangle l_4^2 + \triangle l_5^2 + \triangle l_6^2}}{t_l(i) - t_l(i-1)} \quad (13)$$

where $\Delta l_q = l_q(i) - l_q(i-1)$,$1 \leq q \leq 3$. The finger joints angular velocities are:

$$\dot{g}_n(i) = \frac{|g_n(i) - g_n(i-1)|}{\Delta t_g}, \ n = 1, 2, \ldots, 22 \quad (14)$$

In (14), Δt_g is the average period of the glove's output (10 ms).

D. Threshold Calibration

For the threshold calibration 2 samples of motion features (10 seconds each), during static and moving poses, were acquired. The window size was kept fixed at 20 frames (200 ms) in order to minimize segmentation delay. The calibration was implemented using Solver's Evolutionary GA [20]. The algorithm was run for each feature individually with a population size of 100 and a mutation rate of 0.075 (parameters determined by trial and error). The convergence time of the GA is about 1 minute. The local minima are shown in Fig. 6. Some of the features have very high objective function values (error next to 1000), which is explained by poor correlation between the respective feature and the samples' segmentation ground truth (features 3, 4 and 21). This is most likely caused by low activation of some sensors by the sampling movements. Nevertheless, when all the thresholds are applied concurrently, the objective function drops to the size of the window size. This amount of error is anticipated by the method and is the minimum attainable, so the authors considered the process to be complete. For testing it was considered a value of 3.0 to the threshold sensitivity factor k_s.

E. Performance Parameters

This study proposes the following quantitative parameters to measure segmentation accuracy: average start delay $\overline{\Delta s}$ (ms), average end delay, $\overline{\Delta e}$ (ms), segmentation error S_{error}, segmentation accuracy SA, and extend level EL.

The start delay represents the delay between the start of the motion (ground truth) and the timing in which the proposed system detects such motion, including all processing and communications delay. It is calculated by:

$$\overline{\Delta s_i} = \frac{\sum_i SS_i - GS_i}{N} \cdot \frac{1000}{FR} \quad (15)$$

As shown in Fig. 7, SS_i is the segmentation start frame number for gesture sample i, GS_i is the ground truth gesture start frame, FR is the video capture frame rate, and N is the total number of samples.The end delay represents the delay between the end of the motion (ground truth) and the time of which the proposed system detects the end, including all processing and communications delay:

$$\overline{\Delta e_i} = \frac{\sum_i SE_i - GE_i}{N} \cdot \frac{1000}{FR} \quad (16)$$

where SE_i is the segmentation end frame number for gesture sample i and GE_i is the ground truth gesture end frame.

The segmentation error S_{error} is the fraction between the number of segmentation errors (the sum of the number of times a gesture is oversplit and false segments of motion) and the number of samples:

$$S_{error_i} = \frac{\text{\# of segmentation errors}}{N} \cdot 100 \quad (17)$$

For window size comparison, we define the segmentation accuracy for a given sample i as:

$$SA_i = \frac{\max(\text{RSE}) - \text{RSE}}{\max(\text{RSE}) - \text{ESE}} \cdot 100 \quad (18)$$

where RSE stands for the number of reported start events and ESE is the number of expected start events.

The extend level, EL, measures the discrepancy in length between the ground truth and the system output. This indicates that relevant data for the gesture recognition is captured but some noise is also captured.

$$EL_i = \frac{SD_i - GD_i}{GD_i} = \frac{\overline{\Delta e_i} - \overline{\Delta s_i}}{GD_i} \quad (19)$$

F. Results and Discussion

For a window size of 20, the average S_{error} was 1.3% for subject A, 3.6% for subject B and 3.8% for subject C, Table I. The discrepancy between the results for each subject are justified by their familiarity with the gestures. Subjects B and C did not calibrate the system not practiced the gesture sequence, which led to hesitation and pauses during the gestures (oversegmentation). If a larger window size were used, the number of errors would have been significantly lower for subjects B and C. Fig. 8 shows the plotted motion features over time during a sampling session performed by subject A.

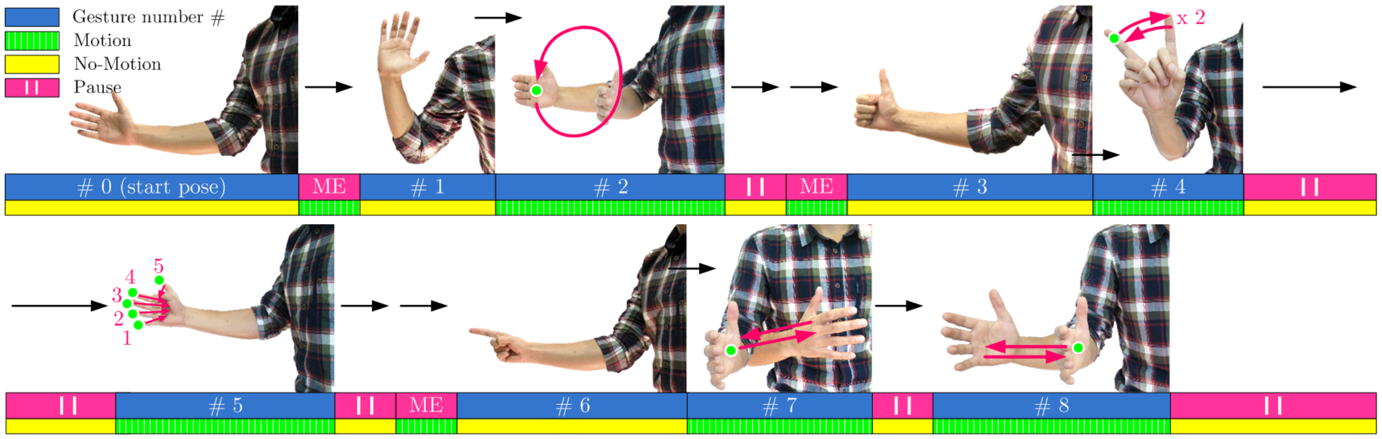

Fig. 5. Gesture sequence performed during the sampling process to create the dataset. ME represents movement epenthesis with associated motion segments.

TABLE I
PERFORMANCE PARAMETERS FOR THE SAMPLES IN THE DATASET. GN INDICATES THE GESTURE NUMBER IN THE DATASET.

Performance parameters	Subject	Method	Gn 1	Gn 2	Gn 3	Gn 4	Gn 5	Gn 6	Gn 7	Gn 8	Average
$S_{\text{error}}(\%)$	A	$Unsupervised$	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	1.3
$S_{\text{error}}(\%)$	B	$Unsupervised$	0.0	0.0	9.5	4.8	9.5	0.0	0.0	0.0	3.6
$S_{\text{error}}(\%)$	C	$Unsupervised$	0.0	0.0	0.0	20.0	0.0	0.0	10.0	0.0	3.8
$S_{\text{error}}(\%)$	A	$Supervised$	5.0	0.0	0.0	0.0	25.0	5.0	0.0	0.0	4.4
$S_{\text{error}}(\%)$	B	$Supervised$	30.0	40.0	25.0	65.0	55.0	0.0	35.0	70.0	40.0
$EL(\%)$	A	$Unsupervised$	25.3	-1.8	10.0	2.7	14.8	-10.5	14.5	17.8	9.1

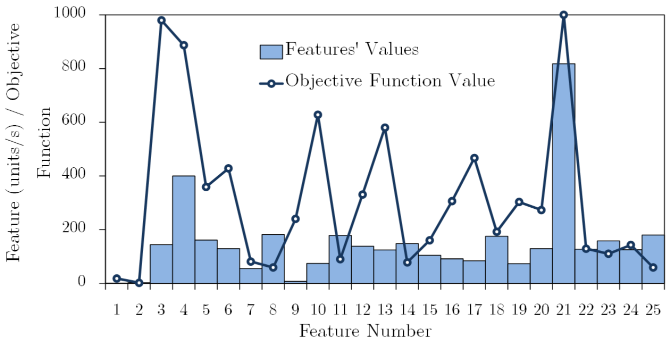

Fig. 6. Feasible thresholds and the objective function values for each of the features.

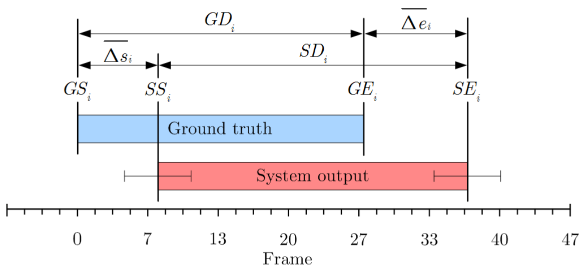

Fig. 7. Example of an output segment and the corresponding ground truth. The GS_i, SS_i, GE_i and SE_i parameters are acquired in the testing phase.

The segmentation output results are superimposed onto the ground truth as seen by the user. False negatives were detected only in gesture 4 due to a long stop during the inversion of movement. The position-related features from the magnetic tracker (feature 1 - $v(i)$, feature 2 - $\dot{v}(i)$ and feature 3 - $\omega(i)$) have higher resolution and accuracy than the other features, so that they are easier to calibrate. Consequently, the thresholds are lower and when the data are normalized, the resulting values are higher than the other features.

The average segmentation start delay for subject A was 230 ms while the end delay was 296 ms (the same behavior for the other subjects). The gesture end delay was higher, as expected, because of the sliding window. Nevertheless, the difference is 66 ms, considerably less than the size of the window (200 ms). The most likely reason for this is asymmetry between the features at the start and end of a gesture, a slow start and a fast end for a given gesture. The total delay should not be problematic for the subsequent classification process, since most of the gesture data is within the output segment, Fig. 9. The segmentation start delay $\overline{\Delta s}$ has two main components: (1) the feature capture/processing delay CD, and (2) the method's delay MD. For this study, MD is the most relevant, since CD changes according to the code efficiency and sensor latency. The total delay can be seen in Fig. 9. The fraction of the delay corresponding to the method itself in our experiments is from 15% to 35%. This originates from the time the features need to reach the threshold while the gesture is speeding up.

Segmentation performance depends largely on the size of the sliding window. The segmentation accuracy SA was

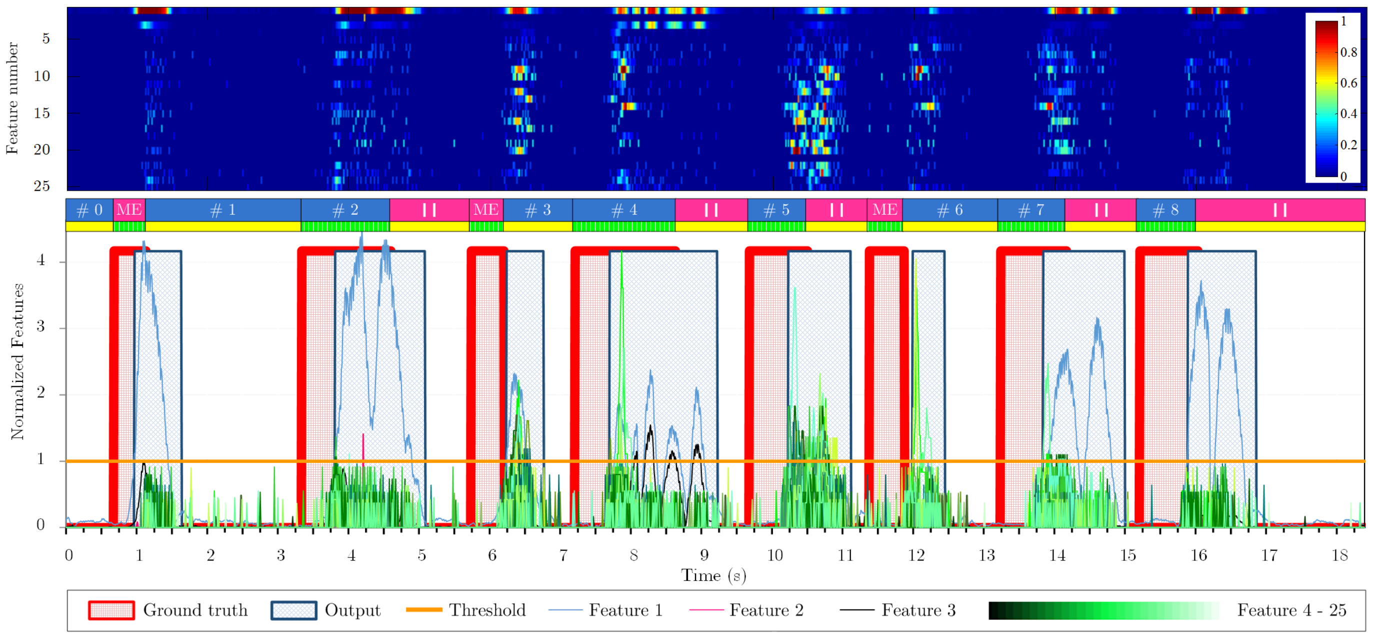

Fig. 8. Data from one of the samples in the dataset (performed by subject A). On the top, the features over time are normalized and colormapped. Below that, the features are normalized by the threshold and plotted over time.

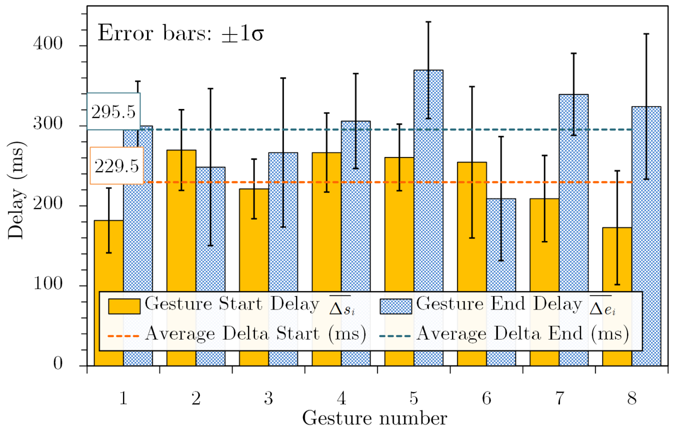

Fig. 9. Gesture-wise average segmentation delay for subject A.

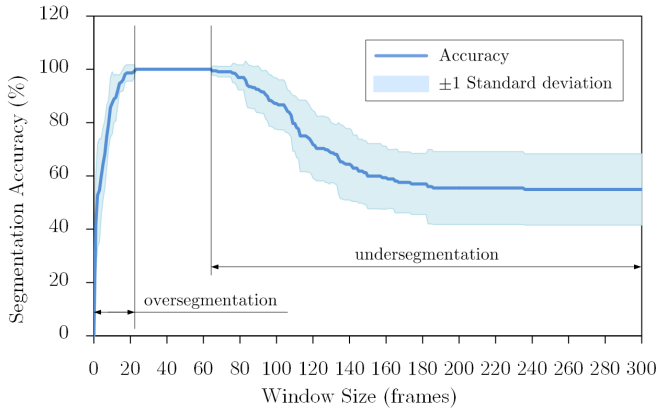

Fig. 10. Segmentation accuracy variation according to sliding window size for the fixed thresholds established in Fig. 6.

measured for different sliding window sizes with a fixed threshold, Fig. 10. Initially, with small sliding windows, there is excessive segmentation (oversegmentation), i.e., every peak of the features is considered a segment. This leads to low accuracy. Nevertheless, the accuracy quickly rises to 100% at a window size of 24 frames. The window size used in the previous tests was below this value (20 frames), hence the oversegmentation errors on gesture 4. The accuracy then plateaus at 100% until window sizes of 65 frames. It then drops, but this time because of undersegmentation. Undersegmentation occurs when a gesture is detected but its end is not. This causes consecutive gestures to overlap. The accuracy plateaus again at higher window sizes, at which point all sample gestures are overlapping into one single segment. Fig. 10 demonstrates that the proposed method behaves as in [4]. The segmentation accuracy is improved initially as the size of the sliding window increases and degrades as the window size increases further. Different classification methods tolerate different segmentation delays for gesture start, gesture end and extend level [18]. The results obtained compare favorably with the best results in literature.

The proposed system was compared with a supervised method, Table I. A one-class feed-forward neural network (FFNN) was used, having as input the sliding window data and a single output neuron outputting a motion index. It was trained with the same calibration data applied in the unsupervised method (only data from subject A like in the unsupervised method). Results compare unfavourably with the proposed unsupervised method due to the frequent oversegmentation, especially for subject B (S_{error} of 40%) that tested the system trained with calibration data from subject A. These results indicate that the proposed unsupervised method can be considered user independent.

IV. Conclusion and Future Work

This paper presented a novel method for unsupervised continuous gesture segmentation by motion detection. It can be concluded that the proposed solution accurately divides a continuous stream of motion data in static and dynamic segments. A quantitative analysis with samples from three subjects indicates an oversegmentation error of 2.70% for a feasible sliding window size of 20 frames, and 0% for an optimal sliding window size (subject A). Segmented data present in most cases a shift-right and extended behavior when compared to the ground truth. The automatic definition of feasible thresholds vector using a genetic algorithm demonstrated reliable behavior. Acceleration features demonstrated to be necessary to achieve better segmentation for gestures with sudden inversions of movement direction, avoiding the appearance of false negatives. The proposed unsupervised segmentation method reduces the noise in input data for subsequent classification.

Future work will be dedicated to testing the proposed solution with other interactions technologies: vision, IMUs and Electromyography (EMG). Additional efforts will be dedicated to improve the accuracy in computing velocity and acceleration features by filtering them.

References

[1] P. Neto, D. Pereira, J. N. Pires, and a. P. Moreira, "Real-time and continuous hand gesture spotting: An approach based on artificial neural networks," *2013 IEEE International Conference on Robotics and Automation*, pp. 178–183, 2013.

[2] R. Yang, S. Sarkar, and B. Loeding, "Handling movement epenthesis and hand segmentation ambiguities in continuous sign language recognition using nested dynamic programming." *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 3, pp. 462–77, mar 2010.

[3] D. Kelly, J. Mc Donald, and C. Markham, "Weakly supervised training of a sign language recognition system using multiple instance learning density matrices." *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 41, no. 2, pp. 526–41, apr 2011.

[4] D. Kim, J. Song, and D. Kim, "Simultaneous gesture segmentation and recognition based on forward spotting accumulative HMMs," *Pattern Recognition*, vol. 40, no. 11, pp. 3012–3026, nov 2007.

[5] H.-D. Yang, S. Sclaroff, and S.-W. Lee, "Sign language spotting with a threshold model based on conditional random fields." *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 7, pp. 1264–77, jul 2009.

[6] J. Kim, "An HMM-based threshold model approach for gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 961–973, 1999.

[7] B. Yao, H. Hagras, M. J. Alhaddad, and D. Alghazzawi, "A fuzzy logic-based system for the automation of human behavior recognition using machine vision in intelligent environments," *Soft Computing*, apr 2014.

[8] H. Kang, C. Woo Lee, and K. Jung, "Recognition-based gesture spotting in video games," *Pattern Recognition Letters*, vol. 25, no. 15, pp. 1701–1714, nov 2004.

[9] O. Banos, M. Damas, H. Pomares, F. Rojas, B. Delgado-Marquez, and O. Valenzuela, "Human activity recognition based on a sensor weighting hierarchical classifier," *Soft Computing*, vol. 17, no. 2, pp. 333–343, jul 2012.

[10] M. J. Mataric, "Sensory-Motor Primitives as a Basis for Imitation: Linking Perception to Action and Biology to Robotics."

[11] J. Beh, D. Han, and H. Ko, "Rule-based trajectory segmentation for modeling hand motion trajectory," *Pattern Recognition*, vol. 47, no. 4, pp. 1586–1601, apr 2014.

[12] X. Cao, B. Ning, P. Yan, and X. Li, "Selecting Key Poses on Manifold for Pairwise Action Recognition," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 168–177, feb 2012.

[13] C. Tran and M. M. Trivedi, "3-D Posture and Gesture Recognition for Interactivity in Smart Spaces," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 178–187, feb 2012.

[14] Y. Song, D. Demirdjian, and R. Davis, "Continuous body and hand gesture recognition for natural human-computer interaction," *ACM Transactions on Interactive Intelligent Systems*, vol. 2, no. 1, pp. 1–28, mar 2012.

[15] H. Junker, O. Amft, P. Lukowicz, and G. Tröster, "Gesture spotting with body-worn inertial sensors to detect user activities," *Pattern Recognition*, vol. 41, no. 6, pp. 2010–2024, jun 2008.

[16] Z. Lu, X. Chen, Q. Li, X. Zhang, and P. Zhou, "A Hand Gesture Recognition Framework and Wearable Gesture-Based Interaction Prototype for Mobile Devices," *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 2, pp. 293–299, apr 2014.

[17] T. Stiefmeier, D. Roggen, G. Ogris, P. Lukowicz, and G. Tr, "Wearable Activity Tracking in Car Manufacturing," *IEEE Pervasive Computing*, vol. 7, no. 2, pp. 42–50, apr 2008.

[18] L.-V. Nguyen-Dinh, A. Calatroni, and G. Tröster, "Robust online gesture recognition with crowdsourced annotations," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3187–3220, jan 2014.

[19] B. Bruno, F. Mastrogiovanni, and A. Sgorbissa, "Wearable Inertial Sensors: Applications, Challenges, and Public Test Benches," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 116–124, sep 2015.

[20] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (2nd edition)*. New York, USA: Springer, 2007.

Miguel A. Simão received the M.Sc. in mechanical engineering from the University of Coimbra, Portugal, in 2014. He was visiting researcher at the Jet Propulsion Laboratory – NASA, California Institute of Technology, USA in 2015. Currently he is Ph.D. student at the University of Coimbra and École National Supérieure d'Arts et Métiers. His research interests include collaborative robotics, gesture recognition and pattern classification.

Pedro Neto received the Ph.D. in Mechanical Engineering from the University of Coimbra, Portugal, in 2012. He is currently Assistant Professor at the University of Coimbra and director of the Collaborative Robotics Laboratory. Pedro Neto is author of several journal and conference publications and participated in two European HORIZON 2020 R&D flagship projects. His research interests include human-robot interaction, collaborative robotics, pattern classification and smart manufacturing.

Olivier Gibaru received the Ph.D. in applied mathematics from ParisTech, Paris, France, in 1997. Currently, he is the Director of the Laboratory of Information Technology and Systems in the Lille campus, CNRS and a Full Professor at the Department of Mathematics and Computer Science at École National Supérieure d'Arts et Métiers, Lille campus, France. His research interests are in applied mathematics, estimation for robotic applications, geometry, control engineering and high precision mechanical systems. He has authored several papers in refereed journals and international conference proceedings. Prof. Gibaru is an active member of the SMAI-SIGMA group, is a national learning society dedicated to Applied Mathematics for Industrial Applications.