ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/330429917

# Online Recognition of Incomplete Gesture Data to Interface Collaborative Robots

**Article** *in* IEEE Transactions on Industrial Electronics · January 2019

DOI: 10.1109/TIE.2019.2891449

CITATIONS
2

READS
393

**3 authors:**

Miguel Simão
University of Coimbra
**22** PUBLICATIONS **144** CITATIONS
SEE PROFILE

Olivier Gibaru
Ecole Nationale Supérieure d'Arts et Métiers
**124** PUBLICATIONS **983** CITATIONS
SEE PROFILE

Pedro Neto
University of Coimbra
**102** PUBLICATIONS **1,104** CITATIONS
SEE PROFILE

**Some of the authors of this publication are also working on these related projects:**

Project Factory of Future - Agile&Flexible Manufacturing View project

Project ComMUnion View project

All content following this page was uploaded by Pedro Neto on 21 January 2019.

The user has requested enhancement of the downloaded file.

# Online Recognition of Incomplete Gesture Data to Interface Collaborative Robots

***Abstract*—Online recognition of gestures is critical for intuitive human-robot interaction (HRI) and further push collaborative robotics into the market, making robots accessible to more people. The problem is that it is difficult to achieve accurate gesture recognition in real unstructured environments, often using distorted and incomplete multi-sensory data. This paper introduces a HRI framework to classify large vocabularies of interwoven static gestures (SGs) and dynamic gestures (DGs) captured with wearable sensors. DG features are obtained by applying data dimensionality reduction (DDR) to raw data from sensors (resampling with cubic interpolation and principal component analysis (PCA)). Experimental tests were conducted using the UC2017 hand gesture dataset with samples from eight different subjects. The classification models show an accuracy of 95.6% for a library of 24 SGs with a random forest (RF) and 99.3% for 10 DGs using artificial neural networks (ANNs). These results compare equally or favourably with different commonly used classifiers. Long Short-Term Memory (LSTM) deep networks achieved similar performance in online frame-by-frame classification using raw incomplete data, performing better in terms of accuracy than static models with specially crafted features, but worse in training and inference time. The recognized gestures are used to teleoperate a robot in a collaborative process that consists in preparing a breakfast meal.**

***Index Terms*—Human-Robot Interaction, Collaborative Robotics, Online Gesture Recognition, Neural Networks**

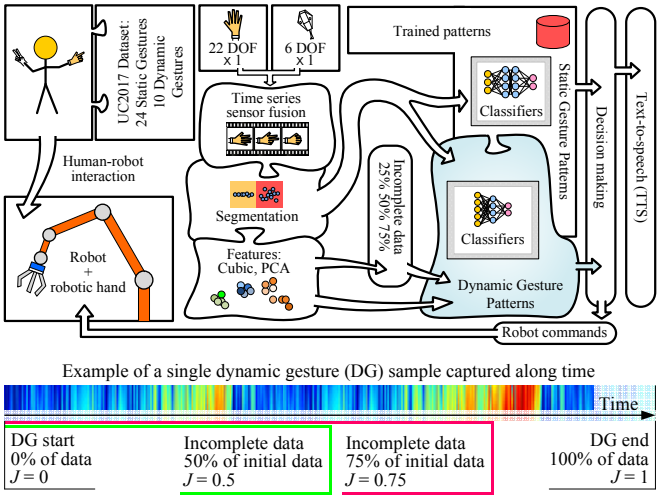


Fig. 1. Overview of the proposed gesture-based HRI framework: data acquisition, segmentation, features, classification and the robot interface. At the bottom it is explained the meaning of incomplete data for DG classification. For example a DG can be classified with initial 50% ($J = 0.5$) of data representing such gesture, i.e., DGs can be classified in anticipation, before the user finishes the gesture in real world.

## I. Introduction

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 where 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. Robots and humans have to understand each other and interact in a natural way (using gestures, speech and physical interaction), creating a co-working partnership. This will allow a greater presence of robots in all domains of our society. 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 in robot programming.

The collaborative robotics market is rapidly growing and HRI interfaces have a main role in the acceptance of robots as partners. Gestures are an intuitive interface to teleoperate a robot since they intuitive to use and do not require technical skills in robot programming to be used [1]. For instance, a human co-worker can use a DG to indicate a grasping position and use a SG to stop the robot [2]. In this scenario, the human has little or nothing to learn about the interface, focusing instead on the task being performed. The robot assists the human when necessary, thus reducing the exposition to poor ergonomic conditions and injury.

This paper proposes an integrated modular gesture-based HRI framework, Fig. 1. Static and dynamic gesture segments, composed of data captured by a data glove and magnetic tracker, are created automatically with a motion detection algorithm applied to a sliding window. Static segments are used as input for SG classifiers. Dynamic segments, which are discriminated by DG classifiers, are subject to data dimensionality reduction (DDR) with resampling based on cubic interpolation (CI) or principal component analysis (PCA). Traditional probabilistic latent variable models, such as PCA, are static linear approaches in which the dynamics and nonlinearities are not properly considered. In [3], the authors propose a weighted linear dynamic system (WLDS) for nonlinear dynamic feature extraction which showed superior prediction accuracy. Nevertheless, the real-time performance is worse when compared to static approaches because it requires more computational time. This is undesirable and limits the online performance of the proposed gesture-based HRI system. The proposed CI and PCA approaches are demonstrated to be computationally inexpensive without sacrificing classification accuracy, even when used with incomplete data. We use large vocabularies of gestures (a total of 34) and a relatively

low number of training samples to simplify and expedite the training process. Experiments demonstrated that standard classifiers, such as artificial neural networks (ANNs), are reliable in both SG and DG classification. Furthermore, they compare equal/favorably to deep learning classifiers (LSTM and Convolutional Neural Networks (CNNs)) in both inference time and accuracy. Finally, a robot task manager maps the classified gestures to robot commands.

### A. Motivation, Challenges and Contributions

A major challenge is the continuous, online and reliable recognition of gestures from real-time data streams. Continuous gesture recognition is the natural way used by humans to communicate with each other, in which communicative gestures (the effective SGs and DGs with an explicit meaning) appear intermittently with non-communicative gestures (pauses and movement epenthesis (ME) – inter-gesture transition periods) in a random order [4]. Many studies do not approach gesture classification in this continuous manner, nor address the negative effect of ME. It is also a challenge to recognize gesture patterns from incomplete data, as well as intuitively map the recognized gestures into robot commands for natural and safe HRI. In this context, the motivations behind this study are:

1) Combine and fuse sensor data from multiple wearable devices in order to capture a person's gestures (hand and arms) accurately, without occlusions;
2) Application of proper DDR methods to increase classification accuracy, reduce the training time, reduce the number of samples required to train the classifiers, while allowing online implementation;
3) Achieve high recognition rates (close to 100%) and ensure generalization capability in respect to untrained samples and new users;
4) Classification of DGs from incomplete data, allowing the classification of a gesture while it is being performed by the human;
5) Intuitive and online interfacing with a robot using gestures.

The proposed system was evaluated by conducting several experiments using wearable/body-worn sensors (a data glove and a magnetic tracker), resulting in the following contributions:

1) The combination of DDR (CI and PCA) and ANNs for DG classification from wearable sensor data resulted in high classification accuracy that compares favorably with standard classifiers, including deep learning LSTM and CNNs. This method is computationally inexpensive, allowing the gesture-based online interaction with the robot. Gestures are recognized with an accuracy of 95.6% for a library of 24 SGs and 99.3% for 10 DGs.
2) The above results were obtained in continuous data, with multiple subjects (user independence) and applied in an unstructured environment;
3) Sequential classification of DGs showed an accuracy that is higher with incomplete data (50% or 75% of initial frames of data that represent a DG) than with 100% of DG data across different classifiers and users. In this context, DGs can be classified in anticipation, before the user finishes a gesture;
4) Framework tested in real unstructured environment where the recognized gestures serve as an intuitive interface to manage an online collaborative process, in which a robot assists the human in the preparation of a breakfast meal.

### B. Related Work

Gesture-based Human-Robot Interaction (HRI) for collaborative robotics is an emerging and multidisciplinary research field. Communicative gestures provide information that is difficult to convey in speech, i.e., command gestures, pointing, gestures addressed to objects or actions, and mimicking gestures [5], [6]. Gestures have been proven to be one of the most effective and natural mechanisms for reliable HRI [7]. They have been used for robot teleoperation and to coordinate the interactive process of cooperation between human and robot. As stated in [8], an interactive robotic task generally consists of individual actions, operations and motions that are arranged in a hierarchical order so that the process can be managed by simple human gestures [9].

An inefficient segmentation process (determining when a gesture starts and ends) results in a classification model that is more likely to fail [10]. The analysis of continuous data streams to solve spatial and temporal segmentation is challenging [11]. The problem is that it is difficult to automate the segmentation process, making gesture recognition in real world scenarios a difficult task [12].

The input features for gesture recognition are normally the hand/arm/body position, orientation and motion [4], often captured from vision sensors. Owing to its naturalness, in opposition to wearable sensors that need to be attached to the human body, vision sensing is the most common interaction technology. Gesture classification from video stream requires large amounts of training data, especially for state-of-the-art deep learning classifiers. Moreover, it is difficult to construct reliable features from only vision sensing due to occlusions, varying light conditions and free movement of the user in the scene [13], [6]. To improve classification reliability, a significant number of studies combine data from vision and wearable sensors. Taking this into account, several approaches to gesture recognition rely on wearable sensors such as data gloves, magnetic tracking sensors, inertial measurement units (IMUs) and electromyography (EMGs), among others. In fact, these interaction technologies have been proven to provide reliable features in unstructured environments. Nevertheless, they also place an added burden on the user since they are worn on the body.

Some gestures, although not all, can be defined by their spatial trajectory, e.g., a circle. Burke and Lasenby succeeded on using PCA and Bayesian filtering for the classification of time series gestures [5]. Hidden Markov Models (HMMs) can be used to find time dependencies in skeletal features extracted from image and depth data (RGB-D) with a combination of Deep Belief Networks (DBNs) and 3D CNNs [14]. Deep learning combined with recurrent LSTM networks

demonstrated state-of-the-art performance in the classification of human activities from wearable sensors [15]. Features are automatically extracted from raw sensor data, avoiding the need for expert knowledge in feature design. The reported results show that this framework outperforms competing deep non-recurrent networks. Various ANNs in series demonstrated superior performance in the classification of a high number of gesture classes [16]. Field et al. used a Gaussian Mixture Model (GMM) to classify human body postures with previous unsupervised temporal clustering [17]. Switching Gaussian Process Dynamic Models (SGPDM) are proposed to capture motion dynamics and to identify motion classes such as walk or run, and smile or angry [18]. Recognition performance on real videos (comparatively low quality, low frame rates and with pose changes) demonstrated that the SGPDM model can efficiently track composite motions with various dynamics. A framework for dynamic hand gesture recognition using Generalized Time Warping (GTW) for alignment of time series is proposed in [19]. Features are extracted from the aligned sequences of hand gestures based on texture descriptors, and the hand motion recognition is performed by CNNs. Autoencoders and stacked autoencoders (SAE) have been successfully used for feature representation in various applications [20].

Recent studies report state-of-the-art methods for hand detection and gesture classification from RGB-D video using deep learning [21]. Generally speaking, deep learning requires large amounts of training data, the models are computationally expensive to train, and it is challenging to determine good hyperparameters, since deep networks are essentially black boxes (it is difficult to know exactly how and why they output certain values). Boosting methods, based on ensembles of weak classifiers, allow multi-class hand detection [22]. Despite all the proposed solutions, it is still challenging to use of gestures as a reliable interaction modality to control a robot/machine in real-time.

The evolution of pattern recognition has been enormous in the last few years. However, many of existing solutions address object or SG classification which is less challenging than sequential classification. Results obtained in well-established datasets have good accuracy in offline classification but are seldom tested online and the processing time is not mentioned. The ability to classify a gesture online is critical to interface with a machine/robot. Finally, no studies approach gesture classification from incomplete data, being normally assumed that more data results in better accuracy.

## II. Gesture Classification

### A. Problem Formulation

Within a continuous data stream, there may be a sequence of Static Gestures (SGs) and Dynamic Gestures (DGs) with no specific order. As new frames are acquired, they are segmented into static or dynamic frames with a motion detection algorithm [10]. This algorithm identifies motion, or lack thereof, including sudden inversions of movement direction which are common in DGs. This is achieved by the analysis of velocities and accelerations numerically derived from positional data. A genetic algorithm is used to compute motion thresholds from calibration data. As a result, we have static and dynamic blocks of frames contiguous in time. Static blocks are SG candidates and dynamic blocks are DG candidates. Therefore, we propose two independent classifiers, one for the classification of SGs and the other for the classification of DGs.

The segmentation function $\Gamma$ based on a motion-threshold algorithm is applied to a window of a stream of data $\boldsymbol{S}$, of dimensionality $d$ and length $n$: $\left\{(\boldsymbol{S}, \Gamma(\boldsymbol{S})) : \ \boldsymbol{S} \in \mathbb{R}^{d\times n} \text{ and } \Gamma(\boldsymbol{S}) \in \{0,1\}^n\right\}$. The static frames indicating no motion (input data for the SG classifier) are defined by $m_i = 0$ and the dynamic frames indicating motion (input data for the DG classifier) by $m_i = 1$.

The dynamic segments are extracted by a search function that finds transitions in $m$ (from 0 to 1 and 1 to 0). Given two consecutive transitions in the frames $i$ and $i + k$ so that $m_{i-1} = 0$, $m_i = 1$, $m_{i+k-1} = 1$ and $m_{i+k} = 0$, a DG sample is defined by:

$$\mathbf{X}^D = [S_{\bullet i}\ S_{\bullet i+1}\ ...\ S_{\bullet i+k-1}], \quad \mathbf{X}^D \in \mathbb{R}^{d\times k} \tag{1}$$

where the $S_{\bullet i}$ vector is the *i*-th column (frame) of the data stream. In terms of matrix notation, being $\mathbf{A} \in \mathbb{M}^{p\times q}$, $\mathbf{A}_{ij}$ represents the element of the array $\mathbf{A}$ with row $i$ and column $j$, $\mathbf{A}_{i\bullet} \equiv [\mathbf{A}_{i1} \cdots \mathbf{A}_{in}]$ and $\mathbf{A}_{\bullet j} \equiv [\mathbf{A}_{1j} \cdots \mathbf{A}_{nj}]^T$, and $\mathbb{M}$ is the notation for a real-valued matrix.

The static gesture samples are considered the first frame after a transition from $m_{i+k-1} = 1$ to $m_{i+k} = 0$:

$$\mathbf{X}^S = S_{\bullet i+k}, \quad \mathbf{X}^S \in \mathbb{R}^d \tag{2}$$

Hereinafter, the notation for a sample independently of its nature (static or dynamic) is $\mathbf{X}$. Static and dynamic samples are differentiated by their dimensionality. The *i*-th sample of a dataset is represented by $\mathbf{X}^{(i)}$. We represent the feature extraction pipeline by $\Pi$, which is used to transform the raw data into the predictors $\mathbf{z}$ that feed the classifiers: $\left\{(\boldsymbol{X}, \mathbf{z} = \Pi\,(\boldsymbol{X})) : \ X \in \mathbb{R}^{d\times n} \text{ and } \mathbf{z} \in \mathbb{R}^b\right\}$, where $d$ is the number of channels of the sample and $n$ its length. The target vectors are one-hot encoded class indexes. For any given sample, the target class has the index $o$ and the target vector $\mathbf{t}^{(o)}$ is defined by $\mathbf{t}_j^{(o)} = \delta_{oj},\ j = 1, ..., n_{classes}$. Therefore $\mathbf{t} \in \{0,1\}^{n_{classes}}$, $\delta$ is the Kronecker delta and $\mathbf{t}_j$ is the *j*-th element of $\mathbf{t}$.

For DGs, the transformation $\Pi$ could yield a long vector, which often makes training the classifier more difficult. Therefore, we introduce DDR at the end of the pipeline $\Pi$, such as PCA and CI. The feature vectors are fed into the respective classifiers.

In this study, our aim is to map the classified gestures into robot actions, such as moving to a target or halting movement. However, before issuing an action command, we must exclude poorly classified patterns from the stream. For

example, we can exclude classifications by context and by applying a threshold to the classification score:

$$\mathbf{o}_i = \begin{cases} \mathrm{SG}_t, & \text{if } p(\mathbf{y}=\mathbf{t}|\mathbf{z}) \geq \tau^S \wedge m_i = 0 \\ \mathrm{DG}_t, & \text{if } p(\mathbf{y}=\mathbf{t}|\mathbf{z}) \geq \tau^D \wedge m_i = 1 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where $\mathbf{o}_i$ is the output gesture class number, $p(\mathbf{y}=\mathbf{t}|\mathbf{z})$ is the likelihood of the classifier's output $\mathbf{y}$ being in the class $\mathbf{t}$ given the $\mathbf{z}$ input, $\tau$ is the likelihood threshold and $m_i$ is the motion variable associated to $\mathbf{z}$.

### *B. Feature Dimensionality Reduction*

For SGs no DDR is proposed since the feature space is relatively small. On the contrary, DDR is beneficial for DGs feature extraction due to the relatively large feature space and to standardize the variability of DG length (reducing the gesture length to a small fixed size – resampling). We propose two forms of dimensionality reduction, using CI and PCA. CI is used to transform any variable-length DG sample $\mathbf{X}^{(i)} \in \mathbb{M}^{d\times n}$ into a fixed-dimension sample $\mathbf{X}' \in \mathbb{M}^{d\times k}$, effectively reducing the sample dimensionality if $k < n$. PCA performs an orthogonal linear transformation of a set of $n$ $d$-dimensional observations, $\mathbf{X} \in \mathbb{M}^{d\times n}$, into a subspace defined by the Principal Components (PCs). The PCs have necessarily a length smaller than or equal to the number of original dimensions, $d$. The first PC has the largest variance observed in the data. Each of the following PCs is orthogonal to the preceding component and has the highest variance possible under this orthogonality constraint. The PCs are the eigenvectors of the covariance matrix and its eigenvalues are a measure of the variance in each of the PCs. Therefore, PCA can be used for reducing the dimensionality of gesture data by projecting such data into the PC space and truncating the lowest-ranked dimensions. These dimensions have the lowest eigenvalues, so that truncating them retains most of the variance present in the data, i.e., most of the information of the original data is kept in the reduced space. In this study, the singular vectors of the samples across time are calculated and used as features. The first singular vector determines the direction in the PC-space in which there is the most significative variance along a DG. This means that the singular vector is a measure of the relative variance of each variable along time and good features for DG classification are expected. Another advantage is that the PCs can be calculated even before a DG is finished (incomplete data), remaining good predictors. As a result, these features are actually time series since we can calculate them on any window of data starting with the first frame and ending with any arbitrary frame of the sample.

### *C. UC2017 Hand Gesture Dataset*

We introduce the UC2017 static and dynamic gesture dataset. Most researchers use vision-based systems to acquire hand gesture data. Despite that, we believe that more reliable results from more complex gestures can be obtained with wearable sensor systems. There are not many datasets using wearable systems due to the plethora of data gloves in the market and their relative high cost. For these reasons, we opted by creating a new dataset to present and evaluate our gesture recognition framework. The objectives of the dataset are: (1) provide a superset of hand gestures for HRI, (2) have user variability, (3) to be representative of the actual gestures performed in a real-world interaction.

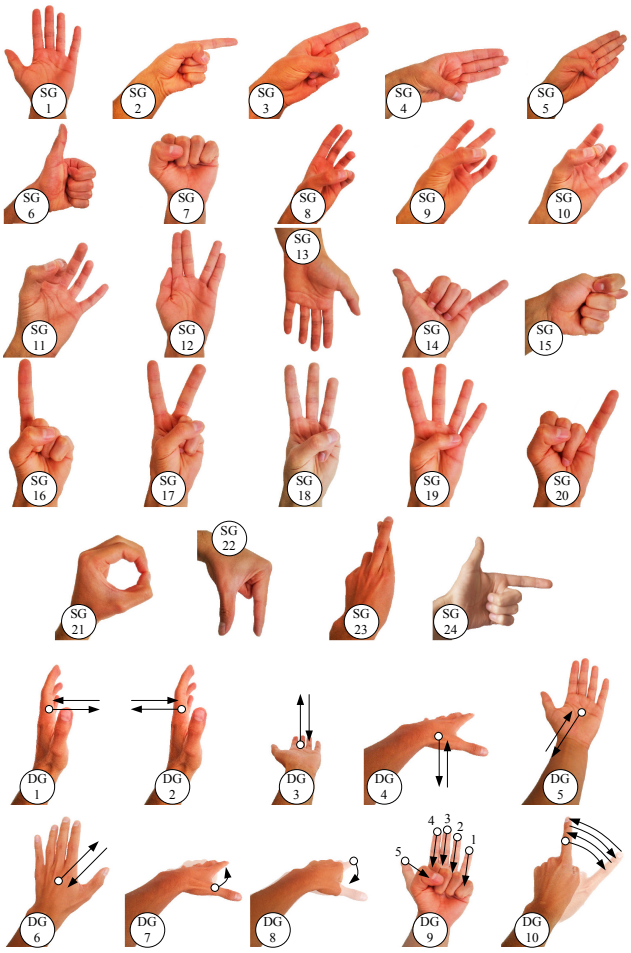


Fig. 2. Representations of the library of 24 static gestures and 10 dynamic gestures of the UC2017 library.

We divide the dataset in two types of gestures: SGs and DGs. SGs are described by a single timestep of data that represents a single hand pose and orientation. DGs are variable-length time series of poses and orientations with particular meanings. Some of the gestures of the dataset are correlated with specific interactions in the context of HRI, while the others were arbitrary selected to enrich the dataset and add complexity to the classification problem.

The library is composed of 24 SGs and 10 DGs, Fig. 2. The dataset includes SG data from eight subjects with a total of 100 repetitions for each of the 24 classes (2400 samples in total). The DG samples were obtained from six subjects and has cumulatively 131 repetitions of each class (1310 samples in total). All of the subjects are right-handed and performed the gestures with their left hand.

A data glove (CyberGlove II) and a magnetic tracker (Polhemus Liberty) are used to capture the hand shape, position and orientation over time. The glove provides digital signals $g_i$ proportional to the bending angle of each one of the 22 sensors elastically attached to a subset of the hand's joints: 3 flexion sensors per finger, 4 abduction sensors, a palm-arch sensor, and 2 sensors to measure wrist flexion and abduction. These 22 sensors provide a good approximation of the hand's shape. The tracker's sensor is rigidly attached to the glove on the wrist and measures its position in Cartesian space and orientation in respect to a ground-fixed frame (a magnetic source cube defines the reference coordinate system frame). The orientation is the rotation between the fixed frame and the frame of the tracker sensor, given as the intrinsic Euler angles yaw, pitch and roll (ZYX). The Cartesian position $(x, y, z)$ is denominated by $(l_1, l_2, l_3)$ and in terms of orientation the roll angle is denominated by $l_4$, the pitch by $l_5$ and the yaw by $l_6$. Sensor data are fused together online since the sensors have slightly different acquisition rates – 100Hz for the glove and 120Hz for the tracker. Tracker data are under-sampled by gathering only the closest tracker frame in time.

A goal was to obtain multiple repetitions of each gesture in the library to build the dataset. We also want the dataset to be representative of real-world conditions, so we must guarantee that the samples are independent.

The magnetic tracker reference was fixed in a location free of magnetic interference. The users are then asked to put on the data glove on their own on their left hand, even though all of our test subjects were right-handed. As a result, the sensors are not carefully placed, which should yield a dataset with larger variance. There is no calibration done in this setup. The subjects follow a graphical interface that shows the representation of the gesture to be performed and press a button to save a sample. The order of the gestures was randomized to prevent order dependencies. Furthermore, the subjects were requested to repeat the sampling for two to three different sessions. We have also implemented an online movement detection algorithm to facilitate the labeling of DGs, namely the starting and ending frames.

A final point should be made about the random sampling of the DGs. We have included in the samples the transition between the ending pose of the previous sample and the starting pose of a sample (movement epenthesis). Owing to random sampling, there is a high likelihood that all of the possible transitions were recorded.

The dataset and accompanying code are publicly available at [Omitted due to blind review].

**Coordinate transformation**: Gesture classification must be independent of the subject's position and orientation in space. Since the user is free to move around in the world reference frame $\{W\}$, we need to make sure that every gesture sample has its feature data reported to their local reference frame $\{L\}$. Origin and orientation of $\{L\}$ are defined in relation to $\{W\}$ at the instant a gesture begins. The proposed transformation is composed of a 3D translation and a rotation around the vertical axis $z$.

We denote $l_i$ and $g_i$ as the *i*-th Degrees of Freedom (DOF) of the tracker and glove, respectively. At the beginning of a DG sample $\mathbf{X}^{(i)}$, the yaw $\psi_0$ ($l_6$) and position $\mathbf{p}_0^W$ ($l_1$ through $l_3$) of the first frame are stored. The yaw angle is used to calculate the rotation matrix for each sample. It allows to consistently distinguish important directions, such as right, left, front and back, in respect to the user. The rotation is applied to every frame of a sample so that we can translate the coordinate system to frame $\{L\}$, Fig. 3:

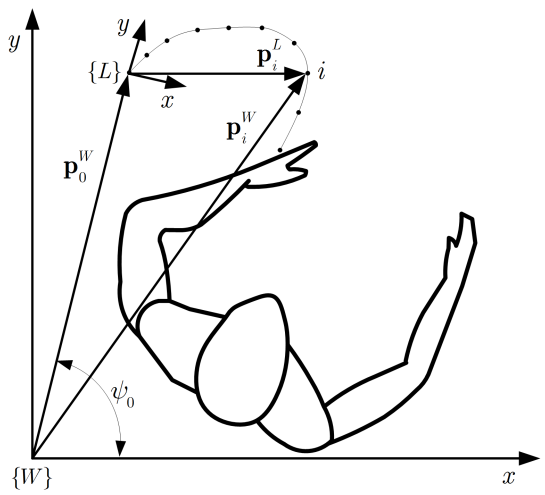


Fig. 3. Representation of the transformation of the world coordinates $\{W\}$ of a gesture to a local coordinate frame $\{L\}$.

$$\mathbf{p}_i^L = {\mathbf{R}_L^W}^T(\psi_0) \cdot \left(\mathbf{p}_i^W - \mathbf{p}_0^W\right) \tag{4}$$

where $\mathbf{p}_i^L$ is the position of the $i$th frame in respect to the local reference frame $\{L\}$, and $\mathbf{R}_L^W(\psi_0)$ is the rotation matrix that represents the rotation around $z$ of the world reference frame $\{W\}$ to $\{L\}$ by $\psi_0$ degrees. The rotation matrix is given by:

$$\mathbf{R}_z(\psi) = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

After the transformation, the yaw angle is $\psi_i^L = \psi_i^W - \psi_0^W$. In summary, we have a transformation function $\Psi$ applied to a DG sample $\mathbf{X}^{(k)} \in \mathbb{M}^{d\times n}$:

$$\begin{aligned} \Psi: \quad & \mathbb{R}^{6\times\bullet} \rightarrow \mathbb{R}^{6\times\bullet} & & \\ & \mathbf{X}_{ij}^{(k)} \rightarrow \left(\mathbf{p}_1^L\ \mathbf{p}_2^L\ \mathbf{p}_3^L\ \mathbf{l}_4\ \mathbf{l}_5\ \psi^L\right)_{\bullet j}^{(k)\,T}, & & \begin{matrix} i = 23, ..., 28 \\ j = 1, 2, ..., n \end{matrix} \end{aligned} \tag{6}$$

where $(...)_{\bullet j}^{(k)}$ corresponds to the *j*-th timestep of sample *k*.

**Dataset split**: The dataset is split before feature extraction. It is shuffled and split in three subsets: training (70%), validation (15%) and test (15%). The training set was used to train the classifiers and to obtain feature scaling parameters, such as mean and standard deviation. The classifiers' hyper-parameters were optimized for accuracy on the validation set. The generalization capability of the model is measured by the accuracy on the test set. The samples of one subject were held-out from the training set to ascertain the performance on

new users. Users that trained the system are designated by "trained users" and users that that did not train the system are designated by "untrained users".

### D. Feature Extraction

For **Static Gestures** the features are all the angles provided by the glove, $g_1$ $g_2$ ... $g_{22}$, and the pitch angle, $l_5$ (to differentiate gestures with similar handshapes and distinct orientation). Thus, the features chosen are simply a subset of the available raw data. Finally, the features are standardized by $x'_i = (x_i - \bar{x}_i)/s_i$, where $x'_i$ is the standardized value of feature i, $x_i$ is the value of the feature, $\bar{x}_i$ and $s_i$ are the mean and standard deviation of the feature in the training set. The validation and test sets are standardized by these same means and standard deviations.

For **Dynamic Gestures** we propose three different sets of features. For all sets, data samples are preprocessed according to $\Psi$, defined in (6):

$$\mathbf{X}'^{(i)}=\Psi(\mathbf{X}^{(i)}), \quad \mathbf{X}^{(i)} \in \mathbb{R}^{28\times n} \tag{7}$$

where *n* is the length of the DG sample. Starting from $\mathbf{X}'$, the first proposed set DG-CI uses the full DG data resized to a fixed length by applying CI. The second set, DG-PV, is based on PCA and represents the extraction of the first principal vector (PV) from DG data. The third set is the preprocessed data, which we call RAW.

For DG-CI, given a DG sample $\mathbf{X}^{(i)}$ with $n$ frames ($\mathbf{X}^{(i)} \in \mathbb{R}^{28\times n}$), the goal is to resample it to a fixed size $n'$. The value for $n'$ can be chosen arbitrarily but higher values have a detrimental effect on the classification accuracy. Training the classifier is faster and often better with less features. For all experiments, the value $n' = 20$ was used because the gesture lengths in the dataset vary between 20 and 224 frames. The lowest length was selected. Applying CI, the result is a matrix $\mathbf{Z} \in \mathbb{R}^{28\times 20}$. By concatenating every frame vertically, $\mathbf{Z}$ is transformed into a vector $\mathbf{z} \in \mathbb{R}^{560}$:

$$\mathbf{z}^{(i)} = \left( {\mathbf{Z}^{(i)}_{\bullet 1}}^T, {\mathbf{Z}^{(i)}_{\bullet 2}}^T, \ldots, {\mathbf{Z}^{(i)}_{\bullet 20}}^T \right)^T \tag{8}$$

In DG-CI the feature extraction involves the whole DG data, so there is a prediction only after the gesture is complete. However, it is beneficial to have an early classification from incomplete data, i.e., before the full gesture data are available. For DG-PV, PCA allows to obtain features from incomplete gesture data and still obtain time-coherent features. We apply this methodology for each timestep of the gesture. The feature vector for sample $i$ at timestep $j$ is calculated by:

$$\boldsymbol{z}^{(i)}_j = pv\left(\left[\mathbf{X}'^{(i)}_{\bullet 1}\ \mathbf{X}'^{(i)}_{\bullet 2}\ \ldots\ \mathbf{X}'^{(i)}_{\bullet j}\right]\right), \quad j > 1 \tag{9}$$

where *pv* is a function that extracts the principal vector from its argument. $\mathbf{X}'^{(i)}$ is the standardized sample $\mathbf{X}^{(i)}$, i.e., with zero mean and unit variance.

A single sample may originate multiple feature vectors depending on the timestep $j$. We used the set $J^{(i)} = \left\{x:\ 2 \le x \le n^{(i)} \wedge x \in \mathbb{N}\right\}$ for training and validation of the classifiers, where $n^{(i)}$ is the sample length. To simplify the display of results, we tested with a subset $J^{(i)} = \{\lceil 0.25n \rceil, \lceil 0.5n \rceil, \lceil 0.75n \rceil, \lceil 1.0n \rceil\}$, where $\lceil\rceil$ represents the ceiling function. Simply put, this means that we are testing the features sets extracted from the samples starting at the first timestep and ending at 25%, 50%, 75% and 100% of gesture length.

The final step of feature extraction for all features sets is feature scaling, i.e., the standardization of the features as described for SGs.

TABLE I
TRAINING AND INFERENCE TIMES OF SEVERAL CLASSIFIERS, ACCURACY ON THE TRAIN, VALIDATION AND TEST DATA SUBSETS FOR SGS. THE TEST SCORES ARE DIVIDED INTO THE SCORES OF THE TRAINED AND UNTRAINED USERS (OTHER).

| Model | Time (s) | | Accuracy (%) | | |
|---|---|---|---|---|---|
| | Train | Test | Train | Validation | Test (other) |
| ANN | 127.0 | 0.1 | 97.9 | 94.2 | 94.6 (87.9) |
| QDA | 0.0 | 0.0 | 99.6 | 91.7 | 94.9 (66.7) |
| RBF SVM | 0.1 | 0.2 | 98.0 | 94.2 | 95.2 (83.3) |
| Gaussian Process | 23.8 | 13.1 | 99.8 | 991 | 94.9 (69.7) |
| KNN | 0.0 | 4.2 | 96.0 | 89.2 | 93.9 (53.0) |
| Naive Bayes | 0.0 | 0.0 | 93.4 | 88.3 | 91.2 (69.7) |
| Random Forest | 0.1 | 0.0 | 99.8 | 94.7 | 95.6 (92.4) |

## III. RESULTS AND DISCUSSION

The accuracy of the classifier models on both SG and DG data was obtained considering a segmentation accuracy estimated to be about 98%. The error is mostly oversegmentation, i.e., pauses in the middle of a DG where the subject slows down or hesitates. In this scenario, the classification of the DGs is more likely to fail due to lack of gesture data.

### A. Static Gestures

The accuracy of several classifiers was evaluated on the UC2017 dataset. The objective is to compare the performance of ANNs with other machine learning models: K-Nearest Neighbors (KNN), Support Vector Machines with a Radial Basis Function kernel (RBF SVM), Gaussian Processes (GP), Random Forests (RF), Gaussian Naive Bayes (NB) and Quadratic Discriminant Analysis (QDA). The training time, inference time and accuracy of the models are measured and evaluated as performance parameters.

The ANN used, implemented with Keras, is a feed-forward neural network (FFNN). Its architecture and hyperparameters were optimized on the validation dataset using random grid search. Grid search and manual search are the most common techniques for hyperparameter optimization [23]. Grid search generates candidates from a grid of parameter values in which every possible combination of values is tested to optimize hyperparameters (detailed parameters and Python code available in supplementary material). The optimal network has two dense hidden layers of 200 neurons each. Between these layers, there is a Gaussian noise layer with $\sigma = 0.6$. The transfer functions of the dense layers are linear and rectified. A final layer implements the *softmax* function to obtain the

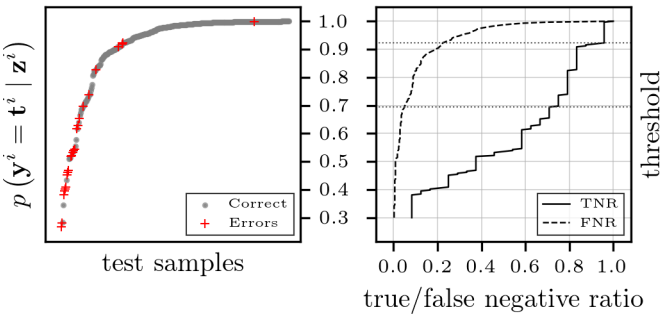


Fig. 4. On the left, sorted activation values of the winning class for each sample of the SG test set. The red crosses correspond to classification errors. On the right, the true and false negative ratios (TNR/FNR) when we apply a threshold to discard errors. The horizontal dotted lines correspond to the 0.696 and 0.923 thresholds.

probability distribution over the target classes. For weight regularization, we used the L2 distance with a factor of 0.005 and a weight decay coefficient of $10^{-7}$. The optimization was done using Stochastic Gradient Descent (SGD) with batches of 32 and a learning rate of 0.001. Furthermore, in order to prevent overfitting, we used early stopping when there is a minimum on the validation loss with a tolerance of 10 epochs. The hyperparameters of the remaining classifiers were optimized using manual search (detailed parameters and Python code available in supplementary material).

The performance of the trained classifiers is shown in table I. The best performance on the test set was 95.6% on the trained users (92.4% on the untrained), obtained with the RF. The ANN was slightly worse, with 94.6% and 87.9% accuracy on the trained and untrained users, respectively, leading to the conclusion that in this case, the RF is generalizing better to new users than the ANN. The next best performance was the SVM, with 95.2% accuracy. The other models performed very well on the trained users, but clearly overfitted the dataset, since their accuracy on untrained users is below 70%.

The accuracy of the ANN model for each individual subject varies between 87.9% (untrained user) and 100.0%, while one of the trained users reached only 88.2%. This subject was one of the authors and was involved in the definition of the gesture library, which may have originated samples that are significantly different from those of the other users. The distribution of errors per class was nearly random, with no gestures being mixed consistently.

We also present the test results of the feasibility of removing poorly classified gestures by their score, Fig. 4. This analysis was done with the ANN classifier, since it outputs a probability distribution over the classes. We present the score of the winning class $p(\mathbf{y}^i = \mathbf{t}^i \mid \mathbf{z}^i)$, i.e., the probability of the ANN output $\mathbf{y}^i$ being the expected target $\mathbf{t}^i$ given the feature vector $\mathbf{z}^i$. It is impossible to define a score threshold to exclude misclassifications without excluding also some good ones. The trade-off is demonstrated in Fig. 4 on the right, via the true and false negative ratio. If we agree that a 5% False Negative Ratio (FNR) is acceptable, the threshold 0.696 reduces miss-classifications by 71%. On the other hand, if we want 95% of miss-classifications to be discarded, we lose 22% of valid classifications with a score threshold of 0.923. The latter is not a particularly good trade-off. In this context, another solution should be found, such as generating false samples so that the classifier learns how to better separate them.

TABLE II
FEATURE SETS CONSIDERED FOR THE EXPERIMENTS. CI REFERS TO CUBIC INTERPOLATION, PV TO PRINCIPAL VECTORS AND RAW TO NO FEATURE EXTRACTION.

| Feature set | Description |
|---|---|
| CI-FULL | CI applied to raw preprocessed data describing a full DG sample |
| PV-FULL | PVs applied to raw preprocessed data describing a full DG sample |
| PV-TS | PVs applied sequentially to a DG sample, starting from its first frame to an arbitrary timestep. |
| RAW-LSTM | Raw preprocessed data classified by LSTMs |
| RAW-CNN | Raw preprocessed data classified by CNNs |

TABLE III
CLASSIFICATION ACCURACY FOR THE FULL DG EXPERIMENTS. THE TEST SCORES ARE DIVIDED INTO THE SCORES OF THE TRAINED AND UNTRAINED (OTHER) USERS.

| | CI-FULL | | | PV-FULL | | |
|---|---|---|---|---|---|---|
| Model | Train | Val | Test (other) | Train | Val | Test (other) |
| ANN | 100.0 | 98.5 | 99.3 (96.2) | 99.7 | 91.3 | 94.4 (66.0) |
| LDA | 100.0 | 98.0 | 97.2 (92.5) | 67.2 | 60.7 | 68.8 (50.9) |
| KNN | 97.9 | 94.4 | 96.5 (86.8) | 90.0 | 81.1 | 84.7 (62.3) |
| RF | 100.0 | 98.0 | 97.2 (86.8) | 100.0 | 92.3 | 88.9 (67.9) |
| SVM | 99.8 | 98.5 | 97.9 (96.2) | 87.9 | 82.1 | 79.2 (60.4) |

### B. Dynamic Gestures

From the UC2017 dataset, four sets of data contemplating different features were considered for the experiments, Table II. CI-FULL and PV-FULL output a single classification per DG, while PV-TS, RAW-LSTM and RAW-CNN output a classification for each timestep of a DG.

The results for the experiments using **CI-FULL** features are shown in Table III. Multiple classifiers were tested and the results report the accuracy achieved by the best ones. The hyperparameters were chosen by manual search for all the classifiers (detailed parameters and Python code available in supplementary material). As an example, the ANN has two hidden layers of 100 and 200 nodes each, their activation function is linear and rectified, and the output is the *softmax* function. The weights were regularized using the L2 distance. For optimization, SGD was used with a batch size of 128 and a learning rate of 0.01.

The accuracy of the classifiers is generally excellent, around 97.0%, in the test set for trained users and up to 96.2% for untrained users. The KNN and RF classifiers did not generalize as well to new users, reaching an accuracy of just 86.8%. This is most likely explained by the size of the feature vector (560) and comparatively low number of samples. On the other hand, the SVM performed nearly as well as the ANN on untrained users (96.2%), but worse on trained users (99.3% vs 97.9%).

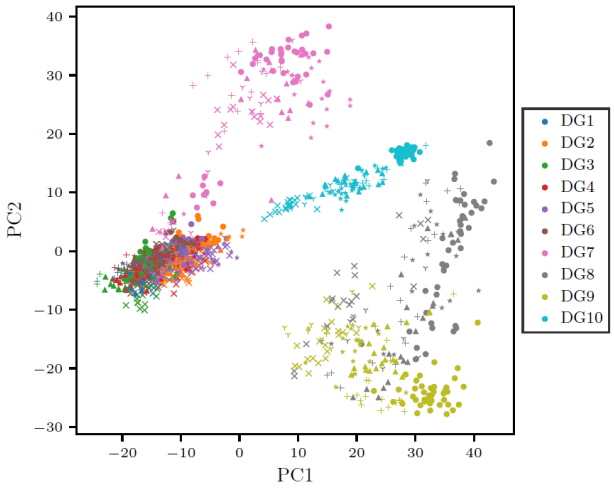


Fig. 5. DGs projected on the plane defined by the first two principal components of the entire dataset - training, validation and test data). The ten DG classes are represented by different colours and markers. From DG1 to DG6 we can observe a cluster and then for DG7, DG8, DG9 and DG10 we have other clusters. This is because the gestures from DG1 to DG6 are all performed with the hand open (no variations in finger angle data).

The results for the experiments using **PV-FULL** features are shown in Table III. Fig. 5 shows the ten DGs projected on the plane defined by the first two principal components of the data. We have tested the same classifiers as in CI-FULL, so that we can establish a comparison between the two feature sets. The accuracy is generally and markedly below of those obtained by CI-FULL. The ANN accuracy decreased by 4.9% for trained users, even with an updated architecture composed by two hidden layers with 300 units each. The generalization to untrained users was poor, with just 66.0% accuracy. The RF achieved slightly better results on the training and validation sets than the ANN, but the test accuracy was lower (88.9%). All of the other models performed significantly worse. As a conclusion, the PV features lose more information about the DGs than the CI features, making classification harder. However, the PV features can be calculated with an arbitrary number of frames of data, without the full gesture data, therefore allowing sequential (online) classification.

The results for the experiments using **PV-TS** features are shown in Table IV. For this case we present the results for the best performing models, a feed-forward ANN, KNN, RF and SVM. The classifiers' hyperparameters were optimized again by manual search, but they remained nearly the same as for PV-FULL, since the problem is similar. However, the ANN architecture was updated to two hidden layers of 512 and 256 nodes. There is also a $0.1\sigma$ Gaussian noise layer after each hidden layer and a dropout layer (50% rate) in between. These noise layers help the network generalize better at all timesteps, since there is now a feature vector for each timestep of each DG sample, which in turn originates much more variability in the features.

We present the training time for each model and the total inference time for the whole dataset. The accuracy results for each dataset split are presented and the test split was divided into trained and untrained users. According to our designation for incomplete data, the columns "0.25" correspond to the PV feature vectors calculated with the first 25% of timesteps of each DG sample. The same logic applies to "0.50" , "0.75" and "1.00". The column "1.00" corresponds to all of the DG timesteps being used, which is the same case as PV-FULL. However, in PV-TS we use precisely the same classification model for all timesteps.

The best classification accuracy on the test set was obtained with the ANN at all timesteps of the test set. The RF and SVM models reached nearly the same accuracy as the ANN, with 90% at the middle of the DGs and 89% accuracy at 75% of DG completion. At the end of the DGs (100% completion), the accuracy is lower for these models than for the ANN (81% vs 85%). The KNN model performed worse at most timesteps, reaching only 83.3%, 81.2% and 81.9% accuracy at 50%, 75% and 100% of DG completion, respectively. Interestingly, for all of the classifiers, the accuracy is better at 50% and 75% of DG completion than at 100%. This is likely due to the way most of the gestures of the library have a motion in one direction and then move back to the initial position. This would mean that the first half of the gesture is a better predictor of the whole gesture than the second half. For example, the second half of DG1 is very similar to the first half of DG2. To aid in visualization, the features are shown in a 2D PC space (for the test set) in Fig. 6. There is considerable noise at $J_{0.25}$, which is unfavorable for classification. However, after that, we see stable clusters, such as classes 8 and 10. On the other hand, class 9 has many samples flipping their position at $J_{1.00}$, which may help explain the drop in accuracy.

**RAW-LSTM** experiments use a LSTM recurrent neural network that is composed of cells that have memory which can be kept or forgotten over-time, depending on the sequence of input data. The LSTM structure and hyperparameters were obtained by manual search. There is a densely connected layer with 512 units and a $0.4\sigma$ Gaussian noise layer, which is followed by a LSTM layer of 256 cells. After that we implement a 50% dropout layer. The output layer has a *softmax* activation function. All the other layers have the hyperbolic tangent as transfer function. Additionally, we increase the weight of the timesteps after 50% of sample completion, so that the model optimizes accuracy at later stages of the gesture. Detailed parameters and Python code is available in supplementary material. In terms of accuracy on the trained users of the test split, the LSTM outperforms all the other models at 50%, 75% and 100% of gesture completion, with 95.1%, 97.2% and 96.5% accuracy, respectively. In respect to generalization to new users, the LSTM is significantly better than all other models when 75% or more data is available. At 75% and 100%, the accuracy on new users is 87.3% and 89.1%, respectively. It compares favorably to the second best performer (ANN) at the cost of a significant increase in training and inference times, due to the large number of parameters of the LSTM model.

TABLE IV
DG CLASSIFICATION SEQUENTIAL ACCURACY FOR THE TIME-SERIES BASED EXPERIMENTS PV-TS, RAW-CNN AND RAW-LSTM AT 25, 50, 75 AND 100% OF DG COMPLETION. THE TEST SCORES ARE DIVIDED INTO THE SCORES OF THE TRAINED AND UNTRAINED (OTHER) USERS.

| Model | Time (s) | | Train accuracy (%) | | | | Validation accuracy (%) | | | | Test accuracy (%) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Train | Test | 0.25 | 0.50 | 0.75 | 1.00 | 0.25 | 0.50 | 0.75 | 1.00 | 0.25 | 0.50 | 0.75 | 1.00 |
| ANN | 94.7 | 0.2 | 95.3 | 96.9 | 95.4 | 91.1 | 74.5 | 86.7 | 88.3 | 85.2 | 77.8 (49.1) | 91.0 (75.5) | 88.9 (71.7) | 84.7 (62.3) |
| KNN | 25.6 | 3.4 | 99.9 | 99.9 | 100.0 | 99.5 | 67.9 | 82.1 | 83.2 | 78.6 | 70.8 (43.4) | 83.3 (69.8) | 81.2 (69.8) | 81.9 (54.7) |
| RF | 54.6 | 0.2 | 100 | 100.0 | 100.0 | 100.0 | 74.5 | 88.8 | 91.3 | 86.7 | 70.1 (54.7) | 88.9 (75.5) | 87.5 (67.9) | 80.6 (67.9) |
| SVM | 102.4 | 8.8 | 97.8 | 97.8 | 97.1 | 93.1 | 73.0 | 83.7 | 87.2 | 83.2 | 75.7 (41.5) | 89.6 (71.7) | 88.9 (69.8) | 81.2 (60.4) |
| LSTM | 390.5 | 69.5 | 87.9 | 96.8 | 99.8 | 99.8 | 78.1 | 92.9 | 97.4 | 98.5 | 81.7 (50.9) | 95.1 (74.5) | 97.2 (87.3) | 96.5 (89.1) |
| CNN | 66.4 | 2.7 | 86.6 | 96.5 | 97.1 | 88.7 | 81.1 | 92.9 | 97.1 | 88.7 | 84.7 (60.4) | 94.4 (84.9) | 92.4 (73.6) | 81.9 (54.7) |

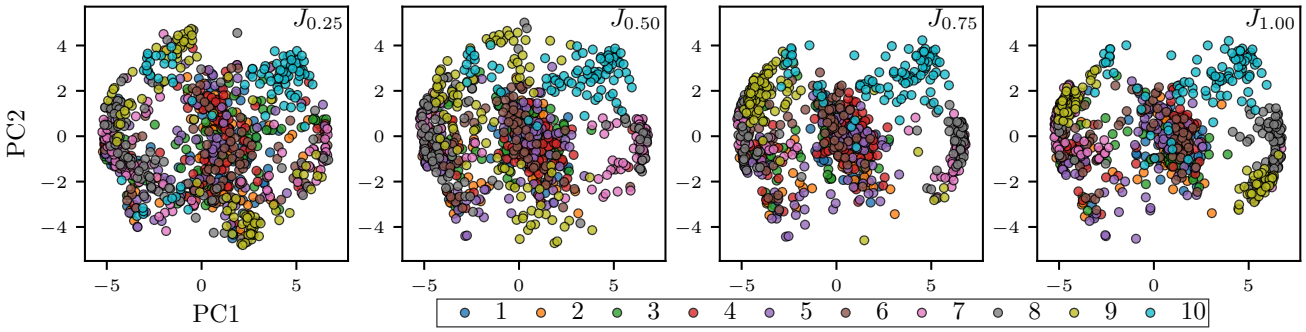


Fig. 6. Plots of the features of the DG training set, with 25%, 50%, 75% and 100% of the data, in a reduced 2D principal component space. Each color represents a different class. Results for each classifier are detailed in table IV.

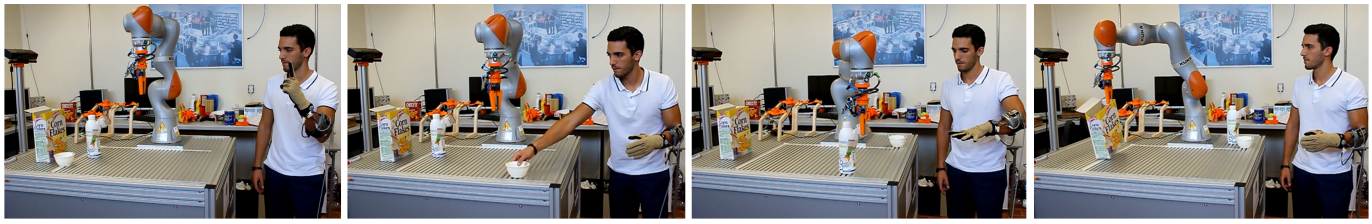

Fig. 7. The human collaborates with the robot to prepare the breakfast meal. The video is available in supplementary material.

We performed the **RAW-CNN** experiment in the same conditions as the previous RAW-LSTM. The CNN used had an initial dense hidden layer with 512 nodes and a *tanh* transfer function. Afterwards, there are two convolutional (1D) layers with 100 filters each, windows of 5 timesteps and rectified linear units as transfer function. Each convolutional layer is followed by Gaussian noise layers of strength $0.2\sigma$. The model is trained by SGD with a learning rate of 0.001. While oftentimes a CNN can be used to model sequential data with performance similar to that of a LSTM, in this case it was worse in accuracy at the later stages of the DGs, and also in terms of generalization to new users. Considering 50% of gesture data, the accuracy on trained users (94.4%) is close to the LSTM's 95.1%. The CNN also generalizes better at that timestep than the LSTM (84.9% vs 74.5%). However, for 75% of gesture completion, it is significantly worse than the LSTM for all users, and at 100% it is worse than the feed-forward ANN with just 81.9% and 54.7% accuracy for trained and untrained users, respectively. A decrease of this magnitude was unexpected and it is possibly due to the data padding that occurs during the convolution operations. The last timestep of a DG is evaluated by a convolution operation on a window centered on that timestep. However, since the window has length 5, that means that 2 timesteps correspond to CNN padding, which is generated, unreliable data, leading to the low score.

The LSTM and CNN approaches have the advantage that the raw sensor data can be fed directly into the model after scaling, unlike all of the others, which require carefully chosen feature extraction methods. For LSTMs, the disadvantage is that the training and inference time are one to two orders of magnitude higher than the other classifiers, due to model complexity. The latter is more concerning because classification must be done online for an efficient human-robot interactive process. The experimental setup demonstrated that the inference time per frame for the LSTM model is about 0.16 ms (about 6300 Hz) on a GPU, so it could be an issue for its implementation in embedded systems.

### C. Robot Interface

We have implemented a gesture-based human-robot interface composed by gestures from the UC2017 dataset. Since there are delays in data acquisition, data stream segmentation, candidate sample preprocessing, classification, decision making and robot communication, we estimate that the total delay between the end of a gesture performed by the human and the robot reaction is about 300 ms.

The collaborative robot is a 7 DOF KUKA iiwa equipped with the Sunrise controller and interfaced using the KUKA Sunrise Toolbox for MATLAB [24]. The attempted collaborative robotic task consists in preparing a breakfast meal composed by subtasks such as grasping a cereal box, a yogurt bottle, and pouring the contents into a bowl, Fig. 7. These tasks were performed by direct robot teleoperation, being the robot actions controlled online by the human gestures and the collaborative process managed by a collaborative robot task manager [9]. The task manager can be setup with a number of required validations so that when a gesture is wrongly classified the system actuates to avoid any potential danger for the human and/or the equipment. From the library of 24 SGs and 10 DGs, three subjects were taught the mapping between gestures and specific robot commands, such as: stop motion, move along X, Y or Z in Cartesian space, rotate the robot end-effector in turn of X, Y or Z, open/close the gripper, and teleoperate the robot in joystick mode. Anytime the user is not performing a given gesture the system is paused.

All users indicated that the interaction process is very natural, since they can easily select the desired operation modes and the system is intuitive to use. During the interactive process, the reached target points can be saved and used in future robot operations. The impedance controlled robot compensates positioning inaccuracies, i.e., the users can physically interact with the robot to adjust positioning. Concerning safety, the subjects indicated that they feel safe in interacting with this robot due to the fact that the KUKA iiwa is a sensitive robot that is able to stop its motion when a pre-defined contact force is reached.

## IV. Conclusion and Future Work

This paper presented an online static and dynamic gesture recognition framework for HRI. Experimental results using the UC2017 dataset showed a relatively high classification accuracy on SGs without feature extraction. For DGs, the use of CI features resulted in high offline classification accuracy using a regular ANN model. The achieved accuracy compares favourably with standard classifiers and with deep learning LSTM on online classification with PCA features. The LSTM uses scaled raw data, therefore being more easily extendable to new datasets. Nevertheless, The LSTM degrades when we compare the training and inference time, which is critical for online implementation. The sequential classification of DGs with either PCA features or raw data showed an accuracy that is higher with partial gesture data (50% or 75% of the initial frames of a DG sample) than with the full DG data. In this context, DGs can be accurately classified in anticipation, even before the user finishes the gesture in real world, thus allowing faster and more efficient gesture-based control of a robot by cutting the processing time overheads.

The human-robot interactive process demonstrated that is feasible to associate the recognized gesture patterns to robot commands and by this way teleoperate the collaborative robot in an intuitive fashion.

Future work will be dedicated to testing the proposed solution with other interaction technologies (vision, IMUs, and EMG). The promising results obtained with the classification from incomplete data will be explored as a way to anticipate robot reaction to human commands.

## References


[1] S. Sheikholeslami, A. Moon, and E. A. Croft, "Cooperative gestures for industry: Exploring the efficacy of robot hand configurations in expression of instructional gestures for human-robot interaction," *The International Journal of Robotics Research*, vol. 36, no. 5-7, pp. 699–720, 2017.

[2] M. Simão, P. Neto, and O. Gibaru, "Natural control of an industrial robot using hand gesture recognition with neural networks," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Oct 2016, pp. 5322–5327.

[3] X. Yuan, Y. Wang, C. Yang, Z. Ge, Z. Song, and W. Gui, "Weighted linear dynamic system for feature representation and soft sensor application in nonlinear dynamic industrial processes," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 2, pp. 1508–1517, Feb 2018.

[4] 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.

[5] M. Burke and J. Lasenby, "Pantomimic gestures for human–robot interaction," *IEEE Trans Robotics*, vol. 31, no. 5, pp. 1225–1237, 2015.

[6] S. Waldherr, R. Romero, and S. Thrun, "A Gesture Based Interface for Human-Robot Interaction," *Autonomous Robots*, vol. 9, no. 2, pp. 151–173, sep 2000.

[7] E. Zahedi, J. Dargahi, M. Kia, and M. Zadeh, "Gesture-based adaptive haptic guidance: A comparison of discriminative and generative modeling approaches," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1015–1022, April 2017.

[8] T. Ende, S. Haddadin, S. Parusel, T. Wüsthoff, M. Hassenzahl, and A. Albu-Schäffer, "A human-centered approach to robot gesture based communication within collaborative working processes." IROS 2011, 25-30 Sept. 2011, San Francisco, California.

[9] N. Mendes, M. Safeea, and P. Neto, "Flexible programming and orchestration of collaborative robotic manufacturing systems," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, July 2018, pp. 913–918.

[10] M. A. Simão, P. Neto, and O. Gibaru, "Unsupervised gesture segmentation by motion detection of a real-time data stream," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 473–481, April 2017.

[11] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, "A unified framework for gesture recognition and spatiotemporal gesture segmentation." *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 9, pp. 1685–99, sep 2009.

[12] M. Simão, P. Neto, and O. Gibaru, "Unsupervised gesture segmentation of a real-time data stream in matlab," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Oct 2016, pp. 809–814.

[13] B. Burger, I. Ferrané, F. Lerasle, and G. Infantes, "Two-handed gesture recognition and fusion with speech to command a robot," *Autonomous Robots*, vol. 32, no. 2, pp. 129–147, dec 2011.

[14] D. Wu, L. Pigou, P. J. Kindermans, N. LE, L. Shao, J. Dambre, and J. M. Odobez, "Deep dynamic neural networks for multimodal gesture segmentation and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2016.

[15] F. J. Ordonez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, 2016.

[16] 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.

[17] M. Field, D. Stirling, Z. Pan, M. Ros, and F. Naghdy, "Recognizing human motions through mixture modeling of inertial data," *Pattern Recognition*, vol. 48, no. 8, pp. 2394 – 2406, 2015.
[18] J. Chen, M. Kim, Y. Wang, and Q. Ji, "Switching gaussian process dynamic models for simultaneous composite motion tracking and recognition," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 2655–2662.
[19] C. A. Torres-Valencia, H. F. García, G. A. Holguín, M. A. Álvarez, and Á. Orozco, "Dynamic hand gesture recognition using generalized time warping and deep belief networks," in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, I. Pavlidis, R. Feris, T. McGraw, M. Elendt, R. Kopper, E. Ragan, Z. Ye, and G. Weber, Eds. Cham: Springer International Publishing, 2015, pp. 682–691.
[20] X. Yuan, B. Huang, Y. Wang, C. Yang, and W. Gui, "Deep learning-based feature representation and its application for soft sensor modeling with variable-wise weighted sae," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3235–3243, July 2018.
[21] C. Monnier, S. German, and A. Ost, *A Multi-scale Boosted Detector for Efficient and Robust Gesture Recognition*. Springer International Publishing, 2015, pp. 491–502.
[22] K. Mei, J. Zhang, G. Li, B. Xi, N. Zheng, and J. Fan, "Training more discriminative multi-class classifiers for hand detection," *Pattern Recognition*, vol. 48, no. 3, pp. 785 – 797, 2015.
[23] G. E. Hinton, "A practical guide to training restricted boltzmann machines." in *Neural Networks: Tricks of the Trade (2nd ed.)*, ser. Lecture Notes in Computer Science, G. Montavon, G. B. Orr, and K.-R. MÃŒller, Eds. Springer, 2012, pp. 599–619.
[24] M. Safeea and P. Neto, "Kuka sunrise toolbox: Interfacing collaborative robots with matlab," *IEEE Robotics Automation Magazine*, pp. 1–1, 2018.

View publication stats