A General Pipeline for Online Gesture Recognition in Human–Robot Interaction

Valeria Villani, *Member, IEEE*, Cristian Secchi, *Senior Member, IEEE*, Marco Lippi, and Lorenzo Sabattini, *Senior Member, IEEE*

***Abstract*—Recent advances in robotics have allowed the introduction of robots assisting and working together with human subjects. To promote their use and diffusion, intuitive and user-friendly interaction means should be adopted. In particular, gestures have become an established way to interact with robots since they allow to command them in an intuitive manner. In this article, we focus on the problem of gesture recognition in human–robot interaction (HRI). While this problem has been largely studied in the literature, it poses specific constraints when applied to HRI. We propose a framework consisting in a pipeline devised to take into account these specific constraints. We implement the proposed pipeline considering, as an example, an evaluation use case. To this end, we consider standard machine learning algorithms for the classification stage and evaluate their performance considering different performance metrics for a thorough assessment.**

***Index Terms*—Classification algorithms, gesture recognition, human-robot interaction.**

I. Introduction

OVER the last years, progress in the design and development of robotic systems has led to advanced solutions that are entering our daily lives in several application fields, such as social assistance, surveillance, tour guidance, rehabilitation, and search and rescue. From the point of view of human–robot interaction (HRI), in addition to the requirements for safe interaction, this trend has further implications on the modalities used to communicate with robots. First, it is important that communication from and to the robot is quick, smooth, colocated and requires low attentional demand. In other words, for efficient HRI, communicating to the robot and understanding its messages should not be a bottleneck and should not increase the overall complexity of teaming up with a robotic agent, as it is usually in human–human communication. Moreover, given the fact that most robotic assistants are intended to be used by people without expertise in robotics, it is also important that intuitive communication is enabled, so that users do not have to learn commands that are specific for the robot. In this regard, gestures represent a valid candidate for intuitive communication with robots.

Gesture-based interaction with machines or robots has been proposed by a large body of the literature, in both everyday and industrial applications [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]. The problem of gesture-based interaction can be split in two specific problems: recognizing gestures and mapping gestures into commands to the interaction system. The focus of our article is on the first specific problem applied to the domain of robotics, that is, gesture recognition in the context of HRI. While gesture recognition has long been an active area of research in computer vision and machine learning [11], the use of gestures for interaction with robots poses specific constraints. These constraints define the prerequisites for successful gesture-based HRI. To improve the use of gestures for HRI it is, then, beneficial that approaches to gesture recognition are compliant by design with such constraints. The second specific problem refers to designing how gestures can be translated into commands to the robot, possibly in an intuitive manner [2]. While these two specific problems are independent, they jointly address the general problem of gesture-based interaction with robots. Building upon these lines, in this article, we propose a pipeline for online gesture recognition in HRI. The approach has been designed to be general and can be applied to any set of gestures for interacting with any robot. To the best of our knowledge, this is the first attempt in this direction, since existing approaches have been designed to address specific use case scenarios. Conversely, we design the proposed pipeline starting from the understanding of how gesture-based interaction, in general, can be applied to HRI; no additional constraints or requirements derived by specific use cases are included to design the pipeline. As a result, the proposed pipeline is application agnostic.

To achieve this, in Section III-A we first discuss the specific constraints of gesture recognition for HRI, which have to be taken into account to use gesture-based interaction in daily life. Then, having these constraints in mind, in Section III-B, we propose an algorithmic pipeline to implement gesture recognition in HRI. To show and discuss the application of the proposed pipeline, we consider the experimental scenario presented in [2], [12], and [13]. In particular, we assess the capability of the pipeline to generalize across multiple subjects having different level of acquaintance with the use of gestures. Indeed, prospectively, the use of gestures for interacting with robots should be designed in such an intuitive way that it is available

Manuscript received 28 September 2022; accepted 3 December 2022. Date of publication 18 January 2023; date of current version 15 March 2023. This article was recommended by Associate Editor Y. Liu. *(Corresponding author: Valeria Villani.)*

This work involved human subjects or animals in its research. The author(s) confirm(s) that all human/animal subject research procedures and protocols are exempt from review board approval.

The authors are with the Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, 41121 Reggio Emilia, Italy (e-mail: valeria.villani@unimore.it; cristian.secchi@unimore.it; marco.lippi@unimore.it; lorenzo.sabattini@unimore.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/THMS.2022.3227309.

Digital Object Identifier 10.1109/THMS.2022.3227309

to any user, and does not require specific prior experience with gestures.

The rest of this article is organized as follows. Section II reports the state-of-the-art on gesture recognition in HRI. In Section III, we discuss how HRI constrains the use of gesture-based interaction and, building upon such discussion, we present the proposed pipeline. Then, Section IV focuses on the classification stage, which is the core of the proposed approach. In Section V, the experimental setting considered to validate the proposed architecture is presented. Then, in Section VI, we report the results of implementing the proposed pipeline to the considered use case. Finally, Section VII concludes this article.

II. Gesture Recognition for HRI

A large part of existing approaches to gesture recognition in the domain of HRI relies on the use of vision systems [14], [15]. In [5], a stereo camera is mounted on the head of a personal service robot for elderly assistance. Whole-body gestures, such as walking or raising a hand, are then recognized from estimated 3-D human body components. Burke and Lasenby [6] proposed the use of a Kinect sensor to detect pantomimic hand gestures that control an unmanned aerial vehicle (UAV). Xu et al. [8] focused on the problem of background subtraction when using RGB-D cameras for hand gesture detection in home-like dynamic environment. Kim et al. [7] proposed a vision-based gesture recognition to address the problem of HRI at a long distance, approximately 5 m from the camera. Sigalas et al. [9] have considered to classify the arm trajectories, seen as sequences of motor primitives. To this end, RGB video sequences are used, with the subject standing in front of the robot in a constrained setup. The same requirement, which is the subject standing in front of the robot, is set by Cicirelli et al. [10], despite the use of multiple Kinect cameras to monitor the surrounding environment. Chandarana et al. [16] used the infrared-based leap motion controller to detect hand gestures for teleoperating UAVs. UAV trajectories are built combining gestures that define portions of flight paths. Generally speaking, vision-based approaches require proper lighting conditions and camera angles, and the user has to be in the field of view of the camera. Hence, these characteristics pose a limitation to the use of such approaches in real-world HRI applications.

To overcome the limitations linked to the use of vision systems and given the recent advances in pervasive computing, wearable devices have been used to detect and recognize gestures [17]. Indeed, unobtrusive, wireless, and inexpensive body worn sensors, such as accelerometers and gyroscopes, possibly integrated in inertial measurement units (IMUs), are available on everyday mobile and ubiquitous systems, such as smartwatches, wrist bands, and mobile phones. They provide information about body movements and, hence, can be used to track user activity [18], [19]. In addition, they have been used for gesture recognition for human–machine interaction in several works [3], [20], [21]. For what concerns gesture-based interaction with robots, Neto et al. [4] proposed the use of five IMUs and an ultra-wideband positioning system to capture the human upper body shape and the relative position between the human and the robot. Villani et al. [2], [12] used inertial data recorded with a smartwatch to control both wheeled and aerial robots. Gestures are used to provide high-level commands, such as take off, land, or stop, whereas robot velocity is determined by mapping user's wrist movements. A similar setting was considered by Carfì et al. [22]. However, their framework was not designed for HRI. Indeed, gestures were executed in steady conditions, which means that the user was in a fixed predetermined pose between consecutive gestures. This condition limits the possibility to use this interaction means while the user is performing everyday activities. In addition, intersubject robustness of the classification approach was assessed to a limited extent since it was tested with gestures performed by the same subjects involved in the training phase.

Furthermore, wearable sensors based on surface electromyography (sEMG) are being used for gesture recognition [23], [24], [25]. Quite often, IMU and EMG data are combined together to improve gesture recognition. Indeed, while inertial data provide information about hand position, EMG sensors allow to fully understand complex finger or hand gestures. This is the case, for example, of the work by Jiang et al. [26], where sEMG and IMU sensing fusion allows to recognize several air and surface gestures with two distinct force levels. Georgi et al. [27] proposed the simultaneous usage of IMU and EMG sensors for gesture-based interfaces and Hidden Markov Models are used as classifiers to discriminate between the defined gesture classes.

From an algorithmic point of view, most approaches to gesture recognition resort to machine learning techniques to deal with high-dimensional, multimodal streams of data that are characterized by a large variability. Different machine learning algorithms applied to gesture recognition have been compared in [24], [28], and [29]. Comparisons by Trigueiros et al. [28] and Wahid et al. [24] focused on hand gestures, as most of the approaches proposed in the literature (e.g., [20], [21], [23], and [27]), whereas realistic daily life activities are considered by Sagha et al. [29].

The abovementioned approaches consider the use of gestures for HRI in specific case studies and cannot be easily scaled to other applications. Our aim in this article is to address the problem of gesture-based HRI from a general perspective, identifying the specific constraints of this application domain and proposing a pipeline that can be applied to any case study, with some fine tuning.

III. Proposed Architecture

We hereby describe the proposed pipeline for online gesture recognition in HRI. It addresses several constraints specific for gesture-based interaction with robots. A diagram representing the proposed architecture is depicted in Fig. 1.

A. Specific Constraints of Gesture Recognition in HRI

As introduced in Section I, when dealing with gesture recognition in HRI, it is important that using gestures does not limit the user interacting with the robot, insofar gestures are not perceived as a slowdown for the interaction. As a result, the use of gestures in the context of HRI poses the following specific constraints for the problem of gesture recognition.

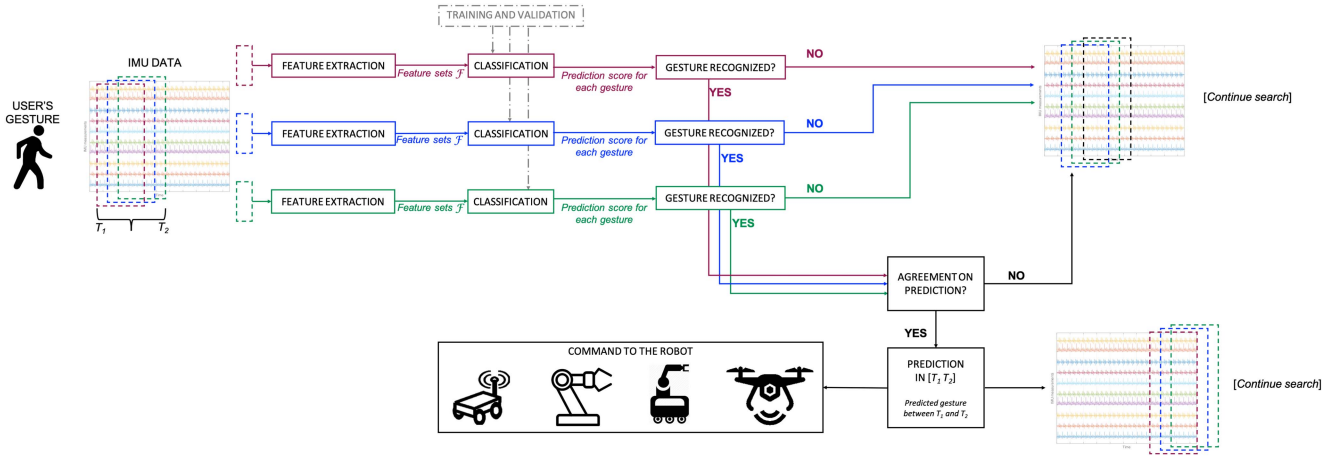

Fig. 1. Flowchart for the proposed pipeline for online gesture detection and classification in HRI. The proposed pipeline can be implemented in ROS, which can deal with data acquisition, data processing, gesture recognition, and communication with the robotic platform.

1) Gesture-based interaction should rely on a lean infrastructure that requires poor or no installation and does not limit the user's freedom to move around and/or with the robot.
2) Gesture recognition should be performed online and not introduce any perceivable delay between the execution of a gesture and its effect, meant as command to the robot. In other words, the algorithmic pipeline in charge of gesture recognition should be fast enough, requiring limited computational burden. Moreover, the architecture should guarantee immediate communication of a detected gesture to the robot.
3) It should be possible to detect gestures online and in dynamic conditions, while users are performing other ordinary activities and movements.
4) The system should be easily adapted to different users, thus requiring robust generalization capabilities.
5) While seeking to improve gesture recognition performance, it is particularly important to minimize the rate of false positives since they would initiate an unintended communication towards the robot.

It is noteworthy that some of these constraints are general for gesture-based interaction and apply also to other domains, as discussed, for example, in [14] and [30]. Nevertheless, as regards HRI, they define the prerequisites for successful gesture-based HRI. These constraints are summarized in the left column of Table I, whereas the right column describes how the proposed system tackles them. More details are given in the following sections.

TABLE I
SPECIFIC CONSTRAINTS OF GESTURE RECOGNITION IN HRI AND PROPOSED APPROACHES TO COMPLY WITH THEM

Specific constraints in HRI	Proposed solution
C1 – Wearable and lean infrastructure	Smartwatch or other wrist-worn device embedding an IMU
C2 – Online gesture recognition	Choice of efficient classification stage
C3 – Online application in dynamic conditions (e.g., moving user)	Sliding windows and specific decision rules
C4 – Robustness to inter-subject variability	Different user's confidence with the gestures under analysis
C5 – Minimization of false positives	Model selection with specific performance metrics

B. Overview of the Proposed Pipeline

To track the movements of the user, we consider inertial data recorded by a wrist-worn device, and thus focus on forearm gestures. This allows to comply with constraint *C1* since wearable devices recording inertial data are not cumbersome to wear and are easily available on the market. Inertial data are then analyzed, by considering sliding windows of fixed length, with one sample shift. Recorded data are continuously processed and the occurrence of a gesture is continuously verified, thus complying with *C3*. In particular, to search for gestures, for each sliding window a set of statistical features are computed for each of the measured signals. The extracted features represent the input for the classification stage. In this article, we start considering different classical classification algorithms, such as K-nearest neighbors (KNN), support vector machines (SVMs), random forests (RF), and neural networks (NNs). Among them, we then select the algorithm that best complies with the specific constraints for gesture-based HRI, in a validation use case. To select the most appropriate algorithm for the considered application domain, different performance metrics are considered, tailoring the severity of false positives in HRI, so as to comply with *C5*. Moreover, starting from the results achieved in our case study, the determination of the classification stage is driven by considerations related to the need for intersubject robustness and computational burden, thus complying with *C2* and *C4*. The right column of Table I summarizes how the proposed pipeline complies with the specific constraints for gesture-based HRI.

With reference to Fig. 1, all the considered algorithms share the same general structure. They require previous training and validation with respect to the specific gestures of the considered use case. As output, each algorithm provides a score referred to the probability that one of the considered gestures occurred in the current sliding window. A threshold is set on such scores; the threshold can be tuned in order to balance between false positives

and false negatives. If the confidence of the classifier is below such threshold, then the process moves to the following sliding window. Otherwise, in case the confidence of the classifier is above the threshold for at least one gesture, a decision is not taken immediately, but the system checks for the same condition also in the subsequent sliding windows. If the condition is met, then a decision is taken and the corresponding command is sent to the robot. The rationale behind this behavior is to improve the robustness of the detection phase. Since consecutive sliding windows are shifted by one sample, it is likely that a gesture occupies more than one window, depending on the sampling frequency of the recording device. Hence, it is likely that when a gesture is performed, it is recognized by the classification algorithm in several consecutive windows, whose number depends on the duration of the gesture and the device sampling frequency. On the other side, if a gesture is detected in a window only, it is likely to be a false positives and should be discarded.

IV. Classification Stage

The classification stage of the proposed architecture consists in a machine learning module that receives as input the statistical features computed on the current sliding window and provides as output a decision whether a gesture is recognized or not. In this section, we overview the general setting of supervised machine learning, and we briefly describe four classic algorithms that have been tested in our experimental evaluation.

Broadly speaking, in a classification task, the goal is to predict the category $y \in \mathcal{Y}$, sometimes also named class, label, or *output*, of a given observed example $x \in \mathcal{X}$, which instead represents the *input* of the system. In a supervised setting, we are given a collection $\mathcal{D}$ of m input/output pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{m}$ and we aim to fit a function $f : \mathcal{X} \to \mathcal{Y}$ so that we can then predict the category $\hat{y}$ of a given, novel (not previously seen) example $\hat{x}$. Clearly, different machine learning approaches exploit different definitions for function f, as well as different algorithms and techniques to learn such a function. To identify the best approach for the proposed HRI pipeline, we have compared classic machine learning approaches, namely KNN, SVMs, RF, and NNs. However, other different classification algorithms can be considered. In all these cases, we have considered a setup where each input instance x is described by a vector of real numbers, which are the characteristics, or *features* of that instance.

The parameters chosen to implement each approach are discussed in Section V-C, with reference to the considered datasets.

A. K-Nearest Neighbors

Based on the concept of distance between examples, the KNN classifier is not properly a learning algorithm. In fact, given a test example $\hat{x}$ to be classified, the algorithm looks for the K examples in the training set that are the closest ones to $\hat{x}$ according to a chosen metric (e.g., the Euclidean distance). The prediction is then performed via a majority voting procedure among the classes of the KNN. Although very simple, this algorithm can work well in practice, when the distance computed on the feature vectors is highly discriminative of the target class. The parameter K defines the size of the neighborhood to consider for classification.

B. Support Vector Machines

SVMs are another classic method for supervised classification in machine learning. Considering binary classification, SVMs are trained to learn a discriminative function that best separates positive and negative examples, with the maximum possible margin [31]. Given a collection of m training samples, such a discriminative function is computed as

$$\mathfrak{f}(x) = \sum_{i=1}^{m} \alpha_i \mathcal{K}(x_i, x) + b \tag{1}$$

where α_i are the learnable model parameters, and the kernel function $\mathcal{K}(\cdot, \cdot)$ aims to measure the similarity between examples. Function $\mathcal{K}$ is, in general, a nonlinear function so that the classifier can model nonlinear dependencies between features and class. Decision function $\mathfrak{f}$ only depends on those training examples whose corresponding coefficient α_i is different from zero: these are called *support vectors*. In general terms, the decision function $\mathfrak{f}$ defines a hyperplane, which constitutes the decision boundary that used to classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. This idea can be extended to the case of multiclass classification, breaking down the multiclassification problem into multiple binary classification problems. Two approaches can be selected to this end: In the one-versus-one approach, a binary classifier is set per each pair of classes, regardless of the other classes; in the one-versus-rest, a binary classifier is set per each class, to distinguish it from the rest of data.

A commonly used kernel function for SVM classifiers is the radial basis function (RBF). The RBF kernel function for two points x_1 and x_2 computes the similarity or how close they are to each other, as a function of their Euclidean distance. It is specified by means of two parameters: C and γ. While C sets a tradeoff between misclassification of training examples and simplicity of the decision surface, γ defines the width of the radial functions.

C. Random Forests

An RF [32] consists in a collection of individual Decision Trees (DTs) [33], whose predictions are combined typically through a voting process. A DT inductively learns a set of explainable classification rules by imposing conditions of the values of the features describing the examples. When creating a RF, each DT is trained from a distinct set of n examples randomly sampled from the original training set, and by testing only m out of M features at each node in the tree. The process of combining the outcome of individual classifiers into a single prediction is usually named as an *ensemble* approach, and it is known to typically improve the performance of the overall system, as well as to reduce overfitting. Each individual DT in the RF produces a class prediction and the class with most votes becomes the prediction of the model, as in classic ensemble approaches.

Fig. 2. Gestures considered in this study: from left to right, *up*, *down*, *circle*, *left,* and *right*. Animated examples can be seen in the multimedia attachment to [2].

D. Neural Networks

An NN [34] is a nonlinear function transforming a set of input variables in a set of output variables via a set of adjustable parameters. In particular, an NN is a combination of nonlinear basis functions. Each basis function is itself a nonlinear function (called activation function) of a linear combination of the inputs, and the coefficients in such combination are adaptive weights that can be learned to fit the training data. Several layers of adaptive weights can be stacked to form a deep network.

V. Experimental Implementation

In this section, we present the experimental setting that was considered to implement the pipeline shown in Fig. 1. We first introduce the validation use case and describe different sets of features extracted from the raw signals; then, we illustrate the adopted training and evaluation procedure, and finally the performance metrics employed at prediction time.

A. Use Case

To validate the proposed pipeline, we considered the gesture-based HRI approach introduced in [2]. In particular, the scenario consists in recognizing the following $N_{\text{gest}} = 5$ gestures, depicted in Fig. 2:

1) *up:* sharp movement upward in a plane parallel to the sagittal one;
2) *down:* sharp movement downward in a plane parallel to the sagittal one;
3) *circle:* movement in a circular shape in a plane parallel to the frontal one;
4) *left:* sharp movement to the left, from sagittal plane to frontal one, in a plane parallel to the transverse one;
5) *right:* sharp movement to the right, from sagittal plane to frontal one, in a plane parallel to the transverse.

Examples can be seen in the multimedia attachment to [2]. These gestures are meant to be performed with the right arm, with the subject wearing an IMU on the right wrist. As our input device, we considered a commercial multipurpose smartwatch, namely the Samsung Gear S device. Data used in the analysis presented in this article consist of triaxial inertial measurements recorded by the smartwatch, namely $x \in \mathbb{R}^{10}$, and include timestamp, angular velocities, and linear accelerations (raw and with automatically compensated gravity). Data are accessed by means of a Tizen interface and are provided on an uneven sampling grid. On average, approximately 25 samples per second are provided.

As regards the overall software architecture, data are recorded via Tizen and sent via Wi-Fi to an external computer, for ease of implementation. The classification stage is implemented in Python, using the `scikit-learn` library [35] and `Tensorflow 2.0.0` [36].

Robotic Operating System (ROS) can be, then, used for implementing the remaining of the pipeline [37]. It is an open-source hardware-independent middleware widely used in robotics and consists in a set of software libraries and tools that allow communication with robots, actuators, sensors, and other devices commonly used in robotic applications. Since it supports Python, it can deal with data acquisition, data processing, gesture recognition, and communication with the robotic platform. To this end, a publish–subscribe pattern can be used that can efficiently handle communication events, such as arriving messages and inform the robot about the detection of a gesture.

B. Feature Extraction

Starting from the data provided by the wrist-worn device, synthetic features are extracted and passed to the classification stage as input data. We considered three sets of features, which were compared as possible different inputs for the classification algorithms. The first two feature sets, denoted in the following as $\mathcal{F}_1$ and $\mathcal{F}_2$, were defined as follows. Set $\mathcal{F}_1$ includes standard statistics computed in the domain of time, for each inertial quantity: mean value, standard deviation, maximum value, and minimum value. These are customarily used in many gesture recognition applications (for example, see work in [38]). Following [24] and [26], set $\mathcal{F}_2$ includes mean absolute value, count of slope sign changes, count of zero crossings, and waveform length, given as follows:

1) mean absolute value: $\frac{1}{L}\sum_{k=1}^{L}|x_k|$;
2) slope sign change: $(x_k - x_{k-1})(x_{k+1} - x_k) < 0$;
3) zero crossing: $x_k x_{k+1} < 0$;
4) waveform length: $\sum_{k=2}^{L}|x_k - x_{k-1}|$;

where L is the number of samples in a sliding window.

Set $\mathcal{F}_3$ is obtained from the union of $\mathcal{F}_1$ and $\mathcal{F}_2$ and, hence, consists of eight features.

As a result, for each feature set, each input sequence for classification (both for training and test datasets) is represented by 36 features for $\mathcal{F}_1$ and $\mathcal{F}_2$ and 72 features for $\mathcal{F}_3$, since the abovementioned features were computed for each of the nine inertial quantities measured by the smartwatch.

C. Training and Evaluation

All the considered classification algorithms were trained by considering a labelled dataset of 300 gestures, which included 60 trials per gesture. Such gestures were all performed by the same subject, denoted in the following as S_0, standing in steady state. The dataset contains inertial samples that refer to the execution of the gestures, only: samples between two consecutive gestures were manually excluded. The composition of the training dataset is detailed in Table II. In order to tune the hyperparameters of the different classifiers, an inner k-fold cross-validation on the training set was conducted.

TABLE II
COMPOSITION OF THE TRAINING AND TEST DATASET

	Label	Instances	Duration ([samples])
Training	*Up*	60	29.5 ± 2.9
	Down	60	29.9 ± 3.3
	Circle	60	32.6 ± 4.8
	Left	60	25.9 ± 2.5
	Right	60	28.9 ± 2.5
	$\mathbf{N_{train}}$	300	

		Instances	Subject
Test	$\mathbf{N_{test}}$	200	90 by S_1 10 each by $S_2, \ldots, S_{12}$

The duration is expressed in average number of samples, with its standard deviation.

TABLE III
ARCHITECTURES FOR NNS CONSIDERED IN THE CLASSIFICATION STAGE

	Layer	Type	Nodes	Activation function
Model 1	Input	Dense	100	ReLU
	Hidden	Dense	50	ReLU
	Hidden	Dense	25	ReLU
	Output	Dense	6	Softmax
Model 2	Input	Dense	100	ReLU
	Hidden	Dense	50	ReLU
	Output	Dense	6	Softmax
Model 3	Input	Dense	50	ReLU
	Hidden	Dense	25	ReLU
	Output	Dense	6	Softmax

Learning rate was set to 1e-3, the number of training epochs to 500, and the batch size to 100 for all the considered architectures.

Based on preliminary experiments, in the analyses reported in this article, for KNN we tested values for $K = 5, 7, 10, 15$. Since the two hyperparameters, C and γ, have to be jointly chosen, we exploited a grid search with an internal cross-validation[1] as customary in this kind of applications. We finally selected $C = 0.1$ and $\gamma = 10^{-6}$. As for RF, we considered a total of 100 trees in our model. For NN, we considered three different architectures, summarized in Table III. Each network was trained using the Adam optimizer [39] with batch normalization, and the hyperparameters (learning rate, momentum, and number of training epochs) were set with Bayesian optimization.[2]

To test the algorithms under analysis, we considered streams of inertial data containing gestures. In other words, consecutive gestures were separated by a nonconstant number of samples in which the user moved in a natural manner, as described in the following. The goal was to replicate real-life scenarios in which the subject is free to move while interacting with a robot. Occurrences of gestures were manually annotated to serve as ground truth. To deal with online classification, and following [2], input data were provided to the algorithms by considering sliding windows of length $L = 35$ samples, with one sample shift. As a result, the test set includes $N_{\text{test}} = 200$ gestures performed with the subjects moving in a natural manner between consecutive gestures (e.g., walking, waving, and drinking). These gestures were performed by 12 subjects, namely $S_1, S_2, \ldots, S_{12}$, different from the one contributing to the training set. They were not told the movements to execute between gestures, but were left free to chose. An equal number of the five gestures under consideration was included. The composition of the test set is detailed in Table II.

All the algorithms were implemented in a multiclass configuration. This implies that the detection of a gesture is accepted only if the confidence of the classifier is above a certain threshold that can be tuned in order to balance between false positives and false negatives. To this end, different thresholds have been considered. In the case of RF, performance has been computed considering three different thresholds, namely σ_1, σ_2, and σ_3, on the predicted probabilities for each class, to be used as a confidence level for gesture recognition. Specifically, for each input sequence, N_{gest}-predicted probabilities $P_i, i = 1, \ldots, N_{gest}$ are given as output, each representing the likelihood for that sequence to contain one of the considered gestures. For each threshold σ_j, gesture i is, then, detected if $\max P_i > \sigma_j$, with $j = 1, 2, 3$. In a similar manner, for SVM, a one-versus-one approach for multiclass classification is selected and five thresholds are considered, denoted as θ_k, $k = 1, \ldots, 5$. A score for each sample in relation to each gesture is provided and the threshold θ_k is set with respect to such score. The results achieved for RF and SVM with the different thresholds σ_j and θ_k ($j = 1, 2, 3$ and $k = 1, \ldots, 5$) are reported in Section VI.[3] A threshold is considered also in the case of KNN: it was set to $\tau = 1$ for any value considered for K, and, hence, is not reported in Section VI. Alternatively, when dealing with a specific case study, the optimal threshold can be set with a cross-validation step.

Finally, as introduced in Section III-B, a gesture was detected only if the same outcome was predicted by a classifier throughout several consecutive sliding windows. Given the average duration of the considered gestures (reported in Table II) and the approximate sampling frequency of the smartwatch of about 25 Hz, we set such number of consecutive sliding windows to 25.

D. Performance Metrics

A set of established metrics has been computed to compare the performance of the considered algorithms [40], [41]. The selection of these metrics was guided by the need to guarantee high recognition performance in general, while taking into account specific HRI constraints as well. We name *Recall* the percentage of gestures in the test set that have been correctly detected by the classifier, whereas *Precision* is the percentage of predicted gestures that are correct.[4] These two measures account for different kinds of error: namely, *Recall* considers false negatives since it takes into account those gestures that are not recognized by the system, whereas *Precision* considers false

[1]We used the `GridSearchCV` class of `scikit-learn`.

[2]We used the library `Keras Tuner`.

[3]We set $\sigma_1 = 0.55$, $\sigma_2 = 0.60$, and $\sigma_3 = 0.65$, and $\theta_k = 4.240 + k \cdot 0.005$, with $k = 1, \ldots, 5$.

[4]In general, in multiclass problems, *Precision* and *Recall* are more frequently computed on a per-class basis, whereas in our application scenario, we prefer to report classifier-level metrics in order to focus on the global performance of the system.

positives, which are gestures that the system wrongly recognizes. The latter are much more dangerous in HRI, where a gesture starts a command to the robot, as highlighted by constraint *C5* in Section III-A. Hence, in the need for a compromise between performance on *Precision* and *Recall*, it is important that *Precision* is as high as possible. Low *Precision* means that the risk of providing unintended commands to the robot is high, which is clearly unacceptable. Conversely, a false negative in HRI means that the user has performed a gesture to command the robot, but this command was not received by the robot. Although annoying for the user, this circumstance is less dangerous than the previous one, since it does not have direct consequences in terms of unintended behaviour of the robot.

Precision and *Recall* are usually combined in the $F_\beta$score, defined as

$$F_\beta\text{score} = (1+\beta^2)\frac{\text{Recall}\cdot\text{Precision}}{\text{Recall}+\beta^2\text{Precision}}. \tag{2}$$

The parameter β quantifies the importance of *Recall* over *Precision* and is typically set to 1. Since, in our context, it is more important that *Precision*, rather than *Recall*, is large, we set $\beta = 0.5$, to attribute more importance to *Precision* over *Recall* [40], [41]. Thus, in our analysis, we consider $F_{0.5}$score, following constraint *C5*.

In order to better highlight the difference between the two main error categories (i.e., missed or misclassified gestures) and comply with constraint *C5*, we introduce two additional metrics. The *misclassification gesture rate* (MGR) is the percentage of real gestures that are assigned to the wrong category, whereas the *undetected gesture rate* (UGR) is the percentage of real gestures that are not detected. We hereby remark that Recall + MGR + UGR = 1. Following the line of abovementioned reasoning, it is more important to minimize MGR than UGR, since the former accounts for unintended swaps among commands to the robot, whereas the latter refers to the need, for the user, to repeat a gesture.

Finally, to guarantee real-time gesture recognition (see constraint *C2* in Section III-A), algorithms have been compared also in terms of computational burden. To this end, the classification time has been considered. With respect to Fig. 1, this amounts to considering the time required from the selection of a sliding window (dashed rectangles on the left-hand side) to the output of classification algorithms for that window (boxes "*Gesture recognized?*" in the figure).

VI. Analysis of the Experimental Data

In this section, we analyze and discuss the experimental data collected implementing the proposed pipeline in the use case introduced in Section V-A. In particular, besides comparing the performance of the considered four classical machine learning approaches with different feature sets, we will focus on the generalization capabilities of the classifiers across different subjects, with varying levels of familiarity with the selected gestures and inclusion in training set. The idea behind this analysis is to assess the amount of user's contribution, meant in terms of participation in the training set or gained acquaintance with the gestures, that is needed to achieve satisfactory performance for gesture recognition. To this end, we consider the three different conditions:

a) subjects with different familiarity with the considered gestures, whose data were not included in the training set;
b) a subject with previous experience on the use of the selected gestures, but whose data were not included in the training set;
c) the same subject as in the previous condition, whose data contributed, in small part, to the training set.

a) Multiple subjects, expert, and novel, not included in the training set: Table IV reports the performance of the classification stage, for the different algorithms considered, in the case of subjects with and without prior familiarity with the considered set of gestures. With reference to Table II, numbers in Table IV refer to 200 gestures performed by all the subjects in the test set (from S_1 to S_{12}). We remark that these subjects were not represented in the training set and had received different amount of instructions on how to execute the gestures. Specifically, subject S_1 had previous experience with the considered set of gestures, whereas all the others had not, and received little training soon before recording the test set. The table shows that satisfactory results can be achieved with RF (with features $\mathcal{F}_3$) for *Precision* and *MGR*, which are the most relevant metrics for HRI applications, as discussed in Section V-D. Nevertheless, performance achieved for the other metrics is extremely low. The performance achieved with the other algorithms is quite poor, too.

b) Expert subject, not included in the training set: To improve the performance of gesture recognition, we limited the test set to gestures performed by an expert subject, who had prior experience in using them, namely S_1. Indeed, although the considered gestures consist in simple and natural movements, they are meant as sharp movements, as described in Section V-A, and require some experience to get familiar with movement speed and initial and final positions. Table V reports the results achieved in this condition. As for the previous analysis, the subject contributing to the training set was not included in the test set. The table shows that a notable increase in performance is achieved. However, classification performance is still quite low for all the considered algorithms. In particular, while very good performance can be achieved in terms of *Precision* and *MGR*, *Recall* is still quite low. Satisfactory *Recall* is achieved with NNs (Model 1 and features $\mathcal{F}_1$), at the expenses of low *Precision* and $F_{0.5}$ *score*. There is no algorithm, among those considered, that returns satisfactory results for both *Recall* and *Precision*.

c) Expert subject, included in the training set: As a further attempt to improve the performance of gesture recognition, we increased the training set adding some gestures by the same subject in the test set. In particular, with reference to Table II, the training set was augmented adding 100 new gestures (20 per type) by subject S_1. Regarding the test set, we considered 90 gestures performed by S_1 [i.e., the same as condition *b)*]. Table VI reports the performance achieved in this condition. The best classification performance is achieved with SVM with features $\mathcal{F}_1$, which proves successful for all the considered metrics. All the thresholds θ_k, with $k = 1, \ldots, 4$, return similar results, whereas θ_5 is less performing. Considering $F_{0.5}$ *score*, which is the weighted combination of *Recall* and *Precision*, the

TABLE IV
CLASSIFICATION PERFORMANCE FOR GESTURES PERFORMED BY MULTIPLE SUBJECTS, EXPERT, AND NOVEL, NOT INCLUDED IN THE TRAINING SET

	Features $\mathcal{F}_1$					**Features $\mathcal{F}_2$**					**Features $\mathcal{F}_3$**				
	R	*P*	*UGR*	*MGR*	$F_{0.5}$	*R*	*P*	*UGR*	*MGR*	$F_{0.5}$	*R*	*P*	*UGR*	*MGR*	$F_{0.5}$
RF (σ_1)	29.1	72.2	62.8	8.1	33.0	20.9	31.1	55.6	23.5	22.4	27.0	67.9	62.8	10.2	30.7
RF (σ_2)	20.9	91.1	77.0	2.1	24.7	16.8	63.5	74.0	9.2	19.7	21.0	90.7	76.8	2.2	24.8
RF (σ_3)	17.3	91.9	81.2	1.6	20.7	8.2	69.6	88.3	3.5	9.9	12.8	**92.6**	86.2	**1.0**	15.4
KNN ($K = 5$)	44.4	56.5	33.7	21.9	**46.4**	21.9	62.3	66.8	11.3	25.2	29.6	68.2	59.7	10.7	33.4
KNN ($K = 7$)	43.4	60.7	38.3	18.3	46.0	14.3	60.9	77.6	8.1	16.9	25.5	70.4	65.8	8.7	29.2
KNN ($K = 10$)	40.3	67.5	44.9	14.8	43.8	11.2	62.9	82.7	6.1	13.4	20.9	75.9	73.0	6.1	24.5
KNN ($K = 15$)	37.2	75.3	52.0	10.8	41.4	5.6	55.0	90.3	4.1	6.8	14.3	71.8	80.6	5.1	17.0
SVM (θ_1)	51.5	35.1	15.3	33.2	47.1	44.4	26.0	10.7	44.9	38.9	50.0	27.9	**9.7**	40.3	43.2
SVM (θ_2)	48.5	33.5	17.9	33.6	44.5	42.3	24.5	**9.7**	48.0	37.0	47.4	28.4	10.7	41.9	41.9
SVM (θ_3)	45.9	34.1	20.9	33.2	42.9	37.8	23.6	12.2	50.0	33.7	41.8	27.4	11.2	46.9	37.9
SVM (θ_4)	43.4	44.0	31.6	25.0	43.5	34.2	23.0	11.7	54.1	31.2	37.2	26.6	14.3	48.5	34.5
SVM (θ_5)	39.8	70.3	50.0	10.2	43.6	32.1	24.0	15.8	52.1	30.1	32.7	28.4	24.5	42.8	31.7
NN (Model 1)	**51.9**	35.3	28.8	19.3	37.3	35.2	25.4	35.0	29.8	26.7	47.2	32.0	27.3	25.5	33.9
NN (Model 2)	50.1	29.6	26.6	23.3	31.9	41.5	34.7	35.2	23.3	35.4	49.4	33.2	26.1	24.5	35.2
NN (Model 3)	46.8	29.2	27.2	26.0	31.3	38.2	29.2	36.8	25.0	30.5	47.2	30.3	25.9	26.9	32.4

R: Recall, P: Precision, UGR: undetected gesture rate, MGR: misclassification gesture rate, F0.5 : F0.5 score.
Bold entities denotes the best classification algorithm.

TABLE V
CLASSIFICATION PERFORMANCE FOR GESTURES PERFORMED BY AN EXPERT SUBJECT, NOT INCLUDED IN THE TRAINING SET

	Features $\mathcal{F}_1$					**Features $\mathcal{F}_2$**					**Features $\mathcal{F}_3$**				
	R	*P*	*UGR*	*MGR*	$F_{0.5}$	*R*	*P*	*UGR*	*MGR*	$F_{0.5}$	*R*	*P*	*UGR*	*MGR*	$F_{0.5}$
RF (σ_1)	43.0	84.1	51.2	5.8	47.7	30.2	28.6	37.2	32.6	29.9	41.9	80.0	47.7	10.4	46.3
RF (σ_2)	34.9	90.9	61.6	3.5	39.8	24.4	63.6	61.6	14.0	27.8	33.7	90.6	62.8	3.7	38.6
RF (σ_3)	30.2	92.9	67.4	2.4	34.9	16.3	73.7	77.9	5.8	19.3	23.3	95.2	75.6	1.1	27.4
KNN ($K = 5$)	59.3	60.0	23.3	17.4	59.4	33.7	82.9	62.8	3.5	38.3	46.5	83.3	50.0	3.5	51.0
KNN ($K = 7$)	58.1	64.9	29.1	12.8	59.4	20.9	90.0	77.9	1.2	24.7	40.7	89.7	58.1	1.2	45.7
KNN ($K = 10$)	57.0	77.8	34.9	8.1	**60.2**	17.4	93.8	81.4	1.2	20.8	33.7	**100.0**	66.3	**0.0**	38.9
KNN ($K = 15$)	53.5	88.5	43.0	3.5	58.1	9.3	**100.0**	90.7	**0.0**	11.4	24.4	100.0	75.6	**0.0**	28.8
SVM (θ_1)	68.6	31.9	**3.5**	27.9	55.8	57.0	27.8	4.7	38.3	47.1	68.6	33.5	4.7	26.7	56.7
SVM (θ_2)	66.3	33.3	9.3	24.4	55.3	52.3	25.4	4.7	43.0	43.2	62.8	31.6	5.8	31.4	52.4
SVM (θ_3)	60.5	39.1	20.9	18.6	54.5	46.5	24.4	5.8	47.7	39.4	53.5	29.9	7.0	39.5	46.2
SVM (θ_4)	57.0	56.3	33.7	9.3	56.8	40.7	22.6	5.8	53.5	35.1	45.3	27.1	9.3	45.4	40.0
SVM (θ_5)	51.2	83.0	47.7	1.1	55.4	40.7	23.8	5.8	53.5	35.6	41.9	28.1	12.8	45.3	38.1
NN (Model 1)	**83.6**	44.5	9.8	6.6	48.8	41.8	25.3	32.2	26.0	27.5	78.4	44.9	8.1	13.5	48.8
NN (Model 2)	76.8	32.5	10.0	13.2	36.6	57.8	37.9	19.9	22.3	40.5	80.7	43.4	5.4	13.9	47.6
NN (Model 3)	70.5	34.8	13.3	16.2	38.6	52.2	32.1	25.5	22.3	34.8	79.0	42.6	5.6	15.4	46.6

R: Recall, P: Precision, UGR: undetected gesture rate, MGR: misclassification gesture rate, F0.5 : F0.5 score.
Bold entities denotes the best classification algorithm.

threshold θ_1 returns the highest value. RF provides slightly better performance for *Precision* and *MGR* than SVM, but $F_{0.5}$ *score* for RF is quite low. Good performance, in terms of *Recall*, *UGR*, and *MGR*, is also achieved with NNs with features $\mathcal{F}_1$, although *Precision* and $F_{0.5}$ *score* are quite low. In summary, it is possible to argue that better performance can be achieved when using SVM with threshold θ_1 and features $\mathcal{F}_1$.

Building upon this result, we tested this classifier considering gestures performed during intense motion conditions or moderate physical activity. To this end, subject S_1 was asked to perform $N_{\text{test}}^{\text{IM}} = 20$ gestures during running and leg-skipping sessions and arm circumductions. These movements were intended as spurious noise for the classification algorithms. Nevertheless, SVM, with threshold θ_1 and features $\mathcal{F}_1$, has returned quite good performance also in this extreme motion condition: $R = 100\%$, $P = 57.1\%$, UGR $= 0.0\%$, MGR $= 0.0\%$, and $F_{0.5}$ score $= 87.0\%$. While *Recall* is very high, the effect of intense motion can be seen in lower *Precision*, due to an increased number of false positive gestures.

A. Discussion

Tables IV–VI report the classification performance achieved considering a possible implementation of the proposed pipeline. To this end, we considered four standard classification algorithms and implemented them with different parameters. The achieved results show that, to achieve satisfactory classification performance, it is needed that the user receives some sort of training on how to execute the selected gestures. In particular, even if natural and easy movements are considered, the user should practice them, in order to get familiar with range and speed of motion and other specific features. To this end, including, in the classification algorithm, some amount of user specific training is beneficial. It is noteworthy that this is not in contrast with constraint *C4* of Table I. Indeed, in Table VI, we considered the case

TABLE VI
CLASSIFICATION PERFORMANCE FOR GESTURES PERFORMED BY AN EXPERT SUBJECT, INCLUDED IN THE TRAINING SET

	Features $\mathcal{F}_1$					Features $\mathcal{F}_2$					Features $\mathcal{F}_3$				
	R	*P*	*UGR*	*MGR*	$F_{0.5}$	*R*	*P*	*UGR*	*MGR*	$F_{0.5}$	*R*	*P*	*UGR*	*MGR*	$F_{0.5}$
RF (σ_1)	61.6	98.1	38.4	**0.0**	66.6	36.0	81.6	58.1	5.9	40.6	45.3	92.9	52.3	2.4	50.5
RF (σ_2)	52.3	**100.0**	47.7	**0.0**	57.8	29.1	**100.0**	70.9	**0.0**	33.9	30.2	96.3	68.6	1.2	35.0
RF (σ_3)	40.7	**100.0**	59.3	**0.0**	46.2	24.4	**100.0**	75.6	**0.0**	28.8	27.9	**100.0**	72.1	**0.0**	32.6
KNN ($K = 5$)	74.4	52.9	18.6	7.0	68.8	25.6	53.7	69.8	4.6	28.6	40.7	62.5	54.7	4.6	43.8
KNN ($K = 7$)	73.3	56.3	19.7	7.0	69.1	14.0	50.0	84.9	1.1	16.3	33.7	64.4	64.0	2.3	37.3
KNN ($K = 10$)	73.3	64.9	23.2	3.5	71.4	11.6	62.5	87.2	1.2	13.9	26.7	79.3	73.3	**0.0**	30.8
KNN ($K = 15$)	64.0	75.3	36.0	**0.0**	65.9	7.0	85.7	93.0	**0.0**	8.5	18.6	94.1	81.4	**0.0**	22.2
SVM (θ_1)	**96.5**	91.2	**2.3**	1.2	**95.4**	66.3	47.5	7.0	26.7	61.4	88.4	67.9	4.6	7.0	83.3
SVM (θ_2)	94.2	95.3	5.8	**0.0**	94.4	65.1	51.9	8.2	26.7	61.9	88.4	74.5	5.8	5.8	85.2
SVM (θ_3)	94.2	96.4	5.8	**0.0**	94.6	62.8	54.0	10.5	26.7	60.8	84.9	76.0	9.3	5.8	83.0
SVM (θ_4)	94.2	97.6	5.8	**0.0**	94.8	62.8	55.1	10.5	26.7	61.1	80.2	77.5	14.0	5.8	79.7
SVM (θ_5)	83.7	98.6	16.3	**0.0**	86.3	59.3	58.0	17.4	23.3	59.0	77.9	83.8	18.6	3.5	79.0
NN (Model 1)	94.6	62.9	3.4	2.0	67.3	55.0	41.0	24.4	20.6	42.9	90.8	63.4	6.3	2.9	67.0
NN (Model 2)	94.8	54.9	**2.3**	2.9	59.8	56.1	43.8	26.4	17.5	45.5	88.9	59.8	4.9	6.2	63.3
NN (Model 3)	94.6	62.9	3.4	2.0	67.1	58.9	42.7	17.6	23.5	45.0	87.0	50.6	5.5	7.5	54.9

R: Recall, P: Precision, UGR: undetected gesture rate, MGR: misclassification gesture rate, F0.5 : F0.5 score.
Bold entities denotes the best classification algorithm.

TABLE VII
AVERAGE COMPUTATIONAL TIME FOR GESTURE CLASSIFICATION ON A SINGLE TIME WINDOW

Model	RF	KNN	SVM	NN Model 1	NN Model 2	NN Model 3
Time (ms)	23	3	2	16	17	18

that the algorithms were trained with data from a subject not in the test set in large part and, in small part, with data from the same subject in the test set. This is customary in devices running, for example, algorithms for voice or handwritten text recognition, which usually require some fine training by the user in charge.

As Table VI shows, if the user is familiar with the selected set of gestures and algorithms are trained also with her/his data, classification performances become highly satisfactory and allow an efficient use of gestures in HRI. This is confirmed also by the classification performance achieved in the case of intense motion condition: although this represents a quite unusual condition in HRI, these results show that gestures can be used without restrictions to any other tasks the user might be carrying out while interacting with a robot.

B. Analysis of Computational Burden

The computational burden of the proposed pipeline was assessed to verify its applicability to real-time gesture recognition (constraint *C2* in Table I). To this end, we computed the average classification time required by each of the considered algorithms. The measured times, averaged over 5000 sliding windows, are reported in Table VII.

We hereby make two observations. First, according to the decision rule presented in Sections III-B and V-C, a gesture is detected only if it is recognized in 25 consecutive sliding windows. In other words, the recognition of a gesture requires that 25 windows elapsed. Hence, since times in Table VII refer to classification for a single time window, the recognition of a gesture implies a delay that is 25 times the one reported in the table, from the beginning of its execution. Second, for ease of computation, we implemented the classification stage on an external computer[5] running Python. Computation times reported in Table VII refer to this setting. In a real-world operational setting, an hardware-driven software implementation should be considered, possibly relying on the computational capacity of the robot.

Table VII shows that all the algorithms are quite fast, with SVM being the fastest among those considered. Even considered the need to process 25 consecutive windows, the delay introduced by the classification operation does not affect the fluency of gesture-based interaction. As a result, the selection of the most suited algorithm for the proposed classification can be guided by classification performances only.

VII. CONCLUSION

In this article, we considered the problem of forearm gesture recognition for HRI. The ultimate goal is that of providing commands to robots by means of intuitive gestures. To this end, we proposed a pipeline for gesture recognition specifically designed for HRI applications. As input data, we consider wrist inertial movements, which can be recorded with any commercial device mounting an IMU. Gesture detection and classification is performed by a classification stage that relies on machine learning algorithms. To this end, we compared the performance of several machine learning algorithms used in classification problems: RF, KNN, SVM, and NN. An extensive evaluation was performed, including also an analysis of computational burden for real-time gesture recognition. Different performance metrics were introduced to provide a thorough assessment of algorithms and to highlight the specific needs of gesture recognition in HRI context. In particular, we highlighted that the cost of false positives and misclassifications (wrongly recognized gestures) is much higher than that of false negatives (gestures not detected at all), thus motivating the need for models with a high precision, even though at the cost of a lower recall.

An evaluation use case was selected to show an implementation of the proposed pipeline. It consisted of five gestures,

[5]The computer used for this analysis embeds an Intel i5 3.2 GHz CPU and 8 GB RAM, and it runs Ubuntu 16.04.7.

recorded with a smartwatch. In total, 12 subjects were included in the validation, considering different confidence with the selected gestures. First, we analyzed whether a general, meant as opposite to user tailored, training of classification algorithms could be sufficient to achieve satisfactory recognition performance. Then, to improve recognition performance, we considered the need of training algorithms also with a small amount of data from the subject in charge.

As concluding remarks, it is noteworthy that the proposed pipeline is general. Indeed, it can be used implementing other classification algorithms, or with other parameters. In addition, it can be used to interact with any kind of robots and with other sets of gestures.

ACKNOWLEDGMENT

The authors would like to thank Anna Bellodi and Ilias Maite for their contribution to the preliminary analysis of the proposed approach.

REFERENCES

[1] A. Chaudhary, J. L. Raheja, K. Das, and S. Raheja, "Intelligent approaches to interact with machines using hand gesture recognition in natural way: A survey," *Int. J. Comput. Sci. Eng. Surv.*, vol. 1, no. 2, pp. 122–133, 2013.

[2] V. Villani, L. Sabattini, G. Riggio, C. Secchi, M. Minelli, and C. Fantuzzi, "A natural infrastructure-less human-robot interaction system," *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1640–1647, Jul. 2017.

[3] V. Villani, L. Sabattini, N. Battilani, and C. Fantuzzi, "Smartwatch-enhanced interaction with an advanced troubleshooting system for industrial machines," *IFAC-PapersOnLine*, vol. 49, no. 19, pp. 277–282, 2016.

[4] P. Neto, M. Simão, N. Mendes, and M. Safeea, "Gesture-based human-robot interaction for human assistance in manufacturing," *Int. J. Adv. Manuf. Technol.*, vol. 101, no. 1-4, pp. 119–135, 2019.

[5] H.-D. Yang, A.-Y. Park, and S.-W. Lee, "Gesture spotting and recognition for human–robot interaction," *IEEE Trans. Robot.*, vol. 23, no. 2, pp. 256–270, Apr. 2007.

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

[7] D. Kim, J. Lee, H.-S. Yoon, J. Kim, and J. Sohn, "Vision-based arm gesture recognition for a long-range human–robot interaction," *J. Supercomputing*, vol. 65, no. 1, pp. 336–352, 2013.

[8] D. Xu, X. Wu, Y.-L. Chen, and Y. Xu, "Online dynamic gesture recognition for human robot interaction," *J. Intell. Robotic Syst.*, vol. 77, no. 3-4, pp. 583–596, 2015.

[9] M. Sigalas, H. Baltzakis, and P. Trahanias, "Gesture recognition based on arm tracking for human-robot interaction," in *Proc. IEEE/RSJ Int. Conf. Intelli. Robots Syst.*, 2010, pp. 5424–5429.

[10] G. Cicirelli, C. Attolico, C. Guaragnella, and T. D'Orazio, "A kinect-based gesture recognition approach for a natural human robot interface," *Int. J. Adv. Robotic Syst.*, vol. 12, no. 22, 2015.

[11] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Trans. Syst., Man, Cybern., Part C (Appl. Rev.)*, vol. 37, no. 3, pp. 311–324, May 2007.

[12] V. Villani, L. Sabattini, G. Riggio, A. Levratti, C. Secchi, and C. Fantuzzi, "Interacting with a mobile robot with a natural infrastructure-less interface," in *Proc. IFAC 20th World Congress Int. Federation Autom. Control IFAC*, 2017, vol. 50, no. 1, pp. 12753–12758.

[13] V. Villani, B. Capelli, C. Secchi, C. Fantuzzi, and L. Sabattini, "Humans interacting with multi-robot systems: A natural affect-based approach," *Auton. Robots*, vol. 44, no. 3, pp. 601–616, 2020.

[14] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: A survey," *Artif. Intell. Rev.*, vol. 43, no. 1, pp. 1–54, 2015.

[15] J. Suarez and R. R. Murphy, "Hand gesture recognition with depth images: A review," in *Proc. IEEE 21st Int. Symp. Robot Hum. Interactive Commun.*, 2012, pp. 411–417.

[16] M. Chandarana, A. Trujillo, K. Shimada, and B. D. Allen, "A natural interaction interface for uavs using intuitive gesture recognition," in *Proc. Adv. Hum. Factors Robots Unmanned Syst.*, 2017, pp. 387–398.

[17] D. Roggen et al., "Collecting complex activity datasets in highly rich networked sensor environments," in *Proc. IEEE 7th Int. Conf. Netw. Sens. Syst.*, 2010, pp. 233–240.

[18] H. Junker, O. Amft, P. Lukowicz, and G. Tröstera, "Gesture spotting with body-worn inertial sensors to detect user activities," *Pattern Recognit.*, vol. 41, pp. 2010–2024, 2008.

[19] N. C. Krishnan and D. J. Cook, "Activity recognition on streaming sensor data," *Pervasive Mobile Comput.*, vol. 10, pp. 138–154, 2014.

[20] H. Han and S. W. Yoon, "Gyroscope-based continuous human hand gesture recognition for multi-modal wearable input device for human machine interaction," *Sensors*, vol. 19, no. 11, 2019, Art. no. 2562.

[21] M. Kim, J. Cho, S. Lee, and Y. Jung, "IMU sensor-based hand gesture recognition for human-machine interfaces," *Sensors*, vol. 19, no. 18, 2019, Art. no. 3827.

[22] A. Carfi, C. Motolese, B. Bruno, and F. Mastrogiovanni, "Online human gesture recognition using recurrent neural networks and wearable sensors," in *Proc. IEEE 27th Int. Symp. Robot Hum. Interactive Commun.*, 2018, pp. 188–195.

[23] M. E. Benalcázar et al., "Hand gesture recognition using machine learning and the MYO armband," in *Proc. IEEE 25th Eur. Signal Process. Conf.*, 2017, pp. 1040–1044.

[24] M. F. Wahid, R. Tafreshi, M. Al-Sowaidi, and R. Langari, "Subject-independent hand gesture recognition using normalization and machine learning algorithms," *J. Comput. Sci.*, vol. 27, pp. 69–76, 2018.

[25] Z. Zhang, K. Yang, J. Qian, and L. Zhang, "Real-time surface emg pattern recognition for hand gestures based on an artificial neural network," *Sensors*, vol. 19, no. 14, 2019, Art. no. 3170.

[26] S. Jiang et al., "Feasibility of wrist-worn, real-time hand, and surface gesture recognition via sEMG and IMU sensing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3376–3385, Aug. 2018.

[27] M. Georgi, C. Amma, and T. Schultz, "Recognizing hand and finger gestures with IMU based motion and EMG based muscle activity sensing," in *Proc. Int. Joint Conf. Biomed. Eng. Syst. Technol.*, 2015, pp. 99–108.

[28] P. Trigueiros, F. Ribeiro, and L. P. Reis, "A comparison of machine learning algorithms applied to hand gesture recognition," in *Proc. IEEE 7th Iberian Conf. Inf. Syst. Technol.*, 2012, pp. 1–6.

[29] H. Sagha et al., "Benchmarking classification techniques using the Opportunity human activity dataset," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2011, pp. 36–40.

[30] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-based hand-gesture applications," *Commun. ACM*, vol. 54, no. 2, pp. 60–71, 2011.

[31] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, Jul./Aug. 1998.

[32] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[33] J. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[34] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[35] F. Pedregosa et al., "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[36] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, *software available from tensorflow.org*. Accessed: Dec. 13, 2022. [Online]. Available: https://www.tensorflow.org/

[37] M. Quigley et al., "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, 2009.

[38] J. Kim, S. Mastnik, and E. André, "EMG-based hand gesture recognition for realtime biosignal interfacing," in *Proc. 13th Int. Conf. Intell. User Interfaces*, 2008, pp. 30–39.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[40] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 11, pp. 37–63, 2011.

[41] A. Tharwat, "Classification assessment methods," *Appl. Comput. Informat.*, vol. 17, no. 1, pp. 168–192, 2021.