Gesture Recognition in an IoT environment: a Machine Learning-based Prototype

Mariagrazia Fugini and Jacopo Finocchi

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza L. da Vinci, 32, Milano, Italy

mariagrazia.fugini@polimi.it, jacopo.finocchi@mail.polimi.it

Abstract: The spread of IoT and wearable devices is bringing out gesture interfaces as a solution for a more natural and immediate human-machine interaction. The "Seamless" project is an industrial research and development experience, aimed to build a virtual environment where data collected by IoT sensors can be navigated through a *gesture interface* and *virtual reality* tools. This paper presents the portion of the project concerning gesture analysis, focused on the problem of automatically understanding a set of hand gestures, in order to give commands through a wearable control device. The tackled issue is to build a *real time gesture recognition system* based on inertial data, that can easily adapt to different users and to an extensible set of gestures. This *gesture data variability* is addressed by means of a supervised Machine Learning approach, that allows adapting the system response to different gestures and to various ways of performing them by different people. A context-aware adapter allows interfacing the gesture recognition system to various applications.

Keywords: Human-Machine Interaction, Inertial Measurement Unit, Gesture Recognition, Machine Learning, Neural Networks, Context Aware Architecture.

1. Introduction

This paper presents an industrial research and development experience dealing with gesture recognition as part of the "*Seamless*" Project, funded by Regione Lombardia (Italy) and partially by the H2020 WorkingAge EU Project[1].

The Project, conducted in cooperation between Politecnico di Milano - DEIB, Next Industries[2] and Digisoft System Engineering (DSE)[3], was aimed to build a *virtual*

[1] This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No.826232, project WorkingAge (Smart Working environments for all Ages).

[2] https://www.nextind.eu

[3] https://www.gruppodse.it

environment where a set of three-dimensional structural data can be navigated through virtual reality tools and a gesture interface. Structural data regarding constructions, e.g., bridges, buildings, risky frameworks, etc., were collected by IoT sensors, monitoring the vibrations and movements of the structures.

In particular, recognition of movements was performed using a wearable device to give commands by drawing gestures in the air. The spread of IoT and wearable devices is bringing out gesture interfaces as a solution for Human-Machine Interaction (HCI) in a more natural and immediate way [1, 2].

The work presented here is framed in *Seamless* as follows. DSE developed a *virtual environment* where multiple subjects can interact collaboratively by connecting remotely through the web to a cooperative space of IoT "objects". This environment can be set up quickly in any room, even with no pre-setting or knowledge of the physical layout, by using a simple mobile equipment consisting of a set of IoT devices. It is controlled by a software application based on the Unreal Engine, which is interfaced with the Tactigon-Skin^{TM} provided by Next Industries, as a gesture control input device. The Tactigon-Skin is developed by Next Industries as an IoT device to be worn on the user's hand and is equipped with a 6-axis Inertial Measurement Unit (IMU).

When analyzing the inertial data collected through Tactigon-Skin, we observed a remarkable *data variability*, attributable to the difference between gestures and for hand gestures are performed differently depending on people and circumstances. For instance, the gesture has different intensity of speed and acceleration (e.g., rapid acceleration and slow braking or vice versa), different duration amplitude (e.g., in relation to the length of the user's arm), trajectory or slight hand vibrations. The need to handle such data variability led us to adopt a supervised Machine Learning (ML) approach, which is described in the paper.

The gesture recognition task focused on the problem of automatically recognizing a set of hand gestures, to enable users to send commands to applications through the Tactigon-Skin device. The analysis aims at building a real-time *hand gesture recognition system*, that can adapt to different users, able to recognize a set of gestures that can be defined and extended in different application environments.

Under these premises, our goal was to develop a prototype software capable to perform gesture recognition under four constraints:

- based on *inertial data only*, with no additional sensors, whereas most of the previous works rely on other types of sensors, especially vision-based [3, 4];
- in *real-time*, so that the recognized gesture prediction occurs within a predetermined time interval from the hand gesture conclusion, to allow for a smooth man-machine interaction;
- *user independent*, so as not to require a specific user training activity for each new user, after the completion of the general training sessions;
- with *no explicit signals* marking the gesture "start" and "end" (like pressing or releasing a button), in order to achieve a better user experience.

During our experimentation, two different solutions were tested, that rely on different ML techniques: an approach based on the Dynamic Time Warping (DTW) technique and a classical neural network approach.

The paper is organized as follows. In Section 2 we present relevant related works. In Section 3, we introduce our gesture recognition solution, illustrating the inertial data and the gesture recognition process. Section 4 describes the inertial data pre-processing stage and Section 5 the neural network classifier. Section 6 focuses on the context aware adapter that connects the gesture recognition system to applications.

2. Related Work

In the field of gesture recognition, most systems are based on some additional sensor technology, merging the inertial data with another data source, like magnetometers, visual information from a camera, or infrared sensor data.

Various challenges, such as interposition of obstacles, variations in lighting or complex background, make an accurate detection of hand gestures difficult in vision-based approaches.

A vision-based hand gesture recognition system is presented in [5]. Several researchers developed solutions based on depth cameras, as in [6], where an authentication system is proposed. In [7] a binocular vision system provides the depth information needed to build a three-dimensional map that allows to detect the gestures on a complex background. In [8] a reinforcement learning solution is described, based on two neural networks, trained to analyze video images to segment and classify surgical gestures.

The use of neural networks is frequent among systems that use inertial data as well.

In [9] gestures capturing and recognition is based on inertial and magnetic measurements units (IMMU) assembled on a fully cabled glove. The paper describes recognition methods based on Extreme Learning Machine, a kind of feed-forward neural networks, both for static and dynamic gestures classification.

A user-independent hand gesture recognition system is described in [10], based on an accelerometer-based pen device. Basic gestures are classified by a feed-forward neural network, while a segmentation algorithm is proposed to split complex gestures into several basic gestures, through a similarity matching procedure. In [11] a Wiener-type recurrent neural network is used for handwriting trajectory reconstruction, based only on inertial data. Some classifiers are compared in [12] to recognize hand gestures from two different inertial data sets, using various ML methods, such as Extremely Randomized Trees, Gradient Boosting and Ridge Classifiers.

For the analysis of inertial data, other researchers used ML techniques different from neural networks, such as the DTW algorithm, a technique that evaluates the distance between two sequences [13] through a non-linear distortion with respect to an independent variable, such as time.

DTW is particularly suitable for the analysis of time series and actually is used in various fields of application, from Speech Recognition to the recognition of human activities.

It was applied with good results in several inertial data studies, such as in [14], that presents a continuous hand gestures recognition technique, using a three-axis accelerometer and gyroscope. The solution is based on a gesture-coding algorithm, where gestures are recognized by comparing the gesture code against a gesture database, using the DTW algorithm.

The previous stage of developing a gesture recognition solution within *Seamless* Project, under the constraints described in Section 1, was also based on DTW technique [15].

In the second stage of the project, explained in this paper, we tested a neural network architecture and we paid more attention to the application interfacing questions, which also involve aspects of context awareness. ML and context awareness are topics that often appear together in smart technologies applications [16] and context-aware solutions are emerging in different fields and areas, in particular in smart environments and IoT [17]. We are exploring the feature of people movements through gesture recognition, aimed to the development of innovative HCI methods, in the WorkingAge EU Project [18].

The concept that the same gesture may have different meanings depending on the context has been proposed in gesture recognition, starting from [19], that focuses on gesture recognition in the medical field. The aim is to reduce the gesture set and to improve recognition reliability and usability. In [20], context awareness is achieved through a finite state machine, applied to a gesture recognition system based on neural networks.

3. Gesture Analysis Process

3.1 The Gestures Set

The T-Skin device contains an IMU chip, based on Micro Electro-Mechanical Systems (MEMS) technology, which collects inertial data, measuring three-axis acceleration and three-angular velocities with a 50 Hz frequency.

Sensor data are collected on-line from the T-Skin device, trough Bluetooth transmission or through a serial interface. Inertial data can also be acquired off-line from data files recorded during the previous on-line sessions. Recorded sessions allow us to perform a fast system training without the need of a gestures live session and to compare different algorithms under the same conditions.

During the experimentation we tested the recognition process on a set of 15 gestures: the simple translations and rotations along each of the space axes (one-dimensional motions) plus a few more complex gestures (two-dimensional motions), and a short hand movements that we call “grab”, that is performed without moving the wrist.

In detail, the system was trained on the following hand motions:

- Simple translations: Forward, Backward, Up, Down, Right, Left;
- Simple rotations: Pitch Down, Pitch Up, Roll Right, Roll Left;
- Complex gestures: Circle, Square, Slope Up, Slope Down.
- Small gestures: Grab.

These gestures were selected both for the clear inertial characterization and for their suitability to give intuitive commands in a three-dimensional spatial context. A gesture is typically performed with the T-skin device in about half a second for the shorter gestures, up to about one second for the more complex ones, with a short pause between gestures, of the order of a few tenths of a second.

3.2 The Gesture Recognition

Raw inertial data are first processed by a *filtering stage* with the purpose of reducing signal disturbances and perturbations. Then, data are analyzed to *detect and separate* the single gestures, using a segmentation algorithm based on thresholds, developed by the authors. Subsequently, the data set corresponding to each gesture is normalized in length and amplitude and forwarded to the *classification stage*.

The classification is performed by a *supervised ML* system, previously trained through a series of labeled gestures recognition sessions. The recognized gesture is finally made available to the adapter module, that implements the interface with the surrounding *Seamless* virtual environment.

The different steps of the described data analysis process were implemented in the gesture recognition prototype software developed during the project. The software was developed in Octave environment and partially in Python language.

The logical structure of the implemented process is summarized in Figure 1.

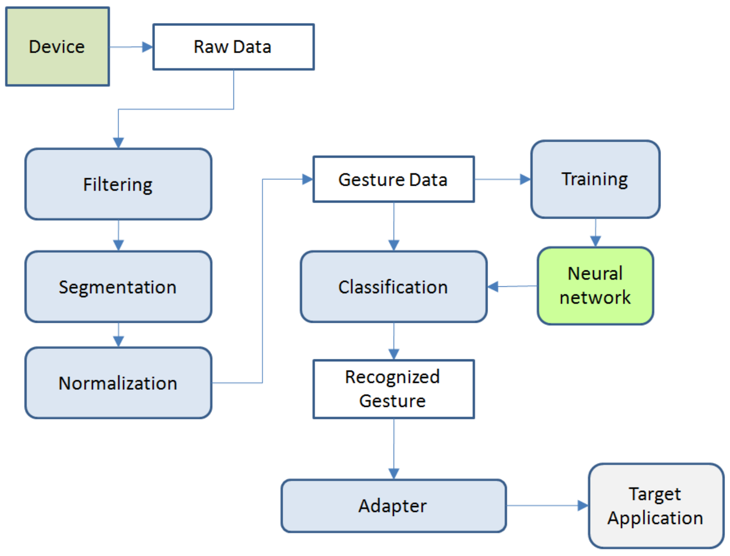

Fig. 1. The gesture recognition process

4. Gesture Pre-processing

4.1 Data Filtering

Raw data acquired by the sensors are subject to various perturbations, namely the gravity acceleration, the drift due to loss of calibration or measurement inaccuracy and high frequency noise, produced by vibrations and sensor thermal noise. For this reason, the acquired signals are firstly processed through a correction and filtering step.

The signal offsets due to gravity are removed by subtracting the projection of gravity along each axis from each acceleration sample.

Then, simple low-pass and high-pass filters are applied to remove noise and fixed components. A low-pass Butterworth filter is applied to remove high-frequency noise and a mild high-pass filter is applied to remove drift effects. The effects of high frequency noise are also mitigated by the normalization step performed after the gesture segmentation.

The drifting effects are limited by the short duration of a single gesture and by levelling the acceleration values to zero before computing each new gesture.

4.2 Gesture Segmentation

The goal of the segmentation stage is to cut in real-time the subset of data corresponding to a single gesture from a continuous inertial data flow.

The current segmentation algorithm is able to detect the short pause that divides consecutive gestures, based on a non-linear combination of angular velocities and accelerations that is evaluated against a value threshold and a duration threshold. The thresholds were obtained experimentally, by comparing the effectiveness of different combinations on the inertial data recorded during mixed gestures sessions and with the help of visual charts generated from the same data. The segmentation algorithm so far developed, works well with wide and fast gestures, separated by a short movement pause, and shows a lower success rate on softer movements.

An example of successful segmentation is shown in Figure 2. The figure shows the inertial data for two of the six sensor axes, provided in particular by one of the accelerometers and one of the gyroscopes.

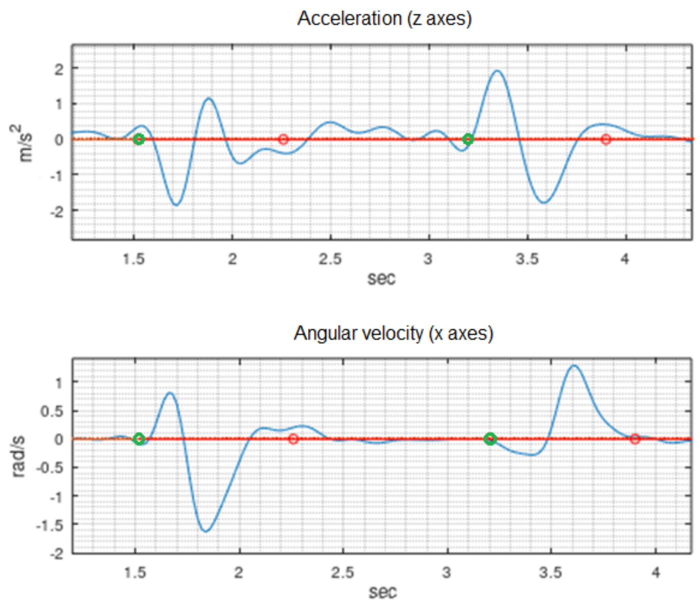

Fig. 2. Segmentation of two successive gestures: Up and Down.

The data section shown as an example, includes two consecutive gestures (an upward translation followed by a downward translation) separated by a short pause. The inertial data shown are already filtered. Green dots mark the gesture start while red dots mark the gesture ending, as detected by our segmentation algorithm.

Only data between the beginning and the end of each segment are considered *significant as gestures* and hence passed to the subsequent processing phases. Other values are considered as not belonging to a gesture and are hence excluded from further processing.

4.3 Gesture Normalization

After the detection of a data segment corresponding to a single gesture, the six inertial values are normalized in length and in amplitude.

First, a length normalization is performed that uniforms the gesture data to a predefined standard number of samples, through an interpolation algorithm. After that, an amplitude normalization is performed by means of a custom algorithm that recalculates each value on all axes to ensure that the peak value is set to 100%.

Figure 3 shows an example of gesture data normalization.

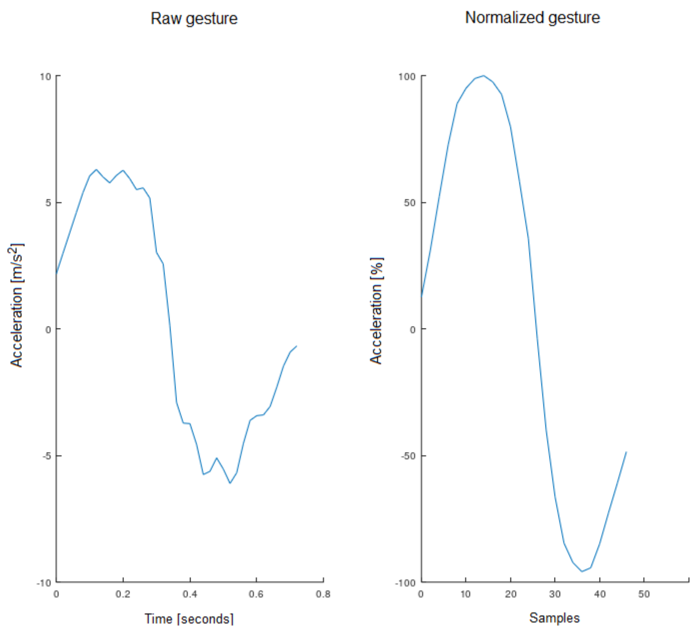

Fig. 3. Gesture raw data and normalized data

Figure 3 displays a simple translation gesture along one of the device axis, where first an acceleration and then a deceleration occur. On the left hand side, the values returned by one of the three accelerometers are depicted, while on the right side the resulting values after the application of the interpolation and normalization algorithms are reported. The first step dampens the small variations due to vibrations or small differences in the path and also the abscissa axis takes on the meaning of the virtual sample number. The second step scales the values along the ordinates to an arbitrarily fixed scale between -100 and 100.

These processing steps allow bringing together data segments corresponding to short and long span gestures as well as faster and slower gestures, that otherwise show very different raw acceleration values.

It should be observed that the standardization of the samples number, by merging adjacent samples, also causes a value smoothing that reduces the effect of noise and outliers.

5. Gesture Classification

The gesture recognition task was carried out as *a problem of classification* of the segmented input data.

As described above, the gesture classification tasks are usually based on ML techniques. We also in our project make use of a ML approach, to manage the high variability in the input data.

During our experimentation, two different solutions were tested, that rely on different Machine Learning techniques. In the first project phase, we experimented a DTW based approach, as described in [15]. The second ML approach tested in the project is based on a classical neural network architecture. We successfully experimented a simple feed-forward neural network, designed as follows.

An output node is set for each gesture to be recognized, where the expected output value is converging to one for the actual gesture and converging towards zero for all other nodes.

As for the features to feed the network with, we found it effective to provide the network with the temporal succession of the values of each dimensional axis, in a single step. After the normalization stage, the detected gesture is in fact represented by a fixed number of samples for each axis of the gyroscope and for each accelerometer, regardless of the actual duration of the gesture. These values are provided directly to the neural network input nodes.

An input node is set for each subsequent data sample on each data dimension. The number of input nodes is therefore equal to six times the standardized gesture length: typically, we standardize the detected gestures to 50 samples and taking into account the three gyroscope axes and the three orthogonal accelerometers, we feed the network with 300 input nodes.

This solution is outlined in Figure 4.

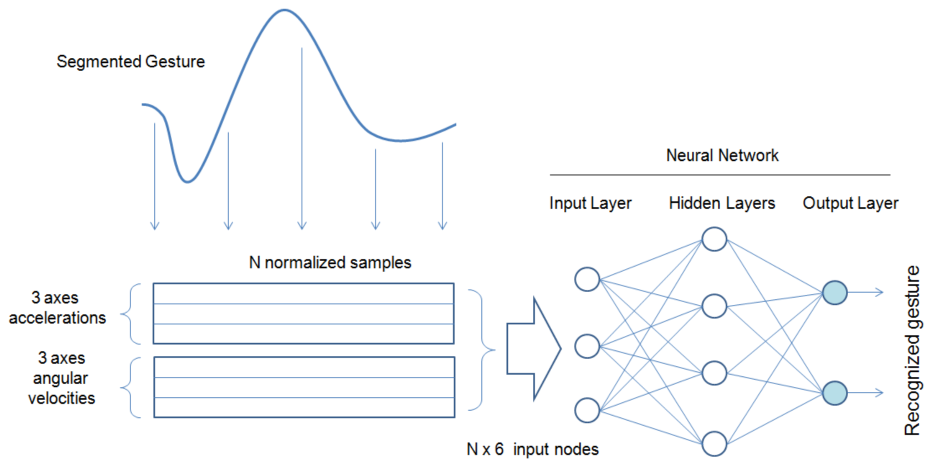

Fig. 4. The neural network classifier used in the gesture recognition process.

Each output node provides a value between 0 and 1 and the node with the highest value shows the gesture recognized by the classifier.

On the output values we considered two classification *confidence indexes*, to rate the result provided by the neural network.

The greatest value itself returned from the output nodes, provides a first confidence index. Then, by sorting the output values in descending order and taking into consideration the two greatest ones, we used the quotient between the second and the first value as a measure of uncertainty of the classification result proposed by the system. The second highest value represents the second most likely gesture according to the classification stage.

Values greater than 0.1 show a slight uncertainty, values greater than 0.5 denote a strong uncertainty. When the classifier should assign the same value to the best two output nodes, we have the same probability that the gesture belongs to one class or another and the uncertainty is equal to 1. For practical use, we choose to consider the results with uncertainty index lower than 0.1 as validly recognized gestures.

As for the management of the data variability depending on the type of gesture, in the current state of experimentation we find the gestures that resulted more difficult to recognize are the slow or small amplitude motions, like the “grab” gesture, which have a less clearly characterized pattern than others. Introducing this kind of motion in the gesture set, we observed that the classifier does not show a low successful recognition rate on that gesture, but rather the recognition rate gets worse on some other gestures, which are often confused with the new one.

Concerning the data variability in dependence on the different ways of performing the same gesture, the system behaviour was tested by setting up different training sessions, conducted by a different number of users.

We performed the training sessions starting from one system trainer up to five trainers. While adding more people to the classifier training sessions, the system reached a better performance rate with 3 trainers and showed a slight decay with 4 or 5 subjects. A possible explanation is that the different way of performing the same gestures reduces the separation between the ensembles of data trajectories that define each gesture, though on this aspect we believe that further experimentation is necessary.

The system was trained with a set of 15 gestures, as described in Section 3.1. Many real time test sessions were carried out in the company laboratory. The recognition software reached a performance ranging from 86% to 97% of correctly recognized gestures, depending from the single user and the gesture performing conditions.

Each test session included a sequence of 10 to 40 different gestures. In the longest test sequences, users showed a decreasing accuracy in the hand movement execution, leading to a slight decline in the recognition success rate.

The user independence of the recognition system was also successfully tested on random people at an exhibition fair, after a brief instruction about the gestures they could perform.

6. Context-Aware Adapter

After the recognition phase, an *adapter module* is placed which acts as a joint with the application interfaced, to enable a flexible association of learned gestures to an arbitrary target system.

This adapter is performing the function of a *semantic layer*, which gives the gestures a meaning that is relevant in the context of the specific interfaced application, translating gestures into actions that make sense for the application. This layer therefore allows giving the same gesture a different meaning according to the controlled application (e.g., the selection of an option in a graphic user interface or a command issued to a robot).

A sort of vocabulary, or correspondence map, translates each gesture into an action performed on the application. This vocabulary is used to define and adapt the gesture-based interface, based on the given usage context.

We however noticed that even dealing with a single application, the set of available gestures is often too limited to control all the desired functions. We have two possible solutions to this problem: i) to provide a wider gestures set (e.g., the letters of an alphabet) or ii) to translate the same gesture in different ways according to the application workflow context. This second solution seems to preserve the naturalness of the gesture, which is a characteristic of gestural interfaces, without requiring the user to learn the associations of various meanings with a series of abstract symbols.

The meaning generated by the semantic layer therefore depends both on the events generated by the user (gestures) and on the state of the application context, such as, for example, the presence or absence of an item, or the effects produced on the environment by previous gestures.

In our system, this solution is implemented by introducing the definition of different "modes" for each application and associating a different vocabulary to each mode. In this way, a gesture produces a certain effect depending on the current mode. For each mode, a separate configuration is set up. A mode can model an application workflow stage or a context situation, so that the same gestures set can be used for different purposes based on the context.

The solution operates like a finite state machine, where the transitions between modes can take place either through one action set in the vocabulary or based on the state of the interfaced application.

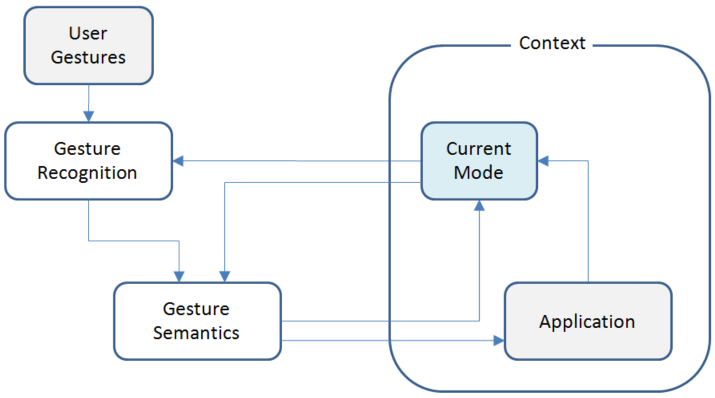

Fig. 5. Interactions of the context-aware gesture semantics module.

The semantic layer is implemented by an adapter module that controls the specific behaviour of a single application. As shown in Figure 5, this module is triggered by user gestures, taking into account the current context, to generate the actions that drive the application and possibly change the mode state. The current mode may affect the recognition process too, limiting the valid gestures to a reduced set of allowed options.

The first version of our application adapter was configured to control a 3D modelling software, adding as actions to the vocabulary the mouse pointer movements, the mouse click and some keyboard shortcuts. For this purpose, the application adapter software directly controls the mouse and keyboard actions of a generic graphic user interface. In this scenario, the "Shift", "Move" and "Edit" modes were defined, which respectively allow to move the pointer and create or select an object, to rotate or move an object and to modify an object by extruding its size along one axis.

The system response can be adapted based on the application workflow state and on the current environment situation. The same gesture, for example a rightwards translation, can mean to "move the pointer", or to "move the object", or to "extrude the object", depending on the situational awareness about the objects placed in this virtual space the user can interact with, and on the previous actions performed on them (e.g., selecting the object or activating a menu option).

A reduction in the gestures set, by increasing the separation between the features space of each gesture, enhances the user independence and the recognition accuracy.

7. Future Works

Further planned developments include enhancing the ability to face gesture variability, in particular by improving the user independence and the recognition accuracy of *smaller or slower movements*.

The user independence, or more generally, the ability to deal with the differences within each gesture class, will be tackled by introducing an automated clusterization stage, where the different ways of performing the same gesture can be grouped as if they were different gestures. The recognition rate of small and slow movements will be tackled by enriching the features provided to the neural network with more information about each segmented gesture, such as the gesture duration and peak values, that are currently lost during the normalization stage.

Another aspect under development is the design of a *configurable adapter* module, to easy interface the gesture controller with different applications, without limiting the gesture recognition solution to the scope of the *Seamless* environment. A visual tool is now under study to allow customers to easily define the application adapter configuration, defining the different modes and associating for each mode the available gestures with the desired actions.

8. Concluding Remarks

The paper has presented a prototype about gesture recognition, relying on a ML approach, based on a neural network classifier. In particular, we have illustrated the processing steps and the obtained results.

The adoption of a supervised ML technique has proved effective in handling the variability of different types of gesture patterns, performing a continuous data flow analysis in real time. The prototype tested the feasibility of a user-independent gesture recognition that is based only on inertial data.

The design of a context-aware adapter allowed the gesture control system to adapt to different application contexts based on its configuration and provided the system the ability to switch between different operating modes, so as to issue a larger set of instructions with fewer gestures.

References

1. Vuletic, T., Duffy, A., Hay, L., McTeague, C.P., Campbell, G., Choo, P.L., Grealy, M.: Natural and intuitive gesture interaction for 3D object manipulation in conceptual design. In: DS 92, Proceedings of the DESIGN 2018 15th International Design Conference, pp. 103-114 (2018).
2. Roda-Sanchez, L., Olivares, T., Garrido-Hidalgo, C., Fernández-Caballero, A.: Gesture control wearables for human-machine interaction in Industry 4.0. In: International Work-Conference on the Interplay Between Natural and Artificial Computation, pp. 99-108, Springer, Cham (2019).
3. Al-Shamayleh, A.S., Ahmad, R., Abushariah, M.A., Alam, K.A., Jomhari, N.: A systematic literature review on vision based gesture recognition techniques. Multimedia Tools and Applications, 77(21), 28121-28184 (2018).
4. Rautaray, S.S., Agrawal, A.: Vision based hand gesture recognition for human computer interaction: a survey. Artificial intelligence review, 43(1), 1-54 (2015).
5. Singha, J., Roy, A., Laskar, R.H.: Dynamic hand gesture recognition using vision-based approach for human–computer interaction. Neural Computing and Applications, 29(4), 1129-1141 (2018).
6. Zhao, J., Tanaka, J.: Hand gesture authentication using depth camera. In: Future of Information and Communication Conference, pp. 641-654, Springer, Cham (2018).
7. Jiang, D., Zheng, Z., Li, G., Sun, Y., Kong, J., Jiang, G., Liu, H.: Gesture recognition based on binocular vision. Cluster Computing, 22(6), 13261-13271 (2019).
8. Gao, X., Jin, Y., Dou, Q., Heng, P.A.: Automatic gesture recognition in robot-assisted surgery with reinforcement learning and tree search. arXiv preprint, arXiv:2002.08718 (2020).
9. Fang, B., Sun, F., Liu, H., Liu, C.: 3D human gesture capturing and recognition by the IMMU-based data glove. Neurocomputing, 277, 198-207 (2018).
10. Xie, R., Cao, J.: Accelerometer-based hand gesture recognition by neural network and similarity matching. IEEE Sensors Journal, 16(11), 4537-4545 (2016).
11. Hsu, Y.L., Chou, P.H., Kuo, Y.C.: Drift modeling and compensation for MEMS-based gyroscope using a Wiener-type recurrent neural network. In: 2017 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL), pp. 39-42, IEEE (2017).

12. Krishna, G.G., Nathan, K.S., Kumar, B.Y., Prabhu, A.A., Kannan, A., Vijayaraghavan, V.: A generic multi-modal dynamic gesture recognition system using Machine Learning. In: Future of Information and Communication Conference, pp. 603-615, Springer, Cham (2018).
13. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD workshop, vol. 10, No. 16, pp. 359-370 (1994).
14. Gupta, H.P., Chudgar, H.S., Mukherjee, S., Dutta, T., Sharma, K.: A continuous hand gestures recognition technique for human-machine interaction using accelerometer and gyroscope sensors. IEEE Sensors Journal, 16(16), 6425-6432 (2016).
15. Fugini, M., Finocchi, J., Trasa, G.: Gesture recognition using dynamic time warping. In: 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), IEEE (2020).
16. Fugini, M., Bonacin, R., Martoglia, R., Nabuco, O., Sais, F.: Web2Touch 2019: Semantic technologies for smart information sharing and web collaboration. In: 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 255-258, IEEE (2019).
17. Raibulet, C., Drira, K., Fugini, M., Pelliccione, P., Bures, T.: Software architectures for context-aware smart systems (2019).
18. Fugini, M., et al.: WorkingAge: Providing Occupational Safety through Pervasive Sensing and Data Driven Behavior Modeling. In: 30th European Safety and Reliability Conference (ESREL 2020), pp. 1-8, Research Publishing (2020).
19. Bigdelou, A., Schwarz, L., Benz, T., Navab, N.: A flexible platform for developing context-aware 3D gesture-based interfaces. In: Proceedings of the 2012 ACM international conference on Intelligent User Interfaces, pp. 335-336 (2012).
20. Hakim, N.L., Shih, T.K., Kasthuri Arachchi, S.P., Aditya, W., Chen, Y.C., Lin, C.Y.: Dynamic hand gesture recognition using 3DCNN and LSTM with FSM context-aware model. Sensors 19.24, 5429 (2019).