Using semantics to automatically generate speech interfaces for wearable virtual and augmented reality applications

# Using Semantics to Automatically Generate Speech Interfaces for Wearable Virtual and Augmented Reality Applications

Fabrizio Lamberti, *Senior Member, IEEE,* Federico Manuri, Gianluca Paravati, *Member, IEEE,* Giovanni Piumatti, and Andrea Sanna

***Abstract*—This paper presents a framework for automatically generating speech-based interfaces for controlling virtual and augmented reality applications on wearable devices. Starting from a set of natural language descriptions of application functionalities and a catalog of general-purpose icons, annotated with possible implied meanings, the framework creates both vocabulary and grammar for the speech recognizer, as well as a graphic interface for the target application, where icons are expected to be capable of evoking available commands. To minimize user's cognitive load during interaction, a semantics-based optimization mechanism was used to find the best mapping between icons and functionalities and to expand the set of valid commands. The framework was evaluated by using it with see-through glasses for AR-based maintenance and repair operations. A set of experimental tests were designed to objectively and subjectively assess first-time user experience of the automatically-generated interface in relation to that of a fully personalized interface. Moreover, intuitiveness of the automatically-generated interface was studied by analyzing the results obtained through trained users on the same interface. Objective measurements (in terms of false positives, false negatives, task completion rate and average number of attempts for activating functionalities) and subjective measurements (about system response accuracy, likeability, cognitive demand, annoyance, habitability and speed) reveal that the results obtained by the first-time users and experienced users with the proposed framework's interface are very similar and their performances are comparable to those of both the considered references.**

***Index Terms*—Virtual reality, augmented reality, wearable devices, automatic user interface generation, speech, semantics.**

## I. Introduction

RECENT years witnessed tremendous developments of technology for virtual and augmented realities (VR and AR), with an ever increasing number of big IT players entering the market in different ways. Some companies developed their own all-in-one devices, such as Epson with its Moverio[1] glasses, or Microsoft with its HoloLens[2] project. Other companies adopted a kind of integration strategy by designing products that need to be paired with users' smartphones, such as the Gear VR[3] by Samsung or the Cardboard[4] by Google. There are also companies that acquired enabling technology developers, such as Apple that acquired Metaio[5], or Facebook that recently bought Oculus[6].

To date, VR and AR have already been exploited in a wide variety of fields, encompassing entertainment [1], health [2], engineering [3], manufacturing [4], logistics [5], and transportation [6], to name a few and new successful use cases are advertised every day. With the continuous expansion of related technologies at the consumer level, it is not difficult to foresee an explosion of opportunities for VR and AR in the near future, leading to the creation of even more powerful solutions and their application in a growing number of domains.

Despite such huge potential, a lot remains to be done to let users properly interact with virtual and augmented contents in all possible usage scenarios. Interesting reviews of existing user interfaces (UIs) for VR and AR are available [7], [8], where interfaces are generally categorized into the following main groups: text-based, tangible, haptic and tactile, gaze-based, visual (e.g., relying upon gesture recognition), and aural (e.g., using speech recognition). It is worth noting that such reviews did not consider other UIs that are currently being experimented, e.g., the UIs based on brain or other biological signals.

The extreme heterogeneity of envisioned application scenarios does not allow any specific category of UIs to be identified as the dominant one. As a consequence, the last few years have witnessed an incredible proliferation of UIS implemented on ad-hoc basis, which renders reusability of results extremely hard to pursue. Although a number of guidelines have been developed [9], [10], often derived from general human-computer interaction design principles, turnkey solutions for developing UIs comparable to those available for developed UIs, e.g., for creating graphic UIs, are still scarce. A conventional approach is to provide the developers with APIs implementing basic functionalities (e.g., for gesture recognition, eye tracking, etc.) and then leave them with the burden of building the UI.

Moving from the above considerations, this paper aims at presenting the design of a framework that renders creation of interfaces for controlling VR and AR applications easy. Given the growing demand for mobile and wearable solutions,

Authors are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Torino, Italy. E-mails: {fabrizio.lamberti, federico.manuri, gianluca.paravati, giovanni.piumatti, andrea.sanna}@polito.it

Manuscript received Xxxx XX, XXXX; revised Xxxx XX, XXXX.

[1]http://www.epson.com/moverio

[2]https://www.microsoft.com/microsoft-hololens/

[3]http://www.samsung.com/eg/gearvr

[4]https://www.google.com/get/cardboard/

[5]https://www.metaio.com/

[6]https://www.oculus.com

it was decided to focus the research on speech UIs, which are applicable to both the contexts.

Speech interaction is natural [11] and can be combined with other interaction modes to devise multi-modal UIs. It also allows users to handle hands-free tasks, which might be essential in a number of application fields, like maintenance, medical practice, vehicles operation, etc. [12]. However, speech interface has its own drawbacks [13] relating mainly to response time (given the serial nature of voice commands), recognition of errors (due to ambient noise or conflicting audio patterns) and cognitive load (as the user needs to learn and recall correct vocabulary and grammar). Some of these drawbacks can be addressed by technological advances, but others only by an appropriate design methodology [14]. The commands should be so chosen as to be close to the intended meaning, and the functionality should be the same for more than one command. Moreover, the commands should be accompanied by graphical representations that can leverage on the user's visual memory [15]. Unfortunately, the developers are provided with only the basic tools in the above steps [16], [17].

To effectively meet the developers' needs, the proposed framework was so designed that it can support the creation of speech interfaces where application functionalities can be activated by a flexible set of voice commands, which can also be represented visually in a concrete graphic interface. The developers were asked to provide a description of the interface and its functionalities by means of short phrases. The symbols to be used in the graphic interface were selected from a library containing icons and associated meanings. The library was developed from a publicly-available set of icons, and by asking the users, who were unaware of the final goal, to provide a description of their content or functionality by way of responding to an online questionnaire.

Functionality and icon descriptions were processed by a semantic-based optimization mechanism, which selected the icons with the highest probability of evoking a given functionality and simultaneously minimized the use of icons with implied meanings, shared by other icons, besides preserving visual consistency throughout the interface. The phrases associated with the selected icons and the corresponding functionalities were then syntactically and semantically expanded (to enhance flexibility in recognition) and their ambiguities removed (to enforce a one-to-one command in functionality mapping). The results were then considered as valid commands for the speech recognizer.

The framework was designed to work independent of the application to be controlled and the technology used. The effectiveness of the framework was evaluated by using it in an ongoing investigation of the European Project EASE-R3[7]. The designed framework was exploited particularly to automatically generate a speech interface for a wearable-based AR application running on the Epson Moverio BT-200[8] smart glasses, used in maintenance and repair of machine tools [18]. The quality of the automatically-generated interface was assessed by comparing it with a user-generated interface, in terms of the impact of visual cues and semantic processing, and by evaluating the first-time user experience (FTUE).

The remaining part of this paper is organized as follows: Section II provides a review of related work; Section III presents the overall architecture of the designed framework, and Section IV discusses the experimental observations. Finally, Section V sums up the conclusions drawn from this study, followed by suggestions for future research.

## II. Related Work

Literature relevant to the current work encompasses various research areas, ranging from interaction with VR and AR applications, through speech interfaces, and finally to automatic interface generation techniques.

### A. Interaction with VR and AR Applications

Spread of VR and AR remained stifled for a long time by technological constraints, but recent developments in mobile and wearable technologies significantly boosted the creation of VR and AR solutions. Today, affordable, head-mounted and see-through displays are available for many consumer-oriented purposes, like gaming, tourism, education, etc. [19], [20], [21].

While opening up enormous possibilities, the technological evolution also brought to the fore significant challenges regarding human- machine interaction. In fact, many paradigms used in traditional settings are no more applicable to mobile and wearable devices. Hence, researchers are experimenting with different modalities, often combining some of them into mixed interfaces and performing comprehensive user studies to determine their applicability in relevant scenarios [22].

Multi-modal interfaces proved their viability in supporting complex interaction tasks [23]. It is well known that, in dealing with these tasks, if the design is accurate, it can benefit from the advantage of a specific interaction modality, besides simultaneously limiting the impact of its drawbacks. For instance, mouse and gestural commands can be effectively exploited for direct object manipulation (e.g., for positioning virtual elements using 2D or 3D coordinates), while voice can be used at the same time for descriptive tasks (e.g., for choosing what to work with). Thus, for instance, Irawati *et al.* used speech and hand gestures for interacting with virtual furniture in a wearable-based AR application in [24]. A similar approach was adopted by Koelsch *et al.*, who used voice commands in combination with a hand-held trackball to interact with AR contents during maintenance and emergency rescue tasks [25].

On the other hand, uni-modal interfaces are generally exploited in simpler interaction scenarios. Common uni-modal approaches for interacting with large-scale VR solutions exploit speech, wands, laser pointers, as well as hand and body gestures, whereas AR applications on mobile devices are often managed by means of native touch controls and voice commands [26], [27]. The foregoing modalities are not always applicable to setups based on wearable devices. In fact, some situations require only hands-free operations, e.g., when executing maintenance, medical or other hazardous tasks [14] or improving service productivity and efficiency [28]. In these situations, generally speech interaction is considered.

[7]http://www.easer3.eu/

[8]http://www.epson.com/cgi-bin/Store/jsp/Product.do?sku=V11H560020

### B. Speech Interfaces

Speech interfaces can be exploited to design two broad classes of applications, namely dictation and command & control [29]. Applications relevant to the current work belong to the second class. The spectrum on command & control interfaces ranges from simple prompt and response interfaces to full sentence conversational agents, which have been experimented in a wide range of application scenarios encompassing air traffic control [30], medical practice [31], military operations [32], etc.

Independent of the targeted application, using a speech interface has to address several challenges, such as converting the acoustic voice signal into a sequence of words and translating voice strings into a format that is understandable to machines. Besides such technological challenges, which are beyond the scope of this work, a key conceptual issue is cited as one of the major causes for poor user experience of those who use speech interfaces: "How to make the users aware of what they can say?" [14]. This problem is the same as the problem one faces in passing commands from command-line to graphic interfaces. In command-line interfaces, the boundaries of what can and what cannot be done, i.e., the functionalities available become evident only on knowing the overall set of issuable commands [29]. Hence, graphic interfaces were designed with the goal of visually displaying such functionalities to the user through evocative graphic signs.

In many speech-only applications, like automatic interactive voice response systems, the above problem is tackled by helping the users in the interaction with available commands, using the so-called vocal prompts [33]. But, this approach has a limitation: the speech output is slow, and the users might, therefore, find it difficult to remember all the available commands [34].

When a speech interface is used in combination with a display, cues about available commands may, actually should, be provided. The cues can leverage on the user's visual memory, thus reducing the cognitive load and limiting, at the same time, interference with the verbal composition processes. This principle is ignored in what is often referred to as a "voice command bar" [15], where text labels are used to display valid commands. The bar acts both as a reminder and as an education tool, making the learning curve smoother and the commands easier to remember. Effectiveness of this paradigm, first introduced by Kurzweil Applied Intelligence[9] and known as "say what you see", has been demonstrated in different scenarios [35], [36]. Building on the importance of providing a visual representation for voice commands, Danis *et al.* argue that, for creating a successful speech interface, the commands must be made mentally available in the easiest way, not only by always displaying the words on the screen, but also by showing to the user more examples of what can be said [37]. This approach has been successfully adopted in the context of AR also [14].

As the number of commands increases, implementing a text-based voice command bar becomes harder. Hence, text labels have to be replaced by icons [38], assuming that each icon is suggestive of one or more voice commands. However, even after using visual cues, mapping the issued commands to the expected functionalities is not a straightforward task. With this perspective, Shriver and Rosenfeld created the so-called universal speech interface by statistically analyzing users' preferences of the commands to be used for activating common functionalities [39].

[9]http://www.kurzweiltech.com/kai.html

### C. Automatic Interface Generation Techniques

Associating an icon with one or more voice commands is a rather common practice on desktop computers, where accessibility features are available in major operating systems. A similar consideration applies to the mobile world, where popular speech recognition-based personal assistants let the users control their devices using voice commands. However, although voice commands (with some exceptions) can be exploited in principle to control any application functionalities, their use is often limited to native applications, because third-party developers need to code all the interactions explicitly [40]. Addressing the above limitations, several approaches have been developed to automatically create a speech interface by interpreting what is displayed on screen [41], [40].

Automatic generation of UIs is pursued not just for accessibility purposes or only in the context of speech interfaces. In fact, the focus has been more on graphic UIs, with the aim of reducing development costs [42], enhancing usability [43], maximizing performance [44], supporting personalization [45] and fostering portability on heterogeneous devices [46].

Although, in most cases, the techniques for automatic interface generation have been designed to arrange graphic elements in a suitable layout or to let the user choose the best widget for a given interaction task, there are several works operating at the level of individual graphic elements and, more specifically, of icons. Icons play, today, a key role in the graphic interface, as they can convey, in a condensed form, the meanings of the functionalities they represent. Icons are especially important for novice users who use interactive systems only infrequently and are assumed to be capable of transcending language barriers [47].

However, drawing or selecting icons for an interface is not a trivial task. Ideally, icons should be capable of activating proper mental models in the users, but this is not always the case, mainly because of the phenomenon known as icon "ambiguity": different individuals may interpret the same icon in different ways, thus associating it with different functionalities. In fact, icon interpretation depends both on the actual user's mental model and on other factors, such as context, culture and experience.

A common approach adopted in selecting or drawing icons is to ensure that the "articolatory distance" between the physical form and the implied meaning is minimal [48]. For instance, the icon that shows a person discarding trash will have a small articolatory distance, because the iconic object is self-explanatory. On the contrary, for an icon with a power button, the distance has to be much higher, because the graphic symbol is rather abstract and its effect on the controlled application

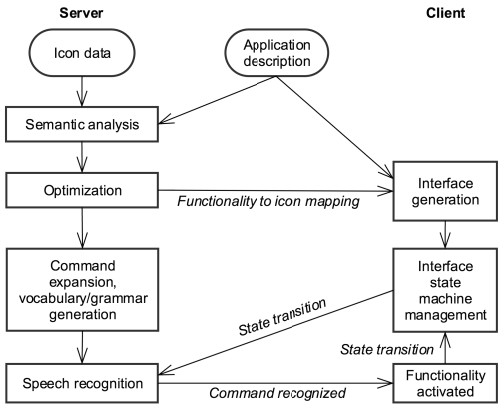



Fig. 1. High-level architecture and interactions between client and server.

needs to be learned or discovered. The steps that need to be taken for minimizing this distance and for making the criteria for an icon to be effective, include, among others, legibility, distinctiveness, comprehension, memorability, familiarity, and reaction time [49], [50]. The strategies proposed for assessing icons date back to 1960's and include appropriateness, preference rating, naming and matching, comprehension, recognition and recall as well as intuitiveness tests [51], [52], [53].

Given the complexity of design and selection tasks, the research efforts have been devoted to finding a way to automate the process. For instance, Morioka proposes an approach to automatically generate complex icons by combining basic graphic patterns [54]. Keogh *et al.* develops a methodology for replacing the icons of a file system with automatically-created icons that reflect the actual content of files [55]. Oda and Itoh propose a somewhat similar strategy to automatically generate icons for music files by matching tunes with images [56]. Setlur and Mackinlay designs a method to automatically generate large icon libraries for use in the visualization of categorical data [57], whereas Pickering addresses the problem of icon selection, with the aim of enforcing consistency in the usage of graphic signs within, as well as across, applications, and enhancing text expressiveness [58].

## III. Proposed Framework

The proposed framework was structured on the lines of client/server architecture, as shown in Fig. 1. The server is responsible for generating the target application interface, including the vocabulary and grammar for the speech recognizer, as well as the corresponding icon-based visual cues. This off-line process can be performed only once per each application. At run-time, the server manages the speech recognition engine and communicates to the client the functionalities to be activated, based on the commands issued. The client reacts by activating the specific functionality and notifying the server about interface changes. The design strategy adopted here significantly reduces the dependence on the client, thus allowing the target application to be implemented using different technologies and/or programming languages.

### A. Server Architecture

The off-line interface generation process, on the server side, receives as input a description of functionalities of the target application together with data about the available icons. Its output is a mapping that associates each icon (to be displayed on the application interface) to a specific functionality and a set of implied speech commands (each expected to be evoked by the corresponding icon).

Application functionalities are described by the developer, using short phrases of one or more words. Icon data comes from an annotated catalog, which was created by interviewing generic subjects and collecting short descriptions of their possible meanings. Application and icon descriptions are semantically analyzed to measure the correlation between functionalities and icons (defining the extent to which their meanings are related), as well as the distance among icons (defining the extent to which their meanings are unrelated). The mapping is determined by an optimization method, which considers the outcome of the semantic analysis to maximize interface flexibility and robustness, while preserving visual consistency. Finally, voice commands are generated by lexically and semantically expanding phrases associated with the functionalities and icons included in the mapping. In the following pages, the modules and methodologies developed for realizing the above process are described in detail. To facilitate easy understanding, the presentation will often refer to the case study of AR maintenance, introduced in Section I.

*1) Application Description:* The developer is expected to provide a description of all client application functionalities using a state machine model. The state machine was assumed to represent the flow of application interface, indicating which functionalities would be available at any given moment and how they affect the interface when activated. Fig. 2 is a graphical representation of the state machine for the application considered in the case study.

The state machine is described using an extension of the W3C's State Chart XML (SCXML) notation[10]. Fig. **??** is an excerpt of the SCXML description that could be provided by the developer for the considered application. Functionalities are described as events (attribute `event`) that activate transitions (tag `<transition>`) between states (tag `<state>`). For each functionality, one or more phrases are provided using the `<phrase>` tag. Moreover, the developer may specify relationships among functionalities by defining the so-called affinity groups with the `<affinity-group>` tag. For instance, "next_step" and "previous_step" in Fig. 2 are two affine functionalities, as they are complementary. Similarly, "play_video", "pause_video" and "stop_video" are affine, because they are all related to video control.

*2) Icon Data Collection:* The proposed methodology requires a catalog of icons, annotated with implied functionalities. The catalog should, in principle, be sufficiently comprehensive to include all the icons required to create the interfaces of various applications. Indeed, a number of icon sets exist in which each icon is assigned one or more significant names. However, what the proposed method needs

[10]http://www.w3.org/TR/scxml/

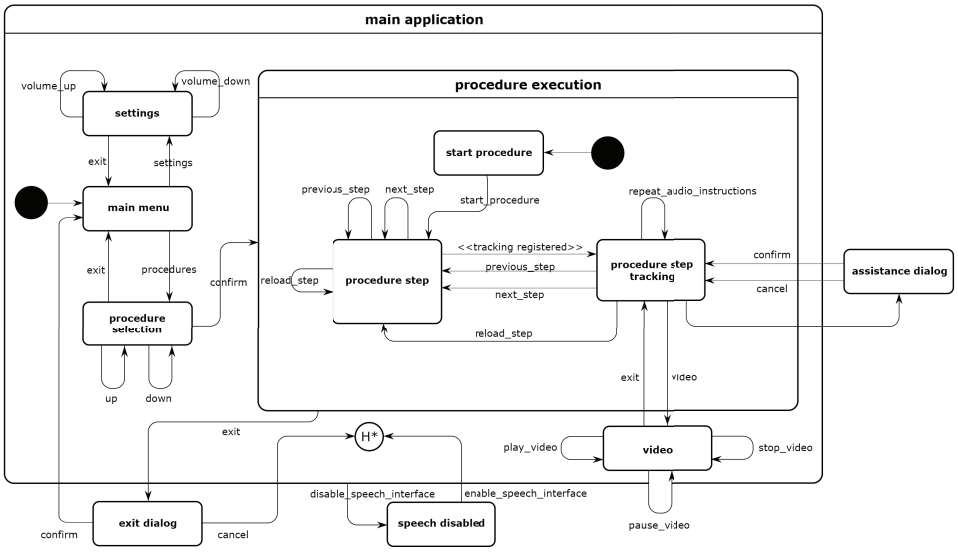



Fig. 2. State chart representing the interface of the application for AR maintenance considered in the used case (see Section I).

```
<scxml initial="main menu" app="ARMaintenance">
  <state id="procedure step tracking">
    <transition event="previous_step" target="procedure step"/>
    <transition event="next_step" target="procedure step"/>
    <transition event="assistance" target="assistance dialog"/>
    <transition event="video" target="video"/>
    <transition event="exit" target="exit dialog"/>
      ...
  </state>
  ...
  <event name="next_step">
    <phrase value="avanti"/>
    <phrase value="passo successivo"/>
  </event>
  ...
  <affinity-group type="move state">
    <event value="previous_step"/>
    <event value="next_step"/>
  </affinity-group>
</scxml>
```

Fig. 3. Developer-provided state machine for the interface in Fig. 2 showing the functionalities (and text-based descriptions) and relations among them.

is a collection of meanings, each expressed as a phrase of one or more words, including frequency statistics. Such an annotated catalog was created by using approaches that are rather common in usability tests of graphic interfaces [53], where one or more sets of alternative visual signs are evaluated through questionnaires and user studies. Instead of having the icons drawn from scratch by a graphic designer or assembling them from multiple sources on the Web, a publicly-available set named Clear Icons[11], comprising 50 general-purpose icons, was used. Then, an online questionnaire was designed, presenting the icons in a random order. Participants (selected among university students) were asked to provide, for each icon, one or more phrases describing the functionality that could be possibly activated in a generic device, application or appliance. The questionnaire was returned by 28 subjects and each icon received, on average, 41 descriptions[12].

*3) Semantic Analysis:* The process of semantic analysis exploits information from the MultiWordNet 1.5.0 lexical database [59], an Italian version of the well-known WordNet thesaurus [60]. As has been mentioned, the goal was to calculate the functionality for icon correlation and the icon to icon distance measure for exploitation in the optimization module. Phrases associated to every icon and functionality were processed to remove stop words. Then, the remaining words were linked to lemmas in the lexical database by applying stemming and lemmatization mechanisms. For each word, the part of speech was recorded, together with the details about how the lemma was obtained (e.g., word found as is, conjugated verb, etc.). With such information, correlation $w_{ij}^{corr}$ between icon $g_i$ and functionality $f_j$ was computed as follows. First, considering the lemmas, all the synonyms and antonyms were extracted. Then, a similarity index $sim(p_s, p_t)$ for two generic phrases $p_s$ and $p_t$ composed of $N_s^W$ and $N_t^W$ words was defined by considering the number of matching lemmas ($n_{st}^L$), synonyms ($n_{st}^S$) and antonyms ($n_{st}^A$), for each word in the two phrases. If $n_{st}^A > 0$, then $sim(p_s, p_t) = 0$. When $n_{st}^A = 0$ and $n_{st}^L = N_s^W$, then $sim(p_s, p_t) = 1$. Otherwise, if $min(n_{st}^L + n_{st}^S, N_s^W) = N_s^W$, then $sim(p_s, p_t) = (n_{st}^L + 0.5 \cdot n_{st}^S)/(n_{st}^L + n_{st}^S)$; if $n_{st}^L + n_{st}^S \neq 0$, then $sim(p_s, p_t) = (n_{st}^L + 0.75 \cdot n_{st}^S)/[(n_{st}^L + n_{st}^S) \cdot max(N_s^W, N_t^W)]$.

The weights in the above equations were empirically defined to account for the fact that synonyms could include meanings that are far from those of the original word. Moreover, considering the fact that the number of words in the two sentences could be different, they also took into account the possible meaning of missing information. Finally, the correlation was obtained as

$$w_{ij}^{corr} = \frac{1}{N_i^P \cdot N_j^P} \sum_{u=1}^{N_i^P} \sum_{v=1}^{N_j^P} sim(p_u^i, p_v^j) \qquad (1)$$

where $p_u^i$ is the $u$-th phrase of icon $g_i$, whereas $p_v^j$ is the $v$-th phrase of functionality $f_j$. $N_i^P$ and $N_j^P$ are the total number of phrases associated to icon $g_i$ and functionality $f_j$, respectively.

Distance $w_{i_1 i_2}^{dist}$ between icons $g_{i_1}$ and $g_{i_2}$ was computed similarly, as 1 minus the icon-to-icon correlation value.

[11] http://appzgear.com/products/clear-icons.htm

[12] http://intelligenthci.altervista.org/questionario/

*4) Optimization:* The mapping process between icons and functionalities was formulated as an optimization problem. To define the problem, application description was considered as having been composed by $N^S$ states and $N^F$ functionalities. By defining $N^G$ as the overall number of available icons, a binary decision variable $x_{ij} \in [0,1]$ was introduced so that

$$x_{ij} = \begin{cases} 1, & \text{if icon } g_i \text{ is associated to functionality } f_j \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

The following constraints were enforced

$$\begin{aligned} \sum_{i=1}^{N^G} x_{ij} &= 1 \ \forall j = 1, 2, \dots N^F \\ \sum_{j=1}^{N^F} x_{ij} &\leq 1 \ \forall i = 1, 2, \dots N^G \end{aligned} \quad (3)$$

ensuring that each functionality was mapped exactly to one icon, and the same icon was not associated to more than one functionality. Two helper functions were also defined as

$$\begin{aligned} q_{sol}(i) &= \sum_{j=1}^{N^F} x_{ij} \\ q_{state}(i,s) &= \sum_{j \in \{j | a_j \in S_s\}} x_{ij} \end{aligned} \quad (4)$$

where $q_{sol}(i) = 1$ if icon $g_i$ is in the solution (i.e., associated to a functionality) and $q_{state}(i,s) = 1$ if icon $g_i$ is present in state $S_s$ (i.e., mapped to a functionality in that state).

The solution needs to maximize the correlation between functionality and associated icon, the icon-to-icon distance in a given state, and the so-called icon affinity (defined later in this Section).

The metrics for correlation and distance were derived in the semantic analysis step (see Section III-A3). Cost functions for correlation and distance are defined as

$$\begin{aligned} obj_{corr} &= max \sum_{i=1}^{N^G} \sum_{j=1}^{N^F} w_{ij}^{corr} x_{ij} \\ obj_{dist} &= max \sum_{s=1}^{N^S} \sum_{i_1=1}^{N^G} \sum_{\substack{i_2=1 \\ i_1 \neq i_2}}^{N^G} w_{i_1 i_2}^{dist} q_{state}(i_1, s) q_{state}(i_2, s) \end{aligned} \quad (5)$$

Overall correlation is the sum of the $w_{ij}^{corr}$ coefficients of all the functionality-icon pairs. Distance was calculated for every state, considering the distance coefficient $w_{i_1 i_2}^{dist}$ for each pair of icons $g_{i_1}$ and $g_{i_2}$ appearing in that state. Icon affinity is related to the visual aspect of icons. In fact, the icons' styles should be mutually consistent, especially when they are associated to related functionalities. For example, if the functionality "next_step" is associated to a "right arrow" icon, then the "previous_step" functionality, if present, should be associated to a "left arrow" icon with the same style. As for functionalities, icons can be associated to one or more affinity groups.

Affinity $w_{i_1 i_2}^{aff}$ between two icons $g_{i_1}$ and $g_{i_2}$ is calculated as the ratio between the number of affinity groups shared and the total number of groups defined for icons. By referring to $F_i^{aff}$ as the $i$-th functionality affinity group and to $N^A$ as the total number of groups specified for functionalities, the following cost functions can be defined

$$\begin{aligned} obj_{aff}^{overall} &= max \sum_{i_1=1}^{N^G} \sum_{\substack{i_2=1 \\ i_1 \neq i_2}}^{N^G} w_{i_1 i_2}^{aff} q_{sol}(i_1) q_{sol}(i_2) \\ obj_{aff} &= max \sum_{k=1}^{N^A} \sum_{i_1=1}^{N^G} \sum_{\substack{i_2=1 \\ i_1 \neq i_2}}^{N^G} \sum_{\substack{j_1, j_2 \in \{j | f_j \in F_k^{aff}\} \\ j_1 \neq j_2}} w_{i_1 i_2}^{aff} x_{i_1 j_1} x_{i_2 j_2} \end{aligned} \quad (6)$$

The first function represents the affinity of all icons belonging to the solution, and the second one the affinity among icons associated to functionalities of a particular affinity group. Although the overall affinity is already being maximized by the first function, the second one helps to speed up algorithm convergence by avoiding icons that do not match with affine functionalities.

The problem was solved by using a multi-objective meta-heuristic optimization approach to provide scalability in terms of number of available icons and number of functionalities, because solution space has a size of $N^G!/(N^G - N^F)!$. Implementation was created by using the jMetal [61] framework. The solution was represented by an array of integers, indicating icon indexes, whereas position in the array is associated to the functionality. The NSGA-II genetic algorithm was used with a polynomial mutation and a SBX crossover operator. The algorithm can obtain high-quality solutions, though convergence speed can still be improved, e.g., by adopting a different strategy for representing the solution.

*5) Command Expansion:* Once the optimal mapping was found, the last step in the generation of the interface consists in building the vocabulary and grammar to handle commands by the speech recognition engine. As conflicts can arise among commands, they were confined to a single state, and expansion was performed separately for each state. For each state and for each functionality-icon pair in that state, the algorithm considered all the phrases associated to both functionalities and icons. Lemmas linked to each word were expanded into synonyms and the verbs were conjugated to imperative form. Expanded terms were combined into new phrases by computing meaningful combinations. Phrases were weighted to account for their frequency and length by considering the way they were generated. Finally, the phrases were converted into commands by replacing some words (e.g., articles such as "a", "the", etc.) with wildcards that match any word the user pronounces, thus increasing flexibility during recognition. Once all the functionality-icon pairs in a state were processed, the associated commands were filtered, based on their weight. Potential overlaps among commands were solved so that, at the end of the process, each functionality would be associated to an independent set of commands.

*6) Speech Recognition:* This module was developed using the Microsoft Speech Platform[13], which presently supports 26

[13]https://msdn.microsoft.com/en-us/library/jj127858.aspx

languages. Implementation for this study operated in Italian language to limit recognition errors possibly caused by mispronunciation.

### B. Client Architecture

The client side of the system was implemented by a library that manages the user interface and the communications with the server. The target application adopted in the considered case [18] was developed, using the Metaio SDK [62] AR framework. Specifically, it was built using the AREL technology, which lets the user create the interface as an HTML page with the logic defined in JavaScript. Hence, the client library was implemented in Javascript. Nonetheless, it can be easily implemented by using different programming languages and in different environments, thus allowing the deployment of the devised interface generation mechanism on heterogeneous platforms.

Initially, the application configures the client library with the state machine of its interface (see Fig. 2, for the case selected to use). It also sets up hooks between interface functionalities and the codes that actually implement them. Once the application is started, the Javascript library connects to the server, requests the functionality to icon mapping, and creates the user interface. Finally, it moves to the first state of the interface (displaying the corresponding icons) and notifies the server.

When a state update is received, the server loads the associated vocabulary and grammar and starts the speech recognition engine. Every time a command is recognized with enough confidence, the client is informed of invoking the matching functionality, triggering its activation.

Fig. 4 depicts a screenshot of the client application (obtained by disabling see-through mode). Maintenance operations are carried out in stereoscopic mode, which allows the user to focus both on the object to be maintained and the 3D assets used as AR hints. However, it was discovered that most users find it difficult to focus the icons shown in that mode, because they lie on a virtual plane at distances different than those of both virtual and real contents. Therefore, by adopting the same approach as the one pursued for native icons of the Moverio glasses, stereoscopy was disabled for icons. Icons are rendered on the sides, in such a way that, for instance, the icons on the left can be seen only with the left eye.

## IV. Experimental Results

The following Section assesses the effectiveness of the proposed framework in supporting the selected case in terms of wearable AR-based maintenance and repair operations.

Assessment encompassed three experimental tests. The first test has been designed to evaluate framework ability to create an interface where icons used are capable to evoke voice commands that, when issued by the user, will be recognized and will activate the expected functionality. The second test was carried out to measure the first-time user experience of the automatically-generated interface. In this case, intuitiveness and usability were evaluated through a set of performance indicators by comparing the first-time use of the application

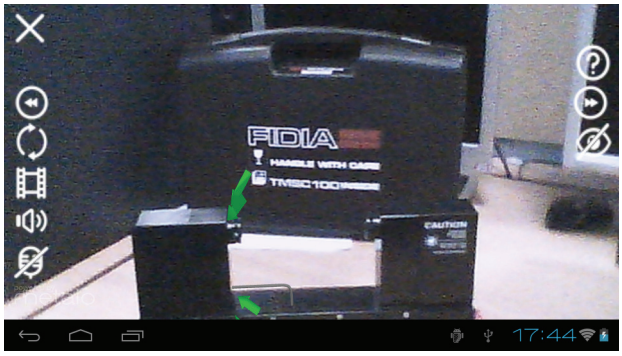



Fig. 4. Client application interface in the "procedure step tracking" state.

with a trained/experienced use. The third test was carried out to compare the FTUE of the interface created by the proposed framework with that of a sub-optimal fully-personalized one. In this case, the users were allowed to freely decide the set of icons to be used for building the graphic interface, as also the mapping between voice commands and application functionalities. To ensure fairness in comparison, expansion techniques were applied to both the command sets.

The hypothesis to be verified is that the first-time user performance of the interface generated with the devised framework is comparable to the trained user performance that can be achieved on the same interface, by using the sub-optimal interface.

### A. Setup

For this study, 45 participants ($24\%$ female, $76\%$ male) of an average age of 25 years ($\sigma = 4.8$) were selected among university students and equally distributed among the three tests. Less than half ($44\%$) of the participants had prior experience with speech recognition systems (mainly on smartphone devices and gaming consoles) and a few of them ($11\%$) with AR applications and/or wearable devices. Considering this information, the participants were evenly distributed among the three tests. None of them had any experience in maintenance and repair of machine tools. Though all the participants were native Italian speakers and the Italian language was adopted for the speech recognizer, the rate of command recognition could still be influenced by other factors, such as the difference between the backgrounds and origins of some participants. However, these participants were evenly distributed among the three tests. As a consequence, it was assumed that these differences in speech recognition performance did not influence the overall considerations. The setup for each test is described in detail below.

*1) First Test:* This test included a preparatory phase, in which the users were driven by an instructor into exploring the automatically-generated icon-based interface. The icons were indicated to the user, in a pre-defined order, with the aim of activating all the functionalities in the state machine (see Fig. 2). The user was asked to activate the functionality associated to each icon by choosing commands that he or she feels are evoked by the icon. In the following, this phase will be referred to as $T1^{(prep)}_{autom}$, where superscript $(prep)$ refers

to the preparatory nature, and subscript $autom$ to automatic creation of the interface.

The core of the first test was actually represented by the second phase in which, after the above training, the user was asked to complete the whole maintenance procedure without any direct supervision. In particular, the task consisted in performing regular maintenance of a high-precision laser meter. The task requires the user to follow a sequence of steps, from unscrewing and removing some parts of the tool to cleaning the internal lens. The user was shown a video that visually explains the operation[14]. Since the users already got acquainted with the interface during the first phase of the test, they could be considered as experienced users for the second phase. Hence, in the test that follows, this phase is referred to as $T1^{exp}_{autom}$, where $exp$ stands for experienced users.

*2) Second Test:* In this test, the users were verbally introduced to all the available functionalities, but they were not apprised about the available icons or about how to activate the associated functionalities (i.e., which are the commands recognized). After such introduction, they were asked to wear the AR glasses and to autonomously complete the above mentioned maintenance procedure by looking at the icons in the automatically-generated interface, deducing the associated functionalities, activating them and performing the operations suggested by the AR hints. Because there was no training on the interface, as in the first test, this test is referred to as $T2^{ft}_{autom}$, where superscript $ft$ refers to first-time users.

*3) Third Test:* In the preliminary step of this test, the users were asked to manually create their preferred interface. To this end, the users were presented with a description of all the application functionalities, and then asked to select, for each functionality, an evocative icon and a list of voice commands. The commands were lexically and semantically expanded. Afterwards, the users were asked to complete the same maintenance procedure as that of the previous tests by using the customized interface in an autonomous way. Hence, this test is referred to as $T3^{ft}_{custom}$. The $ft$ superscript indicates that, as in the previous test, first-time users are considered, whereas the subscript $custom$ refers to the use of the customized interface.

### B. Assessment Strategy

For each test, both objective and subjective measures were collected during interaction with the AR application, as well as after the execution of the maintenance procedure.

During the test, a report on system behavior was produced for each participant by logging the data about system behavior and recording the associated audio tracks containing the commands issued. A cross-analysis between data logs and audio tracks was carried out to extract quantitative information about user interaction and performance of the speech recognition module, including false positive and false negative error rates. Additional objective measurements made during the test include the average number of total attempts for activating a given functionality and the task completion rate. For the supervised phase in Test 1, a task was considered completed if all the functionalities corresponding to the icons indicated to the user were correctly activated. For this purpose, the maximum number of attempts per functionality was defined. When the system was used autonomously to carry out the designed task (i.e., in the second phase of Test 1, and in Tests 2 and 3), the task was considered completed if the participant succeeded in reaching the last step of the procedure without any help from the supervisor, say when he or she failed to activate a critical functionality, using no more than the finite number of attempts. A functionality is considered critical when, if not activated, it would not be possible to complete the task (in Fig. 2, critical functionalities are “procedures”, “confirm”, “start_procedure”, “next_step” and “exit”).

After the test, each participant was asked to evaluate the usability of the speech interface through a subjective questionnaire, based on SASSI methodology [63]. The questionnaire had 34 statements to be evaluated on a 7-point Likert scale. The statements refer to six usability factors: System Response Accuracy (SRA), Likeability (LIKE), Cognitive Demand (CD), Annoyance (AN), Habitability (HAB) and Speed (SPE).

### C. Evaluation Metrics

The tests were aimed at evaluating the number of recognized functionalities. A functionality is considered recognized whenever a participant succeeds in activating it. However, during the activation, several possible situations may occur, such as true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). In the following paragraphs, each situation is defined considering the perspectives of both the user and the system, which, in some cases, may differ. The user perspective considers the expected system behavior when a command is issued, whereas the system perspective considers the behavior of speech recognition.

A true positive (TP) occurs when a functionality is activated and, in fact, this is the one the user wanted to activate. This indicated that the system worked properly and the user perceived it as the correct behavior. This is the case, for instance, of a user who wanted to disable the vocal user interface. He or she saw an icon with a banned microphone and said “spegni il microfono” (“turn off the microphone”). The utterance was correctly recognized (correct behavior of the system) and mapped on the corresponding command, which activated the right functionality.

False detections (both positive and negative) can be divided into two categories based on the causes of their occurrence. When, because of a failure in the command expansion, the system activates the wrong functionality, the false detection is attributed to a semantic cause (SC). False detection can also be attributed to a technological cause (TC) when, for instance, noise or possible configuration issues negatively affect the performance of the speech recognizer.

A false positive (FP), or command switch, occurs when the activated functionality is not what the user expected. This kind of error occurs in two situations. The user might have pronounced the command that is actually mapped to the intended functionality but, for some reason, the system

[14]https://youtu.be/xAL0w4LgiIo

TABLE I
PERFORMANCE OF AUTOMATICALLY-GENERATED INTERFACE MEASURED AS THE ABILITY OF THE SYSTEM TO SELECT ICONS THAT CAN EVOKE PROPER MAPPING WITH RECOGNIZED COMMANDS AND EXPECTED FUNCTIONALITIES, EXPRESSED IN TERMS OF CORRECT BEHAVIOR ($TP$), ERRORS ($FP$ AND $FN$), COMPLETION RATE ($CR$) AND AVERAGE NUMBER OF ATTEMPTS PER FUNCTIONALITY ($n_a$).

| Test | $TP$ | $FP_s$ | $FP_t$ | $FN_s$ | $FN_t$ | $CR$ | $n_a$ |
|---|---|---|---|---|---|---|---|
| $T1^{(prep)}_{autom}$ | 69,1% | 2,3% | 0,6% | 14,8% | 13,1% | 73% | 1.50 |

TABLE II
PERFORMANCE OF THE AUTOMATICALLY-GENERATED INTERFACE, CONSIDERING BOTH FIRST-TIME AND EXPERIENCED USERS, IN TERMS OF CORRECT BEHAVIOR ($TP$), ERRORS ($FP$ AND $FN$), COMPLETION RATE ($CR$) AND AVERAGE NUMBER OF ATTEMPTS PER FUNCTIONALITY ($n_a$), REFERRED TO THE EXECUTION OF THE MAINTENANCE PROCEDURE.

| Test | $TP$ | $FP_s$ | $FP_t$ | $FN_s$ | $FN_t$ | $CR$ | $n_a$ |
|---|---|---|---|---|---|---|---|
| $T1^{exp}_{autom}$ | 83,8% | 1,5% | 1,0% | 2,5% | 11,1% | 86% | 1.18 |
| $T2^{ft}_{autom}$ | 71,9% | 0,9% | 0,3% | 15,7% | 11,1% | 85% | 1.42 |
| $T3^{ft}_{custom}$ | 63,4% | 2,5% | 0,4% | 30,6% | 3,1% | 60% | 1.61 |

recognized a different command, mapped to another functionality ($FP_t$). This may happen because of ambient noise or poor performance of the voice recognition module. Command switches also occur when the user pronounces a command that is not associated to the intended functionality ($FP_s$). For instance, to disable the voice interface, the user might say "volume" ("volume"), to which the system would respond by changing the headset volume.

A false negative (FN) occurs when the user tries to activate a functionality in the correct way (i.e., a command that should have been recognized the way it was issued), but the system fails to recognize it and, at the same time, the command it recognized, if any, has not been routed to activate another existing functionality. False negatives can be due to a technological reason ($FN_t$), e.g., when recognition confidence is too low. They could also occur when the speech recognizer returns a result that does not correspond to the user's command. For instance, the user might say "video" ("video"), but because of assonance, the system recognizes the command as "vedova" ("widow"), which is not mapped to any functionality. On the other hand, the reason for false negatives could be semantic ($FN_s$) when the utterance is not associated to any valid command. For instance, the user might want to start reproducing a video by saying "riproduci" ("reproduce"), but command to be used is "play video".

Finally, a true negative (TN) occurs when the system does not map the command issued to any functionality. The recognition is not successful, because the pronounced command is not valid. The icon is probably not appropriate, although the system worked properly. This could be the case, for instance, when the user sees an icon with a plus sign, which is actually mapped to the functionality to increase the headset volume, but the user says "zoom" ("zoom"). However, from the user's point of view, a true negative cannot exist, since every time he or she issues a command associated to a given icon, he or she expects the system to activate the corresponding functionality. Therefore, in the user's perspective, a true negative is considered a false negative with a semantic cause ($FN_s$).

### D. Objective Evaluation

To facilitate easy understanding and comparison, the objective data gathered during the tests is summarized in two separate Tables. Table I presents the results of the preparatory phase of Test 1 ($T1^{(prep)}_{autom}$), which can be analyzed to assess system's ability to create an interface with evocative icons by observing user's operations in supervised conditions. The data in this Table cannot be compared with the data pertaining to the second phase of Test 1 or with the data collected in Test 2 and Test 3, where the users were asked to interact with the system in an autonomous way. The data relating to autonomous interaction is presented in Table II. In this case, the data describes the execution of the same maintenance procedure under three different conditions, i.e., by experienced ($T1^{exp}_{autom}$) or first-time users ($T2^{ft}_{autom}$ and $T3^{ft}_{custom}$), using the automatically-generated (subscript $autom$) or by the customized (subscript $custom$) interface.

In executing the tests, each interaction was manually annotated to keep track of the errors and correct behaviors, which were classified, based on the user's perspective as defined in Section IV-C. For each test, columns 2–6, in both the Tables, give the number of true positives ($TP$), semantic and technological false positives ($FP_s$ and $FP_t$), and false negatives ($FN_s$ and $FN_t$), whereas columns 7 and 8 present the average task completion rate CR and the average number of attempts $n_a$ required for activating each functionality.

During the preparatory phase of Test 1 (Table I), the task completion rate was $73\%$, indicating that, on average, three users out of four were successful in activating all the functionalities in the automatically-generated interface. The participants committed errors mainly during the last phase of the task, and the errors belong to the false negative category, i.e., the participants were not successful in immediately activating a given functionality, consequent to which their commands were ignored by the system. This kind of error was equally distributed between the semantic ($FN_s$) and technological ($FN_t$) false negatives. Semantic false negatives occurred, for instance, when the user completed a particular step of the maintenance procedure and wanted to proceed further by activating the "next step" functionality, but said "fatto" ("done"), but the command was ignored by the system. Technological false negative occurred, for instance, when the user tried to open the settings panel by using the command "proprietà" ("properties"), which was falsely recognized by the system as "attività" ("activities").

In this case, false positives were scarce ($2.9\%$ of the total number of attempts) and this holds good for other scenarios too. In general, $2.3\%$ of the command switches were $FP_s$ errors. These errors occurred mainly in two practical situations: (i) When some participants used the command "opzioni" ("options") to activate the functionality for displaying the list of available procedures, and the system reacted by opening the settings menu, rather than the expected list; (ii) When the participants used the command "parti", in Italian, to activate the maintenance procedure, which means either "start" or

"leave", depending on the context to which the system responded by erroneously activating the functionality for exiting from the current state. Technological errors $FP_t$ constituted only the $0.6\%$ of the total number of attempts. An example of misunderstanding by the speech recognizer is distortion of a command like "ripeti" ("repeat") into an existing command like "parti" ("leave") and again forcing the exit from the current state.

Intuitiveness of the proposed user interface can be evaluated by comparing the results obtained by first-time users operating with the automatically-generated interface in $T2^{ft}_{autom}$ with those obtained by the two control groups represented by the users involved in $T1^{exp}_{autom}$ and $T3^{ft}_{custom}$.

In $T2^{ft}_{autom}$, the participants were asked to execute the maintenance procedure without prior training. The participants, belonging to the control group $T1^{exp}_{autom}$, were considered experienced users because of their learning in $T1^{(prep)}_{autom}$. This was confirmed by the fact that, as expected, the number of true positives largely increased between the first ($T1^{(prep)}_{autom}$) and the second ($T1^{exp}_{autom}$) phases of the first test. This trend was confirmed by the number of semantic false negatives ($FN_s$), which decreased drastically in the second phase of the test indicating that the number of failures in functionality activation attempts was low.

First-time users of the automatically-generated interface achieved performance comparable to that of trained users. This is confirmed by the average completion rate, which is almost the same in $T2^{ft}_{autom}$ and $T1^{exp}_{autom}$ ($85\%$ vs. $86\%$), attained at the cost of a higher average number of attempts per functionality due to a higher number of semantic false positives ($1.42$ vs. $1.18$, on average). The participants had to try several commands to activate a given functionality but, in the end, they completed the maintenance procedure just as experienced users.

During the test $T3^{ft}_{custom}$, the users built their own personalized interface by selecting icons and commands for each functionality. Even then, the completion rate was higher when the automatically-generated interface was used in $T2^{ft}_{autom}$ ($85\%$ vs. $60\%$). This is reflected by the lower number of user attempts for activating a functionality ($1.42$ vs. $1.61$, on average), which means that, on average, the users had to issue fewer commands to complete the task with the automatically-generated interfaces than with the personalized one. Similar inferences can be drawn by looking at error percentages. The most common problems in $T3^{ft}_{autom}$ were due to the fact that the participants forgot the commands and tried to use the commands associated to other functionalities, disregarding the importance of visual cues.

### *E. Subjective Evaluation*

Subjective usability scores collected with the SASSI questionnaire [63] were mapped on a better-to-worse scale (7-to-1) and averaged per factor. The results obtained show that the user experience was not the same during the three tests (see Fig. 5).

Considering the aggregated scores (bars labeled with TOT.AVG. in the plot), two considerations could be made intuitively.

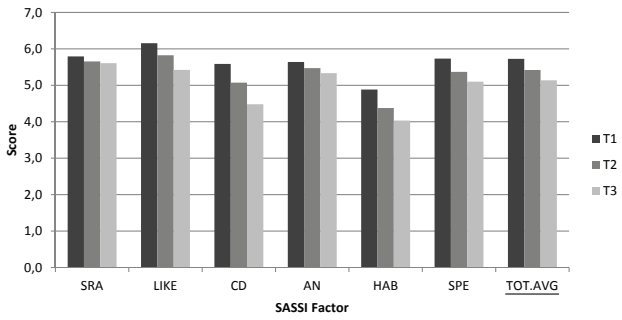



Fig. 5. Results of subjective evaluation based on SASSI methodology [63].

First, the system apparently performed better in Test 1 than in Test 2. This could be because, during Test 1, the participants had the chance to get acquainted with the system before using it autonomously. Second, in Test 3 (i.e., with the control group that used the personalized interface), the usability was lower than that in Tests 1 and 2 (i.e., when the automatically-generated interface was used). In fact, although participants were allowed to choose their preferred icons and commands, their satisfaction was low. The system failed to adequately expand the commands, and the manually-selected icons could not evoke the selected command. Consequently, only a subset of the commands issued were actually recognized by the system.

To confirm the statistical significance of the foregoing results, a one-way repeated measures ANOVA test was carried out. The null-hypothesis $H_0 : \mu_{T1} = \mu_{T2} = \mu_{T3}$ was that the users had the same overall experience during the three tests. The mean square between evaluations is $MS_B = 1.298$ and that within evaluations $MS_W = 0.221$. Therefore, the $F$ statistic is $F = MS_B/MS_W = 5.88$. The results reveal that the average scores are significant at $\alpha < 0.05$ level, $F(2, 42) = 5.14$, $p = 0.01$. Since the $p$-value obtained from the $F$-distribution is lower than the significance level $\alpha = 0.05$, the null hypothesis was rejected, concluding that the three tests did not have the same mean preference. A post-hoc analysis was performed by applying the Tukey HSD (Honestly Significant Difference) test to determine which of the three means are statistically different. The results of the critical values for the Studentized range statistic $Q$, $q(0.05, 3, 45) = 3.43$, reveal that only the difference between the results of Tests 1 and 3 are significant in the $95\%$ confidence interval.

In summary, it can be concluded that the average subjective evaluations of Tests 1 and 3 are unequal, and this confirms the second intuitive observation. A preference for Test 1 w.r.t. Test 2, or for Test 2 w.r.t. Test 3, could not be found from the analysis of variance, and this confirms the hypothesis that first-time users of the automatically-generated interface (Test 2) had a user experience, comparable to the experience of both the experienced participants (Test 1) and the subjects using the personalized interface (Test 3), which is considered a sub-optimal case.

In Table III, disaggregated results are presented to enable a more in-depth analysis of individual factors and related statements. For each test, the mean and the variance of SASSI

TABLE III
SUBJECTIVE RESULTS COLLECTED THROUGH SASSI METHODOLOGY [63] IN A BETTER-TO-WORSE (7-TO-1) SCALE.

| Statement | Test 1 | | Test 2 | | Test 3 | |
|---|---|---|---|---|---|---|
| | $S_{mean}$ | $\sigma^2$ | $S_{mean}$ | $\sigma^2$ | $S_{mean}$ | $\sigma^2$ |
| *System Response Accuracy (SAR)* | | | | | | |
| 1. The system is accurate. | 5.60 | 0.97 | 5.62 | 0.42 | 5.27 | 0.92 |
| 2. The system is unreliable. | 6.47 | 0.70 | 6.58 | 0.27 | 6.33 | 0.67 |
| 3. The interaction with the system is unpredictable. | 5.87 | 0.84 | 6.17 | 0.52 | 5.87 | 0.84 |
| 4. The system didn't always do what I wanted. | 5.33 | 2.81 | 4.58 | 1.72 | 5.67 | 2.52 |
| 5. The system didn't always do what I expected. | 5.20 | 3.46 | 5.25 | 2.02 | 5.80 | 1.74 |
| 6. The system is dependable. | 5.87 | 0.98 | 5.92 | 0.63 | 5.40 | 0.40 |
| 7. The system makes few errors. | 6.07 | 0.92 | 5.42 | 2.27 | 4.93 | 2.78 |
| 8. The interaction with the system is consistent. | 6.13 | 1.12 | 5.67 | 1.15 | 5.40 | 0.40 |
| 9. The interaction with the system is efficient. | 5.60 | 1.97 | 5.69 | 1.06 | 5.80 | 0.60 |
| | **5.79** | **1.58** | **5.65** | **1.12** | **5.61** | **1.29** |
| *Likeability (LIKE)* | | | | | | |
| 10. The system is useful. | 6.33 | 0.38 | 6.42 | 2.08 | 6.13 | 1.12 |
| 11. The system is pleasant. | 5.93 | 0.92 | 5.50 | 3.36 | 5.20 | 1.46 |
| 12. The system is friendly. | 5.87 | 1.12 | 5.64 | 1.45 | 5.27 | 0.64 |
| 13. I was able to recover easily from errors. | 6.13 | 0.84 | 5.67 | 1.52 | 4.60 | 1.69 |
| 14. I enjoyed using the system. | 6.27 | 0.35 | 6.08 | 2.08 | 6.00 | 0.86 |
| 15. It is clear how to speak to the system. | 6.33 | 0.38 | 5.33 | 1.15 | 5.33 | 0.52 |
| 16. It is easy to learn to use the system. | 6.73 | 0.35 | 6.00 | 1.09 | 5.27 | 1.64 |
| 17. I would use this system. | 5.80 | 1.17 | 6.25 | 2.20 | 5.60 | 2.11 |
| 18. I felt in control of the interaction with the system. | 6.00 | 1.14 | 6.00 | 0.91 | 5.40 | 1.97 |
| | **6.16** | **0.77** | **5.82** | **1.67** | **5.42** | **1.44** |
| *Cognitive Demand (CD)* | | | | | | |
| 19. I felt confident using the system. | 6.13 | 1.27 | 5.42 | 1.72 | 4.87 | 1.41 |
| 20. I felt tense using the system. | 5.53 | 2.12 | 5.00 | 2.36 | 4.33 | 2.10 |
| 21. I felt calm using the system. | 5.47 | 2.84 | 5.50 | 2.09 | 4.53 | 1.70 |
| 22. A high level of concentration is required when using the system. | 4.60 | 2.40 | 3.83 | 2.33 | 3.87 | 0.70 |
| 23. The system is easy to use. | 6.20 | 1.03 | 5.62 | 0.76 | 4.80 | 0.74 |
| | **5.59** | **2.16** | **5.07** | **1.82** | **4.48** | **1.39** |
| *Annoyance (AN)* | | | | | | |
| 24. The interaction with the system is repetitive. | 4.27 | 2.64 | 4.38 | 3.59 | 4.53 | 1.84 |
| 25. The interaction with the system is boring. | 5.67 | 2.10 | 5.25 | 2.93 | 5.53 | 1.41 |
| 26. The interaction with the system is irritating. | 6.00 | 1.57 | 6.17 | 0.88 | 5.87 | 1.41 |
| 27. The interaction with the system is frustrating. | 6.33 | 0.81 | 6.25 | 0.93 | 5.80 | 2.03 |
| 28. The system is too inflexible. | 5.93 | 2.07 | 5.00 | 2.36 | 4.93 | 1.78 |
| | **5.64** | **2.26** | **5.47** | **1.98** | **5.33** | **1.87** |
| *Habitability (HAB)* | | | | | | |
| 29. I sometimes wondered if I was using the right word. | 3.47 | 3.27 | 3.08 | 2.24 | 1.87 | 1.55 |
| 30. I always knew what to say to the system. | 4.47 | 2.70 | 3.75 | 0.93 | 2.93 | 2.21 |
| 31. I was not always sure what the system was doing. | 5.67 | 2.10 | 5.17 | 1.97 | 5.27 | 2.21 |
| 32. It is easy to lose track of where you are in an interaction with the system. | 5.93 | 1.64 | 5.67 | 2.24 | 6.07 | 0.50 |
| | **4.88** | **3.29** | **4.38** | **2.57** | **4.03** | **4.47** |
| *Speed (SPE)* | | | | | | |
| 33. The interaction with the system is fast. | 5.73 | 1.21 | 5.15 | 1.64 | 4.80 | 1.46 |
| 34. The system responds too slowly. | 5.73 | 1.21 | 5.58 | 1.54 | 5.40 | 1.69 |
| | **5.73** | **1.17** | **5.37** | **1.31** | **5.10** | **1.61** |

scores were calculated. Considering System Response Accuracy (answers 1 to 9), it can be observed that the participants perceived the automatically-generated interface (Tests 1 and 2) as slightly more accurate and reliable than the personalized interface. Likeability (answers 10 to 18) of the personalized interface was affected by the difficulty involved in remembering the chosen commands, thus rendering the automatically-generated interface to be more appealing. The users found that the automatically-generated interface is more effective for recovering from errors, besides being easier to interact with. This is confirmed by Cognitive Demand (answers 19 to 23) and, in particular, by the 23th statement that the automatically-generated interface is easier to use than the personalized one. Concerning Annoyance (answers 24 to 28), the users perceived the automatically-generated interface as more flexible than the personalized one. As regards Habitability (answers 29 to 32), the participants felt uncomfortable with the personalized interface, because they often wondered if they were using the right words. Finally, as regards speed (answers 33 and 34), the users perceived the automatically-generated interface as faster than the personalized one.

## V. CONCLUSIONS AND FUTURE WORK

This paper has presented the design and implementation of a framework that supports automatic generation of speech interfaces for controlling VR and AR applications. The generated interfaces include icon-based representations of application functionalities, which are expected to be capable of evoking the relevant commands, thus reducing the cognitive load for the user. The framework includes visual cues that are automatically selected by means of a semantics-based optimization strategy, which aims at maximizing the match between application functionality and icon description, while limiting

possible overlaps between meanings implied by different icons and fostering consistency in terms of graphic style.

The automatically-generated interface was tested by first-time users in a case study concerning maintenance of the machine tool, using a wearable AR application. The performance obtained by using this configuration was compared to that achieved in two different scenarios where, in one scenario, the same interface was used by experienced users and, in the other, a personalized interface was manually created by the users. Objective data, in terms of task completion rate and average number of attempts for activating functionalities, shows that first-time user experience with the automatically generated interface is comparable to that measured with the two reference scenarios. This is confirmed by subjective evaluation. Overall, the automatically-generated interface proved to be more usable and intuitive for first-time users, thanks to the lower cognitive load of activating application functionalities. In the case of personalized interface, intuitiveness of the automatically-generated interface was confirmed by its higher task completion rate and fewer activation attempts. Concerning usability, the interface generated by the proposed framework has the advantage of reducing the number of errors (in terms of false negatives) in activating functionalities.

Future research work will be aimed at testing the proposed methodology and the tools developed in different scenarios with new applications and with richer icon sets. Moreover, to improve the quality of semantic mapping and enhance the performance of the speech recognizer, novel strategies will be evolved for extracting context information from icon and functionality descriptions and for expanding the valid voice commands set, preserving the framework's robustness and consistency.

## ACKNOWLEDGMENT

This work was partially funded by the EASE-R$^3$ project: Integrated framework for a cost-effective and ease of Repair, Renovation and Re-use of machine tools within modern factory, FP7, FoF.NMP.2013-8, Grant agreement no: 608771.

## REFERENCES

[1] M.R. Mine, J. van Baar, A. Grundhofer, D. Rose, and Y. Bei, "Projection-based Augmented Reality in Disney Theme Parks," *IEEE Computer*, vol. 45, no. 7, pp. 32-40, 2012.

[2] C. A. Linte, J. White, R. Eagleson, G.M. Guiraudon, and T.M. Peters, "Virtual and Augmented Medical Imaging Environments: Enabling Technology for Minimally Invasive Cardiac Interventional Guidance," *IEEE Reviews in Biomedical Engineering*, vol. 3, pp. 25-47, 2010.

[3] S. Zollmann, C. Hoppe, S. Kluckner, C. Poglitsch, H. Bischof, and G. Reitmayr, "Augmented Reality for Construction Site Monitoring and Documentation," *Proc. of the IEEE*, vol. 102, no. 2, pp. 137-154, 2014.

[4] V. Raghavan, J. Molineros, and R. Sharma, "Interactive evaluation of assembly sequences using augmented reality," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 435-449, 1999.

[5] M. Ehmann, "Evaluating Customer Expectance of Mixed Reality Applications in Order Picking," in *Proc. ACM Conference on Pervasive and Ubiquitous Computing*, pp. 1475-1478, 2013.

[6] K.F. Hussain, E. Radwan, and G.S. Moussa, "Augmented Reality Experiment: Drivers' Behavior at an Unsignalized Intersection," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 2, pp. 608-617, 2013.

[7] P. Ducher, "Interaction with Augmented Reality," *Advances in Embedded Interactive Systems*, vol. 2, no. 4, pp. 23-29, 2014.

[8] J. Jankowski, and M. Hachet, "A Survey of Interaction Techniques for Interactive 3D Environments," in *Proc. 34th Annual Conference of the European Association for Computer Graphics*, pp. 65-93, 2013.

[9] A. Duenser, R. Grasset, H. Seichter, and M. Billinghurst, "Applying HCI Principles in AR Systems Design," in *Proc. 2nd Int. Workshop on Mixed Reality User Interfaces: Specification, Authoring, Adaptation*, 2007.

[10] R. Teather, and W. Stuerzlinger, The Challenge of 3D Interaction: Guidelines for Intuitive 3D Manipulation Techniques, Presentation at Interacting with Immersive Worlds, 2007.

[11] A. W. Stedmon, H. Patel, S. C. Sharples, and J. R. Wilson, "Developing Speech Input for Virtual Reality Applications: A Reality Based Interaction Approach," Developing Speech Input for Virtual Reality Applications: A Reality Based Interaction Approach, vol. 69, no. 1-2, pp. 3-8, 2011.

[12] A. W. Stedmon, "Developing Virtual Environments Using Speech as a Input Device", *Human Computer Interaction, Theory and Practice*, pp. 1193-1197, 2003.

[13] N. Yankelovich, G.A. Levow, and M. Marx, "Designing SpeechActs: Issues in Speech User Interfaces," in *Proc. SIGCHI Conf. on Human Factors in Comp. Sys.*, pp. 369-376, 1995.

[14] S. Goose, S. Sudarsky, Z. Xiang, and N. Navab, "Speech-enabled Augmented Reality Supporting Mobile Industrial Maintenance," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 65-70, 2003.

[15] D, Newman, "Speech Interfaces that Require Less Human Memory," *AVIOS Proc. of the Speech Technology and App. Expo*, pp. 65-69, 2000.

[16] A. K. Sinha, S. R. Klemmer, J. Chen, J. A. Landay, and C. Chen, "SUEDE: Iterative, Informal Prototyping for Speech Interfaces," in *Proc. Ext. Abs. on Human Factors in Computing Systems*, pp. 203-204, 2001.

[17] Voice Command Checklist, Guidelines for Defining Commands to be Used in Glass Applications. [Online] https://developers.google.com/glass/distribute/voice-checklist

[18] A. Sanna, F. Manuri, G. Piumatti, G. Paravati, F. Lamberti, and P. Pezzolla, "A Flexible AR-based Training System for Industrial Maintenance," in *Proc. of the 2nd Int. Conf. on Augmented and Virtual Reality*, 2015.

[19] M. Billinghurst, and A. Duenser, "Augmented Reality in the Classroom," *IEEE Computer*, vol. 45, no. 7, pp. 56-63, 2012.

[20] W. Broll, I. Lindt, I. Herbst, J. Ohlenburg, A.K. Braun, and R. Wetzel, "Toward Next-gen Mobile AR Games", *IEEE Computer Graphics and Applications*, vol.28, no. 4, pp. 40-48, 2008.

[21] L. Baraldi, F. Paci, G. Serra, L. Benini, and R. Cucchiara, "Gesture Recognition Using Wearable Vision Sensors to Enhance Visitors' Museum Experiences," *IEEE Sensors Journal*, vol. 15, pp. 2705-2714, 2015.

[22] M. Billinghurst, H. Kato, and S. Myojin, "Advanced Interaction Techniques for Augmented Reality Applications," in *Proc. 3rd International Conference Virtual and Mixed Reality*, pp. 13-21, 2009.

[23] P. Kay, "Speech-driven Graphics: A User Interface," *Journal of Microcomputer Applications*, vol. 16, pp. 223-231, 1993.

[24] S. Irawati, S. Green, M. Billinghurst, A. Duenser, and H. Ko, "An Evaluation of an Augmented Reality Multimodal Interface Using Speech and Paddle Gestures," in *Proc. 16th International Conference on Artificial Reality and Tele-existence*, pp. 272-283, 2006.

[25] M. Koelsch, R. Bane, T. Hoellerer, and M. Turk, "Multimodal Interaction with a Wearable Augmented Reality System," *IEEE Computer Graphics and Applications*, vol. 26, no. 3, pp. 62-71, 2006.

[26] R. W. Lindeman, J. L. Sibert, and J. K. Hahn, "Towards Usable VR: An Empirical Study of User Interfaces for Immersive Virtual Environments," in *Proc. SIGCHI Conf. on Human Factors in Comp. Sys.*, pp. 64-71, 1999.

[27] G. Serra, M. Camurri, L. Baraldi, M. Benedetti, and R. Cucchiara, "Hand Segmentation for Gesture Recognition in EGO-vision," in *Proc. 3rd ACM International Workshop on Interactive Multimedia on Mobile & Portable Devices*, pp. 31-36, 2013.

[28] SAP SE, Improving Field Service Performance with Augmented Reality Software and Smart Glasses. Walldorf, Germany, 2014.

[29] S. R. Klemmer, A. K. Sinha, J. Chen, J. A. Landay, N. Aboobaker, and A. Wang, "SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces," in *Proc. 13th Annual ACM Symposium on User Interface Software and Technology*, pp. 1-10, 2000.

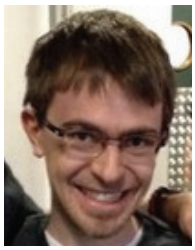
[30] J. Ferreirosa, J.M. Pardo, R. de Crdoba, J. Macias-Guarasa, J.M. Montero, F. Fernndez, V. Samac, L.F. Haro, and G. Gonzlez, "A Speech Interface for Air Traffic Control Terminals," *Aerospace Science and Technology*, vol. 21, no. 1, pp. 7-15, 2012.

[31] D. Sonntag, S. Zillner, C. Schulz, M. Weber, T. Toyama, "Towards Medical Cyber-physical Systems: Multimodal Augmented Reality for Doctors and Knowledge Discovery about Patients," in *Proc. 2nd International Conference, Design, User Experience, and Usability*, pp. 401-410, 2013.

[32] J. Woodard, and E. Cupples, "Selected Military Applications of Automatic Speech Recognition Technology," *IEEE Communications Magazine*, vol. 21, no. 9, pp. 35-41, 1983.

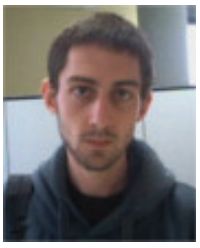
[33] J. White, and M. Duggirala, "Speech-Interface Prompt Design: Lessons from the Field," in *Proc. 7th International Conference on Information and Communication Technologies and Development*, art. 66, 2015.
[34] N. Yankelovich, "How Do Users Know What to Say," *ACM Interactions*, vol. 3, no. 6, pp. 33-43,1996.
[35] M. Yin, and S. Zhai, "The Benefits of Augmenting Telephone Voice Menu Navigation with Visual Browsing and Search", in *Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp. 319-328, 2006.
[36] S. Gamm, and R. Haeb-Umback, "User Interface Design of Voice Controlled Consumer Electronics," *Philips Journal of Research*, vol. 49, no. 4, pp. 439-454, 1995.
[37] C. Danis, L. Comerford, E. Janke, K. Davies, J. DeVries, and A. Bertran, "Storywriter: A speech oriented editor," in *Proc. Conference Companion on Human Factors in Computing Systems*, pp. 277-278, 1994.
[38] M. J. Payne, R. Coelho, and M. H. Hawash, "Color as a Visual Cue in Speech Enabled Applications," Patent Appl. US20040030559, 2004.
[39] S. Shriver, and R. Rosenfeld, "Keywords for a Universal Speech Interface," in *Proc. Extended Abstracts on Human Factors in Computing Systems*, pp. 726-727, 2002.
[40] Y. Zhong, T.V. Raman, C. Burkhardt, F. Biasdsy, and J. P. Bigham, "JustSpeak: Enabling Universal Voice Control on Android," in *Proc. 11th Web for All Conference*, art. 36, 2014.
[41] J. P. Bigham, C. M. Prince, and R. E. Ladner, "WebAnywhere: Enabling a Screen Reading Interface for the Web on Any Computer," in *Proc. 17th international conference on World Wide Web*, pp. 1159-1160, 2008.
[42] L. Xudong, and W. Jiancheng, "User Interface Design Model", in *Proc. 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed*, pp. 538-543, 2007.
[43] T. Mori, T. Nonaka, and T. Hase, "Automatic GUI Generation on AV Remote Control using Genetic Algorithm," in *Proc. IEEE International Symposium on Consumer Electronics*, pp. 1-3, 2010.
[44] A. Sears, "Layout Appropriateness: A Metric for Evaluating User Interface Widget Layout," *Software engineering*, vol. 19, no. 7, pp. 707-719, 1993.
[45] K. Z. Gajos, J. O. Wobbrock, and D. S. Weld, "Automatically Generating User Interfaces Adapted to Users' Motor and Vision Capabilities," in *Proc. 20th Annual ACM Symposium on User Interface Software and Technology*, pp. 231-240, 2007.
[46] F. Lamberti, and A. Sanna, "Extensible GUIs for Remote Application Control on Mobile Devices," *IEEE Computer Graphics and Applications*, vol. 28, no. 4, pp. 50-57, 2008.
[47] S. Caplin, "Icon Design: Graphics Icons in Computer Interface Design," Watson-Guptill Publications, Inc. New York, NY, USA, 2001.
[48] D. Norman, and S. Draper, "User Centered System Design: New Perspectives on Human-Computer Interaction," CRC Press, 1986.
[49] S. Blackenberger, and K. Hain, "Effects of Icons on Human-Computer Interaction," *Int. Journal of Man-Machine St.*, vol. 35, pp. 363-377, 1991.
[50] B. Schneidreman, "Designing the User Interface: Strategies for effective Human Computer Interaction," Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1997.
[51] W. Howell, and A, Fuchs, "Population stereotypy in code design," *Organizational Behav. and Human Perf.*, vol. 3, no. 3, pp. 310-339, 1968.
[52] E. Heard, "Symbol Study - 1972," SAE Technical Paper 740304, 1974.
[53] D. P. T. Piamonte, J. D. A. Abeysekera, and K. Ohlsson, "Understanding Small Graphical Symbols: A Cross-cultural Study", *International Journal of Industrial Ergonomics*, vol. 27, no. 6, pp. 399404, 2001.
[54] M. Morioka, "Automatic Icon Generation System," Patent, US5367626 A, 1994.
[55] E. Keogh, L. Wei, X. Xi, S. Lonardi, J. Shieh, and S. Sirowy, "Intelligent Icons: Integrating Lite-Weight Data Mining and Visualization into GUI Operating Systems," in *Proc. 13th International Conference on Data Mining*, pp. 912-916, 2013.
[56] M. Oda, T. Itoh, "MIST: A Music Icon Selection Technique Using Neural Network", in *Proc. NICOGRAPH International*, 2007.
[57] V. Setlur, and J. D. Mackinlay, "Automatic Generation of Semantic Icon Encodings for Visualizations," in *Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp. 541-550, 2014.
[58] H. Pickering, Auticons. [Online] http://heydonworks.com/auticons-icon-font/
[59] E. Pianta, L. Bentivogli, and C. Girardi, "MultiWordNet: Developing and Aligned Multilingual Database," in *Proc. 1st International Conference on Global WordNet*, pp. 293-302, 2002.
[60] G. A. Miller "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39-41, 1995.
[61] J. J. Durillo and A. J. Nebro, "jmetal: A Java Framework for Multiobjective Optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760771, 2011.
[62] Metaio, the Augmented Reality Company. [Online]. http://metaio.com/
[63] K.S. Hone, and R. Graham, "Towards a Tool for the Subjective Assessment of Speech System Interfaces," *Natural Language Engineering*, vol. 6, no. 3-4, pp. 287-303, 2000.

**Fabrizio Lamberti** (M'02-SM'14) is an Associate Professor at Politecnico di Torino, Italy, from where he received his M.Sc. and the Ph.D. degrees in computer engineering in 2000 and 2005, respectively. His main research interests are in the areas of computational intelligence, semantic processing, human-computer interaction, computer graphics, and visualization. He serves as an Associate Editor for IEEE Transactions on Emerging Topics in Computing and for IEEE Consumer Electronics Magazine.



**Federico Manuri** received his B.Sc. and M.Sc. degrees in computer engineering from Politecnico di Torino, Italy, in 2008 and 2011, respectively. He is currently a Ph.D. student at the Dipartimento di Automatica e Informatica of Politecnico di Torino, Italy. His research interests include human machine interaction, computer graphics and augmented reality.

**Gianluca Paravati** (M'14) is an Assistant Professor at Politecnico di Torino, from where he received his B.Sc. and M.Sc. degrees in electronic engineering and Ph.D. degree in computer engineering in 2005, 2007, and 2011, respectively. His research interests include real-time image processing, collaborative virtual environments, human-machine interaction, remote visualization, and distributed systems.



**Giovanni Piumatti** received his M. Sc. degree in computer engineering from Politecnico di Torino, Italy, in 2014. He is currently a Ph.D. student at the Dipartimento di Automatica e Informatica of Politecnico di Torino, Italy. His research interests are on robotic and phygital gaming, human-computer/human-robot interaction and augmented reality.

**Andrea Sanna** received his Ph.D. degree in computer engineering from Politecnico di Torino, Italy, where he now serves as an Associate Professor. His research interests include computer graphics, virtual reality, parallel and distributed computing, scientific visualization, and computational geometry. He is a senior member of ACM. He has been the General Chair of the 7th International Conference on Intelligent Technologies for Interactive Entertainment 2015.