CoWoOZ – A cloud-based teleoperation platform for social robotics

Gergely Magyar[*], Peter Sinčák[*], Ján Magyar[**], Kaori Yoshida[***], Alessandro Manzi[****], Filippo Cavallo[****]
[*]Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, TU Košice, Slovakia
[**]Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, TU Košice, Slovakia
[***]Department of Human Intelligence Systems, Kyushu Institute of Technology, Kitakyushu, Japan
[****]The Biorobotics Institute, Scuola Superiore Sant' Anna, Pontedera, Italy

***Abstract*—This paper presents our cloud-based teleoperation platform, called CoWoOZ – ClOud-based Wizard of OZ. It describes similar systems, such as Robot Management System (RMS), OpenWoZ, CoWoOZ's predecessor Telescope and the first version of the discussed platform. A special emphasis is given to the description of new functionalities added to the system, e.g. scenario management, custom script builder, client application, etc. The paper also contains a comparison of these systems, which can help researchers to choose the right one for their work.**

Keywords—cloud computing, human-robot interaction, teleoperation, Wizard of Oz

I. Introduction

Since the early 2000s the cloud computing paradigm had a great impact on various fields and robotics was not an exception. In 2010 James Kuffner at the International Conference on Humanoid Robotics introduced the concept of 'cloud robotics' [1]. He claimed, that robots using cloud computing could offload CPU-heavy tasks to remote servers and use smaller onboard computers. Since then many studies utilized the advantages of this paradigm. For a comprehensive summary of research done in recent years please refer to [2] [3] and [4] respectively. However, one can notice, that many of the applications were focusing on navigation and manipulation tasks. Since our work is focused toward social and assistive robotics here we provide a short review of papers dealing with the use of cloud computing in human-robot interaction (HRI) scenarios.

In [5] the authors used "cloud-connected social robots to measure and model the effect of LEGO Engineering and its collaborative nature on the development of social skills in children with Autism Spectrum Disorder". Another study [6] used the cloud infrastructure to enhance the affective skills of a humanoid robot. In this setup, the data collected by the robot were processed in the cloud environment and based on the results the robot got an instruction what it should do in the given situation. In [7] the robot KuBo "relied on cloud services extending its capabilities for human interaction and environmental sensing to provide services such as Google Calendar, Google Speech Recognition or Acapela VaaS (Voice as a Service). Besides testing the feasibility of the cloud service the goal of the experiment was to help elderly with various tasks in their home. In [8] a cloud-assisted pillow robot was introduced which utilizing cloud resources was able to recognize the user's emotions and communicate with other robots of the same type.

All the above mentioned studies used various cloud services, except teleoperation, however, it is a crucial part of HRI experiments these days. Most of those (52% of publications at the annual HRI conference in the last three years according to [9]) uses the Wizard of Oz (WoZ) technique [10]. In general, we can say that it "is a research experiment in which subjects interact with a (computer/robot) system that subjects believe to be autonomous, but which is actually being operated or partially operated by an unseen (hidden) human being" [11].

To fill this gap we developed CoWoOZ, a cloud-based Wizard of Oz system which is intended to become a common platform for HRI researchers applying the WoZ method in their work.

II. Related Work

Although the field of cloud-based teleoperation in human-robot interaction is not very broad, in recent years many systems were developed for similar purposes (not exclusively using cloud computing).

A. Robot Management System

Firstly, we can mention the Robot Management System (RMS) presented in [12]. In the authors' own words "RMS is an open-source framework that allows researchers to quickly and easily install, configure and deploy a secure and stable remote laboratory". It also allows users to create accounts and this way gain access to robots and participate in research studies. The main goals when creating the system were:

- Robot and interface-independent design
- Support for easy creation and management of new interfaces
- Secure user authentication and authorization
- Creation, management, logging and analysis of multi-condition user studies
- Website content management

The architecture of RMS can be seen in Fig. 1.

978-1-5090-5655-2/17/$31.00 ©2017 IEEE

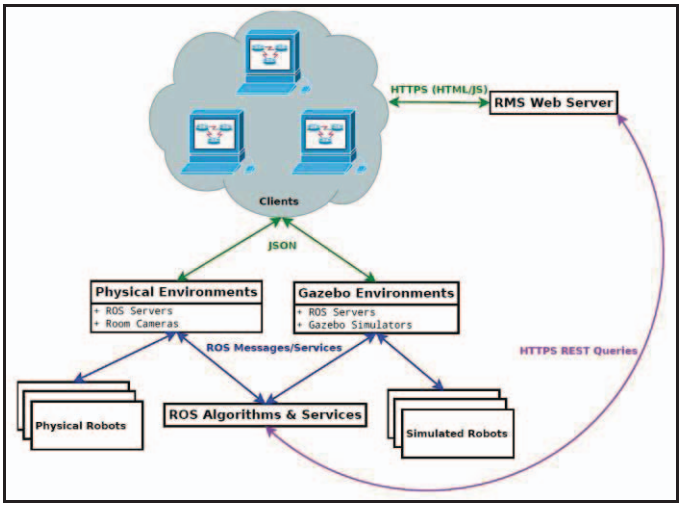

Fig. 1 Architecture of the Robot Management System [12]

The system works with Robot Operating System (ROS) enabled robots. It was successfully tested with PR2 (Willow Garage), youBot (KUKA) and Rovio (WowWee). The feasibility of the system was validated by human participants who were remotely controlling a robot in a series of object retrieval tasks.

B. *OpenWoZ*

Another example is OpenWoZ by G. Hoffman [13]. In general, it is a framework "designed to be updated during operation by an expert". OpenWoZ is implemented as a thin HTTP (Hypertext Transfer Protocol) server running on the robot and a cloud-backed multi-platform client schema. The WoZ server accepts REST (Representational State Transfer) requests from the clients simultaneously (see Fig. 2). This feature allows the addition of commands, new sequencing of behaviors, and adjustment of parameters, all during runtime.

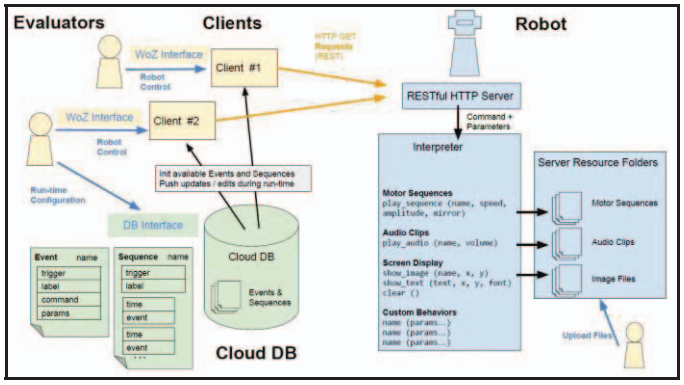

Fig. 2 Architecture of OpenWoZ [13]

As it was mentioned before, the clients communicate with the server via requests. E.g. the command `/move/wave?speed=.5` sent to the robot executes a wave movement at 50% speed. The command `/sound/hello?volume=1` makes the robot say "Hello" at full volume.

The system was tested on a newly constructed robot with capabilities to move, play and record sounds and display data. However, no 'commercial' robot was connected to the system neither it was used in a real-world HRI scenario.

C. *Telescope*

The predecessor of CoWoOZ was Telescope, which was developed at the Center for Intelligent Technologies at the Technical University of Košice as a server-based modular system for teleoperation of connected devices [14]. In this context, a device is any electronic appliance which can communicate over a network and can be accessed programmatically.

When designing the system a special effort was made to make the system platform-independent and so allow the teleoperation of devices connected to a heterogeneous environment. By heterogeneous environment we mean a system consisting of devices with different APIs (Application Programming Interface), programmable in different languages or with different user interfaces. This was solved by introducing a wrapper (similar to a device driver) which translated Telescope's commands into a format understandable by each device. Such a wrapper was developed for each type of device.

In Fig. 3 Architecture of Telescope the architecture of the system is shown. It consists of three parts:

1) Wrapper on the devices – this block consists of the teleoperation and telediagnostics module. The first one serves for translating commands for remote control. The latter is providing information about the overall state of the diagnostics device and sends those to the telediagnostics database.
2) Event and diagnostics servers and databases – the core of Telescope, which ensures the communication between the teleoperator and the teleoperated devices. It is worth noting, that all communication was realized using websockets. The diagnostics database store data which serve as a basis for possible error and fault detection.
3) Webpage – a simple webpage was implemented as a teleoperation interface and for visualizing diagnostics data.

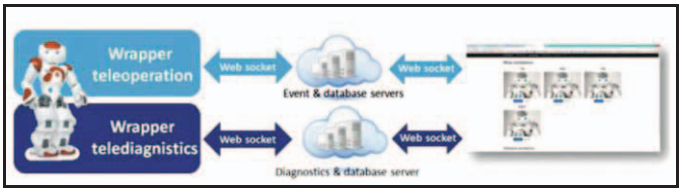

Fig. 3 Architecture of Telescope [14]

The system was tested using the NAO humanoid robot. Teleoperators were able to move the robot's joints, set its stiffness, use its text to speech feature and launch pre-programmed behaviors. In addition to that, Telescope allowed adding, removing and sharing devices between users, making it possible to control one device by multiple teleoperators.

III. CoWoOZ – Cloud-based Wizard of OZ

In general, CoWoOZ is a cloud-based teleoperation platform for various types of robots built on top of Microsoft Azure. Unlike its predecessor Telescope, this system uses the advantages of the cloud environment, such as scalability or the lack of need to take care of a physical server, since these are ensured by the cloud provider.

The system consists of two main building blocks, the frontend, and the backend, as it is shown in Fig. 4.

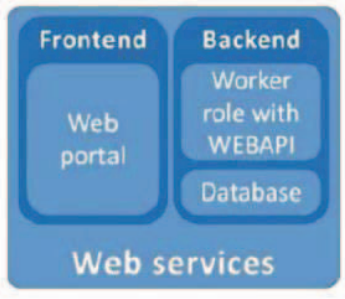

Fig. 4 Architecture of CoWoOZ [15]

The frontend is represented by a webpage which serves as an interface for teleoperation. It allows the user to carry out CRUD (Create, Read, Update, and Delete) operations over the robot behaviors and types and also play the chosen behaviors on a real robot.

The backend consists of the following two parts:

- Worker role with WEBAPI ensures the communication between the server and the robots and also between the server and the web page via HTTP protocol.
- The database contains a table for each type of robot and a blob container (a MS Azure service for storing large amounts of unstructured object data, that can be accessed from anywhere in the world via HTTP or HTTPS) for each table entry for storing robot behaviors.

After clicking the "Play behavior" button on the webpage, the name of the movement and the robot's type are added to a dictionary. The wrapper on the robot is constantly sending requests to the cloud service to find out what it should do. When a request arrives to the service it checks the dictionary if it contains anything. If so, the appropriate script is selected from the database and sent to the robot.

The pairing mechanism between a specific robot and the cloud service is very simple. On the webpage, the user assigns a unique number to the robot which matches the one in the wrapper (and vice versa). This unique number is also added to the dictionary, this way we can ensure, that the chosen behavior is played on the right robot.

Currently, the system can be used in three different ways:

1. **1** user controls **1** robot
2. **1** user controls **n** robots
3. **n** users control **1**

CoWoOZ currently supports three robotic platforms: NAO (Softbank Robotics), Milo (Hanson Robotics) and Q.bo (theCorpora). Experiments for measuring the response times for these platforms were carried out. The results can be found in [15].

IV. Advanced Functionalities of CoWoOZ

Although the system presented in Section 3 was sufficient as a proof of concept of teleoperation of robots from a cloud environment, in order to use it in HRI research it needed some new features. These will be further described in this section.

A. Web interface

In the first version of the system, operators had at their disposal just a list of names of the available behaviors. In order to make their life easier we recorded the movements and created short animations. The resulting interface can be seen in Fig. 5. Please note, that the webpage's design is still in development and in the future will be further improved.

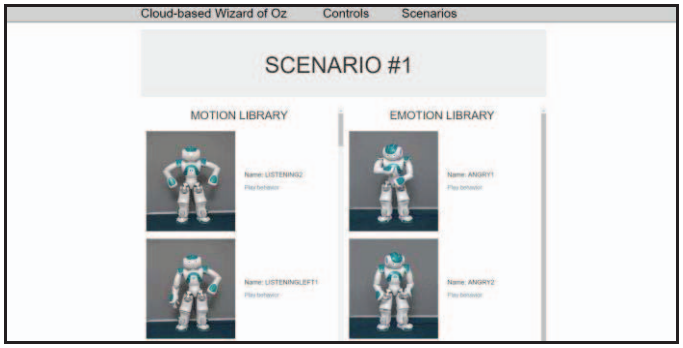

Fig. 5 CoWoOZ's teleoperation interface

1) Security

Another issue which was not solved in the previous version was security and management of user accounts. The original system didn't contain any authentication process. Since we didn't want to completely rewrite parts of the system, we chose a less invasive way of user authentication, using a custom solution.

The security of user login data is now ensured by hashing and salting them because we always have to assume that users ay (and most likely will) use the same login information elsewhere.

2) Updated database functionality

CoWoOZ already offered basic database operations, such as adding behaviors or new types of robots, reading and updating them, however, it lacked the option to delete entries. The usual approach handling this problem is to set a flag marking deleted content and so both deletion and restoration are done simply by changing the flag. We solved this issue by moving the deleted entries to a new table, hence they can't be queried anymore without the need to filter all entries based on flags.

3) Asynchronous web interface

Besides the improvements mentioned in the introduction of this subsection, we introduced asynchronous method calling. This means that the application is not reloaded all the time an action is taken because the browser remembers the state of the web and replaces only desired elements (if any). In practice it means that after executing a movement on the robot, the interface stays in the state it was when the movement was selected. This feature is very convenient for operators since they don't have to wait for the webpage to reload after each action taken.

4) Scenario management

Another new requirement for the system was a more effective management of scenarios. For this, the database model was revised. It now enables filtering the robot behaviors based on scenarios, robot types, and type of behavior. When uploading a new behavior, the user must indicate its purpose. There are two basic categories: motion and emotion. The former represents basic movements for the robot and are vital in accomplishing the desired goal, while the latter are primarily aimed at interaction. Emotion behaviors are further divided based on the emotions on Plutchik's wheel of emotions (joy, trust, fear, surprise, sadness, anticipation, anger, and disgust).

The database structure also enables behavior sharing between users but the functionality is yet to be implemented. It makes it possible for system users to make their tentative robot behaviors private or add them into a public library, creating an environment for effective cooperation between institutions.

The primary navigation structure is based on scenarios, which represents an HRI session and it contains every behavior a robot might need during that interaction. The system's user can add robot behaviors from their computer or from the system's library of behaviors. More users can have access to a given scenario and they can assign robots to the scenario. This creates an easy-to-use interface for scenario management.

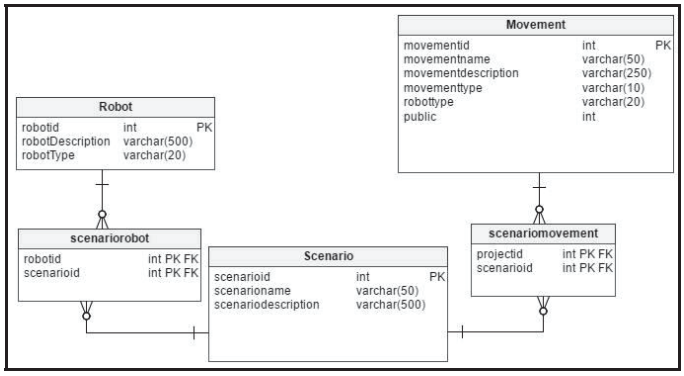

Fig. 6 The updated database structure

Behaviors can be played on the robots accessible from the scenario directly from the system. The system was designed in a way that it should be able to determine which robots are online and offer the user a list of available ones but this functionality hasn't been added yet.

B. Thin client

While the web page provided a great way to control robots remotely, it had some limitations when controlling multiple robots by one user. In this case, the operator was only able to send the same instruction to all connected robots. To overcome this difficulty we decided to develop a custom-tailored thin client.

1) Design

The teleoperation center for which this client was meant consists of a single computer with 6 monitors, placed in a laboratory with 4 IP cameras. However, it is fully functional in any other setup. The thin-client was designed to offer the full functionality of the web portal in addition to camera streams, direct control of the robot, and a script builder for creating own scripts.

The intended usage of the client is to have one of the screens used for the main interface, up to four occupied by cameras and the remaining monitors (amount depends on actual camera usage) unused by the client, allowing the teleoperator to run other assistive programs simultaneously.

In Fig. 7 the main window of the client can be seen when a NAO robot is connected. On the left, the robot's type and its unique identifier can be set and also information about the robot's battery level and processor temperature is displayed. In the middle users can start the stream of the IP camera. On the right, the actions from the cloud database for the connected robot can be found.

Fig. 7 Teleoperation interface of the thin client

2) Communication with the cloud service

The original solution utilized standard HTTP communication with a response unsuitable for the client. That's why a new controller was added to the system with two methods. One for handling the client's requests and the other one ensuring two-way communication with the robot during direct control. The messages between the client and the service have a JSON (JavaScript Object Notation) format. On both sides of the communication, a parser is used to decode the received message.

3) Camera streams

IP cameras in our laboratory already contained a web interface with controls in addition to the video stream. Our goal was to integrate it with the client which is a local application. It

was done by using the GeckoFX library which implements a Firefox browser into a Windows Forms application.

4) Direct control

In some cases, especially in mobile robotics research, it is not enough for the operator to execute predefined scripts on the robot. That's the reason why we implemented the direct control module. By which, we mean moving the robot in the environment in real-time.

Unlike the teleoperation discussed so far, this module is not meant to work with a group of robots with the same ID as it removes the action from the dictionary when retrieving it. In our first approach, the robot moved until it received a signal to stop. This, however, had proven problematic due to varying (but generally high) response times resulting in the possibility to damage hardware or the environment. Because of that, we moved to our current solution, where the robot is allowed to move only a few centimeters and then it has to check whether it can move further. If so, the process repeats, instead of the continuous movement implemented first. As a downside, we have to mention the need of another wrapper for direct control. In the case of the NAO robot, unlike the teleoperation wrapper which requires streaming of each action from the server, the wrapper for direct control has pre-programmed actions at its disposal, which are calling relevant modules from the naoqi library based on the parsed JSON message from the server. In addition to movement, text to speech postures are also available for this type of robot.

Q.bo uses similar logic, but the text to speech functionality is not supported. The direct control is also limited to movement of the robot and rotation of its head. Since Q.bo is using ROS, the responsible ROS topics are called while being directly controlled.

Milo's constraints with movement render it unsuitable for the same type of control as with NAO or Q.bo so its direct control is only based on directly lined scripts from the teleoperation module.

5) NAO live status updates

NAO being one of the most used humanoid robots in HRI research, we believe that a feature providing the operator with real-time information about the status of the robot's hardware can be useful. In practice, it means that a NAO robot running a modified wrapper will report desired values within each iteration of the main loop. Currently, it reports its movement status (idle or in action), battery level and processor temperature. When using direct control, this report is included within the request for its next action. The live status update feature is designed to be handled dynamically so the client doesn't have to be recompiled to reflect the changes and is able to work with any robot which reports its status.

6) Script builder

The thin client also contains a demo implementation of a script builder, currently supporting the NAO robotic platform. Its goal is to provide a simple graphical user interface for users who would like to create a custom script for the robot (see Fig. 8). It does so by providing templates for the robot's behavior. This allows people without any prior programming knowledge to create their own scripts for the supported robotic platforms. However, programmers wanting to add more functionality are able to do so by selecting "custom" and manually typing the script.

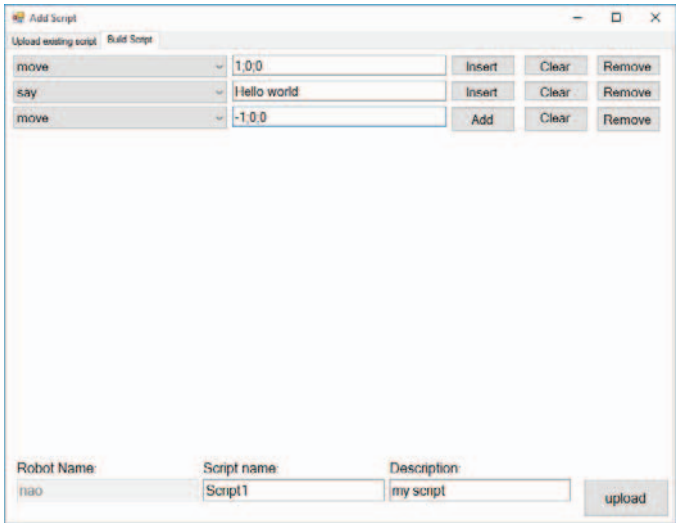

Fig. 8 Script Builder's interface

V. COMPARISON OF THE DISCUSSED PLATFORMS

This section contains a comparison of the following systems: OpenWoZ, Telescope, CoWoOZ (basic version), updated CoWoOZ based on various criteria. We decided to exclude Robot Management System from the comparison because it is ROS-based and can't be directly compared with the other ones. The comparison was done by the following criteria:

- architecture – main architecture of the system
- generality – whether the system supports different robotic platforms
- diagnostics – whether the operator is informed about the current state of the robot's hardware
- direct control – whether the robot can be controlled directly by using a keyboard or a joystick
- control interface – where the main controller interface is located
- video stream – whether the platform comes with a feature for streaming video (either from the robot or from an external camera)
- scenario management – whether users can create their own scenarios with robot behaviors of their choice
- building own scripts – whether the system is suitable for creating new, custom behaviors and play them on a robot

TABLE 1 COMPARISON OF SELECTED PLATFORMS

	OpenWoZ	Telescope	CoWoOZ v1	CoWoOZ
Architecture	Server-client	Server-client	Cloud-client	Cloud-client
Generality	No	Yes	Yes	Yes
Diagnostics	No	Yes	No	Only for NAO
Direct control	No	Yes	No	Yes
Control interface	Local	Web	Web	Web and local
Video stream	No	No	No	Only in client
Scenario management	No	No	No	Yes
Building own scripts	Yes	No	No	Yes
Sharing behavior	No	No	No	Yes

From the table it is evident, that the current version of CoWoOZ has the most features compared to the other systems, however, many of those are currently in experimental phase or are supporting just the NAO robotic platform.

VI. CONCLUSION AND FUTURE WORK

In this paper, we described our cloud-based Wizard of Oz system CoWoOZ and then compared it with its predecessor Telescope, its first version, and the OpenWoZ system. The comparison shows that the current version of our system has the most functionalities from which many were introduced in the second version. We believe that these new features can further increase the Wizard's productivity and can be helpful for HRI researchers.

Although our system covers most of the needs of a Wizard in a social HRI scenario, we are always aware of its weaknesses. The most urgent issue which has to be addressed is the user interface. Our goal is to find the right balance between aesthetics and functionality while allowing users to design their own interfaces according to the scenario for which they are building it. In the future, we also want to extend the GUI of the thin-client by adding the animations into it (currently, these are available only on the web page).

Regarding the other functionalities of the thin-client (direct control and live status updates), we want to add support for the other robotic platforms as well, namely Q.bo and Milo. Also, Script Builder must be extended in order to create more complex robotic behaviors and not only for NAO.

Our ultimate goal is to equip the existing teleoperation platform with learning capabilities as it was proposed in [16]. We have already done experiments for proving that interactive reinforcement learning can be used for learning robotic behavior from the Wizard in social HRI [17][18]. We also have to note that the experiments were carried out using a simulation of HRI. In the future, we plan to test the system (with learning) in an assistive HRI scenario.

ACKNOWLEDGMENT

This research work was supported by the Slovak Research and Development Agency under the contract No. APVV-015-0731 and research project supported from 07-2016 to 06-2019.

REFERENCES

[1] J. J. Kufner, "Cloud-enabled robots," in IEEE-RAS International Conference on Humanoid Robotics, 2010.

[2] K. Goldberg, B. Kehoe, "Cloud robotics and automation: A survey of related work," in EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-5, 2013.

[3] B. Kehoe, et al., "A survey of research on cloud robotics and automation," in IEEE Transactions on Automation Science and Engineering, vol. 12, no. 2, pp. 398-409, 2015.

[4] Cloud robotics and automation [online], http://goldberg.berkeley.edu/cloud-robotics

[5] J. Albo-Canals, et al., "Modelling social skills and problem solving strategies used by children with ASD through cloud-connected social robots as data loggers: First modelling approach," in Conference Proceedings New Friends 2015: The 1st International Conference on Social Robots in Therapy and Education, pp. 1-2, 2015.

[6] Y. Ma, et al., "Cloud-assisted humanoid robotics for affective interaction," in 2nd International Conference on Control, Automation and Robotics (ICCAR), pp. 15-19, 2016.

[7] A. Manzi, et al., "Design of a cloud robotic system to support senior citizens: The KuBo experience," in Autonomous Robots, pp. 1-11, 2016.

[8] M. Chen, et al., "Cp-robot: Cloud-assisted pillow robot for emotion sensing and interaction," in International Conference on Industrial IoT Technologies and Applications, pp. 81-93, 2016.

[9] P. Baxter, et al., "From characterising three years of HRI to methodology and reporting recommendations," in The 11th ACM/IEEE International Conference on Human-Robot Interaction, pp. 391-398, 2016.

[10] L. D. Riek, "Wizard of Oz studies in HRI: A systematic review and new reporting guidelines," in Journal of Human-Robot Interaction, vol. 1, no. 1, 2012.

[11] B. Hannington, M. Bella, "Universal methods of design: 100 ways to research complex problems, develop innovative ideas, and design effective solutions, Rockport Publishers, 2012.

[12] R. Torjs, D. Kent, S. Chernova, "The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing," in Journal of Human-Robot Interaction, vol. 3, no. 2, 2014.

[13] G. Hoffman, "OpenWoZ: A runtime-configurable Wizard of Oz framework for human-robot interaction," in 2016 AAAI Spring Symposium Series, 2016.

[14] L. Mizenko, et al., "Telescope: System Overview," in Developments in Virtual Reality Laboratory for Factory of the Future, pp. 100-105, 2014.

[15] T. Cadrik, et al., "Cloud-based robots and intelligent space teleoperation tools," in Robot Intelligence Technology and Applications 4, pp. 599-610, 2017.

[16] G. Magyar, M. Vircikova, "Learning from the Wizard of Oz on cloud in social human-robot interaction: Proposal," in IEEE 19th International Conference on Intelligent Engineering Systems, pp. 107-111, 2015.

[17] G. Magyar, M. Vircikova, P. Sincak, "Interactive Q-learning for social robots that learn from the Wizard: A pilot study," in Proceedings of the 2016 IEEE International Conference on Systems, Machine and Cybernetics, 2016.

[18] G. Magyar, M. Vircikova, P. Sincak, "Increasing the robot's level of autonomy in social human-robot interaction through interactive reinforcement learning," in New Friends 2016: 2nd International Conference on Social Robots in Therapy and Education, 2016.

.