Experiences with an Approach to Abstract Handling of Content for Human Machine Interaction Applications

Richard Schmidt, Johannes Fonfara, Sven Hellbach and Hans-J. Böhme*

Artificial Intelligence Lab, University of Applied Sciences, Dresden, Germany
{schmidtr;fonfara;hellbach;boehme}@htw-dresden.de

Abstract. Current robotic software frameworks lack a mean to aid in the generation, validation and presentation of high quality content for user interaction. This paper introduces a new approach to extend a basic robotic software framework with a layer for content management. This layer has capabilities for controlling the content presentation subsystems already integrated. We introduce abstract *dialog acts* as a centerpiece for creating and handling robot behavior including user interactions. A simple file format is used to edit the dialog act structure and it allows the delegation of the dialog creation to domain experts within the desired field. We demonstrate that the creation of different sets of dialog acts allows the implementation of completely different use cases without requiring any changes to existing software components.

Keywords: dialog content · content creation · corpus building · human machine interaction

1 Introduction

Within recent years, the field of human machine interaction (HMI) has drawn more and more attention within the robotics community. Interactions with human users play a key role in numerous disciplines such as robotic guidance, entertainment, and ambient assisted living.

There are plenty of ways for robotic platforms to communicate with human users. The most common ones are to display text, images, and videos on a mounted screen, and output speech – either prerecorded or generated by a text-to-speech system. Touch screens and automated speech recognition systems along with dialog management systems receive and process the input from the user. A schematic overview can be seen in Fig. 1.

For researchers and developers working in this field, the following three problems usually arise:

* This work was supported by ESF grant number 100076162.

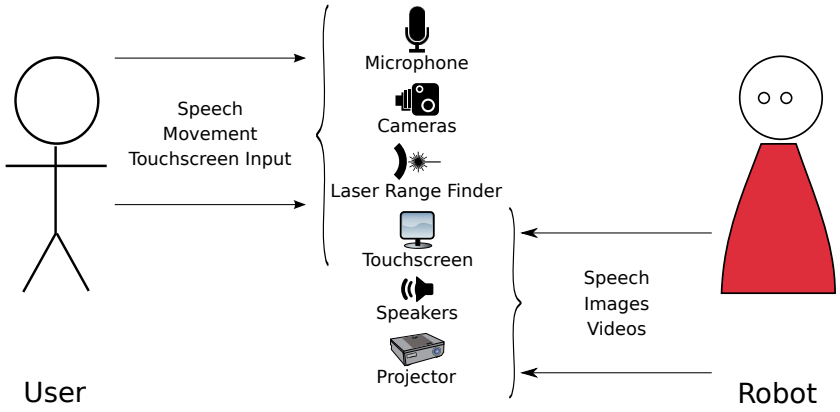

Fig. 1: Schematic overview with means of user communication for human machine interaction.

Missing Presentation Middleware In the robotics community several software packages like ROS [10], Player/Stage [7] and MIRA [5] aid the implementation process of real-world robotic applications. These frameworks are very helpful to abstract hardware access, interconnect software modules and develop algorithms even up to a behavioral level. Yet they all lack focus regarding the interaction with humans, as they were never designed specifically for this purpose.

Content Creation In real world applications, the *dialog content*[1] that the robot can present has to be gathered, edited, and evaluated. Unfortunately, the developers of a robotic application commonly do not have the expert knowledge to generate high quality dialog content for the robot's operational scenario. So *domain experts* should be enabled to author such content instead. Furthermore, the authored dialog content usually has to be tested and evaluated in real world scenarios, as this may reveal additional ways to improve the content.

Corpus Building Log data from previous deployments might be needed to create a data collection allowing further analysis, also known as a *corpus*. This corpus can be used to develop and tune speech recognition or dialog management systems and adapt them to the content. But as long as these systems are not yet capable of performing a user-satisfactory dialog autonomously, reliable data is hard to obtain.

In the following section, we formulate the requirements for a HMI-capable museum tour guide robot (see Fig. 2). In Sect. 3, we describe our proposed framework extension to fulfill the requirements. We continue with a discussion of how we applied our extension to multiple real world use cases in Sect. 4. Sect. 5 con-

[1] The term *dialog content* herein comprises everything that is used for user interaction such as speech, text, images, videos and even interactive applications like games.

(a) (b)

Fig. 2: (a) Tour guide robot leading a group. (b) Attached digital image projector is used to present content.

cludes this paper with an evaluation of our approach and gives an outlook for possible subsequent work.

1.1 Robotic Platform

For our experiments we used a Scitos G5 robot by MetraLabs GmbH[2], as shown in Fig. 2. Its anthropomorphic qualities, such as its life-sized proportions and a movable head, make this platform adequate for HMI applications. A sonar array, two laser range finders (front and back), microphones, a 360° camera array and a depth camera are the sensors on the platform. Speakers, a touchscreen and a digital video projector are the devices that allow presentation of information towards visitors.

1.2 Related Work

Several approaches for multimodal dialog management systems for robotic application like MuDiS [8] and the dialog system of the BIRON project [12] exist. Their goal is to enable a natural interaction with robot applications by interpreting input from different modalities, fusing the input and generating dialog output accordingly. Commonly not addressed are the aspects of dialog content authoring, evaluation and presentation that we focus on in this paper.

The Artificial Intelligence Markup Language (AIML) [13] is a markup language that serves as the knowledge base for HMI applications. It shares similarities with a markup language proposed in this paper, but lacks means of handling multimodal dialog content while being more complex. In [3,2] the Multimodal Interaction Markup Language (MIML) is introduced. The language abstracts

[2] `http://metralabs.com`

global tasks, means of interaction and low level modalities. MIML itself, beeing a language concept, does not solve the mentioned real world problems that we are going to address in this paper.

2 Requirements

Our goal is to use the robot as a tour guide in a museum. In this real world application, the robot has to inform and entertain visitors that were not trained for interaction with the device. Therefore, we think that spoken natural language is the best mean of interaction. The main reason why robots in public areas still lack complex dialog capabilities is that speaker-independent speech recognition is still a challenging task. Having this problem in conjunction with a dialog system still being in its development phase, a satisfactory spoken user interaction is not within short term reach. In order to still be able to gather dialog data for research and evaluate our already created subsystem under real world conditions, we decided to deploy a so called *Wizard of Oz* setup [11,9], in which a human operator remotely controls the application. This creates the illusion of an already completely operational system with spoken dialog interactions in a manner and quality that we aim to achieve eventually with a completely autonomous system.

For this Wizard of Oz extension to our existing platform the following main requirements were formulated:

Framework Integration The extension has to be implemented on top of an already employed robotic framework without requiring extensive modifications to the framework itself or existing subsystems. This enables the evaluation of these subsystems, for example navigation and people tracking, in a real – possibly crowded – environment.

Remote Operation A *remote operator* must be able to control certain high level aspects of the interaction and the robot behavior, for example triggering dialog reactions letting the robot navigate to waypoints. Therefore the operator has to take a remote location, where video and audio data are streamed from the robot's sensors via wireless connection.

Multimodal Dialog Presentation We intend to present dialog contents mainly by natural language outputs being generated by a text-to-speech system on the robot, together with a touch screen and a digital video projector presenting images, videos and text contents. The touch-capability of the screen should be used to allow browsing through a graphical user interface.

Content Creation In our scenario, the expert knowledge and media files about museum exhibits are not directly available to the developers. Therefore, the wish emerged to hand over certain aspects of the content authoring process to the domain experts of the exhibition. A template structure for all the content has to be created, easy enough to be filled by the experts without requiring background information about the software framework. The development of additional content authoring tools should be avoided for simplicity reasons.

Beside letting domain experts author the content, real world deployment sometimes requires the ability to adapt content without much effort, for example to react to unforeseen changes in the environment.

Contextual Dialogs It should be possible to provide a dialog text statement in different alternatives. This is necessary to adapt dialogs to the current operational context of the robot for an socially acceptable behavior. For example, groups should be addressed differently than a single person and facts should be explained more easily understandable for children.

Migration to Dialog System From the software developer point of view, the extension to our framework has to work independently of whether the dialog is directed by the remote operator or a dialog management system. A parallel deployment of a human operator and a dialog management system needs to be possible as well, which is desired for an iterative test-and-development cycle. Then more and more tasks of the human operator can be gradually taken over by an autonomous dialog management system.

Corpus Building A corpus of speech and interaction between robot and visitors is needed as the foundation to develop and train a dialog system as noted in [6]. The extension should aid in the process of building such a corpus.

3 Design and Implementation

We decided to base our dialog system on our *General-Robot* framework, whose design is heavily influenced by the actor model [1] and thus allows the concurrent processing of internal messages.

Depending on the design of the robot software framework in use, concurrent processing might not be necessary or even desired at the message passing and dispatching level. For an adaptation of our approach to different frameworks, a simple observer pattern, whereby messages are passed to observers, should be sufficient.

The General-Robot framework, maintains a set of *states*, *state containers* and *state processors*. Data objects representing messages, for example sensor data, are encapsulated into state objects. These encapsulated state objects are then enqueued into state containers which retain a certain constant number of states or alternatively all states within a certain time horizon. State processors can observe these containers, in which case they are notified about new states. We will discuss certain aspects of the design further in this section.

3.1 Dialog Acts

As mentioned in Sect. 1.1, our robot can present dialog content as uttered speech, on a touchscreen and as projected images or videos, of which speech is the most common and important mode. For natural language generation we use static text blocks. In our speech module, we use a third party text-to-speech system

to transform text blocks on the robot directly into an audible signal. Whereas a system with prepared audio files would also be possible, we use this approach as it avoids the process of generating new audio files even after minor text changes.

Formally, a dialog is made up of a series of *dialog acts*. Every dialog act consists of one or more atomic system *commands* and has a unique label that also serves as a short description. When they are triggered by either the dialog system or the remote operator, the *dialog act dispatcher* sends the associated commands to the respective subsystems where they are processed accordingly (see Fig. 3). In order to formalize dialogs, we use a clear text markup language with a focus on human readability which is shown in Fig. 4. This allows to externalize the creation of the dialog text as mentioned in Sect. 2 and also circumvents the creation of additional tools for authoring.

The text blocks that the robot should utter are directly embedded into the dialog acts file. Although a stricter separation between structure and content – to which the text belongs – of a dialog act might appear desirable, we find that the convenience of being able to editing both in one common place is worth the structural breach and allows the required fast changes to text as mentioned in Sect. 2.

As shown with dialog act *OK* in Fig. 4, it is possible to offer several alternative texts for one statement. The dialog act dispatcher chooses randomly from the alternatives. This avoids a tedious listening experience to often repeated dialog acts like *YES* and *NO*.

To allow different text alternatives for different dialog contexts as required in Sect. 2, we added an extra layer of differentiation as shown on dialog act *WHERE_FROM* which is available in the alternations named *Text*, *TextGroup* and *TextFormally*. *Text* is the default and has to exist for all dialog acts where text utterance is desired. Other alternatives can be created and named freely. The dialog act dispatcher will choose the one preferred by the dialog management system.

3.2 Command Dispatching

A command represents a single task to be executed by the robot. Every command has a certain command type and may or may not carry arguments. There are command types for every aspect of our existing robotic platform that need to be controlled remotely in our museum scenario. An overview of the types is shown in Tab. 1. Within the framework, commands and arguments get encapsulated within a state.

Software components can instruct the dialog act dispatcher to trigger dialog acts by their label. The dispatcher then looks up in its in-memory representation of the dialog acts file and resolves the acts into commands. Submodules – which are also state processors – can listen to the dispatcher's commands state container in order to receive notifications when new commands arrive (see Fig. 3).

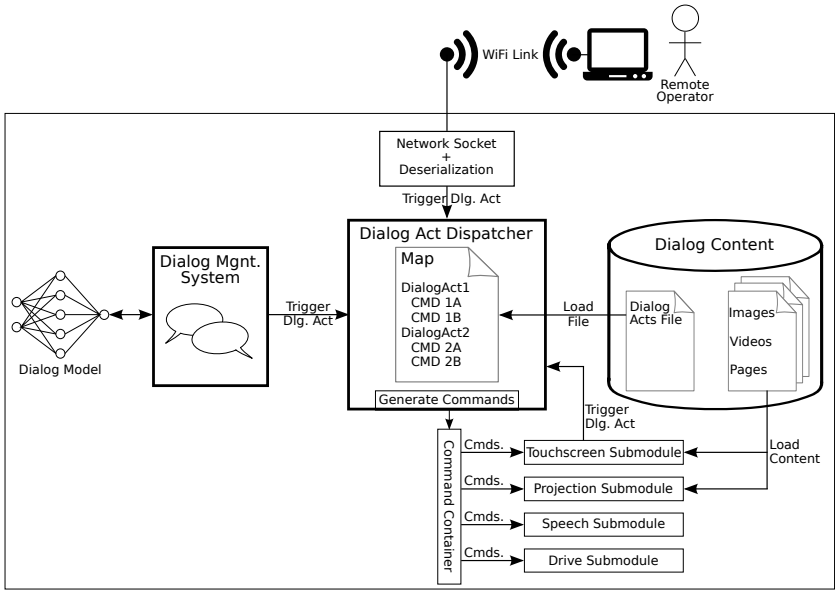

Fig. 3: Schematic overview of the usage of dialog acts in our system.

3.3 Media Content

We use the touchscreen and the projector to present visual dialog content to visitors. Similar to the speech submodule, the corresponding submodules are controlled by commands from the dialog act dispatcher. But here, the commands carry a file path to media files as argument. It is up to the submodules to render the files on the output devices appropriately.

The projection submodule takes care of finding projection regions and perspective correction which is further described in [4]. For touchscreen content, we decided to not only resort to plain images and videos, but also to use a HTML renderer. This allows the creation of interactive pages, through which users can browse by touch gestures. We extended the HTML render with the possibility

Table 1: Most commonly used commands for our robot applications.

Command	Argument	Submodule	Description
SAY	text	Speech	Let the robot speak the given text
PROJECT	file path	Projection	Let projector display image/video
DISPLAY_PAGE	file path	Touchscreen	Show HTML page on touchscreen
SET_WAYPOINT	coordinates	Drive	Set target coords. in environment map
DRIVE_START	none	Drive	Start/Continue drive to waypoint
DRIVE_STOP	none	Drive	Stop drive to waypoint

```
- Label: WELCOME
  Text:
        - Hello!
  Cmds:
        - DISPLAY_PAGE Welcome/Welcome.html

- Label: OK
  Text:
        - OK!
        - Great!
        - Splendid!

- Label: WHERE_FROM
  Text:
        - Where do you come from?
  TextGroup:
        - Where are you from?
  TextFormally:
        - May I mask from where you are?

- Label: RUN_VIDEO_EXHIBIT
  Text:
        - Let me show you a video about this exhibit.
  Cmds:
        - PROJECT Exhibit/video.mpg
```

Fig. 4: Example listing of a dialog acts file consisting of four dialog acts.

to trigger dialog acts, which enables more complex reactions to touch gestures, for example speech output.

It should be noted, that the capabilities of the projection and touchscreen submodules are not limited to preexisting media files, as we can display everything that could be rendered to a pixel buffer. Therefore, the modules allow the presentation of runtime generated media content – for example interactive games – which we plan to integrate in the future.

3.4 Client/Server Communication

We developed a remote control client that connects to a server component on the robot over the network. The server itself is an extension to an existing robot software stack, acquiring live camera images and audio from the sensors and streaming them over the network to the client, which plays them back to the remote operator.

On startup, the client loads and parses a dialog content file. The remote operator can trigger each dialog act by clicking the corresponding button. A screen shot can be seen in Fig. 5. Then the client forwards the triggered dialog act over network to the dialog act dispatcher on the robot. There, the dialog acts get resolved into commands which are sent to the listening subsystems. It should be noted that is does not matter for the subsystems where the commands come from, which allows a seamless migration between remote and autonomous dialog operation as required in Sect. 2.

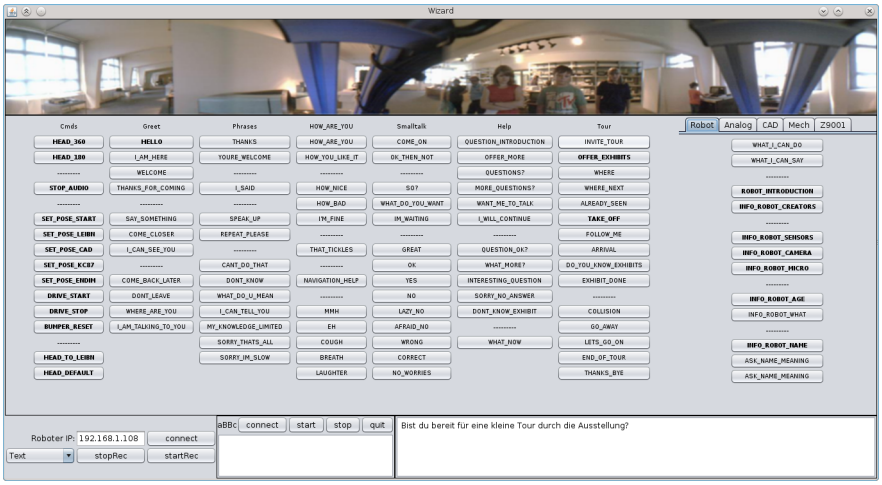

Fig. 5: Screen shot of the remote operation software. On the upper side, the camera image can bee seen. Below are the buttons to trigger dialog acts.

4 Discussion

In this section, we will discuss the usage of our dialog content extension in real world applications, regarding the tour guide use case and the building of a corpus for dialog management systems. Over time other use cases emerged, that we wanted to realize with our existing robot hard- and software. Our extension proved to be quite flexible and could also handle these new use cases with little to no modification to the existing setup. We will also discuss two additional use cases in the Sect. 4.3 and Sect. 4.4.

4.1 Tour Guide

We deploy our system in an exhibition of vintage computer hardware. As developers do not have access to all the resources and knowledge of the museum staff, a major part of the content authoring for the tour guide was done by the staff. To ease the process to them, we provided documentation and a template dialog acts structure that could be used as a building block for different exhibits.

We used the remote operation capability to give personalized tours to single visitors or smaller groups. In this setup, the remote workstation is located hidden from the visitors and connected to the robot via Wireless LAN. Our approach proved very suited to provide entertaining tours to visitors and gather real live data of dialog interactions. Also, we were able to evaluate already existing sub modules, like people tracking and path planning, in a candid real world environment.

Remote operating the robot has shown to be a complex task, as the operator has to choose an appropriate dialog option from a wide range of possibilities in a short period of time [9].

4.2 Corpus Building

To build a corpus, we use the logging capabilities of the robotic framework to record audio data and camera video streams. The dialog interactions triggered by the remote operator get recorded by a simple extension to the framework's logging capabilities. Due to lacking reliability of current speech recognition systems in our operational scenarios, it is unavoidable to resort to manual annotation of the recorded sensor data to comprehensively record the dialog interaction from the visitors towards the robot.

During several sessions we gathered a corpus of about six hours audio and video material, showing genuine interactions between robot and visitors. The corpus consists of 133 dialogs involving 378 test subjects. We annotated the corpus distinguishing about 30 different dialog situations, in which we transcribed all spoken utterances from the visitors. Additionally, major movement actions, the location and attention of the user were labeled.

The corpus analysis was very helpful in many ways. Firstly, it gave us a general feeling for the type of behavior to expect from visitors interacting with our system. We were able to designate four main classes of interaction behavior: *interested*, *chatting*, *passively interested*, and *not interested*. Surprisingly, most people reveal a chatting behavior, which included a lot of small talk before the interest shifts towards the museum exhibits.

Secondly, we computed various statistics of user behavior which were used to train a user simulator. The simulator model used is described in [6]. Using this simulator, we were able to reproduce versatile interactions of the tour guide scenario and train a dialog management system.

Thirdly, having all the user utterances annotated, we tested several algorithms for text classification. This allowed us to build a natural language processing module that can make use of a large-vocabulary speech recognizer to recognize broad range of speech inputs.

4.3 Info Terminal

We deployed a robot as an advanced info terminal at a variety of venues. There, the robot ought to provide basic information, such as schedules and maps of the location, to users using the touch screen. Remote dialog operation was not used.

This use case has been implemented without modifications to our software, only by creating dialog content. We had to write a dialog acts file and appropriate browsable pages for the touchscreen. The ability to trigger dialog acts from page elements (see Sect. 3.3) like buttons, allowed not only to let users progress between pages, but also to let the robot verbally utter descriptions of the pages.

We tested our info terminal application on various occasions and it behaved as expected. But to further improve this use case, a simple dialog system could be added, that employs data from our people tracker to allow the robot to react to nearby persons and automatically advertise itself as a source of information.

4.4 Poster Presenter

We also wanted to use our robot in an entertaining way as a presenter for posters at exhibitions, workshops and conferences. To present a poster, the robot highlights a certain area on a poster pinned to a wall using its projector, while uttering speech towards its audience. The touchscreen is used to show supplementary information. After a poster area has been explained, the robot proceeds to the next one.

For this setup, the projection submodule is used to simply project black images containing white patches matching the areas of the poster. Every poster section is represented by a dialog act. The dialog progress is controllable either remotely by an remote operator or automatically. For the automatic progression, we use JavaScript-Timers in the touchscreen HTML page to trigger the following dialog act.

Both variations were tested successfully on various occasions. However, the additional flexibility that currently only the operator can warrant, allows for a sometimes desirable variation from an otherwise static flow of information towards the user. Further information about the projection setup of this application are presented in [4].

5 Conclusion

In this paper, we proposed a concept to deal with the problem of dialog content handling in robotic applications. The described software stack did not only regard the organization of dialog content and its presentation, but also the authoring phase. By involving domain experts with the required domain knowledge in the authoring phase, the dialog content can become more useful and achieves a higher quality. In the end, this will increase the usefulness of the resulting robotic application to the user and might improve his impression of how interesting and pleasant the interaction with the robot turns out to be.

The presented approach proved highly versatile and flexible, as it allowed the realization of different applications by merely authoring additional content. Uttered texts are the main focus, but also on-screen content, images, and videos are considered.

By being able to remote control the dialog flow, the approach allows to build a corpus of real world dialog data needed for the further improvement of dialog systems. The migration to a completely autonomous dialog system can directly be done utilizing the existing implementation.

Further Work The extension of the framework towards our application is fairly complete. However, there is still room for improvement. As we are employing predefined text blocks as the foundation for spoken utterances, an extension towards less static representations of text might be desirable for further iterations of our dialog system.

In regards to the requirements noted in Sect. 2, the content creation phase, and especially the externalization aspect, could be optimized further. Even if we

employed a very simple markup language to hold dialog actions, a special written editing software could still prove more user friendly. Such a software could easily be integrated into the current workflow.

References

1. Agha, G.: ACTORS: A Model of Concurrent Computation in Distributed Systems. MIT Press, Cambridge, MA, USA (1986)
2. Araki, M.: Proposal of a markup language for multimodal semantic interaction. In: Proceedings of the 2007 Workshop on Multimodal Interfaces in Semantic Interaction. pp. 58–62 (2007)
3. Araki, M., Tachibana, K.: Multimodal dialog description language for rapid system development. In: Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue. pp. 109–116 (2006)
4. Donner, M., Himstedt, M., Hellbach, S., Böhme, H.J.: Awakening history: Preparing a museum tour guide robot for augmenting exhibits. In: Proceedings of the European Conference on Mobile Robots (ECMR). pp. 337–342 (2013)
5. Einhorn, E., Langner, T., Stricker, R., Martin, C., Gross, H.M.: Mira - middleware for robotic applications. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS). pp. 2591–2598 (2012)
6. Fonfara, J., Hellbach, S., Böhme, H.J.: Learning Dialog Management for a Tour Guide Robot using Museum Visitor Simulation. In: Proceedings of the Workshop - New Challenges in Neural Computation 2012 (NC2). pp. 61–68 (2013)
7. Gerkey, B.P., Vaughan, R.T., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: Proceedings of the 11th International Conference on Advanced Robotics. pp. 317–323 (2003)
8. Giuliani, M., Kaßecker, M., Schwärzler, S., Bannat, E., Gast, J., Wallhoff, F., Mayer, C., Wimmer, M., Wendt, C., Schmidt, S.: Mudis - a multimodal dialogue system for human-robot interaction. In: Proc. 1st Intern. Workshop on Cognition for Technical Systems (2008)
9. Poschmann, P., Donner, M., Bahrmann, F., Rudolph, M., Fonfara, J., Hellbach, S., Böhme, H.J.: Wizard of Oz revisited: Researching on a tour guide robot while being faced with the public. In: 21th IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN). pp. 701–706 (2012)
10. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
11. Shiomi, M., Kanda, T., Koizumi, S., Ishiguro, H., Hagita, N.: Group attention control for communication robots with wizard of oz approach. In: Proceedings of Conference on Human-Robot Interaction (HRI). pp. 121–128 (2007)
12. Shuyin, I.T., Toptsis, I., Li, S., Wrede, B., Fink, G.A.: A multi-modal dialog system for a mobile robot. In: Proc. Int. Conf. on Spoken Language Processing. pp. 273–276 (2004)
13. Wallace, R.S.: The anatomy of a.l.i.c.e. In: Epstein, R., Roberts, G., Beber, G. (eds.) Parsing the Turing Test, pp. 181–210. Springer Netherlands (2009)