Wizard-of-Oz Interfaces as a Step Towards Autonomous HRI

Nikolas Martelaro
Stanford University, Center for Design Research
nikmart@stanford.edu

Abstract

Within the HRI community, Wizard-of-Oz (WoZ) methods have been extensively used to explore robot interactions. However, their wide usage has also been criticized as not giving good representation of how real, autonomous robots can interact. In this paper, I argue that thoughtfully designed Wizard-of-Oz control interfaces can enable robot designers to explore the interaction design space fur human robot interaction *and* also move them towards autonomous robot development. I provide two case studies of WoZ controlled robots and their interfaces and describe the design rationale behind them. I then provide an initial set of design considerations for future WoZ controllers as a way to improve how WoZ interactions can inform the design of autonomous behavior and move closer to fully autonomous robots.

Introduction

Developing appropriate and effective Human-Robot Interaction often requires the design of various elements of robotic behavior. Dialogue, proximity, body movement, language recognition, and timing all need to be carefully considered to create positive and impactful experiences with social robots. Although robot designers often aim to create fully autonomous robots, it is often impossible to effectively explore the interaction design space for robot behavior while focusing on autonomous function. Interaction designers often employ Wizard-Of-Oz (WoZ) techniques (Dahlbäck, Jönsson, and Ahrenberg 1993) to explore possible robot behaviors such as natural language interfaces, non-verbal feedback, and motion (Riek 2012). In a review of WoZ within HRI (Riek 2012), most WoZ studies constrained robot behaviors, however few limited the sensing capabilities presented to the human Wizard. This raises issues with WoZ studies and their applicability to creating fully autonomous robot interactions as complex, high fidelity sensing and planning are often unrealistic for even advanced robots.

To help reconcile the issues of effectively exploring the interaction design space using WoZ control and developing truly autonomous robot interactions, more focus and attention should be given to the creation of Wizard control interfaces. A well designed Wizard control interface can limit the Wizard's interactions and sensing capabilities, allowing them to explore not only possible behaviors but also gain a better understanding of the robot's limitations. These constraints can allow for better approximations of autonomous behavior while still allowing rapid, iterative testing of robot behaviors.

Currently though, the creation of Wizard control interfaces are often custom pieces of GUI software or hardware controllers (i.e. joysticks, gamepads). There have been some efforts to create standardized interfaces for creating Wizard control interfaces, such as the Polonious project. Polonious uses a high level markup language and Finite State Machines built upon ROS to present a Wizard with possible next robot actions based on the current system state (Lu and Smart 2011). Although this system allows for easier creation of control interfaces, there appears to be little development on the project[1]. At the same time, web frameworks developed in JavaScript have reduced the complexity in developing interactive websites and could serve as a good foundation for creating control interfaces. Indeed, the ROS community has already created JavaScript libraries for interacting with ROS enabled robots, including keyboard control, navigation, and visualization[2]. However, there does not seem to be a specific toolkit for creating Wizard control interfaces that expose both the capabilities and limitations of the robot while supporting robot designers in moving towards autonomous robot development.

In order to allow for better robot interaction design that ultimately leads to autonomous HRI, I argue that robot platforms should provide a usable and easily developed framework for creating Wizard control interfaces. In the rest of this paper, I present two case studies of the design of custom robot control interfaces, discuss the underlying

[1] http://wiki.ros.org/polonius
[2] http://robotwebtools.org

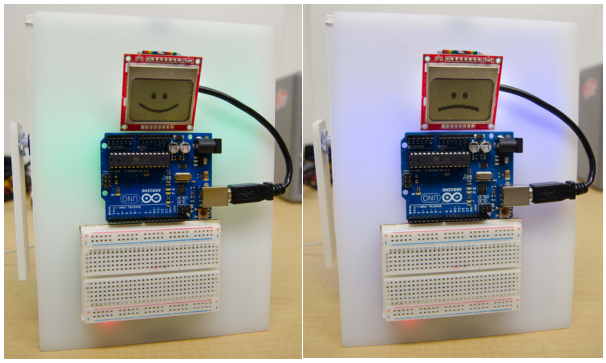

Figure 1 – Robot tutor; an expressive, electronics teacher

tools to develop the interfaces, and elaborate on design considerations for future robot WoZ control interface tools.

Case Studies

Robot Tutor

The Robot Tutor (Figure 1) is an interactive robot to help students learn how to build and program electronics. The robot interacts with students by guiding them through a tutorial via voice conversation, emotional expression via an LCD and colored lights, and arm movement. The system is controlled via Wizard-of-Oz from a workstation in another room.

In order to better approximate autonomous behavior, the robot's speech and motions were scripted. The Wizard was primarily concerned with the timing aspects of the tutorial and only initiated the next robot action based on when the student stopped speaking, said "repeat" to repeat the last thing the robot said, or said "I don't know" if the Wizard could not understand the student's speech. Aside from the "repeat" keyword or poor audio, the student's speech content did not influence the robot's behavior. The Wizard was provided a high quality audio/video channel from cameras inside the room, allowing the Wizard to clearly hear and see the student's actions.

A networked wizard control GUI (Figure 2) was written using Python and ZeroMQ, enabling one button control of both audio and emotion expression from a remote workstation. A local Python script on the robot, controlled by a Raspberry Pi, received messages from wizard control program and passed serial commands to an Arduino controlling the physical entities of the robot.

This interface allowed for rapid exploration of students interacting with a robot tutor without the need for developing a natural language processing engine. The scripted nature of the control interface minimized the Wizard's sensing responsibilities in a way that could plausibly implemented in an autonomous robot with a robust speech detection algorithm. This being said, the Wizard still had much higher fidelity sensing capabilities than would be available to an autonomous robot. A future interface may benefit

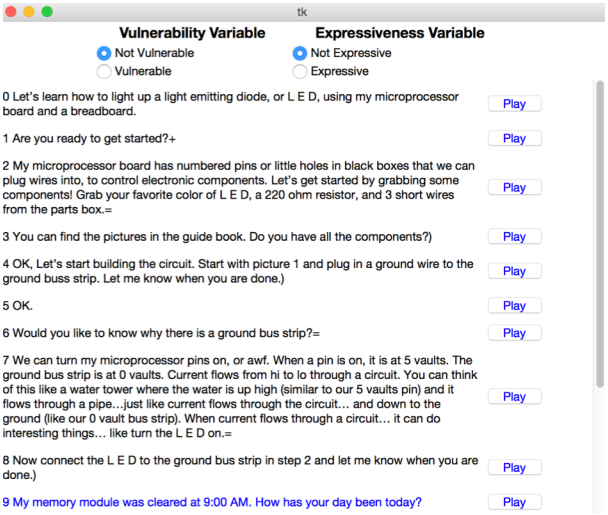

Figure 2 – SofaBot control interface

from having a lower quality audio/video feed built into the robot. Additionally, if the robot had a built in speech detection algorithm, the interface could show the Wizard what the robot interpreted and act mostly autonomously, with the Wizard providing oversight. This would provide a robot interaction designer the ability to better understand the robot's sensing limitations and explore interactions in a WoZ manner with these limitations in mind.

SofaBot

The SofaBot (Figure 3) was developed to explore how everyday objects, such as sofas, could be made expressive and interactive. To develop a robotic Sofa, we used an ODROID U3 single board Linux computer running a node.js webserver to serve a motion control webpage directly from the couch. The webpage pulls a live video stream from a USB webcam connected to the computer using 'mjpg-streamer[3],' a command line tool for streaming live video. The webpage also features information about whether someone is sitting on the couch using analog force sensitive resistors connected via an Arduino Nano. The sofa is controlled remotely via the webpage by keyboard strokes and on-page buttons (Figure 4). The node.js webserver code sends simple messages encoding speed and direction to the Arduino Nano, which controls high power motors that move the sofa. All systems are battery powered, making the unit entirely mobile.

The goal of the SofaBot was to explore how people would physically interact with the couch to inform potential motion planning and visual processing algorithms. The SofaBot control interface provided the Wizard with a realistic webcam view from the robot's perspective and no audio. This forced the Wizard to work with a time-lagged, narrow field of view image, similar to what could be processed by an on-board computer vision algorithm and to

[3] http://sourceforge.net/projects/mjpg-streamer/

Figure 3 – SofaBot: an expressive, web-controlled sofa

ignore or be influenced by vocal commands. Additionally, the interface provided binary information about the SofaBot's "person sitting" sensors, providing the Wizard with the same information known by the robot.

Although the control interface provided realistic sensing capabilities, high level motion planning and robot behavior was still done by the Wizard. A future version of this control interface could show various filtered image views that better represent machine vision and could aid the designer in developing their vision algorithms. Additionally, the interface could log all actions made by the Wizard, allowing for the logs to be used in creating autonomous motion control algorithms. This control was implemented simply for a "wiggle" motion, however future interfaces could record and list more complex motion actions.

Considerations for WoZ Interface Tools

The two previous case studies show examples of WoZ interfaces and describe some of the motivations for their design. From these examples, I propose a set of design considerations for creating Wizard control interfaces.

On-board control interfaces using established tools

A robot should provide its own control interface that gives easy access to and displays the robot's sensing and action capabilities. Although the Robot Tutor interface approximated autonomous robot behavior well, the interface itself required significant coordination between the multiple Python scripts running on the robot and the control workstation. The SofaBot case study provides a better example where the control interface is served via JavaScript by the robot itself, requiring less code coordination and allowing for easier display of the robot's actual sensing capabilities. By providing an on-board system and a widely used programming interface, robot designers can more quickly and easily create controller interfaces. This can allow them to

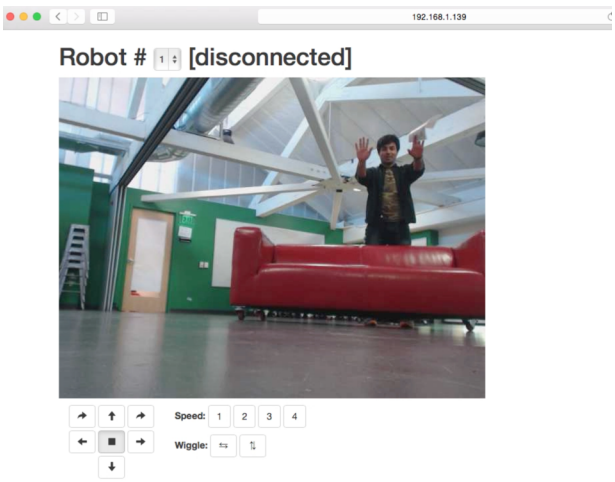

Figure 4 – SofaBot control interface

focus more on how the design of these interfaces can help them explore interactions and move towards autonomy.

Realistic sensing presentation

In order to move designers closer towards autonomous HRI, control interfaces should provide a representation of actual sensor data. This will allow the robot designer to better understand the limitations of the robot and can help to designer develop the actual autonomous algorithms they could program into the robot. Although the Wizard in the Robot Tutor case study was trained to only respond to stopped speech or a few keywords, an interface that showed something like audio levels or even speech-to-text could allow for a more realistic middle ground between rapid interaction exploration and robot autonomy. For the SofaBot, the use of the actual camera feed forced the Wizard to deal with the limitations of the video stream, but still provided a very human readable image. If the system could provide computer vision filters to the video, the designer may better be able to understand the different features that could be useful for autonomous algorithms.

Adjustable robot behavior control

The control interface should provide the ability to freely control or fix robot behaviors. The robot tutor showed an example of highly controlled robot behavior. While this better approximated autonomous behavior, it did not allow for the Wizard to explore interactions in a more freeform manner. To develop the fixed robot behaviors, iterative pilot testing was done where the Wizard had more control over the speech, facial expression, and movement separately. This informed the design of the fixed behaviors but required a new interface to be developed. A future control interface tool should allow for the Wizard to tune the amount of control they have, allowing them more free expression during early interaction explorations, and more fixed control when testing specific robot behaviors.

Wizard action logging

The control interface should log the Wizards actions for future analysis and autonomous algorithm development. Just as macros can record user interactions to help automate tasks in computer software, a Wizard control interface could also record and playback interactions. Indeed, this idea has been explored by Knox et. al. as Learning from the Wizard (LfW) (Knox, Spaulding, and Breazeal 2014). By using the control interface, those without extensive programming background can design robot behaviors and have them play back in an autonomous fashion. For example, the movements of the SofaBot could be recorded during exploratory interaction sessions. In future sessions, the captured movements could be played back using single buttons rather than through intricate control by the Wizard. As development continued, these recorded motions could be linked to specific sensor readings provided be the control interface. Knox et. al. demonstrated a version of this type of system where a robot would collect data from the environment and associate that with the Wizard control to learn a model of socially appropriate behavior. We can take inspiration from Sauppé and Mutlu's *interaction design patterns* (Sauppé and Mutlu 2014). Their work creating reusable interaction blocks and a visual authoring environment allowed interaction designers to quickly create robot interactions. However, their interaction blocks were pre-scripted. By allowing for recording of WoZ interactions, we can explore and create interaction blocks at the same time, allowing for faster development.

Conclusion

The creation of autonomous robots often takes significant development time. Although autonomy is often an end goal for robots, focusing solely on autonomy can hinder exploration of the interaction design space. To reconcile these goals of free interaction exploration and robot autonomy, I propose that Wizard-of-Oz control interfaces should be designed with both in mind and should provide designers with tools to guide them towards autonomous robots. By having robots provide on-board control interfaces, designers can create controllers that can give them wide control of the robot while also providing an accurate representation of the robot's capabilities. While I have only provided a small set of design considerations, I hope that they can act as a starting point for the development of future WoZ control interfaces. By providing easy tools for creating control interfaces based on the robot's capabilities, HRI researchers and designers can better use WoZ as a stepping stone towards autonomy. In doing so, we can move away from common complaints of WoZ studies and focus on the goal of creating more impactful HRI experiences.

References

Dahlbäck, Nils, Arne Jönsson, and Lars Ahrenberg. 1993. "Wizard of Oz Studies: Why and How." In *Proceedings of the 1st International Conference on Intelligent User Interfaces*, 193–200. ACM. http://dl.acm.org/citation.cfm?id=169968.

Knox, W. Bradley, Samuel Spaulding, and Cynthia Breazeal. 2014. "Learning Social Interaction from the Wizard: A Proposal." In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*. http://web.media.mit.edu/~samuelsp/resources/mlis_14.pdf.

Lu, David V., and William D. Smart. 2011. "Polonius: A Wizard of Oz Interface for HRI Experiments." In *HRI '11*, 197–98. HRI '11. New York, NY, USA: ACM. doi:10.1145/1957656.1957729.

Riek, Laurel D. 2012. "Wizard of Oz Studies in Hri: A Systematic Review and New Reporting Guidelines." *Journal of Human-Robot Interaction* 1 (1). http://www.humanrobotinteraction.org/journal/index.php/HRI/article/view/9.

Sauppé, Allison, and Bilge Mutlu. 2014. "Design Patterns for Exploring and Prototyping Human-Robot Interactions." In *CHI '14*, 1439–48. CHI '14. Toronto: ACM. doi:10.1145/2556288.2557057.