Polonius: A Wizard of Oz Interface for HRI Experiments

David V. Lu[*]
davidlu@wustl.edu

William D. Smart[*†]
wds@cse.wustl.edu

[*]Washington University
One Brookings Drive
St. Louis, MO 63130
United States

[†]Willow Garage
68 Willow Road
Menlo Park, CA 94025
United States

ABSTRACT

Polonius is a robot control interface designed for running Wizard of Oz style experiments. It is designed to be easy enough to be used by the non-programmer collaborators of roboticists. The program acts as an intermediary between the robot and a wizard interacting with a GUI based on a pre-defined script. Polonius also eliminates the need for coding the video after experiments by integrating a robust logging system.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Human-Robot Interaction; Graphical User Interface*

1. MOTIVATION

Wizard of Oz interfaces [4] allow experiments to be performed using behaviors not yet implemented. In a Wizard of Oz interface, the robot's actions are controlled by a "man behind the curtain" who acts as a puppeteer. This facilitates the interaction in two key ways. First, it allows for the intelligence of the system, i.e. the decision making, to be delegated to the human, who is more often capable of making complex decisions more quickly than an algorithmic decision maker. Second, it often allows the wizard to act in lieu of or in conjunction with the robot's sensors, eliminating the need for complex sensor interpretation algorithms to try to determine what the human has done.

While many people use Wizard of Oz interfaces in human robot interaction (HRI) research, there are currently no frameworks systematized for running such experiments. Other fields have developed frameworks with some success, such as SUEDE [5] for prototyping voice based interfaces. Also, since the field of HRI thrives on collaborations with non-programmers, there is a need to create tools which offer a high level interface to controlling and re-tasking robots.

Furthermore, one of the most tedious parts of running experiments is coding the video and other sensor streams after the fact. Most experiments need to measure some component of the human's reactions to the robot, including how they reacted and when. The process of gathering this information usually entails watching a video of the interaction after the experiment is over and painstakingly recording the details. This is an expensive process, both in terms of money and time.

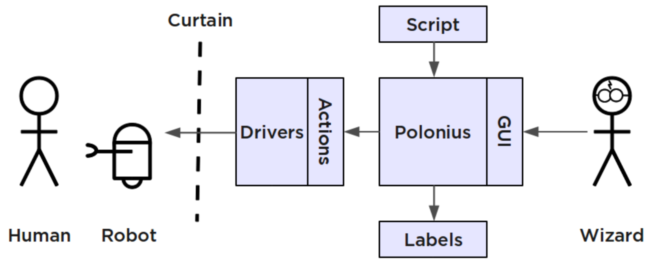

Figure 1: Program Flow

In addition to HRI experiments, there are many other scenarios where a robot must perform a set of scripted actions with a human in the control loop, such as demos or other public exhibitions. One case in an emerging field is controlling a robot on the stage in a play. There are numerous examples of robots interacting with humans on stage, performing with their human counterparts [3, 1, 2]. In this context, it is often much easier to have a human control the robot than to program the robot to act autonomously. Additionally, using a Wizard of Oz interface has no ill affect on the audience, since the control happens quite literally from behind the curtain.

Polonius is a front-end interface to control robots in an easy, scriptable manner and to simultaneously code the interaction during the actual interaction. Polonius is named for the wise character who hides behind a curtain in Act V of the Shakespeare tragedy Hamlet [8].

2. FRAMEWORK

Polonius is built using the ROS Framework [6], allowing for a high level of modularity and adaptability for many different types of robots. Specifically, the `smach` and `actionlib` libraries[1] are used in the back-end of Polonius.

The core program flow is shown in Figure 1. Prior to the experiment, the script is developed, defining all of the possible actions for the human and the robot in the form of a finite state machine (FSM). During the interaction, the wizard is presented with different actions, which they can command the robot to do, and with different labels, which they can select to record what the user is doing. These options are defined by the script and by the current state of

[1]`http://www.ros.org/wiki/smach`
`http://www.ros.org/wiki/actionlib`

```
child: {start: Child starts gesture, label: Child Gesture,
        end: {Gesture A: doA, Gesture B: doB,
              Gesture C: doC, Finish: wave}}
doA: {start: 2, label: Gesture A, action: Pose(0), end: child}
doB: {start: 2, label: Gesture B, action: Pose(1), end: child}
doC: {start: 2, label: Gesture C, action: Pose(2), end: child}
wave: {label: Wave Goodbye, action: Wave() }
```

Figure 2: Gesture Imitation Game Script

the FSM. After the interaction, the labels and actions can be viewed through the log file.

To better understand how the system is used, consider the Wizard of Oz interface used in Robins et al.'s Gesture Imitation Game experiment [7]. The core of the experiment had a humanoid robot imitating gestures made by a child. The child would make one of the three gestures that the robot could make, and then the robot would make the same gesture, and then the process would repeat, ending when the robot waved goodbye. Researchers were curious whether a two second delay between the child's gesture and the robot's gesture would effect the child's timing. A script for such an experiment is show in Figure 2, and we use it here to illustrate the capabilities of Polonius.

The script is specified in a text file, using a high-level language, representing a finite state machine. Here, we define five states: the first for the child's action, three for the robot doing one of the three gestures, and one for the concluding gesture. Formulating the script as a finite state machine allows for linear, branching and looping scripts. Each state is defined by up to five properties: identifier, label, action, start, and end. The identifier is the name of the state (`child, doA...`), used internally, and the label is a more verbose description of what the state is, used in the GUI and the labels.

The action defines what commands to give the robot. The `actionlib` library provides a framework for defining and implementing pre-emptable, parameterized tasks. In our example we use `Pose(x)` to represent a command that moves the robot to the pose for the xth gesture. We could have also specified multiple actions to be done in parallel. If no action is given, then the action is considered to be a human action, where the robot just waits for the wizard to mark the human's action as finished.

The start and end properties define when to move into and out of each state. If the start is defined as a number as in state `doA`, the FSM waits that many seconds before starting the action. However, if the action should not start automatically, a cue can be defined, such that the wizard will wait for the action to happen (the cue), and then press a button when it happens, allowing for the action to proceed. The state `child` defines when the child is doing a gesture, and starts when the wizard clicks a button on the GUI labeled "Child starts gesture." The FSM will then leave the state either automatically, or if the end property is defined, when the wizard clicks on the button. If multiple ends are defined, then the next state is defined by which button the wizard clicks. In the `child` state, if the child does Gesture A, then the wizard would click the corresponding button, prompting the FSM to go into the `doA` state and do Gesture A back.

Once the interaction is complete, labels marking what buttons the wizard pressed and what actions were performed are saved for later processing, either in a text file or using ROS's built-in logging utility. This allows researchers to examine the timings of the starts of the child's actions, which actions were performed, and how long of a pause there was between the robot's gesture and the child's next gesture, all without coding the video and minimal extra work on behalf of the wizard.

3. CONCLUSION

We have created a flexible framework for running robots through a scripted Wizard of Oz style interaction, useful for HRI experiments and other interactions. The system has already been used to run a dramatic presentation involving a robot and a human actor at Washington University, and the code has been released to the ROS community[2]. In the future, we plan to investigate the efficacy of using Polonius vs. coding the video by hand, which we predict will show that Polonius presents an easy interface for controlling robots while simultaneously coding the actions in a timely and accurate way.

4. THANKS

This work as supported by NSF award IIS 0917199.

5. REFERENCES

[1] BBC News. Actor robots take japanese stage. 26 Nov. 2008. `http://news.bbc.co.uk/2/hi/asia-pacific/7749932.stm`.

[2] B. Brantley. In robot world she turns more Hedda than Hedda. *The New York Times*, 18 Feb. 2006. `http://theater2.nytimes.com/2006/02/18/theater/reviews/18hedd.html`.

[3] G. Hoffman, R. Kubat, and C. Breazeal. A hybrid control system for puppeteering a live robotic stage actor. In *Proceedings of the 17th International Symposium on Robot and Human Interactive Communication (RoMan'08)*, pages 354–359, 2008.

[4] J. Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41, 1984.

[5] S. Klemmer, A. Sinha, J. Chen, J. Landay, N. Aboobaker, and A. Wang. Suede: a Wizard of Oz prototyping tool for speech user interfaces. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 1–10. ACM, 2000.

[6] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source Robot Operating System. In *International Conference on Robotics and Automation*, 2009.

[7] B. Robins, K. Dautenhahn, R. Te Boekhorst, and C. Nehaniv. Behaviour delay and robot expressiveness in child-robot interactions: a user study on interaction kinesics. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 17–24. ACM, 2008.

[8] W. Shakespeare and A. Verity. *The Tragedy of Hamlet.* University Press, 1904.

[2]`http://www.ros.org/wiki/polonius`