Brown ROS Package: Reproducibility for Shared Experimentation and Learning from Demonstration

Sarah Osentoski, Graylin Jay, Christopher Crick, Odest Chadwicke Jenkins
Department of Computer Science
Brown University
Providence, RI , 02912 USA
{sosentos | tjay | chriscrick | jenkins }@cs.brown.edu

Abstract

We describe our efforts toward developing infrastructure for shared remote robotics laboratories that allow reproducible robotics experimentation and large scale learning from demonstration (LfD). Our goal is to create a facility where users demonstrate desired tasks by teleoperating robot platforms through video-game style web-based interfaces, providing data that LfD approaches will turn into robot policies. We describe a selection of open source tools, provided by the Brown robotics lab, created in service of building such a remote lab.

Introduction

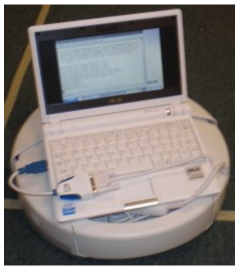
Our research centers around the creation of a remote robotics laboratory to enable large-scale Learning from Demonstration (LfD). Before robots can enter everyday home and workplace environments, roboticists must lift the substantial burden of entry that currently prevents end users from easily interacting with and programming robots. We propose "large-scale robot LfD" with a "demonstrate, test, iterate" development process that uses a repository-type structure in which massive amounts of demonstrated data can be stored, tagged, filtered, and "compiled" into controllers. Our belief is that many LfD approaches (summarized by (Argall *et al.* 2009; Billard *et al.* 2008)) will benefit from access to truly large stores of data, in the same way that simple machine learning techniques in other contexts have become vastly more capable when exposed to the wealth of information now easily available on the internet. Our remotely-accessible laboratory environment will provide a large number of end users the ability to remotely interact with robots, incidentally creating large databases of demonstration data.

Towards these goals we have been working on open resources to enable general users and researchers to run experiments on robots. Our tools are built upon ROS (Quigley *et al.* 2009), Willow Garages's robot middleware system. The software discussed in this paper is available in the brown-ros-pkg[1].

[1]http://code.google.com/p/brown-ros-pkg/

Supported Platforms

Our work has focused on off-the-shelf systems that are available for purchase by other research groups. While much of the work in the Brown ROS Package can be used on any platform running ROS, we also provide drivers for two widely-available platforms: the iRobot Create and the Aldebaran Nao (pictured in Figure 1).


(a) iRobot Create

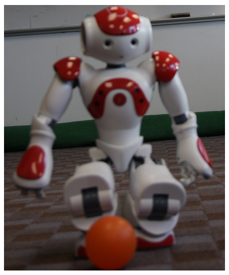

(b) Aldebaran Nao

Fig. 1. Platforms supported in the Brown ROS package .

Tools For Lab Environment

In this section, we describe tools we have built towards creating a remote lab environment and to aid the community in developing robot applications.

Gscam: This package leverages Gstreamer (Gstreamer 2010), a multi-media framework used by applications such as Gnome. It provides compatibility between ROS and cameras supported by Linux. This allows any camera supported by Linux to be used by ROS with ease. Gscam can also make realtime adjustments to video streams in order to add features such as white balancing for cameras that lack such processing abilities.

Augmented Reality Tag Recognition: We have created a package wrapping the ARToolkit (Kato & Billinghurst 2004), a software library for building augmented reality (AR) applications. Robotic tasks often depend upon visual information. However many vision algorithms require precise camera calibration, consistent lighting conditions, and significant knowledge of the environment. AR tags can be added to robotic domains to augment or replace vision algorithms for some tasks. AR tags, pictured on the right of Figure 2, can be used to mark and identify objects of interest, as landmarks for localization, or to command robots in various ways.

This package is fully integrated with the ROS build environment. This enables ROS users to use ARToolkit through a ROS messaging interface. It allows users to use ARToolkit as they would any other ROS package, without linking or a secondary external compilation step.

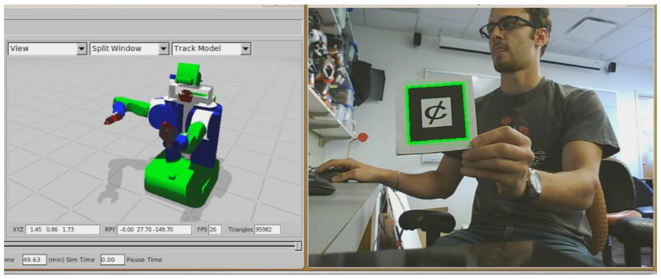


Fig. 2. A user uses an AR tag to teleoperate the arm of a simulated robot.

rosjs: rosjs is a light-weight Javascript (E.C.M.A. International 2009) binding for ROS that allows developers to expose robot functionality as web services. rojs is designed to enable end users and developers to leverage the capabilities of ROS through a standard web browser. Developers of robot applications can leverage the power of HTML to build engaging applications and interfaces for robots quickly without recompiling multiple ROS nodes. Additionally, rosjs allows users to access and run ROS-based applications without the need for *any* installed software beyond a plain browser. rosjs exposes underlying ROS services and topics as objects. It also extends ROS by providing mechanisms for security and data logging.

rosjs consists of a server and a pure Javascript library. It is not tied to any particular web-server or framework; it even works when served remotely. rosjs uses websockets which provide low latency for teleoperation, closed-loop control, or a variety of other robot applications.

Remote Lab

The tools we have discussed in the previous sections are part of our larger goal to create a remote laboratory. We envision users who will be able to schedule time on the robot and interact with the robot using a remote web interface. This interface can be used to demonstrate tasks or to visualize the robot as it performs tasks provided by custom built controllers. During each session data is logged in stored in a repository that is publicly available. Custom controllers and learning algorithms can use the data and provide policies for desired tasks on the robot.

Many of the tools we have discussed focus upon user interaction with and visualization of the robot. Many different types of visualization are possible using rosjs. Figure 3 shows one potential interface created using rosjs. Since

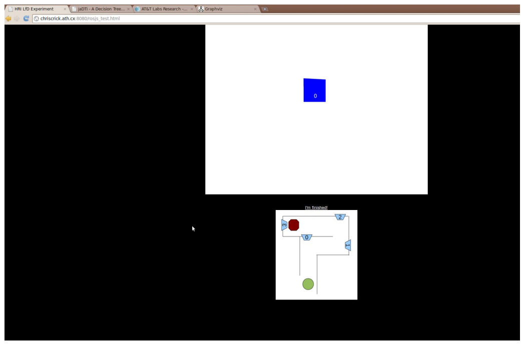


Fig. 3. Example of a remote laboratory interface in which users navigate a robot through a maze with a visualization of the AR tags visible to the robot.

the lab is subject to latency across the internet, visualizations must be carefully selected. High quality video may require a substantial amount of bandwidth and thus create a situation where the image that the user sees is not an accurate reflection of the state of the robot. An additional consideration is that the type of visualization that is provided to an end user may influence the type of commands they provide to the robot. Humans are able to leverage a large amount of information from a video stream that the robot is unlikely to be able to process. However the visualization must be intuitive to humans so that they can provide quality demonstrations.

Acknowledgements

We thank the members of and students who have worked with the Brown Robotics Laboratory, especially Jesse Butterfield, Gal Peleg, and Sam Pucci.

References

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57:469–483.

Billard, A.; Calinon, S.; Dillmann, R.; and Schaal, S. 2008. Robot programming by demonstration. In Siciliano, B., and Khatib, O., eds., *Handbook of Robotics*. Secaucus, NJ, USA: Springer. 1371–1394.

E.C.M.A. International. 2009. ECMA-262: ECMAScript language specification, 5th ed. Online. http://www.ecma-international.org/publications/standards/Ecma-262.htm.

Gstreamer. 2010. Gstreamer: open source multimedia framework. Online. http://www.gstreamer.net/.

Kato, H., and Billinghurst, M. 2004. Developing ar applications with artoolkit. In *ISMAR*, 305.

Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; and Ng, A. 2009. ROS: an open-source robot operating system. In *Proc. Open-Source Software workshop of the International Conference on Robotics and Automation*.