Int J Soc Robot
DOI 10.1007/s12369-012-0157-8

Remote Robotic Laboratories for Learning from Demonstration

Enabling user interaction and shared experimentation

Sarah Osentoski · Benjamin Pitzer · Christopher Crick · Graylin Jay · Shuonan Dong · Daniel Grollman · Halit Bener Suay · Odest Chadwicke Jenkins

Accepted: 13 June 2012

Abstract This paper documents the technology developed during the creation of the PR2 Remote Lab and the process of using it for shared development for Learning from Demonstration. Remote labs enable a larger and more diverse group of researchers to participate directly in state-of-the-art robotics research and will improve the reproducibility and comparability of robotics experiments. We present solutions to interface, control, and design difficulties in the client and server-side software when implementing a remote laboratory architecture. We describe how researchers can interact with the PR2 and its environment remotely through a web interface, as well as develop similar interfaces to visualize and run experiments remotely.

Additionally, we describe how the remote lab technology was used by researchers participating in the Robot Learning from Demonstration Challenge (LfD) held in conjunction with the AAAI-11 Conference on Artificial Intelligence. Teams from three institutions used the remote lab as their primary development and testing platform. This paper reviews the process as well as providing observations and lessons learned.

S. Osentoski (✉) · B. Pitzer
Bosch Research and Technology Center, 4005 Miranda Ave, Palo Alto, CA 94304, USA
e-mail: sarah.osentoski@us.bosch.com

B. Pitzer
e-mail: benjamin.pitzer@us.bosch.com

C. Crick · G. Jay · O.C. Jenkins
Brown University, Box 1910, 115 Waterman St, Providence, RI 02912, USA

C. Crick
e-mail: chriscrick@cs.brown.edu

G. Jay
e-mail: tjay@cs.brown.edu

O.C. Jenkins
e-mail: cjenkins@cs.brown.edu

S. Dong
Massachusetts Institute of Technology, 362 Memorial Dr. W7-203, Cambridge, MA 02139, USA
e-mail: dongs@mit.edu

D. Grollman
EPFL, Ecole Polytechnique Federale de Lausanne, EPFL-STI-I2S-LASA, Station 9, 1015, Lausanne, Switzerland
e-mail: daniel.grollman@epfl.ch

H.B. Suay
Worcester Polytechnic Institute, 100 Institute Dr., 125 Atwater Kent Labs, Worcester, MA 01609, USA
e-mail: benersuay@wpi.edu

Keywords Remote laboratories · Robot learning from demonstration · Robot interfaces

1 Introduction

A significant challenge facing the effort to bring robots into the daily lives of nontechnical users is their limited and inflexible behavioral repertoire. Expert roboticists can program new policies and skills within specialized domains such as manufacturing and lab experimentation, but this approach is not scalable to widespread deployment of robots into homes and offices. Learning from Demonstration (LfD) [3, 5] has been proposed as a potential solution. Under the LfD framework, a user, also known as a teacher, provides demonstrations of a desired task, which are then used to develop a policy through machine learning techniques. A variety of approaches have been proposed as potential solutions

Published online: 26 June 2012

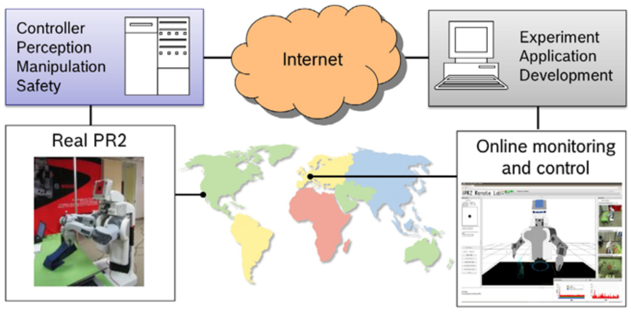


Fig. 1 In the PR2 Remote Laboratory clients connect over the internet, run experiments, and develop robotic applications on a shared PR2 platform

for LfD including supervised learning [4, 7, 11, 19], reinforcement learning [1, 22, 31, 35, 40], and behavior-based approaches [28]. While these and other approaches are evidence of the substantial progress that has been made in the field, a fully realized system that works on a broad set of desired tasks has not yet been achieved.

This article focuses not on the specifics of a LfD technique, but on a means to enable LfD research on a broader scale. We examine how remote web-based robotic laboratories can be used to advance LfD, describing tools for creating such laboratories and web-based robot interfaces. These tools were part of our efforts to create the PR2 Remote Laboratory, sketched in Fig. 1. As illustrated by the high response rate to the PR2 Beta Program, the demand for high-quality research platforms far outweighs the supply. These expensive platforms are difficult to obtain for smaller universities and research groups.

Moreover, researchers have long found it difficult to make direct comparisons between the effectiveness of algorithms and techniques for any number of robotics tasks, including LfD. Approaches are often presented for different applications, on different robots, and only to solve partial problems without integration into complete systems. These factors limit the productivity of the robotics research community and the development of the robotics market. Remote research laboratories can enable researchers to develop, test, and compare robot controllers in a unified setting. This will not only allow for objective comparisons between different solutions to the same problem but also uncover strengths and limitations of current approaches more efficiently.

Previous attempts to create remote lab systems and online robots [8, 9, 17, 18, 34] focused on simple experiments and online learning. By contrast, our approach aims to provide researchers with an environment for remote development and experimentation on more capable robotics research platforms. Most previous work was not intended to foster shared research and experimentation. For this reason, these systems did not build upon shared robot middleware systems or a common software infrastructure. One consequence of this is that researchers could not easily extend previous efforts to build their own labs.

By contrast, the development of the PR2 Remote Lab centered around reusable, open source software. We use ROS, a robotic middleware system [33], in part due to its popularity: currently there are at least 117 ROS repositories and 3119 ROS packages.[1] Not only can we leverage this body of work for our remote lab, but other researchers can use our infrastructure to support their own research.

In this paper, we present technology that allows remote groups access to a shared PR2. A *PR2 Remote Laboratory* requires robust remote control, external sensing, remote safety systems, remote error recovery, static and dynamic environment modeling, visualization, and environment configuration. We have made our solutions to these problems publicly available to other researchers in the hope that these tools can be expanded and adapted to other robot systems and research problems. For example, these tools could be used to collect demonstrations from a large number of remote users.

Remote labs also provide the opportunity to carry out joint experiments and therefore foster collaboration between research labs. We describe the first use case for the remote lab, the facilitation of the 2011 Robot Learning from Demonstration Challenge. This challenge allowed teams to come together to showcase cutting-edge techniques. We describe how three teams participated in the challenge using the PR2 Remote Lab as their primary development and test platform. The participants found the remote lab setting conducive to getting their robot software development process up and running quickly, and were able to share data between teams and algorithms between common robot platforms.

2 Related Work

Remote laboratories offer the possibility to improve distance education, timetabling issues, accessibility and the cost of running laboratories. Li et al. [23] also cite the scarcity of equipment and supervisors as a reason for pursuing remote laboratories. A notable example of a successful remote laboratory installation is the Telerobot of the University of Western Australia (UWA) [37, 38]. This robot has been online for over ten years and has been used by many students as a remote laboratory to test kinematic models and control algorithms for stacking blocks. Other examples of remote laboratory systems can be found in [8, 9, 16]. Previous remote laboratories for online learning focused on experiments such as manipulating simple objects [37, 38], visual servoing [25], basic physics [9], and signal processing [16].

[1] http://www.ros.org/wiki/rosdoc_rosorg

Other online robots have mainly been built for entertainment purposes or as artistic displays. Goldberg et al. placed a robot in a garden and allowed users to view and interact with the robot over the web. Users were able to plant seeds, water, and monitor the garden [18]. Also, Taylor and Trevelyan provided internet access to their UWA robot, allowing the public to play with brightly colored blocks [37].

While the aforementioned work was focused on stationary robots, some work has been done on remote interfaces for mobile robots. Schulz et al. [34] examined the use of web interfaces to remotely operate mobile robots in public places. This work focused on letting remote users interact with humans within the robots' environment and did not examine the effect of the visualizations in a learning task. Burgard and Schulz [6] have explored handling delay in remote operation/teleoperation of mobile robots using predictive simulation for visualization. This work examined how robots could be controlled when there was a large delay in the visualization presented to the user.

Apart from remote laboratories and public online robots, much of the teleoperation work in robotics has traditionally been aimed at tasks where robots operate in environments that are hazardous or difficult for human users, such as robotic surgery [29], search and rescue [10], and outer space [2]. While this form of remote operation is a part of this paper, our goal is to provide a means of allowing researchers to use the robot for remote development.

Recently, work in the machine learning community examined using crowdsourcing approaches to train robots. Chernova et al. [12] examined using a multi-player video game where users collaborate to provide user demonstrations. Crick et al. [14] allowed a large number of users to demonstrate policies on an actual robot within a maze environment using one of two interfaces. The remote lab described in this paper could be used for such experiments, as well as experiments that do not require learning or human user studies.

3 Web-Based Laboratories for Learning from Demonstration

Web-based robotic laboratories can help to advance the state of the art in the LfD field in several ways. Such technologies enable researchers to create a variety of different interactions quickly, efficiently, and in platform-agnostic fashion. Another goal of our approach was to create a system that could be used to recruit subjects at internet scale, enabling the creation of huge demonstration datasets which could scale from single tasks to large numbers of skills.

In addition, the ability to create remote labs enables comparable research results. LfD does not have a set of standard datasets since the types of interaction can vary greatly. However, by using a common remote lab infrastructure LfD researchers can test their learned controllers in a set of standard domains. Additionally, in cases where interactions are similar enough, data can be easily shared.

While remote labs will benefit LfD research, there are limitations. For example, researchers who hope to examine how social interaction plays a role in LfD or researchers who examine LfD in situations where the robot watches humans perform a task will have a difficult time employing remote labs in their research. Even so, a significant portion of LfD research can benefit from such technology. We derive the following requirements for a web based remote laboratory:

1. **Connectivity:** Parts of the system must be able to communicate with each other. Hardware interfaces need to communicate with processing nodes, the remote user needs to communicate with the robot, etc. This communication can be asynchronous, as when changes in the robot's state need to be propagated to other parts of the system, or synchronous, when a client makes a request and waits for a reply before continuing. The communication system must be able to handle a variety of different transport modalities including high-bandwidth channels, such as camera streams, and rapidly-changing data, such as state updates. The protocol should be flexible enough to include new systems and functionality without any change. Clients may have slow connections and not be able to receive all data produced by a system. The system must be able to adapt to the available bandwidth of a client and send appropriate amounts of data.
2. **Web Compliance:** The aforementioned communication must work within a web environment, using HTTP (over TCP) as the underlying networking protocol. Direct TCP communication may be more efficient in some situations; however the client may be behind a firewall which will block any communication other than HTTP. A remote laboratory client should load inside a web browser without requiring additional plugins or software packages. Adhering to the HTML standards recommended by W3C[2] and WHATWG[3] should ensure interoperability with different browsers and different operating systems.
3. **Synchronization:** Synchronization is an important requirement for any distributed system. State changes in one part of the system need to be distributed to other parts of the system. Systems using asynchronous communication over web connections also require synchronization within the client. Generally, a variable delay is caused by data traveling over the web connection. While this delay

[2]http://www.w3.org
[3]http://www.whatwg.org

cannot necessarily be controlled, it can at least be measured. This allows synchronization of data from different sources.
4. **Access and Resource Control:** The remote lab's web connectivity inherently exposes the system to a large number of web users. Parts of the system might only be accessible to certain users, administrators for example. Even if a user is allowed to access the remote lab hardware, control and ownership must be shared with other users. This leads to two requirements: unique authentication of users and a system to organize and schedule the control over the remote laboratory resource.
5. **User Interface:** An important step in designing an interface for a remote laboratory is to consider who will actually be using it. The technology described in this paper can be used by a variety of different users with different needs. For the purpose of this paper, we focus primarily on LfD researchers and LfD demonstrators. Remote laboratories allow researchers remote access to the robot in order to directly participate in state of the art LfD research. Since both the physical hardware and the middleware are shared among different research groups, remote labs should enable experiments that build upon and compare against results on the same platform and in the same environment for common tasks. The interface must include enough functionality to run a variety of experiments and the ability to modify existing and add new components. Also additional functions to aid in testing and debugging may be necessary to allow efficient development. LfD demonstrators may not have significant experience with robots, and therefore require simple and intuitive interfaces that allow them to control the robot to perform a desired task.

The larger goal of this technology is to broaden participation in robotics to robotics researchers, students, and the general public. Researchers can create labs for shared experimentation as well as enabling them to provide access to new research platforms earlier in the development process. They can also carry out joint experiments and foster collaboration between research labs. Remote laboratories could be used to improve distance education, timetabling issues, accessibility, and cost of running laboratories. Members of the general public who are interested in experimenting with robots on a very simple level should also be able to use such laboratories to better understand the robotics field.

4 Tools for Remote Robotic Laboratories

This section describes the technology used to create the web-based remote laboratories.

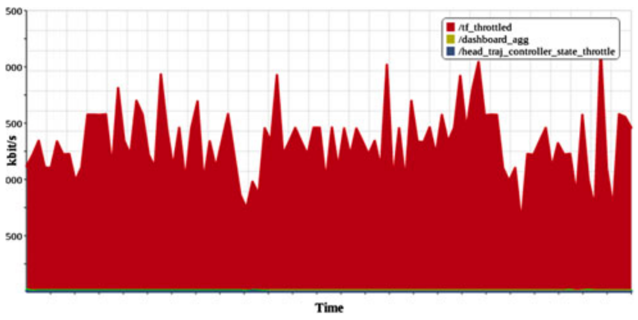
4.1 Rosbridge

ROS provides a structured communications layer above the host operating system and is organized as a peer-to-peer network of *nodes* that are processing data together. The nodes communicate asynchronously by passing messages called *topics* based on publish/subscribe semantics. A synchronous communication model is realized by *services*. ROS addresses the connectivity and synchronization requirements by providing a flexible communication layer. However, it was designed to function in a homogenous, tightly-integrated environment with extremely fast hardware connections, not for slow and unreliable connections over the web. We developed Rosbridge [13], which exposes the capabilities of ROS, such as a robot's data streams and controllers, through web sockets accessible anywhere over the internet, as well as providing security mechanisms and runtime tools for interacting with and maintaining a robot system.

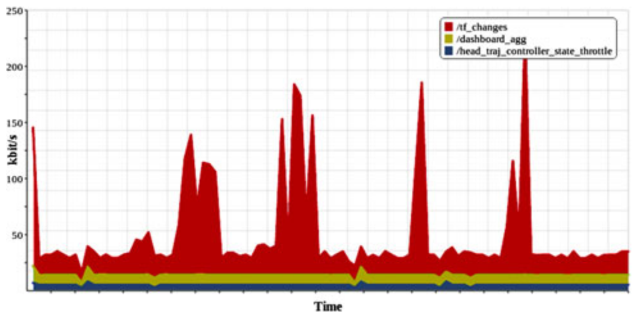
Rosbridge includes several helpful features to assist robot application developers and researchers in creating robot controllers and interfaces that are executed over the network. It provides dynamic access to ROS topics, services and associated types and objects, as well as runtime control of nodes and parameters in the ROS environment. Rosbridge also provides facilities for access control and data logging.

As a Rosbridge client can be any internet host, the exact network characteristics of the connection are not possible to predict in advance. However, tests of a representative selection of hosts can establish likely network characteristics. Limited bandwidth and variable end-to-end delays are a major concern for web connections. We measured the data rate (number of bits per second) being transported through Rosbridge. Note that this does not include video streams since a separate channel is used for this data. Figure 2(a) shows the data rates of the three largest topics sent to the client at the rate of production. The average data rate is about 580 kbit/s dominated by the /tf topic (530 kbit/s). For the PR2, the /tf topic updates the robot's state at 100 Hz. Since most state variables change infrequently, a simple optimization of this data channel consists of transmitting only partial state updates of changing state variables. Figure 2(b) presents the data rates after applying this optimization. The result is a significantly reduced data rate (70 kbit/s) with spikes at times when the robot's arms move. A second optimization performed in this experiment is an active throttling of topics.

The end-to-end delay is estimated by measuring the round trip time for a small topic in ROS sent from the client to a node running on the robot. This is a more accurate measure than network round trip times (ping) since it includes the overhead involved in rosjs (Sect. 4.2.2), Rosbridge, and ROS for processing the messages. For a LAN connection



(a) Data rate for the three largest topics sent at production rate.



(b) Data rate for the three largest topics after optimizing. Here, the /tf topic is transmitted only as partial state update of changing state variables and throttled to 10 Hz. Spikes in the data correspond to joint motion events where multiple states are updated in a short period of time.

Fig. 2 Comparison of data rates before and after optimization

this round trip is about 30 ms; a transcontinental connection increases this tenfold.

4.2 Creating Web Interfaces

Remote user interfaces can enable LfD researchers to conduct and observe experimental results on platforms not available at their institutions. Additionally these tools can be used to perform learning from demonstration experiments on a large scale, gathered from users on the web. Given these constraints, a web-based interface is a natural fit. Tools such as HTML5 and Javascript allow developers to create sophisticated interfaces. In this section, we describe the tools that can be used to visualize the robot's state and sensor data in the browser. An example of an interface created using these tools is shown in Fig. 3.

4.2.1 Video Visualization

Video is a natural and intuitive means of interacting with, providing demonstrations to, and checking on the progress of a robot. MJPEG, or motion JPEG, is a video stream format in which each frame of the stream is separately compressed as an JPEG image. We created an mjpeg server[4] that subscribes to requested image topics in ROS and publishes those topics as MJPEG streams via HTTP to a web browser. While Rosbridge is capable of streaming video, as it is just another message type from ROS, the web browser is optimized to efficiently download images in binary format. Attempting to send high-resolution video streams over the internet caused difficulty for previous remote lab efforts; our video streamer handles them as efficiently as possible.

Users specify their requests to the mjpeg server in the form of a URL. The URL includes the desired topic and specifies video parameters (such as quality and size) to accommodate particular connection speeds and interface designs. This allows the webstream or image to be embedded in a website through the <img> tag.

4.2.2 rosjs

While video streams are useful viewing tools, they are not capable of displaying the wide variety of data available on a robot like the PR2. Additionally, there are many situations in which remote users are likely to want to interact with the robot either as part of data collection or as a normal part of running experiments. We created rosjs [30], a Javascript binding for ROS, to take full advantage of this technology, enabling applications developed in the web browser to communicate with a robot running ROS and Rosbridge. Thus, the web interface has access to the entire ROS middleware ecosystem, exposed in a familiar, platform-neutral environment widely used by even nontechnical users. In addition to providing a means of communicating with ROS topics and services over Rosbridge, rosjs also provides a large collection of visualization tools implemented in Javascript. These tools allow for rapid creation of user interfaces.

4.2.3 3D Web Visualization

A large portion of rosjs is dedicated to enabling 3D visualization of the robot and its environment. 3D interfaces are one of the most useful tools in visualization and development of robot systems, allowing users to examine data from multiple angles and understand the spatial relationships between the robot and its environment. An additional benefit of such interfaces is that the messages required to create them are relatively low bandwidth, especially when compared to live video streams. The remote lab interface described in this paper provides this functionality. Using robot models, poses, maps, and custom visualization markers, the visualization interface can display views of various types of robot data.

The underlying technology for visualizing 3D data on the web is WebGL, a 3D graphics API implemented in a web browser without the use of plug-ins. Similar to OpenGL, WebGL provides a programmatic interface for 3D graphics using the OpenGL ES 2.0 standard. Application developers may use the WebGL library of their choice for visualization.

[4]http://www.ros.org/wiki/mjpeg_server

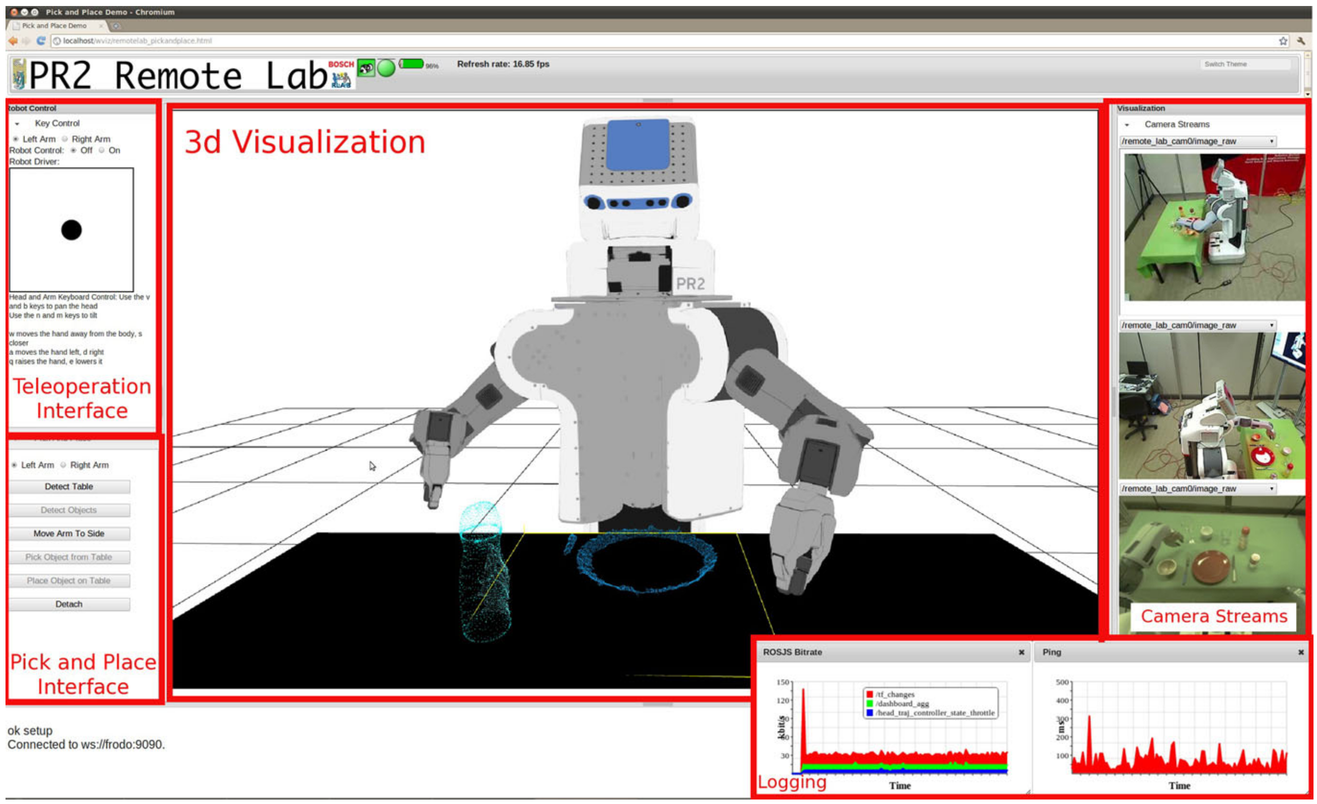


Fig. 3 3D visualization based on WebGL is used to show the robot's state and sensor data. A user can interact with the PR2 by using one of the predefined interfaces or by writing small Javascript programs in order to send control commands

Our 3D visualizer provides an interface to WebGL based on world objects and other high-level classes. It also provides a scene graph, which is an object-oriented representation of the 3D world. Such a representation is especially suitable for robotic development, since data is represented in various interdependent frames. It also simplifies the extension to new data types since scene nodes can be implemented for any data type and inserted into the graph, as long as they adhere to a common interface.

4.2.4 *Widgets*

Much of the infrastructure provided for rosjs is provided in the form of widgets, modular stand-alone applications that can be easily added to the interface. rosjs provides a widget manager that handles registering widgets and their interfaces. Programmers need only specify a class type when creating a section of the HTML document. Most remote lab widgets are robot-independent, though some are specific to the PR2. Many of the widgets are useful for general LfD experiments and remote experimentation.

Data Logging The topic logger provides both a client-side Javascript user interface and a server. In order to record data from experiments, a user can select the topics they wish to record and press buttons to start and stop logging. The desired topics are sent to the server-side node which records the data. Once the user requests logging be stopped, the remote lab makes the data available on a web server for download.

Robot Monitoring Several widgets help monitor the current status of the robot from the web interface. These include robot-independent widgets that monitor the data rate, the frame rate of the display on the client end, and a widget that allows users to list the topics available on the robot. Additionally there are several PR2-specific widgets that allow users to monitor battery levels and the status of the controllers. A run-stop widget, which disables the robot motors, allows users to halt the robot's motion in case of an emergency or unexpected robot behavior.

4.3 User Interaction

In order for the remote lab to be useful, users must be able to interact with the robot on a variety of different levels. Different research projects may require different interfaces. Additionally, frequent tasks may be automated to reduce the burden upon the remote operator. In this section, we describe four types of user interaction we have currently implemented for the remote lab.

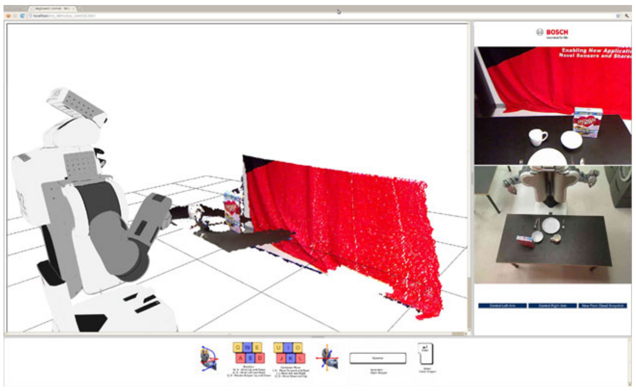


Fig. 4 Example interface with point cloud visualization and keyboard control

4.3.1 Keyboard Teleoperation Interface

The keyboard teleoperation interface allows the user to use the keyboard to control the head's pan and tilt mechanisms, and to control the motion of the arms. Additionally, it allows users to control the base using a virtual joystick controller. In order to use keyboard control for manipulation, the user selects the desired arm through the interface and turns on control. Key commands issued by the user are translated into a new end position of the hand and sent to a Cartesian controller. An example of an interface with instructions for key control is shown in Fig. 4.

4.3.2 Mouse Teleoperation Interface

While accurate, keyboard teleoperation can be tedious, non-intuitive and frustrating for users. As part of the development of the remote lab project, we created a puppeteer interface that allowed users to select the arm of the robot and move it to specified locations. With the introduction of interactive markers,[5] a generic 3D interaction tool recently introduced into ROS, we extended the interface to allow any robot with an interactive marker interface to be controlled through the web. Figure 5 displays a web-based interactive interface for the PR2, along with control markers for manipulating the head, base and right arm.

4.3.3 Prescripted Interface

The third type of interaction allows developers to provide interfaces for frequently used actions. For example, the remote lab includes a pick-and-place interface that allows users to select objects. We extended the pick and place demo[6] provided in ROS. The user interacts with the robot through a series of buttons to command the robot to perform a set of actions, including detecting the table, detecting objects, moving the arms to the front or side, as well as many others. It is easy to imagine similar interfaces for other applications, including navigation where a user could click on points on a map and then command the robot to navigate, or other manipulation tasks within a kitchen where selecting a door would command the robot to perform a door-opening task. These interfaces are simple for users, but they perform only as well as the autonomous algorithms they employ.

4.3.4 Scripting Interface

For advanced users a simple scripting interface consisting of a text box and submit button is available. Users enter and execute Javascript code, including new components and classes, without the need of a server-side script. This provides advanced users with the ability to write complex robot controllers rather than issuing a set of "commands" as in [24]. This enables users to test new code without changing the entire interface, as well as include Javascript code from arbitrary libraries on the fly.

4.3.5 Discussion of Interaction Types

Each type of interaction has its own pros and cons and the selection of an interaction mechanism depends on the intended end user and the context in which they will be operating. The keyboard interface and mouse interfaces can be used by any type of user and provide low level control. Since users are directly controlling the robot they can achieve a wide variety of tasks. The keyboard interface is natural for navigation but it can be difficult to control high degree of freedom manipulators due to the large number of keys required to move the robot. The mouse interface provides more natural control but it requires the ability to reason and control the robot in three-dimensional space, which may be unintuitive to some users. The keyboard interface is lightweight and generally responsive even on slower network connections. The mouse interface requires rendering the robot's pose and environment in 3D, which may impose a performance penalty.

Prescripted interfaces are the easiest of the interfaces to use since they require button clicks. The interfaces are generally do not send large amounts of information. These interfaces are typically used for higher level control and depend upon a great deal of robot autonomy. Thus, it can be difficult to recover from unforeseen situations.

Scripting interfaces require programming experience and are not appropriate for the general public. They offer the ability to develop and test new robot functionality, but are generally too cumbersome for one-time low-level control movements.

[5] http://www.ros.org/wiki/interactive_markers

[6] http://www.ros.org/wiki/pr2_pick_and_place_demos

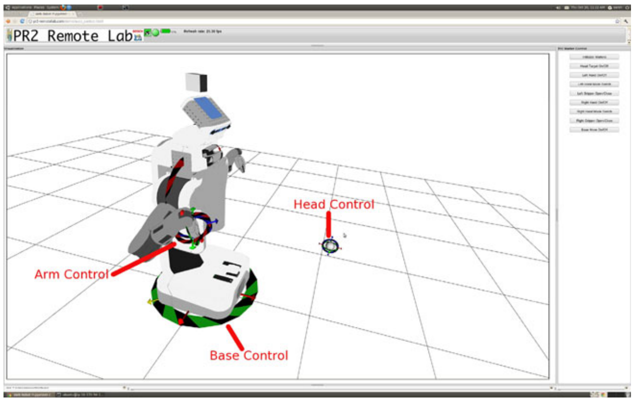


Fig. 5 Web interface with interactive marker control. Users control the robot through the mouse by moving the various controls

5 Shared Experimentation

This section describes the implementation of the PR2 Remote lab and how it was used in the AAAI LfD Challenge.

5.1 PR2 Remote Laboratory

The robot platform used in the remote laboratory described in this paper is the PR2 personal robot [39], a two-armed robot with an omnidirectional base. Equipped with two 7-degrees-of-freedom compliant arms, the PR2 is designed for compliant interaction with the environment. The compliance is an important safety feature for the humans and objects that may be present in the robot's environment, as well as for the robot itself when being used by a remote user. The PR2 has an extensive sensor suite allowing remote users to perform a variety of mobile manipulation tasks.

The remote lab environment shown in Fig. 6 is equipped with additional sensors. Four cameras observe the robot's work-space and allow the remote user to see the PR2 and objects in its environment. In addition to the 2D cameras, a depth camera is used to acquire a 3D view of the remote lab. A PhaseSpace optical motion capture system is used to obtain ground truth pose information for the robot and objects. The PhaseSpace system tracks active LED markers and provides highly accurate, real time motion data which is necessary to update the remote lab's state.

The software is structured on both the client and server side. An overview is presented in Fig. 7. The client side consists of a browser-based user interface implemented in HTML and Javascript. On the server side software is roughly divided into three main layers: hardware interfaces, ROS middleware, and web services. The hardware interface layer comprises a number of software modules concerned with receiving and timestamping the sensor data, as well as interfaces to the robot's actuators.

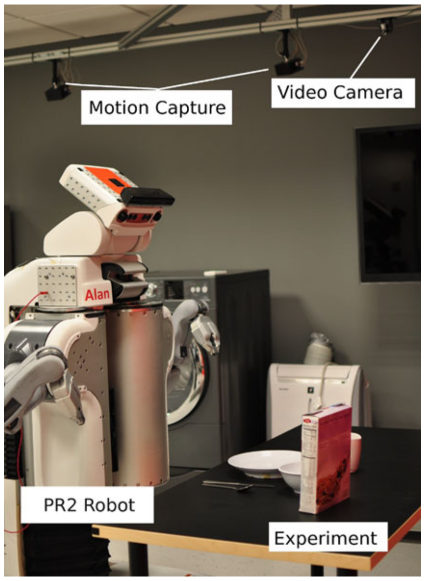


Fig. 6 The PR2 Remote Laboratory the workspace is observed by multiple cameras and a motion capturing system. The remote users see and control the robot through a web-based interface

5.2 AAAI LfD Challenge

The 2011 Robot Learning from Demonstration Challenge was held in conjunction with the AAAI Conference and Robotics Exhibition. The challenge is an opportunity for members of the LfD community to showcase their recent results through presentations and live demos. In previous years, participants demonstrated their research techniques on a set of unrelated tasks. In order to better enable comparisons and cooperation, the 2011 LfD Challenge focused on a single challenge task centered on food preparation. The task required a combination of both low-level and high level-skills, showcasing a variety of LfD techniques.

5.2.1 Shared Task: Breakfast for Two

The robot's goal was to set the table for breakfast for two people. The first person prefers cereal and tea. The second person prefers toast, coffee, and juice. The robot must set the table with objects that will help speed up the breakfast process for the users.

The task involved a standard set of objects and a standard setup. During the task the robot would navigate between four pre-specified positions. At the beginning of the task objects are located in specified initial positions reachable on a preparation table. The robot selects these objects and then moves them to the appropriate location on the nearby dining table. Specifications were provided for the positions of the tables and the possible robot positions. The objects and tables were widely available standard items and exact specifications were provided so that teams with a PR2 that wished

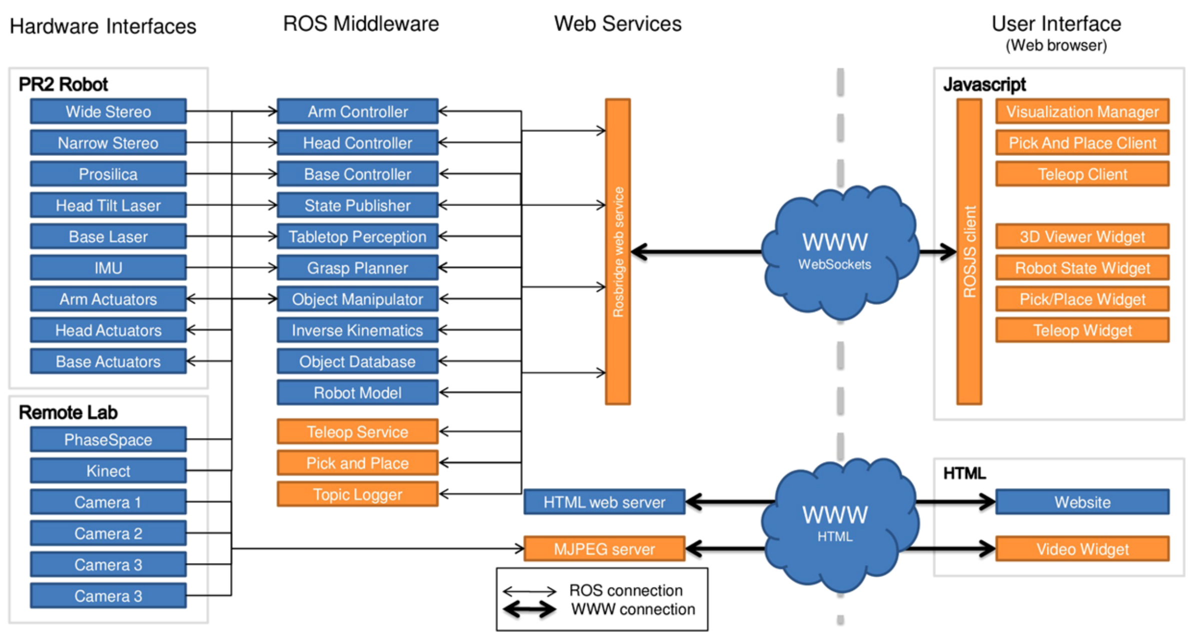


Fig. 7 Flowchart of the remote lab's software system. The software is divided into three layers on the server side (hardware interfaces, ROS middleware and web services) as well as a client-side user interface. *Shaded* software nodes were developed in conjunction with the remote lab described in this paper

to experiment locally could recreate the experimental set up in their own lab.

In order to highlight different LfD techniques, there were three classes of objects. The first class were objects that were part of an object database provided with ROS and had pre-computed grasps available. The second class of objects were part of the object database but had no available grasps. The third class of objects were not part of the object database and had no default recognition or grasps.

5.2.2 *Initial Preparations and Schedule*

Once the challenge task was finalized, we created an initial solution to the task to demonstrate that the robot was physically capable of completing the task. In order to perform the task, we used a simple scripted high-level algorithm. We used open-loop control to pick up and place the objects. The pose of a pre-grasp and grasp position were recorded for each object. When the high-level plan selects the next object the arm moves to the predefined positions and then closes the gripper. With careful placement of the objects we were able to complete the majority of the challenge task. However, this solution is not robust and there is significant room for improvement.

In addition to demonstrating feasibility, the code provided a set of interfaces for the participants. LfD algorithms that used the same interface could easily be run as part of the full system. This allowed algorithms to be evaluated within the context of a complete task, something often neglected in current research efforts. Additionally, researchers gained the benefits of an integrated system. Algorithms could be combined not only with provided scripted code, but also with other LfD algorithms developed during the challenge. For example, a researcher focused on low-level LfD can benefit from improvements in a high-level planning system without investing additional time and effort (and vice versa).

To help teams test code before performing experiments, we provided a simulation of the challenge. The simulator contained a subset of the provided objects and two tables. Teams could run the provided code as well as test their own code within the simulator. The simulator is not currently capable of being used as an exclusive development and testing platform, but it provided an initial testing ground to discover unexpected behavior.

The initial solution to the task was completed in May 2011. Teams began using the remote lab at the beginning of June. For two of these teams the remote lab provided exclusive access to the PR2. The LfD Challenge was held in San Francisco on August 7–9. Teams actively used the remote lab for about two months to develop and test their algorithms.

5.2.3 *Participants*

Three teams (from École Polytechnique Fédérale de Lausanne (EPFL), Massachusetts Institute of Technology (MIT), and Worcester Polytechnic Institute (WPI)) used the PR2 Remote Lab as their primary development platform and testbed in order to participate in the LfD Challenge. The teams varied in location, academic experience and experience with the technology needed to participate in the challenge task.

The approach used by the team from EPFL focused on learning low-level motion primitives, such as reaching for a glass or pouring milk, from demonstration. The learning method, called Stable Estimator of Dynamical Systems (SEDS) [21], models motions as a non-linear autonomous dynamical system and defines sufficient conditions to ensure global asymptotic stability at the target. The participants had not used ROS before but had a completed algorithmic approach that was implemented and tested on another robot. Their task was to port the existing proven technology to a new platform and to demonstrate the generality of the approach.

The LfD approach used by MIT also focused on learning of low-level motion primitives from demonstration. Specifically their approach employed probabilistic flow tubes to infer the desired state region at each time step from the data provided by the demonstrations [15]. The participants had no ROS experience, little experience with Linux, and had an initial system implemented in Matlab. During the course of the challenge this team's task involved improving their proposed algorithm for LfD as well as interfacing their Matlab system with ROS. Additionally, this team had occasional access to the PR2 located at MIT; however, their main testbed was the PR2 Remote Lab.

WPI used Behavior Networks [27] to learn the task as a high-level plan consisting of subgoals. In this approach, the agent continuously compares the current state of the world to check if it has achieved one of the pre-defined subgoals during the demonstration (e.g. holding the orange juice bottle in one hand, while standing in front of the table). The agent then creates a history of the subgoals reached. When asked, the agent executes the actions necessary to reach the subgoals, one by one, eventually accomplishing the main goal of the task. This team had used ROS extensively in previous research projects and had helped develop a widget that displayed currently available topics for the remote lab. This team spent much of the preparation time developing new algorithms.

5.2.4 *Development Using the Remote Lab*

In addition to web interface access, teams were provided with accounts on the robot and remote login access. Individual weekly meetings were set up for each team with Bosch RTC personnel. These meetings were used to discuss progress, troubleshoot problems, and discuss requirements.

We made use of a Google calendar to manage the robot's time. This calendar was shared not only by the teams but also by the research associates and summer interns at Bosch RTC. Timesharing of an expensive robot platform was one of the larger goals of the project. Many of these expensive platforms are often idle. The goal was that researchers would be able to use the remote lab without supervision especially during times when it was unlikely to be used. Teams used approximately 33 % of the scheduled robot hours between June 12 and August 6. 62 % of that time was performed outside of the hours of 7 am–7 pm.

An important aspect of the challenge was providing demonstrations. The team from WPI collected data using calls to interfaces that moved the arms to specified locations via Cartesian control and the base to one of the fixed positions. However, gathering demonstration data for teams learning motion primitives proved more challenging. While the keyboard interface allowed the researchers to control the robot's arms, it was difficult for researcher to perform smooth natural motions. To overcome this difficulty, a participant from EPFL visited Bosch RTC in conjunction with nearby conferences. Researchers at Bosch RTC also provided a small number of demonstrations, around 5–10 examples, for a set of desired tasks. Due to the shared infrastructure both teams learning low-level motion primitives were able to share data.

5.2.5 *Observations and Lessons Learned*

In this section, we report observations and survey responses regarding the value of such labs as a research tool for remote development and LfD. Specifically, the teams were asked how comfortable the users were with the technology before and after they started using the remote lab, how long it took until they were comfortable using the lab, how long it took until they had preliminary demonstrations working, how much time they spent using the remote lab, the simulator, and integrating with remote lab personnel, which monitoring, control interfaces, and ROS features they used, how they thought the remote lab could be improved, and how the remote lab compares to hardware platforms used in the past. These interviews were conducted seven months after the event, and the data below reflect the teams' self-reports from memory.

Working with the remote lab was easiest for the for the group who had prior experience developing interfaces for the remote lab. Generally, working with the remote lab was easiest for groups who had a fully-developed codebase. However, these groups had less desire to invest time into learning new interfaces. The remote lab was a tool for the challenge rather than a potential long term development platform. Thus interfaces needed to be intuitive and easy to use. On average the time for teams to become comfortable using the remote lab was about 6 and a half days. The team with the most remote lab experience took 3 days while the team with the least ROS experience took 10 days. Additionally, two teams reported having preliminary demonstrations within 7 days and the other team reported 14 days.

Two remote lab teams felt that the user interface could be improved. The set of interaction tools available to remote lab participants did allow teams to interact with the robot, such as by using control interfaces to move objects back to desired locations after a preliminary controller failed. However, designing appropriate, intuitive interfaces for learning tasks will form a substantial research agenda in human-robot interaction, one which benefits greatly from a remote lab platform for evaluation. For example, we implemented mouse-based interfaces after the teams started using the remote lab, based on their suggestions.

The team from MIT had limited access to the PR2 at their institution. Even so, using the remote lab provided them with several benefits. It allowed them to leverage the shared infrastructure and manpower discussed earlier in this paper. Additionally, it provided the ability to benchmark a task in a set, well defined environment. Algorithms could be compared on the same setup that other challenge teams were using making future results more comparable. They also were able to easily transfer interfacing code from the remote lab to their local PR2, demonstrating the versatility of this infrastructure.

Integrating the physical remote lab system with information about web usage would be helpful. Physical cues would allow users within the remote lab space know that a remote user was in control of the robot. This would improve safety since local users would be more aware that the robot might move at any moment. It would also enhance resource sharing: local users could see if a remote user had logged off slightly early from their scheduled time. Additionally, with the proper technology remote users could recruit the help of local researchers. For example, if a remote user was performing a manipulation task and an object had fallen to a spot where they were unable to reach it they could ask someone to help them replace the object in the experimental space. This means that any local user could help, rather than the small subset of the researchers with whom the remote users were accustomed to working.

Two remote lab teams felt that the sensing capabilities of the remote lab could be improved. One possibility for extra sensors would be to create tools to enable instrumentation of the remote lab with audio capabilities. This would enable research using voice commands for LfD. Audio feedback would also alert users to unexpected collisions or accidents that were out of the view of cameras.

All teams reported a higher comfort level after the challenge. When compared to previous systems all of the teams felt that the remote lab was "much better at enabling comparable experiments." Teams also felt that the remote lab was better at enabling first-time users to get something working and enabling code sharing. Generally the teams felt the remote lab was about the same as other platforms in terms of providing feedback and sensing information. They did not feel the remote lab was better at controlling mobility or managing manipulators. As researchers continue to build and use remote lab systems, feedback such as this should help in the improvement of interface design, much as we were able to incorporate suggestions during the LfD Challenge.

6 Conclusion and Future Work

This paper presented technology for the creating of remote labs, and documented the specific development of the PR2 Remote Lab. This technology is available as open source ROS packages in the bosch-ros-pkg[7] and the brown-ros-pkg.[8] The goal of this project is to create infrastructure to enable more teams and groups of researchers access to state-of-the-art robotics platforms as well as to enable researchers to expose their research to the public at large through the internet. We described the first test case of the PR2 Remote Lab for shared experimental development in which teams used the system to test and develop LfD algorithms for the 2011 Robot Learning from Demonstration Challenge.

This project highlights several promising areas of future work. The importance of natural intuitive user interfaces was highlighted during our work on the remote lab. Creating interfaces that can control high-degree-of-freedom arms while also being intuitive and usable with commodity hardware, for example a mouse and keyboard, is an important but challenging task. Some recent efforts have begun to examine human-in-the-loop, or shared autonomy interfaces [26, 32]. However, evaluation of the usability of existing interfaces as well as creation of better interfaces is a worthwhile effort. Once such interfaces are possible, we can achieve truly large scale learning from demonstration for manipulation tasks. Given the promising results in the vision [20] and the natural language processing communities [36], it is likely that applying similar large data-driven approaches to real-world policy learning will bear fruit.

Additionally, we hope to extend access to the remote lab to a larger audience. Some technology improvements will be required for a larger number of users. Security mechanisms must be designed to handle large number of users all through a web interface. For example, only the person scheduled to control the robot would have access to some topics while other users may still be allowed to view the data. Scheduling could also be more integrated into the remote interface. While these capabilities have been implemented in Rosbridge more testing and full integration is required. The current means of selecting which widgets and

[7] http://bosch-ros-pkg.sourceforge.net/

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

data types must be viewed is done through code. A more dynamic interface would allow users to select widgets and data they wish to use for an experiment on the fly.

References

1. Abbeel P, Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the 21st international conference on machine learning
2. Aldridge H, Bluethmann W, Ambrose R, Diftler M (2000) Control architecture for the robonaut space humanoid. In: Proceedings humanoids 2000, the 1st IEEE/RAS conference on humanoid robots
3. Argall BD, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Auton Syst 57:469–483
4. Atkeson CG, Schaal S (1997) Robot learning from demonstration. In: Fourteenth international conference on machine learning (ICML), Nashville, TN, pp 12–20
5. Billard A, Calinon S, Dillmann R, Schaal S (2008) Robot programming by demonstration. In: Handbook of robotics. Springer, Secaucus
6. Burgard W, Schulz D (2002) In: Robust visualization for online control of mobile robots. MIT Press, Cambridge, pp 241–258
7. Calinon S, Billard A (2007) Incremental learning of gestures by imitation in a humanoid robot. In: Second conference on human-robot interaction (HRI), Arlington, Virginia
8. Callaghan M, Harkin J, Prasad G, McGinnity T, Maguire L (2003) Integrated architecture for remote experimentation. In: IEEE international conference on systems, man and cybernetics, 2003, vol 5, pp 4822–4827
9. Casini M, Prattichizzo D, Vicino A (2004) The automatic control telelab: a web-based technology for distance learning. IEEE Control Syst Mag 24(3):36–44
10. Casper JL, Murphy RR (2002) Workflow study on human-robot interaction in USAR. In: Proceedings of the 2002 IEEE international conference on robotics & automation, pp 1997–2003
11. Chernova S, Veloso M (2007) Confidence-based policy learning from demonstration using gaussian mixture models. In: International conference on autonomous agents and multiagent systems (AAMAS)
12. Chernova S, Orkin J, Brazeal C (2010) Crowdsourcing HRI through online multi-player games. In: Proceedings of the AAAI 2010 fall symposium on dialog with robots
13. Crick C, Jay G, Osentoski S, Pitzer B, Jenkins OC (2011) Rosbridge: Ros for non-ros users. In: Proceedings of the 15th international symposium on robotics research (ISRR)
14. Crick C, Osentoski S, Jay G, Jenkins OC (2011) Human and robot perception in large-scale learning from demonstration. In: 6th ACM/IEEE international conference on human-robot interaction
15. Dong S, Williams B (2010) Motion learning in variable environments using probabilistic flow tubes. In: 2011 IEEE international conference on robotics and automation (ICRA), pp 1976–1981
16. Ferrero A, Salicone S, Bonora C, Parmigiani M (2003) Remlab: a java-based remote, didactic measurement laboratory. IEEE Trans Instrum Meas 52(3):710–715
17. Fu M, Yeo C, Lin Y, Wang F (2001) Waves: towards real time laboratory experiments in cyberspace. In: IEEE international conference on systems, man, and cybernetics, vol 5, pp 3470–3474
18. Goldberg K, Dreyfus H, Goldman A, Grau O, Gržinić M, Hannaford B, Idinopulos M, Jay M, Kac E, Kusahara M (2000) The robot in the garden: telerobotics and telepistemology in the age of the Internet. MIT Press, Cambridge
19. Grollman DH, Jenkins OC (2008) Sparse incremental learning for interactive robot control policy estimation. In: International conference on robotics and automation (ICRA)
20. Hayes J, Efros A (2008) Scene completion using millions of photographs. Commun ACM 51(10):87–94
21. Khansari-Zadeh SM, Billard A (2011) Learning stable nonlinear dynamical systems with gaussian mixture models. IEEE Trans Robot 27(5):943–957
22. Konidaris GD, Kuindersma SR, Barto AG, Grupen RA (2010) Constructing skill trees for reinforcement learning agents from demonstration trajectories. In: Advances in neural information processing systems 23 (NIPS 2010)
23. Li L, Wang FY, Lai G, Wu F (2003) Online autonomous guidance system for remote experiments in control engineering. In: IEEE international conference on systems, man and cybernetics, vol 3, pp 2444–2449
24. Marín R, Sanz PJ, Del Pobil AP (2003) The uji online robot: an education and training experience. Auton Robots 15:283–297
25. Marín R, Wirz R, Sanz P, Fernández J (2007) Internet-based telelaboratory: remote experiments using the snrp distributed network architecture. In: Advances in telerobotics. Springer tracts in advanced robotics, vol 31. Springer, Berlin, pp 429–444
26. Meeussen W, Marder-Eppstein E, Watts K, Gerkey BP (2011) Long term autonomy in office environments. In: ICRA 2011 workshop on long-term autonomy. IEEE Press, Shanghai
27. Nicolescu M, Matarić MJ (2002) A hierarchical architecture for behavior-based robots. In: 1st international joint conference on autonomous agents and multi-agent systems (AAMAS), pp 227–233
28. Nicolescu M, Matarić MJ (2003) Natural methods for robot task learning: instructive demonstration, generalization and practice. In: 2nd international joint conference on autonomous agents and multi-agent systems (AAMAS), pp 241–248
29. Okamura A (2004) Methods for haptic feedback in teleoperated robot-assisted surgery. In: Industrial robotics, pp 499–508
30. Osentoski S, Jay G, Crick C, Pitzer B, DuHadway C, Jenkins OC (2011) Robots as web services: reproducible experimentation and application development using rosjs. In: Proceedings of the 2011 IEEE international conference on robotics & automation
31. Pastor P, Kalakrishnan M, Chitta S, Theodorou E, Schaal S (2011) Skill learning and task outcome prediction for manipulation. In: Proceedings of the 2011 IEEE international conference on robotics & automation (ICML 2011)
32. Pitzer B, Styer M, Bersch C, DuHadway C, Becker J (2011) Towards perceptual shared autonomy for robotic mobile manipulation. In: 2011 IEEE international conference on robotics and automation (ICRA 2011)
33. Quigley M, Conley K, Gerkey BP, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) Ros: an open-source robot operating system. In: ICRA workshop on open source software
34. Schulz D, Burgard W, Fox D, Thrun S, Cremers AB (2000) Web interfaces for mobile robots in public places. IEEE Robot Autom Mag 7:48–56
35. Smart WD, Kaelbling LP (2002) Effective reinforcement learning for mobile robots. In: 2002 IEEE international conference on robotics and automation (ICRA), pp 3404–3410
36. Talukdar PP, Jacob M, Mehmood MS, Crammer K, Ives ZG, Pereira F, Guha S (2008) Learning to create data-integrating queries. In: Proceedings of the twenty-second international conference on computational linguistics, pp 737–744
37. Taylor K, Trevelyan J (1995) A telerobot on the world wide web. In: National conference of the Australian robot association
38. Trevelyan J (2004) Lessons learned from 10 years experience with remote laboratories. In: International conference on engineering education and research progress "Through partnership"
39. Willow Garage: Personal Robot 2 (PR2). http://www.willowgarage.com

40. Ziebart BD, Maas A, Bagnell JD, Dey AK (2008) Maximum entropy inverse reinforcement learning. In: Proceedings of the twenty-third AAAI conference on artificial intelligence

Sarah Osentoski is a Research Engineer at the Bosch Research and Technology Center. She received her Ph.D. in Computer Science from the University of Massachusetts Amherst and was a postdoctoral researcher at Brown University.

Benjamin Pitzer is a Senior Research Engineer at the Bosch Research and Technology Center and received his Ph.D. from the Institute for Measurement and Control of the Karlsruhe Institute of Technology.

Christopher Crick received his Ph.D. in computer science from the Social Robotics Laboratory of Yale University in 2009, and is now a postdoctoral research associate at Brown University.

Graylin Jay is an independent developer and a researcher at Brown University.

Shuonan Dong is a graduate researcher in the Computer Science and Artificial Intelligence Lab at MIT.

Daniel Grollman is a postdoctoral fellow at the LASA Laboratory at EPFL. He received his B.S. (2003) in Electrical Engineering and Computer Science from Yale University, and his Sc.M. (2005) and Ph.D. (2010) in Computer Science from Brown University.

Halit Bener Suay is a Ph.D. student and a Research Assistant at Worcester Polytechnic Institute, Robotics Engineering. In 2011, he joined Robot Autonomy and Interaction Lab. He has a B.Sc. and a M.Sc. degree in Aeronautics Engineering from Istanbul Technical University and The University of Tokyo.

Odest Chadwicke Jenkins is Associate Professor of Computer Science at Brown University and head of the Robotics, Learning and Autonomy at Brown (RLAB) research group.