Bringing Human-Robot Interaction Studies Online via the Robot Management System

by

Russell Toris - rctoris@wpi.edu

A Thesis

Submitted to the Faculty of the

of the

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

October 2013

Approved

Professor Sonia Chernova, Thesis Advisor

Professor Candy Sidner, Thesis Reader

Professor Craig Wills, CS Department Head

Abstract

Human-Robot Interaction (HRI) is a rapidly expanding field of study that focuses on allowing non-roboticist users to naturally and effectively interact with robots. The importance of conducting extensive user studies has become a fundamental component of HRI research; however, due to the nature of robotics research, such studies often become expensive, time consuming, and limited to constrained demographics. This work presents the Robot Management System, a novel framework for bringing robotic experiments to the web. A detailed description of the open source system, an outline of new security measures, and a use case study of the RMS as a means of conducting user studies is presented. Using a series of navigation and manipulation tasks with a PR2 robot, three user study conditions are compared: users that are co-present with the robot, users that are recruited to the university lab but control the robot from a different room, and remote web-based users. The findings show little statistical differences between usability patterns across these groups, further supporting the use of web-based crowdsourcing techniques for certain types of HRI evaluations.

Acknowledgements

I would like to thank Prof. Odest Chadwicke Jenkins of Brown University for allowing the use of the PR2 robot throughout this study as well as Willow Garage, Inc. and Robert Bosch, LLC. for their development and support of online robotics tools and software. Furthermore, I would like to thank fellow graduate student David Kent for his help conduction the case study. Additionally, I would like to thank Prof. Craig Shue for his help with the methods used for new security procedures used in this work. This work was supported by the National Science Foundation award number 1149876, *CAREER: Towards Robots that Learn from Everyday People*, principal investigator Sonia Chernova, and ONR Grant N00014-08-1-0910 *PECASE: Tracking Human Movement Using Models of Physical Dynamics and Neurobiomechanics with Probabilistic Inference*, principal investigator Odest Chadwicke Jenkins.

Contents

List of Figures

1 Introduction

An overarching theme associated with HRI is the ability to provide users, in particular non-expert users, a way to naturally and effectively train, teach, and interact with robotic agents. Within HRI, convincing research is typically strongly supported by user studies to help prove the effectiveness of new techniques [20, 6, 18, 26, 23]. Integration of user studies into the design, development, and evaluation of new techniques has been shown to result in more usable methods [30], whereas development of algorithms in isolation risks biasing usability towards expert users.

Although user studies dealing with HRI are shown to be effective, they are subject to limitations. Issues such as human safety and robot durability often limit studies to the confines of research labs. Additionally, the high cost of robust, research-grade robots typically limit researchers to using a single robotic platform for a given study. As a result, conducting user studies requires bringing human subjects in one at a time, typically from areas within close proximity of the lab itself. Such studies take days to weeks to perform and are limited to relatively small numbers of participants representing only the demographics of the areas around the labs themselves. Furthermore, the typical research cycle progresses through the stages of method formulation, implementation, user study, and presentation of results, which integrates user studies only at the culmination of a project. This leaves little time for the integration of lessons learned back into the final product.

This work seeks to address several of the above limitations by introducing a web-based framework for HRI experimentation. The goal is to utilize aspects of crowdsourcing, in particular, the use of anonymous and diverse users from across multiple demographics. This work introduces the Robot Management System (RMS), a web-based framework designed to reduce the overhead of running user studies that involve control of the robot. The RMS provides a secure, customizable browser-based interface for robot control, integrated support for testing of multiple study conditions, and support for both simulated and physical environments. Through these capabilities, the RMS gives a large number of online users across the globe the ability participate in user studies by observing the actions of a robot through camera feeds and providing control commands through the keyboard and mouse.

It should be noted that remote teleoperation techniques will not be sufficient to conduct all types of user studies within HRI. In particular, techniques and studies that require physical interaction, sensing or observations of the user, and hard real-time requirements (i.e., methods that are not robust to delays where latency could be an issee) would not be appropriate for this system. Also note that results must be carefully analyzed to take into account the effects of (or lack of) situational awareness [13]. Nevertheless, there still is a large subset HRI studies which can take advantage of modern crowdsourcing techniques. While we cannot enumerate all types of appropriate studies, note that studies which deal with interface design, feedback, or testing particular types of Learning from Demonstration techniques are examples of types of studies that could be appropriate.

In this thesis work, we look at an initial trial of the RMS as a means of conducting user studies by asking users to complete a series of navigation and manipulation tasks with a PR2 robot. The design of this study was inspired by, although not identical to, the HRI work done by Leeper et al. [21]. In this work, metrics are compared across three user study conditions: users that are co-present with the robot (refereed to as the *co-present group*), users that are recruited to the university lab but control the robot from a different room (referred to as the *in-lab group*), and remote web-based users (referred to as the *remote group*). Findings show little statistical differences between different usability patterns across these groups, further supporting the use of web-based crowdsourcing techniques for particular types of HRI evaluation. I again emphasize that the presented framework can not replace face-to-face interactions, and therefore will have limited application in some areas of HRI research, such as proxemics or physical HRI. Instead, I envision the RMS contributing to research in areas such as interfaces for teleoperation, shared autonomy, and Learning from Demonstration by enabling evaluation at unprecedented scale through the web.

We begin by looking at past related work in the field of crowdsourcing and web-based robotics. We then look at the system and outline its important details. Following this, we look at new security measures that are introduced into ROS to better secure these remote environments. Next, we detail a user study and analyze its results to show insight on the validity of using such a system to conduct certain types of HRI research. We then take a closer look at the results from the user study to explore the observations gained from the study as a whole. Finally, we will wrap up with some final conclusions and present directions for future work.

2 Related Work

Crowdsourcing for HRI, or more broadly robotics, and web-based robotics are not new ideas. Over recent years the use of existing crowdsourcing solutions such as Amazon's Mechanical Turk or custom frameworks have been used to solve and break down complex problems.

2.1 Paid Micro-Task Markets

Over the past several years, paid micro-task markets have become a popular means of crowdsourcing. Such a market provides users (or *the crowd*) with micro-tasks which they are asked to complete. Such a task typically lasts no more then several minutes or even seconds (e.g., labeling an image with text). For completing their short task, a small monetary reward is given to each user creating incentive to complete more tasks. Amazon's Mechanical Turk has become the most well known framework in this domain and has seen applications in robotics and HRI [28, 32].

One example that proves the effectiveness of using crowdsourcing solutions to conduct HRI research deals with object manipulation. The problem of grasp planning has been shown to be a challenging issue when dealing with robot manipulation. By using Amazon's Mechanical Turk crowdsourcing framework, work has been done to allow the crowd to help reliably and efficiently solve aspects of this problem. In [28], the crowd was able to help with reconstructing aspects of complex 3D object representations that could later be used in grasp planning algorithms.

Use of Amazon's Mechanical Turk crowdsourcing framework has also proven useful in other areas of robotics. Perhaps one of the most natural ways of interacting with a robot is through verbal commands; however, due to the complexity of human speech, giving such commands requires robust, generalized models to process them. Recent research has utilized crowdsourcing as a way to quickly collect the significantly large amounts of training data needed to build such models [32]. In this approach, the crowd was asked to view a short video of a forklift robot completing an action. The user was then asked to type a one sentence command that describes that action. Such information could then be used to train a speech model.

2.2 Remote Lab Settings and Crowdsourced Robot Control

While the previously mentioned examples have dealt with micro-tasks, more time consuming tasks can also be crowdsourced. As with micro-tasks, it has been shown that adding game mechanics helps to motivate and incentivise users for longer tasks. In recent years, the *Mars Escape* online game has used crowdsourcing to learn robot behaviors [5, 7]. The idea was to form a collaborative collection task between a robot and human pairing. Interactions between the robot and human were recorded and analyzed using different methodologies.

In addition to noting existing crowdsourcing applications and methodologies, it is important to note the motivation of the crowdsourced remote lab setting. The idea behind the development of such a system is motivated by the PR2 remote lab: an open-source remote web interface developed by Robert Bosch LLC. [27, 25] The PR2 remote lab is used for interacting with the PR2 robot by trained and expert users. Their work provides an invaluable starting point for this work. Furthermore, such existing work has been built on top of the widely used Robot Operating System (ROS).

Crowdsourcing has also been used to test different robot learning and teleoperation methods. In [10], an online interface was presented and used to ask the crowd to move a mobile robot through a maze. In this work, data collected from the online users was then used to test if users preformed better in teleoperating and training the robot when presented with an unfiltered live camera feed from the robot, or filtered image data typically used in computer vision algorithms (e.g., the location of image tags placed around the maze). By utilizing the crowd, it was shown that users presented with only the filtered image data were able to train the robot to out perform robots which were trained by users that had access to the unfiltered camera feed.

2.3 Web Based Robotics

In addition to remote lab settings, other work has been done in the area of web robotics. In [15], a remote robot was placed in a garden and online users were allowed to monitor and maintain the garden by planting seeds or watering the garden. The work done in [31] also puts a remote robot arm on the web and allowed users to manipulate a set of colored blocks.

One important part of enabling this technology is a server-side node to serve as the entry point into ROS. A popular choice for this is the *rosbridge* protocol and implementation [9]. The protocol itself defines a JSON (JavaScript Object Notation) specification to gain access to the pub-sub services ROS provides. The key to *rosbridge* is the fact that clients now only have to implement a lightweight protocol as apposed to becoming a full ROS client.

One such client implementation is the ROS JavaScript library suite [24]. This library, developed as part of the Robot Web Tools effort [1], allows web browsers to communicate with and visualize data to and from ROS. These libraries communicate to *rosbridge* via WebSockets, which is built on top of HTTP. This allows remote users, both expert and non-expert, to gain access to robots remotely using a robust, cross-platform solution.

2.4 Cloud Robotics

Cloud robotics is another emerging field that has made use of these techniques. One such project, known as Rapyuta[1], has developed a platform as a service (PaaS) framework for robots using remote, elastic, cloud-based ROS compatible computing environments. Like *rosbridge*, a JSON based specification was developed to connect the remote, cloud processes to the core ROS system.

[1]`http://www.rapyuta.org`

3 The Robot Management System

The work presented here work emphasizes a lack of formalized frameworks aimed at conducting user study research more generally. To progress the field of HRI, it is necessary to make professional grade robots and research available to the masses. Due to costs, expertise levels, and safety concerns associated with current state-of-the-art robotic platforms, using the web to conduct such studies makes logical sense. The Robot Management System is an open-source[2] framework that allows researchers quickly and easily install, configure, and deploy a secure and stable remote lab system. The RMS allows naive users to create accounts, gain access to robots, and participate in research studies. By allowing users access to robotic simulation environments, researchers are able to run experiments in parallel and reduce the risk of naive users damaging the robot or its surroundings.

The framework is designed in a robot, lab, and interface independent manner. At its core, the RMS is a custom content management system written in PHP backed by a MySQL database. Its main goal is to keep track of different ROS enabled robotic environments, interfaces, users, and research studies with little need of additional programming by researchers. Furthermore, the system is fully integrated and takes advantage of the tools and libraries developed as part of the Robot Web Tools effort [1]. By doing so, such a system enables researchers to focus on the goals of their research without needing to spend countless hours testing and implementing a custom web solution. The RMS was developed with the following goals in mind:

- Robot and interface independent design
- Support for easy creation and management of new widgets and interfaces
- Secure user authentication and authorization
- Creation, management, logging, and analysis of multi-condition user studies
- Website content management

[2]Documentation and source code are available at http://www.ros.org/wiki/rms

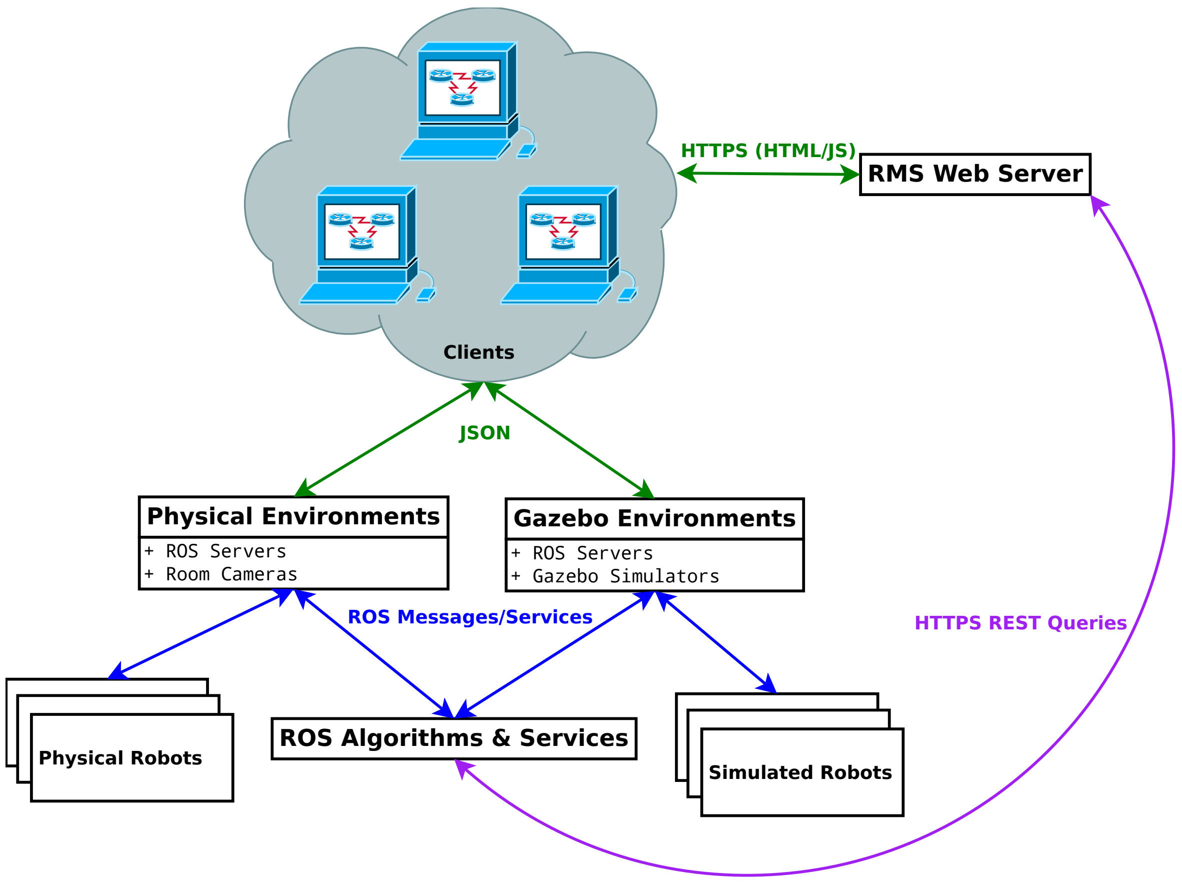


Figure 1: A High-Level Overview of the Communication Between the RMS and ROS

A high level system diagram is depicted in Figure 1. We start by examining the individual goals of the RMS before detailing the complete pipeline.

3.1 Out of the Box Content and User Management

The RMS allows for easy configuration and management of content and users. The purpose of allowing content management is to quickly produce a professional looking website custom to a particular researcher's needs. For example, if a researcher were to deploy the system for use in conducting user studies, content could be added about the researcher's work or host institution which would appear on the same site that participants would be using for their user study. An easy to use GUI back-end allows researchers to add such information without the need of any web programming.

In addition to basic content management, the system also allows for easy user management. Using a simple back-end researchers can easily add both regular and administrative users to the system. Later on, these users can be assigned to particular studies, interfaces, and robotic environments with ease. The RMS stores all passwords securely in the database using the SHA-1 hashing standard [12] with 16-character randomly generated salt strings. Such a technique provides adequate security within the RMS itself.

3.2 Managing Robots and Their Environments

One of the central components of the RMS is the ability to manage robots and robot environments. A robotic environment is defined as a single robot and its associated surroundings. This environment includes any and all sensors present such as cameras around the room. The RMS uses the advantage of generic ROS message and service types to allow the control and sensor displays of an abstract set of robots. Within the RAIL (Robot Autonomy and Interactive Learning) research group at WPI alone, the RMS has been easily connected to complex, research-grade robots such as the PR2 from Willow Garage and the youBot from KUKA, to simple robots such as the Rovio from WowWee and the NAO from Aldebaran Robotics.

3.2.1 Robot & Environment Management

The job of the RMS is to keep track of how to connect to each robotic environment and its associated control and sensor feeds. As with user and content management, easy-to-use control panels are added to allow researchers to quickly configure each environment by adding information such as camera feeds, arm and base control services, or interactive marker [16] topics. This information is stored internally and passed along to each interface when rendering the final webpage.

3.2.2 Physical and Simulated Environments

As mentioned earlier, a motivating factor for this work is to be able to conduct user studies in parallel using both physical and simulated robots. To support this capability, the RMS allows for interfacing with the Gazebo 3D simulation environment through a communication layer between the simulator software and ROS. An example environment is shown in Figure 2 with a KUKA youBot robot. By using this solution, researchers are easily able to model an approximate representation of the real world environment. Such environments can be treated like any other robotic environment within the RMS to allow seamless integration.

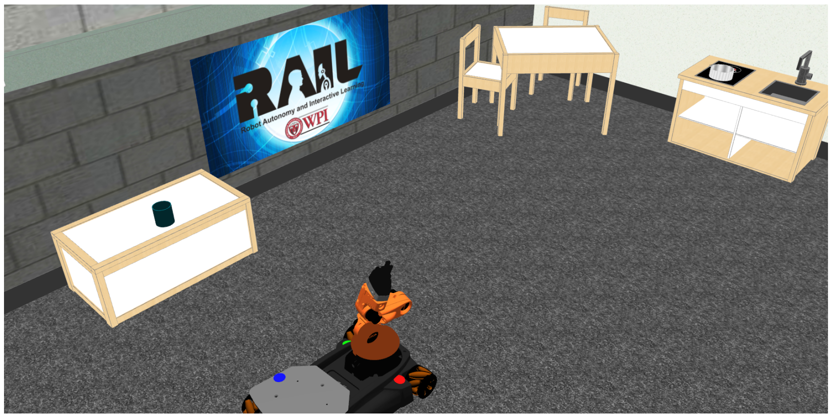


Figure 2: A Simulated KUKA youBot Robot Inside of a Gazebo Simulated Environment

3.2.3 Client-Robot Communication

In addition to keeping track of control services and sensor topics for each environment, the RMS also manages communication between the web client and the environment. Each environment has two entry points to allow communication back and forth to the client's web browser. To allow for high-bandwidth topics such as camera feeds, a server is hosted in the robotic environment to convert ROS image streams into MJPEG streams. Such streams can then be read natively by the browser. Additional sensor streams and control topics are passed through the *rosbridge* [9] protocol via JSON. A server is thus hosted in the robotic environment to manage the communication between the client's JSON messages and the underlying ROS system using WebSockets. The RMS simply keeps track of the IP address and ports associated with each environment and passes this information along to the client which will handle all communication between itself and the robot environment.

3.3 Interface Management

An important feature of the RMS is the ability to manage different interface layouts. This allows the ability to conduct studies such as A/B interface testing or creating interfaces based on varying levels of user expertise. Researchers can rapidly implement an interface using the powerful resources that the RMS provides. By implementing a single PHP script, by the time the script gets called to generate an interface, information such as the user who is using the system, connection information, and the types of control services and sensor streams are simply handed to the script in an encapsulating object.

Generic APIs are provided within the RMS to automatically generate many of the common widget types associated with remote interfaces such as keyboard teleoperation, camera streams, interactive markers, or 2D autonomous navigation.

3.4 User Study Management

With the above infrastructure in place, researchers can now manage, schedule, and analyze user studies for conducting research. As the system up until now can be used to allow users to communicate with different robotic environments using an array of different interfaces, the user study feature becomes an optional, yet crucial, component of the RMS.

Researchers can log into the system and create a new user study. Each study can then have one or more associated conditions. Each condition is also associated with an interface. For example, user study A could contain two conditions: condition X uses an interface that allows users to manually manipulate the robot's arm, and condition Y uses an interface that only allows the use of autonomous grasping methods. Alternatively, conditions could use the same interface and simply be used to label and keep track of different groups.

Finally, researchers are able to schedule individual experiments (trials). Such a trial consists of mapping a user to a given condition (and thus interface), environment, and start/end time. The RMS then keeps track of this information and will allow each particular user to gain access to the appropriate robotic environment using a given interface at the correct times. The users will only be allowed to use the interface once their time slot begins, and will be automatically disconnected once their time has ended.

Additionally, simple function calls can be integrated into interface designs to allow for logging of information (e.g., when a user changes a camera feed or sends the robot to a location in the room) within the RMS database itself. The RMS contains a simple REST (Representational State Transfer) API [14] to allow external, authorized clients to parse and analyze the data. An example client library is provided in C++ as a reference[3].

3.5 The Complete Pipeline

The above pieces fit together as a pipeline, depicted in Figure 3. This pipeline starts with a client connecting to the associated remote lab website via their web browser. From here, they are able to login to the system and authenticate against the RMS's user database. Once logged in, they are given a list of the studies that they are signed up for. If it is time for their study session to begin, buttons will be displayed allowing them to connect. Once this button is pressed, a request is sent back to the RMS to begin their session. On the server's end, the RMS checks which environment the user is assigned to (i.e., which robot they will be using) and which interface they will be using. From here, the RMS gathers all the associated control services and sensor stream topics associated with the given environment. This information is then passed to the associated interface's generation script which uses this information to dynamically create the HTML and JavaScript needed to display the information. This information is then passed back to the client's browser. The browser then renders the interface and creates the WebSocket connection to the robot environment. The user then has control of the robot and access to its sensors until their allocated time has expired.

[3]Documentation and source code are available at http://www.ros.org/wiki/librms

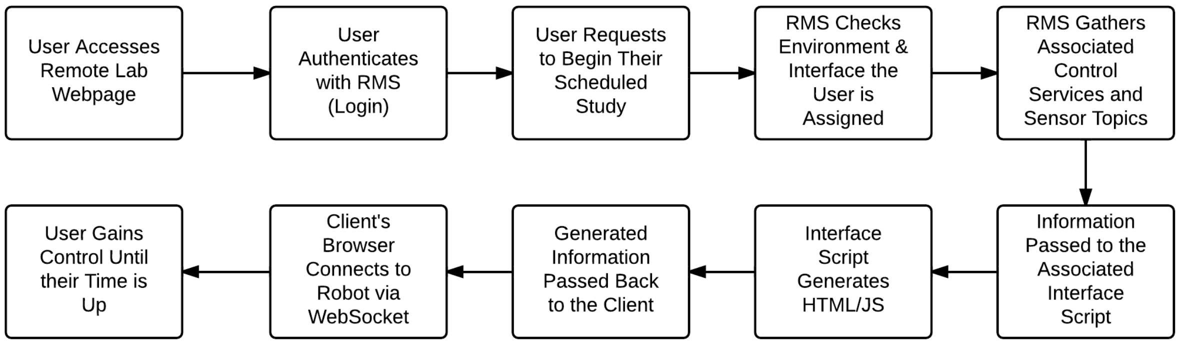

User Accesses Remote Lab Webpage
User Authenticates with RMS (Login)
User Requests to Begin Their Scheduled Study
RMS Checks Environment & Interface the User is Assigned
RMS Gathers Associated Control Services and Sensor Topics
Information Passed to the Associated Interface Script
Interface Script Generates HTML/JS
Generated Information Passed Back to the Client
Client's Browser Connects to Robot via WebSocket
User Gains Control Until their Time is Up

Figure 3: A Pipeline Showing a User Connecting to a Robot Environment via the RMS

As mentioned earlier, the RMS framework itself is designed in a completely robot, lab, and interface independent manner. This allows the system itself to be installed and deployed by all types of researchers by maximizing infrastructure and code reuse. Furthermore, the framework promotes a common ground for sharing interface designs amongst researchers.

3.6 Instances of the RMS

Although I do not explicitly track the installation or use of the RMS, I am aware of several researchers who have adopted it. Examples include the RobotsFor.Me[4] project at WPI [34, 33], the updated version of the PR2 Remote Lab at Robert Bosch LLC.[5], researchers at Brown University, and at Osaka University.

Next, we will look into the new security measures introduced into ROS to enable secure, authorized connections from anonymous remote users.

4 Message Authentication Codes for ROS

As we have seen and as is shown by this work, recent developments in the robotics community has lead to the emergence of cloud-based solutions and remote clients; however, with the increasing use and importance of such ideas, it is crucial not to overlook the critical issue

[4] *https://www.robotsfor.me*

[5] *https://www.pr2-remotelab.com*

of security in these systems. In this section, we discuss the use of web tokens for achieving secure authentication for remote, non-native ROS clients

At its core, ROS is a publish-subscribe (pub-sub) and message passing system which utilizes XML-RPC. This allows native clients from multiple platforms and languages to send and receive data in a peer-to-peer manner. Native clients have been written in C/C++, LISP, Python, Java, Lua, and C# (via Mono)[6].

While many times native client libraries are the best and most robust solution, emerging research in cloud robotics, web robotics, and the use of embedded devices have exemplified the need for using a more lightweight, protocol-based solution. This emergence of non-native ROS clients requires some server-side node to be running within the ROS system itself (e.g., *rosbridge*). This node serves as an entry point into the system and allows clients to connect with a variety of network protocols such as HTTP, WebSockets, or plain TCP, to name a few.

This growing popularity, however, highlights a crucial problem with many systems to-date: the lack of proper security procedures. Many existing remote systems have utilized *security through obscurity* techniques while running live system on the Internet. However, obscurity is rarely an effective defense for valuable targets and the computer security community regards obscurity techniques as inferior to mechanisms with proven security properties, such as cryptographic primitives.

Virtual private networks (VPNs) are often used to create secure connections between the ROS system and remote clients (both native and non-native). While effective in many cases, the need to support anonymous and non-expert remote users in using of ROS systems makes VPNs less appealing due to their complex configuration and need for separate user-side software. Therefore, a bridge must be made between the robotics and security communities to address this growing problem.

Recent work has shed light on the many known vulnerabilities of an out-of-the-box ROS system [22]. In this work, a small robotic toy car running ROS was set up at DEF CON

[6]`http://www.ros.org/wiki/Client%20Libraries`

20, an annual “hacking convention”[7]. The car was set up as a “honey-pot” to find out what weaknesses the system had. In particular, this work points out the use of unsecured TCP ports for ROS-to-ROS (e.g., node-to-node) communication in plain text. This allows for multiple problems such as intercepting and interpreting the plain-text messages, and the ease of spoofing messages into the system. Additional vulnerabilities include the standard use of unencrypted data storage. While this is a valid point, this second area is not the focus of this work.

While the use of non-native clients has opened up the research field to a vast array of new possibilities, the notion of security has been dramatically overlooked. By creating entry points into the ROS system, such as *rosbridge*, we exacerbate the problems outlined in [22]. In order to allow remote users to connect, we now create open, unsecured TCP ports with a clearly defined, lightweight protocol to control an entire robotic system. With the gaining momentum of these technologies, it is critical to begin to solve the issues associated with doing so.

In this work, efforts have been made to develop a system-independent authentication method for remote ROS clients. The method is aimed at being used in a wide array of non-native clients and does not rely on any particular user management system. This critical point allows it to be used in the largest amount of systems and does not force researchers to adhere to, or use, particular authentication systems. Instead, a schema called *rosauth* is developed which utilizes web authentication tokens to verify remote clients via an arbitrary external user management system. Once the methodology itself is described, we then look at its integration into the *rosbridge* protocol and detail an example client use case using the RMS.

4.1 Message Authentication Codes (MACs)

The basis of this schema utilizes the idea of Message Authentication Codes (MACs). The principle behind MACs is quite simple to understand. In order to ensure a message has

[7]`http://www.defcon.org`

come from a trusted source, it is hashed with a known, shared secret key using some known hashing algorithm. For example, assume client C is attempting to send a verified message m to server S. C and S both know some secret key k and agree to use a hashing function called $hash()$ and the concatenation operation "+". C will send a message with the following fields of information:

```
{
  mac:  hash(k + m)
  message:  m
}
```

When S receives the message, it will first compute $hash(k + m)$ and compare it to the received MAC. If they match, then the message received is valid. Otherwise, the message was sent by an untrusted source, was maliciously altered in transit, or was altered due to bit errors in communication. However, these guarantees only hold if a sound hashing algorithm and an appropriate length key is used [36, 35].

MACs have become essential parts of many well known and trusted security measures, including the popular IPSec (Internet Protocol Security) and SSL (Secure Sockets Layer) protocols.

While MACs typically provide message-level security, they can also be used as evidence of user authenticity. In the Kerberos protocol [29], MACs are used to create "tickets" that provide evidence that a client has authenticated using a given user or service's credentials. This approach allows an external authenticator to validate credentials and provide tickets that other remote nodes can use to ensure a user is authentic. Kerberos plays an important role in traditional computer network security, serving as part of the foundation for domain controller software, such as Microsoft's Active Directory system.

4.2 Security Goals

We most note several goals in creating a secure environment:

1. The main ROS core, associated native ROS clients (including the robot itself), and non-

native clients must be able to communicate with guaranteed message authentication, integrity, and confidentiality.

2. Any remote host must be able to request service from the system.

3. An arbitrary external authentication system can be used to provide credentials for internal ROS devices.

4. The system must provide proper authenticity and connection identity, even when multiple clients are multiplexed to the same source IP address (such as when a network address translation (NAT) device is used).

5. Connections that do not successfully authenticate will be terminated.

By meeting these goals, a system can ensure ROS devices can communicate privately, without concern for tampering or alterations from malicious outsiders.

4.3 The *rosauth* Schema

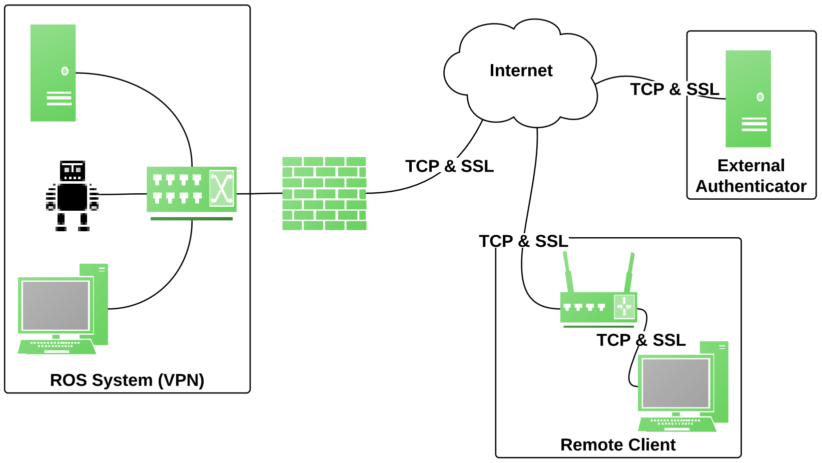


Figure 4: A High-Level Outline of *rosauth*

These security goals implicitly separate the requirements for individual packets from session security. An existing security protocol, such as IPSec or SSL, can be used to create secure tunnels for communication. VPN software is often used for this purpose to allow ROS-enabled components to communicate with each other; however, VPNs often require each party to install specialized software and carefully configure settings. While useful in closed networks, these limitations may hinder participation of external users on non-native clients.

Rather than using VPNs, the SSL protocol will be used to ensure confidentiality, integrity, and authenticity of individual packets. By using certificates issued by trusted certificate authorities, SSL can ensure that external clients know each ROS system, including the external authenticator, is legitimate. SSL also uses MAC and encryption algorithms to provide message integrity, authenticity, and confidentiality. Figure 4 depicts a network that supports both VPN and SSL connectivity for remote users. To the left, the ROS system (including its associated robot) is protected by a firewall and only accessible through a VPN or a single port listening for SSL traffic. Remote clients (bottom) can request the authentication token and associated fields from the external authenticator (top-right) in order to attempt an authenticated connection. A firewall is used to block all traffic that does not use either the VPN or authorized SSL connections.

4.3.1 Authenticating Remote Users

While SSL provides a secure channel for communication, it does not provide any assurances that a connected client is an authorized user. Instead, a system, refered to as the External Authenticator (the top-left machine in Figure 4), must be responsible for user authentication (e.g., the RMS). This server, based off the Kerberos protocol, must maintain a list of users, their associated access levels, and the private credentials, such as a password, associated with each user account. To establish a connection, a client must approach the External Authenticator with a username and credentials that the External Authenticator can use to verify the legitimacy of the client's identity claim. If the claim is valid, the External Authenticator will provide client with a token, which serves as identifying evidence to other systems, that the client can use to gain access to other systems. This approach allows a single server to take on the burden of user authentication while allowing other systems to seamlessly use this identity.

The External Authenticator must carefully construct the security token to prevent attackers from creating counterfeit tokens or altering existing tokens. The token in the authenticator must include:

- `client` (string): The client string contains the IP of the client where this message originated.
- `dest` (string): The destination string contains the IP or host of the server the client is trying to connect to.
- `rand` (string): A random string is added to the hash as a nonce to prevent replay attacks and cookie stealing, and allow for multiplexing (explained further in this section).
- `t` (int): A count of seconds since the start of the Unix epoch is given, indicating the time the original MAC was created.
- `level` (string): A user level string is provided to state what level of user is connected (e.g., admin).
- `end` (int): An end time in seconds is given stating how long the client is authorized to remain connected.

The `rand` value plays several important roles. First, the random value serves as a nonce, indicating that the same random value should not be used multiple times in a given time period. This ensures an attacker cannot simply replay a prior request to be authenticated or steal another client's credentials. This value also enables servers to demultiplex multiple clients that happen to share an IP address (such as those behind a NAT device). Finally, this value must be kept secret to ensure other clients cannot present the token; however, since the entire communication is protected by an SSL tunnel, an adversary would be unable to obtain this value through eavesdropping attacks.

The External Authenticator will use a delimiter and concatenate these fields, in the indicated order into a string, `token_fields`. The authenticator will then produce a hash MAC using a secure MAC function, such as one of the SHA-2 algorithms [17]. The hash MAC will be constructed using `MAC = hash(key + token_fields)`. The External Authenticator will then use a delimiter and concatenate both the `token_fields` and `MAC` fields into a single string, `token`. This token is then provided to the client, allowing it to attest to its identity with supporting evidence.

When the client contacts a server with a token, the server will perform several checks on the token and the client, as outlined in Figure 5. This check is made during the connection establishment phase and the connection is aborted if any of the checks fail. δ represents a small amount of time to account for loosely synchronized system clocks.

```
1: procedure CHECKAUTHENTICATION($mac, data$)
2:     if $sha512(key + data)$ is not $mac$ then
3:         return $False$
4:     else if $data['client']$ is not socket's client IP then
5:         return $False$
6:     else if $data['host']$ is not server's IP then
7:         return $False$
8:     else if $data['t']$ is not current time $\pm\delta$ then
9:         return $False$
10:    else if $data['end'] \leq$ current time then
11:        return $False$
12:    else
13:        return $True$
14:    end if
15: end procedure
```

Figure 5: The *rosauth* Authentication Check Procedure

By using SSL, we need only perform user authentication once per connection. SSL's protection of the connection makes it impractical for an attacker to inject commands into a protected connection or to assume control of another party's session. This allows the server to authenticate a connection with only a single packet from the client.

4.4 Integration with *rosbridge* and the RMS

To illustrate the schema better, we can look at its integration into *rosbridge* and the RMS. For this application, the clients are using a web-browser based interface designed with *roslibjs*[8] and connect to a *rosbridge* server using Secure WebSockets (WSS), which is built on top of HTTPS. Furthermore, the External Authenticator is the Robot Management System (RMS).

4.4.1 Authentication Pipeline

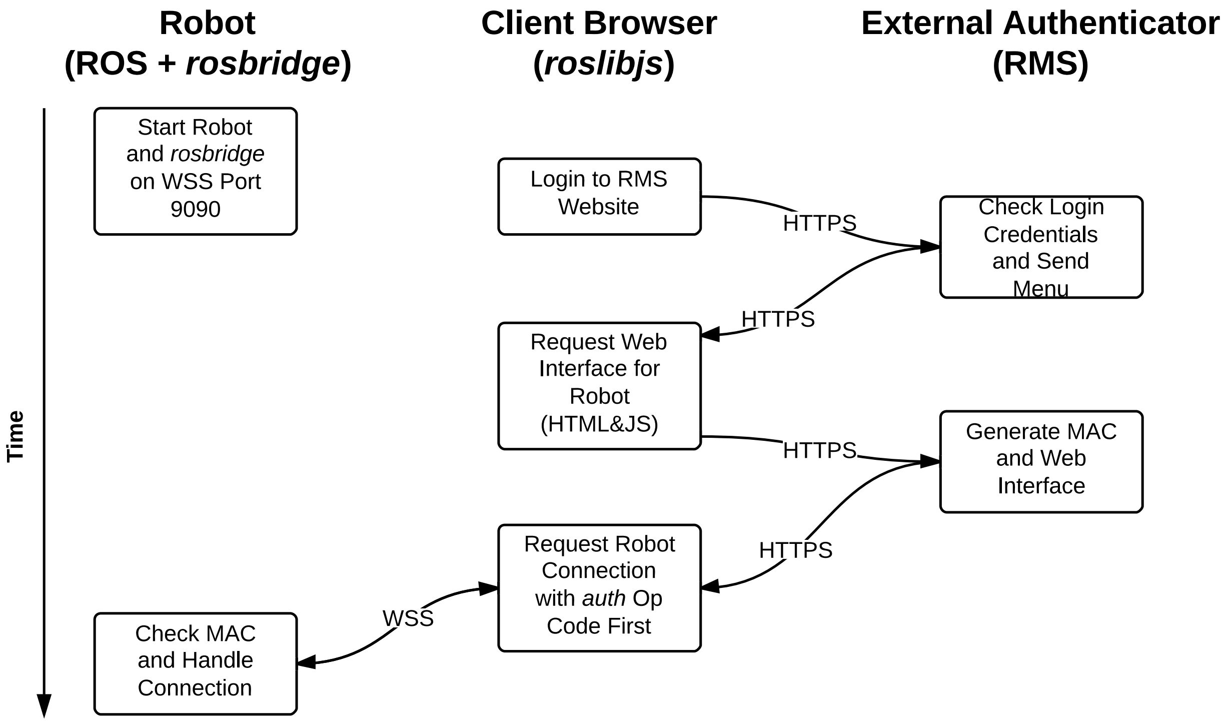

Robot (ROS + *rosbridge*)
Client Browser (*roslibjs*)
External Authenticator (RMS)
Time
Start Robot and *rosbridge* on WSS Port 9090
Login to RMS Website
HTTPS
Check Login Credentials and Send Menu
HTTPS
Request Web Interface for Robot (HTML&JS)
HTTPS
Generate MAC and Web Interface
HTTPS
Request Robot Connection with *auth* Op Code First
WSS
Check MAC and Handle Connection

Figure 6: A High-Level Pipeline of the full *rosauth* Procedure

[8]http://www.ros.org/wiki/roslibjs

We begin by looking at a high-level pipeline of the application flow. This serves to give a general idea of how the system functions before we take a closer look at some of the pieces. This pipeline is depicted in Figure 6.

We start at the top left of the diagram (note that time progresses downwards). To begin, the robot is brought up and a *rosbridge* server is started on the robot. This server runs on port 9090 and uses a Secure WebSocket with a signed SSL certificate. At this point, the robot is fully up and running, and *rosbridge* is listening for any connection.

Next, a user from an external Internet location opens up a web browser and connects to a running instance of the RMS. The connection to this site is made via HTTPS (again, using a signed SSL certificate). The user logs into this site using their login credentials which are verified by the RMS (i.e., external authenticator) itself. Once logged in, the client is redirected via a HTTP *REDIRECT* (status code 302) to the main menu page. This page provides the user with a list of available robots and interfaces. Upon receiving the menu, the user can click on the available robot and a request is sent to the RMS for an interface to control this robot.

Now, the RMS is responsible for supplying the client with the appropriate connection information and MAC string. A deeper discussion of this is explored in Section 4.4.2. Once this information is generated, it is sent back to the user as an HTML and JavaScript page via HTTPS.

At this point, the client now has an HTML page that is filled in with the appropriate connection information. The browser will execute the generated JavaScript and open a Secure WebSocket connection to the ROS system. Here, it passes along the supplied authentication information and the ROS system decides if the information is valid (discussed in Section 4.4.3). Once verified, the connection is kept open and data can stream between the client and the robot.

4.4.2 Client-Side Credential Acquisition

During this process, the RMS is responsible for supplying a correct MAC and associated information so that the ROS system will authenticate the connection request. To do so, it first looks internally at its own secured database to acquire the secret key. This key, a string of 16 characters, is randomly generated upon installation of the RMS and has also been installed on the ROS server. Next, it gathers the rest of the information by checking information such as the client's IP address (which cannot be spoofed due to the TCP handshaking process) and generates a random string. In order to determine the end time, the RMS looks internally at its user database. In this instance, admins were authorized to connect to the robot at any time, and general users were restricted to a specific time frame. This information was determined server-side and sent back to the client and included in the MAC.

All the appropriate information is hashed together using SHA-512 and is filled in as JavaScript that will be sent back to the client. This JavaScript serves as the client's security token.

4.4.3 Server Side Authentication

In order to incorporate the methods developed in this work into the ROS system, the *rosbridge* protocol itself was modified. An additional op code, `auth`, was added to the protocol with the following JSON definition:

```
{
  "op": "auth",
  "mac": <string>,
  "client": <string>,
  "dest": <string>,
  "rand": <string>,
  "t": <int>,
  "level": <string>,
  "end": <int>
}
```
Now, if authentication is enabled on the *rosbridge* server, it will wait for this op code to come in before accepting any ROS messages.

When the request does come in, a ROS service call is made to the *rosauth* node. This generic node takes in the above information and verifies it according to the procedure outlined in Figure 5. The genericness of this node allows these methods to be used in additional server implementations outside of *rosbridge*.

Once the request is checked in *rosauth*, a true or false response is sent back to *rosbridge*. Any false response is treated as an invalid request and results in a severed connection.

4.4.4 Unauthorized Connection Attempts

While the above tests showed the pipeline for a successful connection, it was also critical to test against unauthorized requests. Several tests were made to verify this.

In the first test, an unauthorized instance of RMS was set up to try and connect its own clients to the robot. In this case, all information in the MAC would be valid; however, the secret keys would mismatch. As expected, the connection was immediately dropped.

In additional tests, changes were made to certain parts of the MAC information that did not match the original information used to generate the MAC. This too lead to an appropriately severed connection. In all known previous work, these unauthorized connection attempts would be unconditionally accepted allowing full control of the robot to potentially malicious users.

With these security measures in place, we have now discussed the RMS system in full. We now begin looking at the initial case study which utilized this system.

5 A Case Study of the RMS

In this section we look at a user study conducted using the RMS to determine if remote users can be anonymously recruited to conduct a full length HRI user study. I emphasize that this is by no means a validation across all types of HRI studies, but rather a case study to expose potential weaknesses, strengths, and practicality of such a method.

For this study, participants were divided into two remote and one co-present group. Each group completed the same series of object retrieval tasks, to allow for comparison between the remote groups and the more traditional co-present group. Details about the study and its results follow.

5.1 Participants

The study comprised of 33 participants, recruited from the student body of Worcester Polytechnic Institute (WPI), as well as open recruitment via social media. Students were sent a link to a web registration form through campus email, and the same link was posted publicly through social media services. All participants committed to a one-hour time-slot, and remote participants completed a browser check to determine if they could participate in the user study using their own computer. Participants were both male and female, of multiple ethnicities, and ranged in age from 18 to 32, although no restrictions were placed on recruitment. Participants also had varied experience with robotics and video-games, from no experience to very experienced in either category. Upon completion of the study, participants received a $10 Amazon gift card.

Registration originally called for 45 participants, but due to cancellations the total number of participants became 33. Every time slot was signed up for within 18 hours of publishing the registration link. While many users were students from WPI, some participants registered from as far away as Florida, showing the potential that social media-based recruitment has for expanding the demographic base of the study.

The study was IRB approved, and each participant was given a copy of the informed consent agreement document to read through and sign before participating in the study. Remote users were required to read and digitally agree to the terms before beginning their session.

5.2 Study Conditions

All participants interfaced with the robot through the RobotsFor.Me website[9], our lab's instance of the RMS. Tree study conditions were compared:

1. **Co-present**: participants controlled the robot using a desktop computer located in the same room as the robot itself. In addition to using the on-screen interface these participants could observe the actions of the robot directly.

2. **In-lab**: participants controlled the robot using a desktop computer provided in an on-campus lab in a different location than the robot.

3. **Remote**: participants controlled the robot through their home computers. These participants completed a simple browser test to verify they could render visualizations in 3D, but were not tested on bandwidth limitations. That is, the point was to see how purely anonymous users interacted with the system.

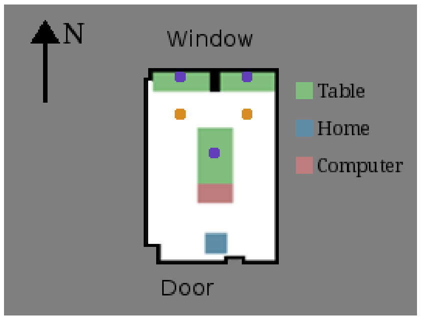
Each study condition had participants connect to the robot using a web browser. The co-present group represents traditional user study conditions, where participants are brought to the lab to participate in the study. The remote group represents the new conditions being tested, where participants do not have to come to the lab, and can simply connect to the robot from anywhere. Additionally, an in-lab group was included, which provides users with the same level of situational awareness as in the remote condition, but includes other important elements of traditional user studies, including standard computer hardware and a more professional and potentially less distracting environment.

The robot used was the PR2 robot from Willow Garage. The PR2 was designed for use in robotic studies, and consists of a mobile base with two 7-DOF arms used for object manipulation. The robot is approximately the height of a human being, and can therefore interact with objects in a normal household setting.

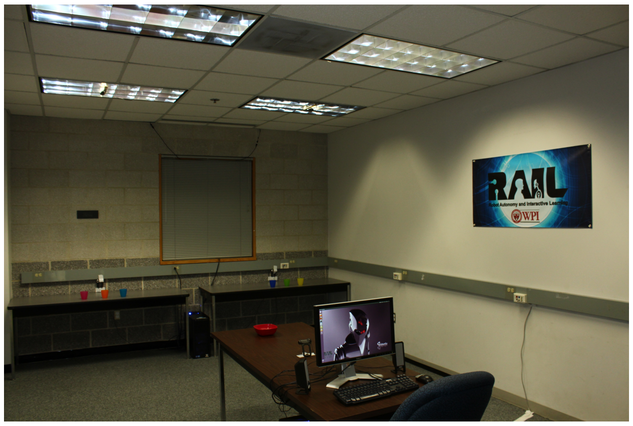
[9] *https://www.robotsfor.me*

5.2.1 The Robot Environment

The study took place in an isolated lab approximately 6.7 meters long by 4 meters wide. The lab contained three tables: two tables placed along the back wall of the room and one table located in the room's center. The back tables each contained three bowls of identical shape situated on the tables' centers. The middle table contained three different objects (a white-board eraser, a tape measure, and a roll of masking tape) on one end and a computer station on the other. The room also included a home location for the robot, marked with a rectangle of white masking tape in the front of the room. A diagram of the lab is shown in Figure 7(a), and a picture of the room taken from the south-east corner is shown in Figure 7(b).



(a) A Diagram of the Robot's Environment



(b) A Photograph of the Robot's Environment

Figure 7: The Robot's Environment Used in the Study

The lab also contained a variety of cameras mounted to provide different view of the room. These included a camera located on the PR2's head, as well as a camera on the end of each forearm. The room itself included five cameras, three of which were placed on each table with a view facing outward, encompassing the three objects on each table, an example of which can be seen in Figure 8. These cameras are represented by purple dots in Figure 7(a). The remaining two cameras were placed on the ceiling giving a view of the east and west sides of the room, which can be seen as the orange dots in Figure 7(a).

Figure 8: Image from the North-West Table Camera

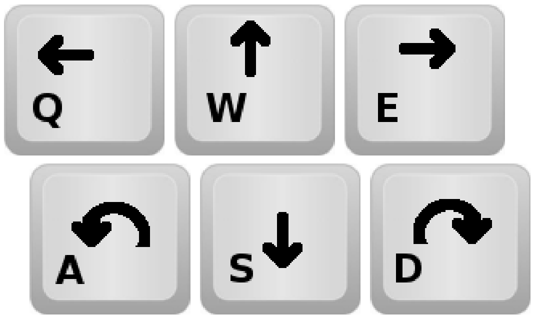


Figure 9: The Mapping Between Keyboard Keys and Navigation Commands

5.2.2 The Remote Interface

The web browser-based interface accessed through the RMS is shown in Figure 10. The interface itself consists of a main control panel on the left hand side which can be used to give high-level commands to the robot, two customizable camera feeds in the bottom-left and bottom-right portions of the interface, a 2D map that displays the robot's location and allows for autonomous navigation in the bottom-center portion, and a 3D visualizer in the center that allows for sensor visualization and manipulation assistance.

Basic Controls To provide manual navigation capabilities to the user, movement commands were bound to the keyboard. The PR2 robot is able to move in any direction, including sideways. As a result of this, the standard arrow keys are not enough to drive the robot around. Instead, the web interface makes use of a series of letter keys on the keyboard. Figure 9 shows a mapping between the keyboard keys and the movement command. Furthermore, the speed of the robot could be adjusted using the speed slider at the top of the left-hand control panel. We note that although this slider allows the user to set speeds from 0%-100%, 100% was throttled to be two-thirds of its maximum speed for safety reasons.

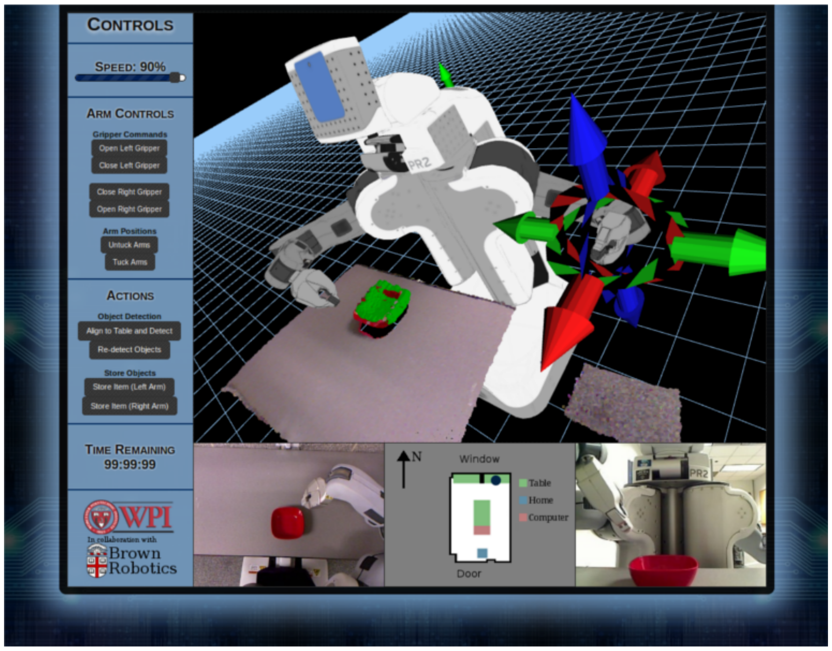


Figure 10: The Web Interface

Cameras Feeds As shown in the interface screenshot (Figure 10), users were able to view two camera feeds at a time (the lower-left and lower-right boxes). As mentioned earlier, there are a number of different camera feeds that could be streamed. In order to change the camera feed in the interface, the user simply hovered over the camera feed they wish to change and selected an *Edit* button that appeared. From here, a drop-down list of available camera feeds appeared that could be used to select the camera feed of their choice.

The 2D Map and Autonomous Navigation The center section at the bottom of the interface displays both a map of the room (the same map given in Figure 7(a)), as well as the robot's position. As the robot moves, the map renders a blinking blue dot at the robot's current location.

In addition to displaying information about the robot's location, this map can be used to give the robot a destination command. By double clicking on a location within the room, users could tell the robot to try to autonomously navigate to that position safely. A status of the robots planning state was displayed in a popup dialog that also allowed users to cancel the action.

The Control Panel On the left side of the interface is the main control panel. This panel contains buttons that can be used to easily control high level action capabilities on the robot. The control panel begins with a speed slider. This slider controls how fast the robot moves when issuing keyboard navigation commands.

The next two sections provide buttons to initiate certain actions of the robot. The first of these sections is the *Arm Controls* section. This section provides a series of useful arm controls such as opening/closing the grippers, and tucking/un-tucking the arms of the robot. Tucking the arms allows the user to drive the robot around the room without the fear of knocking the arms into the environment. The second of these sections is the *Actions* section. This section provided a series of useful controls that would help with object manipulation. The following is a list of the actions that were provided to the user.

- **Align to Table and Detect**: When a user was within close proximity of a table, this button became enabled. Upon clicking this button, the robot automatically aligned to the table and began to look for objects. This consisted of facing the table, un-tucking its arms, driving right up to the table, raising its spine to get an overhead view of the table, and using 3D point cloud data from the Kinect to segment any objects found in the environment [8]. Any objects that are found are rendered as a green, 3D model on the 3D visualizer of the interface. A status of the robots state was displayed in a popup dialog as the action was run.

- **Re-detect Objects**: This action is the same as the final stage of the above action.

- **Store Item (Left Arm)**: Once a user has picked up an object in the robots gripper, they were able to store it inside of a built in pouch that was installed on the robot's base. The use of this button automatically places any objects in the left gripper into the pouch.

- **Store Item (Right Arm)**: This action is the same as the above but uses the right arm instead.

The final section of this panel is the *Time Remaining* section. This countdown let the user know how much time they had left for their study session. Once the clock ran out, they were disconnected from the environment and notified via a popup dialog.

The 3D Visualizer The majority of the interface comprises of a three-dimensional visualization of the robot and its perceived environment. This visualizer is a custom version of the one used in the PR2 Remote Lab [27] known as WVIZ (Web-Visualizer). This visualization can be zoomed, rotated, and panned to provide any view. At all times, a 3D model of the robot is displayed in the main window depicting its current state. At any point, the user could also request that a 3D RBGD point cloud be rendered in the window from the robot's Kinect.

In addition to displaying sensor information from the robot, the visualizer could also be used to control various aspects of the robot. In particular, the robot's end effectors could be directly translated and rotated in 3D space using a set of rings and arrows [21] known as interactive-markers. This is displayed in Figure 10 on the robot's left gripper. This movement, in combination with the gripper controls provided in the control panel, allowed users to manually manipulate and grasp objects.

As mentioned earlier, 3D renderings of any segmented objects are also displayed in the visualizer. Along with being a useful reference point for manual manipulation, this rendering also allowed for autonomous grasping. By right clicking these green markers, the user was able to choose either a left-hand or right-hand pickup. The information was then sent back to the robot environment where grasp planning was performed using existing methodologies [8]. A status of the robots state was displayed in a popup dialog as the action was run and any failures were reported back to the user.

In addition to manipulation control, the visualizer also allowed users to control the angle and direction of the robot's head (and thus the camera feed that was mounted to the head). Users were able to drag a 3D marker around in the visualizer and the robot would point the center of its head at that point.

5.3 Design

Participants were divided into three independent groups each with 11 participants: a co-present group (participants carried out the user study at the computer station located in the lab with the robot), an in-lab group (participants used a computer station located in a separate lab that did not contain the robot), and a remote group (participants completed the user study from their homes). The remote group was selected randomly from the participants who passed the browser check during registration. The remaining participants were randomly divided into the co-present and in-lab groups. Each group was asked to perform the same set of tasks with the robot, and each group used the same interface as specified above.

In order to make the conditions of each group as similar as possible, all instructions for the user study were given in the form of a web-based instruction set. Co-present and in-lab participants were given a brief verbal overview of the user study and were directed to the tutorial, whereas remote users were given the same information in the text of an email at the start of their session. Once the experiment began, we interfered with participants only in the rare event of technical difficulties (i.e., if the simulator program crashed), and there was otherwise no communication with participants in any group.

5.4 Procedure

The web-based instruction set had participants in every group carry out a series of three object retrieval tasks, using different combinations of navigation and manipulation teleoperation techniques for each task. Before carrying out any of these tasks, however, participants first completed a tutorial to familiarize themselves with the PR2 and the interface.

The tutorial comprised of a set of step-by-step textual instructions supplemented with images, such as Figure 10, explaining how to change camera feeds, navigate the robot, and manipulate objects. Each section of the tutorial could be returned to at any point during the user study for review. Participants completed the tutorial by controlling a virtual robot in a virtual representation of the lab simulated using the Gazebo simulator as seen in Figure 11. This simulated environment was a replica of the room the physical robot would be in, but contained a subset of the objects and furniture that was available in its physical counterpart. In particular, the center table and all manipulable objects were removed from the simulated environment. This distinction was made in order to provide users with an environment that would allow them to test pieces of the interface (navigation, table alignments, moving the arms) without being able to practice the final task. The interface itself was identical to that used with the physical robot.

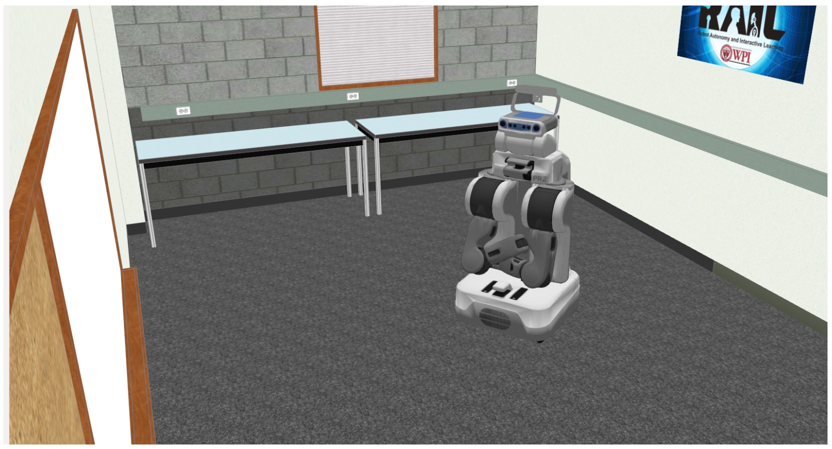


Figure 11: The Simulation Environment Used for the Interface Tutorial

After completing the tutorial, participants completed as many of the following three object retrieval tasks as possible within the time limit of the study.

- **Task 1** The first task involved using only manual teleoperation techniques to retrieve an object from the north-east table. As such, participants navigated the robot to the table manually using keyboard navigation controls, manually moved the robots gripper to pick up an object, and navigated the robot back to the home position with the keyboard controls.
- **Task 2** The second task focused on autonomous teleoperation techniques to retrieve an object from the north-west table. Participants clicked on the map to set goal points

for the robot to autonomously navigate to, and, once at the table, selected an object for the robot to pick up autonomously.

- **Task 3** The final task was open-ended, and instructed the users to navigate to the center table, pick up one object, and navigate back to the home position using their choice of techniques.

The interface provided no restrictions on what participants were allowed to do during each task, and also provided no indication of whether or not the task was completed successfully. As such, interpretation of how to complete the tasks, when to move on to the next part of the instruction set, and whether or not a task had been completed successfully was left up to the participant.

Throughout the duration of the study, a member of the lab was in the room with a robot to ensure that tasks were carried out safely. Participants were informed that a safety operator would halt all motors of the robot if it was placed in a dangerous position (i.e., any situation in which the robot could harm itself or the environment). The operator would then either move the robot to a safe position and re-enable the motors, returning control of the robot to the participant, or terminate the study if the situation was deemed too dangerous.

Once all three tasks were completed, or after the one-hour time slot ended, participants were directed to an exit survey where they filled out questions about demographic information, the ease of which they found different aspects of the study, which methods they used to complete the third task, and other questions about their overall experience with the study in general. At the completion of the survey, participants were then given their gift card as compensation for participating in the study.

5.5 Measures

Throughout the experiment, a vast array of data and observations were logged for each participant. This data was then parsed and explored in detail at the conclusion of the study.

In particular, numerous measures relating to the participant's interaction with the interface were continuously logged. To monitor how participants were interacting with camera feeds, a counter was recorded indicating the number of times each participant changed a feed. Additionally, the amount of time that each participant spent in both the simulator and the physical environment was recorded.

Navigation: To measure navigation performance, several metrics were recorded. For manual navigation, the number of keyboard commands (i.e., button presses) and the total distance traveled using manual navigation were recorded. For autonomous navigation, the number of autonomous navigation commands given was recorded.

Manipulation: To measure manipulation performance, a count was made for the number of interactions (i.e., mouse clicks) with the end effector's interactive marker, as well as the total distance the end effector moved as a result of the interaction. To look at autonomous pickups, the total number of requests for an autonomous pickup was recorded.

Behavioral Observations: In addition to the numerical measures recorded by the system, observational records were kept of each participant by one of the researchers. Such records were made to keep track of the overall progress and performance of the user. For remote participants, observations were made simply based on the performance of the robot. Videos were also recorded from the various room cameras for reference at the conclusion of the study.

Survey: Finally, at the conclusion of a participant's session, they were instructed to fill out an exit survey. This survey was used to keep track of the participant's age, location, robotics and gaming experience on a 7-point Likert scale, difficulty rankings of the manipulation and navigation techniques on a 7-point Likert scale, and general written feedback.

6 Results

While the majority of users completed aspects of the tasks in some manner, looser interpretations of the task descriptions, discussed further in the discussion section, lead to less clear-cut definitions of a participant's *success* (e.g., users picking up all objects in the room instead of the instructed 1 per table). As a result, note that no numerical or observational data was kept on the *success* of a participant. The *success* of the participant thus became an extremely subjective measure. It should be noted, however, that such a phenomenon was present equally amongst all groups as no correction or interference was given to even those participants who were co-present with the researcher.

To analyze and present the results, we break our findings into three distinct sections. The first section compares the recorded measures among each of the three groups. As mentioned earlier, the main goal of this study was to gain insight on the possibility of using anonymous, remote users for a full-length HRI study using the RMS. As a consequence of this, we hope to find little-to-no real difference amongst the three groups.

We emphasize one important note with these findings. Due to relatively small number of participants in each of the three groups, results from the resulting statistical tests are limited and should be taken more as an insight into the similarities and differences rather than that proof.

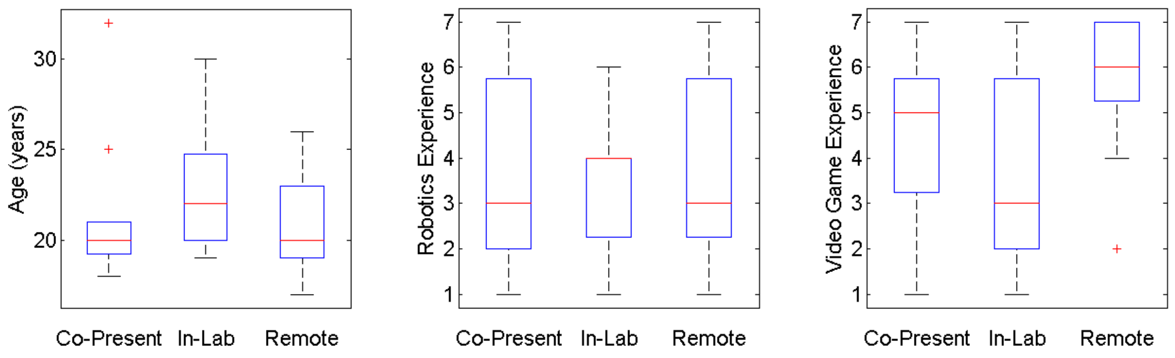
Once these results have been explored, we then move on to looking at the measures amongst the three groups as a whole. This information sheds light on teleoperation navigation, manipulation, and general robotic interaction using a computer interface, or, more specifically, a web interface. Finally, we look at a summary of our observational data. A discussion of these sets of results is presented in the following section.

6.1 A Comparison Among Groups

We begin with a comparison of the general demographics information for each group. Along with their age, participants were asked to rate their previous experience with robotics and with video games on a 7-point Likert scale, from no experience (1) to very experienced (7). A comparison of this data is given in Figure 12. Each group included participants between the ages of 17 to 32, with the majority of users falling into the range of 18 to 25 years old. Robotics experience was also comparable among the groups, as it widely varied within each group.

Video game experience, however, had a notable difference in participants of the remote condition, where participants rated their level of video game experience as generally high.

Without more information, it is difficult to determine the cause of this difference in the remote group.



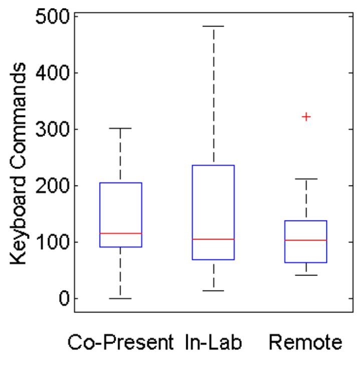
(a) Age (b) Robotics Experience (c) Video Game Experience

Figure 12: Plots of Demographic Information Amongst the Three Groups

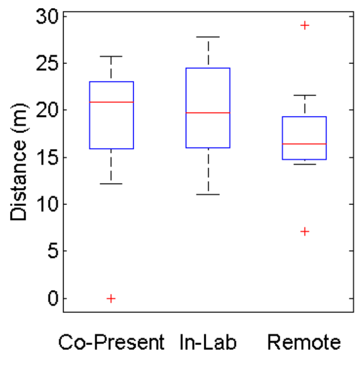
We next look at and compare the numerical measures recorded throughout the study. Note that despite the fact that all three groups were performing the same tasks, it could be the case that co-present users may gain advantages (or disadvantages) by being in the room with the robot which could drastically change results. For example, co-present users who are able to view the robot as it is driving or manipulating objects might make fewer clicks on the interface or send fewer movement commands as they are able to view more of the environment directly. To begin, we look at navigation data. A summary of these results is presented in Figure 13 and in Table 1.

Table 1: A Summary of Navigation Results Amongst the Three Groups

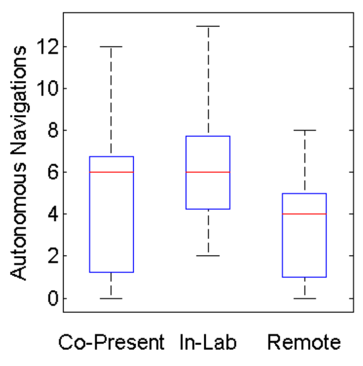
	Manual Commands		**Distance Traveled (m)**		**Autonomous Commands**	
Condition	μ	σ	μ	σ	μ	σ
Co-Present	152.6	130.8	20.2	5.0	6.5	3.1
In-Lab	143.6	83.3	18.4	6.9	4.8	3.6
Remote	119.4	79.7	17.0	5.2	3.8	2.7



(a) # Manual Nav. Commands



(b) Meters Traveled Manually



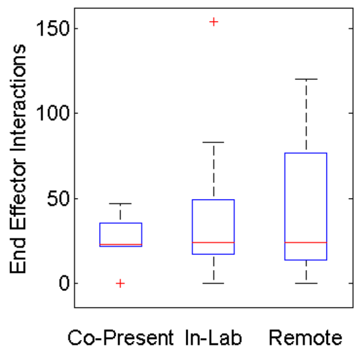
(c) # Autonomous Nav. Commands

Figure 13: Plots of Navigation Results Amongst the Three Groups

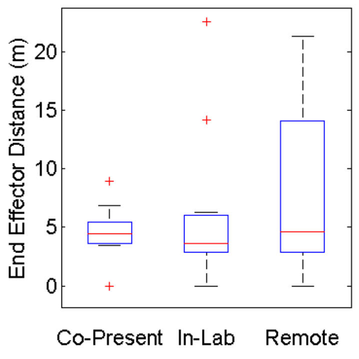
When looking at the number of manual navigation commands (e.g., keyboard navigation keys pressed) (Figure 13(a)), distance traveled via manual navigation in meters (Figure 13(b)), and the number of autonomous navigation commands (Figure 13(c)) given throughout the three groups, we find no strong difference between the groups. We can note a slight difference in the number of autonomous navigation commands given by the remote users; however, in general, using statistical reasoning it is clear that there is no significant difference between these three groups with these measures on average.

Even though the three groups were asked to drive around the same room performing the same task, what these results show is that non-co-present users appear to have the same situational awareness when navigating around the room.

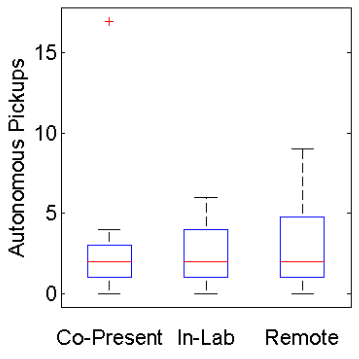
We now move on to measures dealing with manipulation tasks. A summary of these results is presented in Figure 14 and in Table 2. When looking at the number of end effector interactions (e.g., clicks of the end effector's interactive marker) (Figure 14(a)), the total distance the end effector moved as a result of the interaction in meters (Figure 14(b)), and the number of autonomous pickup requests (Figure 14(c)) given throughout the three groups, we again find no significant difference between the groups. The same conclusion can be drawn here as well with regards to situational awareness. We look not at task performance, but emphasize the fact that remote users appear to have no immediate disadvantage for the task at hand. Such observations are crucial in stating the effectiveness and validity of using remote users for conducting particular types of HRI studies.



(a) # End Effector Interactions



(b) Meters End Effector Moved



(c) # Autonomous Pickup Commands

Figure 14: Plots of Manipulation Results Amongst the Three Groups

Table 2: A Summary of Manipulation Results Amongst the Three Groups

	End Effector Interactions		**Distance Traveled (m)**		**Autonomous Pickups**	
Condition	μ	σ	μ	σ	μ	σ
Co-Present	39.9	42.7	6.0	6.4	2.5	1.8
In-Lab	24.0	14.1	4.3	2.5	3.1	4.5
Remote	43.7	40.6	7.8	7.0	2.9	2.8

An interesting observation to make here, however, is the number of end effector interactions and the distance traveled by the end effector for the remote group. Note the large spread of data into larger numbers of interactions and distances traveled in this group as compared to the others. Interestingly enough, such a phenomenon does not occur with the group that was in a separate lab at WPI (i.e., non-co-present, yet still in a lab setting). The cause of this spread is unclear, but it is speculated that this increased spread is due to differences in computing infrastructure. Participants in the co-present and in-lab groups used the same hardware provided by the researches, which ensured that they had access to a high-resolution monitor and a mouse with a middle button (used to move the camera in the 3D visualizer). These conditions for remote users could not be guaranteed, and in fact some remote users performed the study using laptops using trackpads. While the e-mailed instructions did suggest a high resolution screen and the use of a mouse, this could not be enforced. If either condition was not met, control of the 3D visualization of the robot (where all end effector interactions take place) becomes significantly more difficult, and could result in more interactions to reach a desired goal.

To gain further insight into any differences in navigation and manipulation among the three groups, participants were asked to rate the difficulty of the manual and autonomous navigation and manipulation tasks. Figure 15 shows a comparison of the average difficulties reported in each group. Most of the reported difficulties are comparable among the groups, but there are a few differences. The in-lab group perceived manual manipulation as more difficult than the other groups did, the remote group found autonomous manipulation more difficult, and the co-present group found manual navigation easier. In each of these cases, the differences fall within one point of the scale, and so the differences are minor. The implications of these small differences are discussed further in the Discussion section, but for the purpose of analyzing the user study results, these difficulty ratings are considered to be equal for each group.

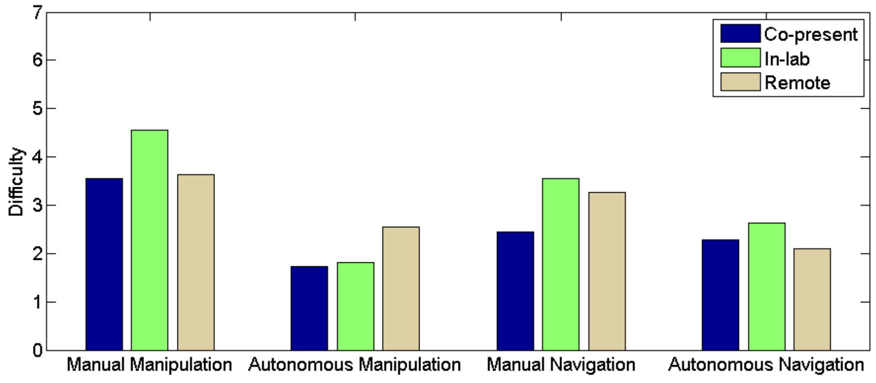


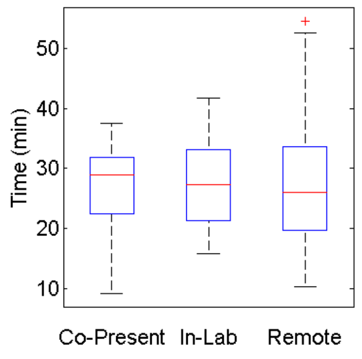
Figure 15: Participant-Reported Difficulty of Manipulation and Navigation Tasks

At this point, we have gained insight that anonymously recruited remote users are able to perform such a study at a similar level as co-present users. Before moving on to the presentation of further findings, note two other numerical measures. The first is the total time each user spent in the physical environment (16(a)). Here we see that the co-present group spent an average of 28.2 minutes ($\sigma = 7.9$), the in-lab group spent an average of 26.4 minutes ($\sigma = 7.8$), and the remote group spent an average of 28.6 minutes ($\sigma = 13.5$) in the environment. Note again the larger spread on the remote group. Once again, in general, using statistical reasoning we find no significant difference between these three groups with this measure.

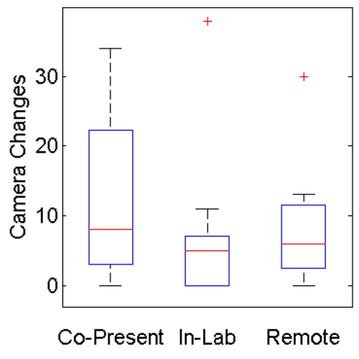
The final measure we look at is the number of times participants in each group changed the camera feeds (16(b)). Here, the co-present group made an average of 7.0 changes ($\sigma = 10.4$), the in-lab group made an average of 12.3 changes ($\sigma = 11.2$), and the remote group made an average of 7.9 changes ($\sigma = 8.1$) in the camera feeds. Note again that by using statistical reasoning we find no significant difference between these three groups. Interestingly enough, however, the group with the largest, and quite noticeable, spread is the co-present group. As with the high spread in the remote group's interactive manipulation measures, the reason for this spread in co-present camera changes is unclear, but we can make some speculations on the subject. The locations of the cameras were likely more apparent to co-present users, who could physically see the locations of the cameras in the lab. In-lab and remote users were provided with the camera locations as seen in Figure 7(a), which was also provided to the co-present users, but we can speculate that being in the room physically better reminded users that more camera views were available. Also of note is that while participants in the room could simply look at the robot, there were many situations where the robot itself occluded the participant's views of the pickup objects, so use of multiple camera views was as necessary in this case as in the remote and in-lab cases.

6.2 Findings Across All Groups

Beyond determining the validity of conducting a remote user study, the study provided an opportunity to gather HRI data on navigation and manipulation techniques using the



(a) Total Time (Minutes)



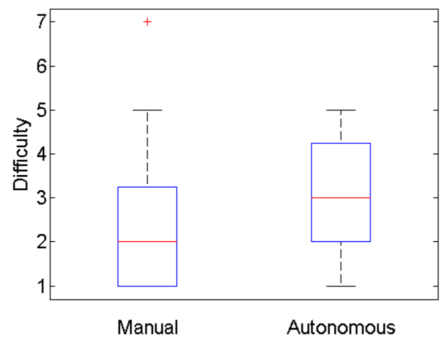
(b) # Camera Changes

Figure 16: Plots of Total Time and Camera Feed Changes Amongst the Three Groups

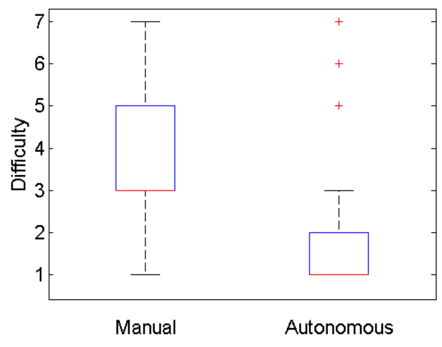
provided interface. Participants were asked to rate the difficulty of manual and autonomous manipulation and navigation using the interface. Furthermore, the open-ended nature of the third task allowed users to choose whether to use autonomous or manual techniques, providing insight into their preferred methods of robot teleoperation.

Difficulty of each teleoperation technique was measured on a 7-point Likert scale, from very easy (1) to very hard (7), the results of which can be seen in Figure 17. Participants rated manual navigation at an average difficulty of 2.3 ($\sigma = 1.8$), autonomous navigation at an average difficulty of 3.1 ($\sigma = 1.4$), manual manipulation at an average difficulty of 3.9 ($\sigma = 1.6$), and autonomous manipulation at an average difficulty of 2.0 ($\sigma = 1.7$). Participants generally agreed that manual navigation was easier than autonomous navigation, with the exception of one outlier. Conversely, participants found manual manipulation more difficult than autonomous manipulation in almost all cases.

Figure 18 shows the choices participants made when completing the third task. For both navigation and manipulation techniques, participants could use either manual, autonomous, or mixed manual and autonomous techniques. Some participants did not complete the third task; these can be seen in the N/A columns of the plots. In both cases, users chose autonomous techniques slightly more often, but mixed and manual techniques were both commonly used.



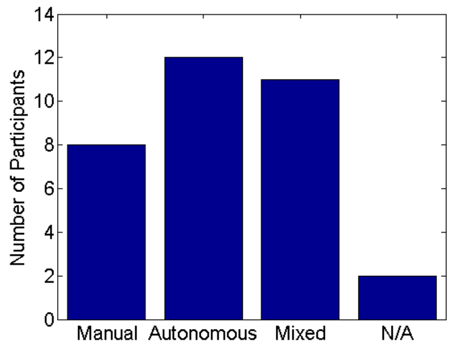
(a) Navigation Difficulty



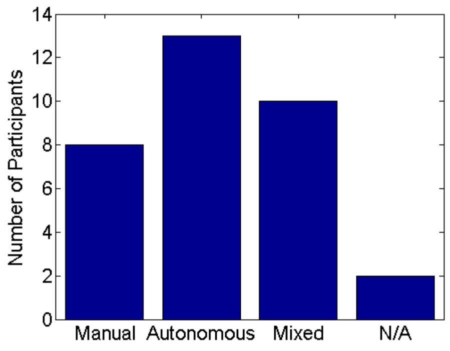
(b) Manipulation Difficulty

Figure 17: Plots of Difficulty of Techniques

Participants were also given additional space in the exit survey to explain why they used their chosen methods. For navigation techniques, a common consensus was that autonomous navigation was easy to use over long distances, but it was imprecise over short distances. As such, participants often used manual navigation or a combination of manual and autonomous navigation to allow for finer control of the robot when approaching the third table. For manipulation techniques, participants agreed nearly unanimously that autonomous manipulation was easier to use. Many participants, however, opted to use manual manipulation because it was more enjoyable, or because they enjoyed the experience of controlling a robot's end effector using interactive markers.



(a) Navigation Technique



(b) Manipulation Technique

Figure 18: Plots of Techniques Chosen for the Third Task

6.3 A Summary of Observations

We now look at a summary of the observational data collected throughout the study. Our first reporting deals with overall proficiency of the interface, robot control, and task completion. It is believed that the variation in overall task completion interpretations is the result of certain users simply skimming or glossing over the instruction set. Efforts were made to keep the instruction set short, concise, and informative; however, it was an inevitable circumstance that not every user would adhere to the instruction set.

This is exemplified by the observations made throughout the tutorial session. Despite spending an average of 15.4 minutes ($\sigma = 6.4$) in the simulator, many participants used more of a trial-and-error approach to learning the interface. While we should not reject the value of using such a technique to learn a new type of interface, it is believed that stronger proficiency of the interface could have been made if users supplemented this with the step-by-step instructions, especially given the time constraint. As a result, participants throughout all groups were uncertain of how portions of the interface worked when given access to the physical robot. This became even more apparent when observing the robot throughout the session (i.e., participants randomly using buttons on the interface to see the result).

In addition, consistent observations were made of participants performing actions that resulted in intentionally harmful scenarios for the robot. For example, numerous participants attempted to tuck the arms of the robot while the robot was flush against a wall or table. The result of such an action attempts to move the arms without any regard of collisions. While an expert user might immediately be aware of this situation and attempt to correct it (e.g., drive away from the obstacle), note that participants simply ignored the collision altogether. In very rare, extreme cases, the researcher would halt the robots motors and move the arms into a safe position. Note that such interference was rare, common across all groups, and occurred no more than twice for a given user. More importantly, the researcher never moved the robot into a position that would assist the user in any way; it was merely a matter of safety.

An additional observation worth noting deals with manual manipulation. Due to the layout of the room, the first manual manipulation task required both remote and co-present users to rely on the camera feeds and sensor information of the robot to perform manipulation. This is due to the fact that by having the table against the north wall, the robot would occlude the objects from the user's view even if they were in the room. This, however, was not the case with the final manipulation task. Here, the objects were placed on the center table opposite of the computer station. These objects where therefore clearly visible to those users who were co-present. For co-present users who chose to use manual manipulation for the final task, a strong tendency to use the on-screen visualizations (both 3D and the associated camera feeds) as opposed to looking directly at the objects was observed.

We should also note that this was the opposite case for manual navigation. Here, an overwhelming majority of co-present users looking directly at the robot while navigating it with the keyboard was observed. Participants in the co-present group also reported that manual navigation was easier as compared to the other groups, as seen in Figure 15. While this result is not too surprising, what is surprising is the fact that there was no clear advantage in doing so. This was justified in previous sections and in Figure 13. This is further justified by the observations made of environmental collisions. In general, collisions with the environment due to navigation were rare. Furthermore, these types of collisions occurred throughout all groups, including those who were co-present. Thus, while it seemed intuitive to look at the robot while navigating, it seemed to provide no clear advantage.

7 Discussion

As explained in Section 6.3, many participants appeared to learn various features of the interface through trial and error using the real robot, as opposed to following the steps of the tutorial and using the simulator. This behavior could be expected, considering there was no verification that the tutorial was completed. Adding step-by-step verification to the tutorial could have improved the participant's learning process, while also alleviating safety concerns of letting non-expert users control an expensive robot.

On the subject of safety, observing some of the dangerous behaviors (i.e. tucking the arms against a table), a better method other than researcher intervention is required. Autonomous safety overrides, such as detecting proximity to obstacles when tucking and un-tucking the PR2's arms, could solve this problem. Note that some autonomous safety overrides for the RMS for robotic platforms other than the PR2 were developed and tested, but for this user study the same safety overrides were not in place for the PR2 robot.

A final safety issue that arose was the need to disconnect participants that repeatedly put the robot into dangerous situations. Over course of the study, this only occurred once, when repeated collisions threatened the safety of the computer in the lab. In this case, a malicious user repeatedly attempted to slam the base of the PR2 into on of the server computers located in the room. After several consecutive occurrences, the operator made the decision to terminate the participant's session. This participant's data was not included in any of the results presented. While the case is rare, this problem should be addressed, especially if any unsupervised studies are to be run in the future.

Note that two of the potential major strengths of using the RMS for conducting user studies was not tested here, and those are the ability to run user studies in parallel and the ability to run user studies continuously and unsupervised. Participants in the remote user group were run through the study directly one after the other, and this worked well as it removed the added time constraint of bringing participants into the lab, but due to some of the safety concerns expressed previously, each participant still needed to be supervised. As for conducting user studies in parallel, this was beyond the scope of the initial research and leave it for future work. That stated, the system would be able to support parallel user studies given enough hardware or via the use of a series of simulators.

8 Known Limitations

The RMS and general usage of remote HRI studies are not without their own set of limitations. As mentioned earlier, there are numerous aspects of HRI research that cannot be addressed with such a system. Particular aspects of physiological and emotional work, in which co-presence and face-to-face interaction are of central importance, are clear examples of this. Additionally, interactive techniques such as kinesthetic teaching [19] become infeasible.

Also note that many of the challenges associated with the current state of online robotics dealing with large amounts of sensor data. In particular, certain streams of data cannot be reliably sent to all remote users, such as live 3D point cloud data or high-resolution camera feeds, due to bandwidth and real-time processing limitations. This problem is further exemplified by the requirement to have a responsive, low-latency interface for teleoperation. Interfaces must therefore adapt to these limitations. We will explore this area in the future work section.

Latency issues can be addressed partially by using a wired connection to the robot as opposed to wirelessly streaming its sensor data. This, however, can lead to limitations of the robot's capabilities (e.g., not being allowed to leave a certain room). In many cases this may not be an issue and can provide a substantial boost in bandwidth.

Also note the limitation in the efficiency of running remote user studies due to safety concerns. In some cases, it may be required to monitor a robot while it is in use. For example, if a large robot is allowed to navigate freely around an environment, it might be necessary to have a researcher present at all times. This limits part of the round-the-clock capabilities of such a system; however, these issues can be addressed by using simulation environments. Nevertheless, simulation environments must also be monitored to a certain extent to maintain their stability. This, however, is much easier to manage and can be done remotely.

9 Conclusion

We have presented a novel system for use in a subset of HRI user studies. This work has developed, deployed, and tested such a system in hopes of showing its potential and validity as a form of an HRI research tool.

In addition to the system itself, we have explored and developed a custom MAC based authentication schema for remote, non-native ROS clients. As expressed in [22] and demonstrated at DEF CON 20, the need for security techniques must be brought into the robotics and ROS communities.

While we should note that some of the methods used in this work could solve broader problems within ROS (discussed further in Section 10), this work focuses on the case of having remote, non-native ROS clients. For native ROS clients, we assume such devices can communicate directly to each other via some trusted, secure network.

This work provides new methods for providing security measures aimed at authenticating remote users from any IP address using non-native ROS clients. As exemplified in work such as [24, 27, 9, 25, 33], the ability to utilize these non-native clients will allow roboticists to utilize a wider range and more diverse group of users and researchers. Nevertheless, with the growing momentum behind such easy to use lightweight protocols and techniques, it becomes ever more important to ensure these environments are run in a safe and secure manner.

The developed security token schema ensures that only clients which have been authenticated from some trusted external authentication source are allowed access to the robot. The development of such a generic schema also allows for a wide array of out-of-the-box (such as the RMS) systems, or custom user management systems to be used to control access to the robot. Furthermore, the *rosbridge* protocol has been further developed and modified to incorporate the developed methods.

Next, in a study presented with 33 participants, we analyzed numerous measures which suggested that little difference was found amongst those users who were remote or co-resent with the robot for the given task. One interesting finding, however, was the difference in spread, or variance, with the remote users in areas such as manipulation. This is accredited to the lack of verification for certain types of hardware, such as a 3-button mouse which make 3D interaction much easier.

We make note that due to the small number of participants, we have not proved any sameness of difference, but instead shed light on the use of anonymous remote users for a full-length HRI study of this type. We also noted the trends and observations found with users of a teleoperation navigation and manipulation interface. The importance of an adequate tutorial and possible verification of steps within such a tutorial has also been outlined.

The Robot Management System is built in a robot, lab, and interface independent manner. Such a system is designed to help researchers quickly and efficiently conduct studies of their own and has been adapted by several research projects and individuals. This work paves the way to help the HRI community conduct low cost, large scale user studies across broader populations, and rapid algorithm development and testing, all of which were previously unfeasible.

In summary, we believe this work has shown that such a system can be appropriately used to conduct a certain subset of HRI studies that has previously been discussed. In particular, it is important that the study selected for crowdsourcing is not effected by gaining a significant advantage from having situational awareness in a co-present scenario, and the control given to the user, or interface, is the same as one that would be used in a co-present study.

10 Future Work

By knowing that the use of the Robot Management System can be used to conduct HRI studies, a vast array of future work can be explored. Such research and work encompasses not only robotics and HRI, but various other areas within Computer Science as well.

One invaluable implication of this work is the possibility of conducting novel work in the field of Learning from Demonstration (LfD) [2, 3]. LfD itself focuses on enabling human, non-expert, users to teach robots new capabilities using various interactive teaching methods as opposed to explicit programming. While significant progress has been made in the field, known limitations such as long training times and exploitation of domain knowledge and pre-programmed knowledge limit its practicality in its current state [30]. The use of RMS systems will allow researchers to quickly tune parameters and gain large training sets by keeping a large, consistent stream of users. Moreover, such advantages can be used throughout other aspects of HRI research.

In terms of security, even with the methods developed in this work, there is still room for a vast array of work to be done in this growing area. To begin, we look again at the case of non-native clients.

While this work provides solutions to authentication, another problem with the system itself is *authorization*. Many times, we do not want to allow certain remote clients access to the entire ROS system. This allows users to send direct commands to the robot which may bypass certain safety constraints. The schema developed in this work provides the ability to expand upon this idea.

Within the security token is a `level` field. This field is associated with the user level of the current client. This arbitrary string is determined by the external authenticator (allowing for a variety of different levels), but the developed authentication schema ensures the client did not tamper with this user level. Thus, an authorization node could be put in place which is configured as a list of ROS topic and service names and the associated user level. If a client attempts to publish or subscribe to a data stream that they are not authorized to access, the request could be ignored.

One additional area of future work involves incorporating the techniques developed in this work for native ROS clients. In this work, we assumed core ROS systems and clients could be wrapped in a VPN; however, as stated earlier, this may not always be possible or desired. The developed technique thus could be adapted into the core ROS system to authenticate any and all *native* ROS clients. The abstractness of the *rosauth* package also makes this a more feasible solution.

In addition to security, this work can be furthered by ongoing research in human-computer interaction, computer graphics, and interactive media and game development communities. Interface design and intuitive, efficient renderings of 3D interfaces are just a few of the areas that could be addressed. Furthermore, by collaborating with experts in such fields, it will be possible to add aspects of gamification [11] to such a system in order to incentivise long-term users.

Furthermore, we should note the importance of latency in such a system. While efforts can be made on the researcher's end to ensure a solid connection, there is no guarantee the client, or user, will also have a fast connection. For the purposes of this study, no attempt was made to filter out low bandwidth users and simply opened up the system to any willing user. This is an important point, but leaves open the possibility of interesting future work. In particular, measuring the latency across a large number of remote users could help to gain an understanding of the average delay one can expect for anonymous, remote users.

A final area of future work worth noting is the possibility of such a system in STEM (science, technology, engineering, and mathematics) fields. Robotics systems for use in education, in particular Computer Science education, are gaining momentum [4]. By providing an easy way to port ROS enabled robots to the web, educators can bring aspects of areas such as robotics or programming into lower-level classrooms.

References

[1] B. Alexander, K. Hsiao, C. Jenkins, B. Suay, and R. Toris. Robot web tools [ROS topics]. *Robotics Automation Magazine, IEEE*, 19(4):20 –23, December 2012.

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

[3] E. Billing and T. HellstrÃűm. A formalism for learning from demonstration. *Paladyn. Journal of Behavioral Robotics*, 1:1–13, 2010.

[4] Grant Braught. dLife: a Java library for multiplatform robotics, ai and vision in undergraduate cs and research. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, pages 33–38, New York, NY, USA, 2012. ACM.

[5] Cynthia Breazeal, Nick DePalma, Jeff Orkin, Sonia Chernova, and Malte Jung. Crowdsourcing human-robot interaction: New methods and system evaluation in a public environment. *Journal of Human-Robot Interaction*, 2(1):82–111, 2013.

[6] Maya Cakmak and Andrea L. Thomaz. Designing robot learners that ask good questions. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, HRI '12, pages 17–24, New York, NY, USA, 2012. ACM.

[7] S. Chernova, N. DePalma, E. Morant, and C. Breazeal. Crowdsourcing human-robot interaction: Application from virtual to physical worlds. In *IEEE International Symposium on Robot and Human Interactive Communication*, Ro-Man '11, July 2011.

[8] Matei Ciocarlie, Kaijen Hsiao, E. Gil Jones, Sachin Chitta, Radu Bogdan Rusu, and Ioan Alexandru Sucan. Towards reliable grasping and manipulation in household environments. In *International Symposium on Experimental Robotics (ISER)*, New Delhi, India, 12/2010 2010.

[9] C. Crick, G. Jay, S. Osentoski, and O.C. Jenkins. ROS and rosbridge: Roboticists out of the loop. In *th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 493 –494, March 2012.

[10] C. Crick, S. Osentoski, G. Jay, and O. Jenkins. Human and robot perception in large-scale learning from demonstration. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI 2011)*, 2011.

[11] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, pages 9–15, New York, NY, USA, 2011. ACM.

[12] D. Eastlake, 3rd and P. Jones. US secure hash algorithm 1 (SHA1), 2001.

[13] M. R. Endsley. Toward a theory of situation awareness in dynamic systems: Situation awareness. *Human factors*, 37(1):32–64, 1995.

[14] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000. AAI9980887.

[15] Ken Goldberg, editor. *The Robot in the Garden: Telerobotics and Telepistemology in the Age of the Internet*. MIT Press, Cambridge, MA, USA, 2001.

[16] David Gossow, Adam Leeper, Dave Hershberger, and Matei T. Ciocarlie. Interactive markers: 3-d user interfaces for ros applications [ROS topics]. *IEEE Robot. Automat. Mag.*, 18(4):14–15, 2011.

[17] Shay Gueron, Simon Johnson, and Jesse Walker. Sha-512/256. In *Proceedings of the 2011 Eighth International Conference on Information Technology: New Generations*, ITNG '11, pages 354–358, Washington, DC, USA, 2011. IEEE Computer Society.

[18] Peter H. Kahn, Jr., Takayuki Kanda, Hiroshi Ishiguro, Brian T. Gill, Jolina H. Ruckert, Solace Shen, Heather E. Gary, Aimee L. Reichert, Nathan G. Freier, and Rachel L.

Severson. Do people hold a humanoid robot morally accountable for the harm it causes? In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, HRI '12, pages 33–40, New York, NY, USA, 2012. ACM.

[19] Petar Kormushev, S. Calinon, and D. G. Caldwell. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25(5):581–603, 2011.

[20] Min Kyung Lee, Jodi Forlizzi, Sara Kiesler, Paul Rybski, John Antanitis, and Sarun Savetsila. Personalization in HRI: a longitudinal field experiment. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, HRI '12, pages 319–326, New York, NY, USA, 2012. ACM.

[21] Adam Eric Leeper, Kaijen Hsiao, Matei Ciocarlie, Leila Takayama, and David Gossow. Strategies for human-in-the-loop robotic grasping. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, HRI '12, pages 1–8, New York, NY, USA, 2012. ACM.

[22] Jarrod McClean, Christopher Stull, Charles Farrar, and David MascareÃśas. A preliminary cyber-physical security assessment of the robot operating system. In *Proceedings of the SPIE: Unmanned Systems Technology XV*, volume 8741, May 2013.

[23] Luis Yoichi Morales Saiki, Satoru Satake, Rajibul Huq, Dylan Glas, Takayuki Kanda, and Norihiro Hagita. How do people walk side-by-side?: using a computational model of human behavior for a social robot. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, HRI '12, pages 301–308, New York, NY, USA, 2012. ACM.

[24] S. Osentoski, G. Jay, C. Crick, B. Pitzer, C. DuHadway, and O.C. Jenkins. Robots as web services: Reproducible experimentation and application development using rosjs. In *Proccedings of the 2011 IEEE International Conference on Robotics & Automation*, 2011.

[25] Sarah Osentoski, Benjamin Pitzer, Christopher Crick, Graylin Jay, Shuonan Dong, Daniel H. Grollman, Halit Bener Suay, and Odest Chadwicke Jenkins. Remote robotic laboratories for learning from demonstration - enabling user interaction and shared experimentation. *International Journal of Social Robotics*, 4(4):449–461, 2012.

[26] Caroline Pantofaru, Leila Takayama, Tully Foote, and Bianca Soto. Exploring the role of robots in home organization. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, HRI '12, pages 327–334, New York, NY, USA, 2012. ACM.

[27] B. Pitzer, S. Osentoski, G. Jay, C. Crick, and O.C. Jenkins. PR2 remote lab: An environment for remote development and experimentation. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3200 –3205, May 2012.

[28] A. Sorokin, D. Berenson, S. Srinivasa, and M. Hebert. People helping robots helping people: Crowdsourcing for grasping novel objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '10)*, October 2010.

[29] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *Usenix Conference Proceedings*, pages 191–202, 1988.

[30] Halit Bener Suay, Russell Toris, and Sonia Chernova. A practical comparison of three robot learning from demonstration algorithm. *International Journal of Social Robotics*, 4(4):319–330, 2012.

[31] Ken Taylor and James Trevelyan. A Telerobot On The World Wide Web. In *1995 National Conference of the Australian Robot Association*, Melbourne, July 1995. Australian Robotics Association.

[32] S. Tellex, T. Kollar, S. Dickerson, M.R. Walter, A.G. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manip-

ulation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, August 2011.

[33] Russell Toris and Sonia Chernova. RobotsFor.Me and Robots For You. In *Interactive Machine Learning Workshop, Intelligent User Interfaces Conference*, pages 10–12, March 2013.

[34] Russell Toris, David Kent, and Sonia Chernova. The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing. *Journal of Human-Robot Interaction*, 2014 (To Appear).

[35] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full sha-1. In *Proceedings of the 25th annual international conference on Advances in Cryptology*, CRYPTO'05, pages 17–36, Berlin, Heidelberg, 2005. Springer-Verlag.

[36] Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.