January 2011

Mixed-Mode Control Interfaces of Mobile Robot Teams

Erik Karulf
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Computer Science and Engineering

Thesis Examination Committee:
Dr. William Smart, Chair
Dr. Caitlin Kelleher
Dr. Christopher Gill

MIXED-MODE CONTROL INTERFACES OF MOBILE ROBOT TEAMS

by

Erik Karulf

A thesis presented to the School of Engineering
of Washington University in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

May 2011
Saint Louis, Missouri

ABSTRACT OF THE THESIS

Mixed-Mode Control Interfaces

of Mobile Robot Teams

by

Erik Karulf

Master of Science in Computer Science

Washington University in St. Louis, 2011

Research Advisor: Assistant Professor William Smart

We propose applying various techniques and ideas, from video games, to the design of a task-level control interface for teams of mobile robots. The control interface, which visually resembles a video game and contains the functional elements of traditional robot control software, was designed and implemented using the ROS software framework. Unlike previous interfaces in the literature, our interface allows the user to control more than one robot and supports "sliding autonomy." This application allows the user to assign differing levels of independence to the robots dynamically, dependent on the situation. We present an overview of the implemented system, a discussion of its various elements, and identify results from some of our preliminary user studies.

Acknowledgments

This research was developed with support from the National Science Foundation, under awards OCI 0939637 and IIS 1037612, and from Willow Garage, Inc. I would like to specifically thank Leila Takayama, of Willow Garage, Inc., for her assistance with statistical analysis.

A very special thanks to Parker Dunton, Marshall Strother, Jim Stevens, Max Witt, Dan Lazewatsky, and David Lu for their help with the development of the RIDE interface.

Lastly, my deepest gratitude to my advisors advisors, Drs. Bill Smart, Caitlin Kelleher, and Chris Gill for their flexibility, support, and guidance.

Erik Karulf

Washington University in Saint Louis
May 2011

Dedicated to my parents,
Richard and Deborah Karulf,
for their love and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Steve Cousins, CEO of the robotics company *Willow Garage*, suggests, "in the next ten to twenty years our culture will experience a personal robotics revolution much like the personal computing revolution of the 1980's and 1990's" [4]. While we agree with Dr. Cousins, we believe the advancement of robotics in our society is limited by control interfaces. There is a growing need for robot control interfaces that enable a single user to effectively control more than one remote robot [11]. The increasing levels of autonomy demonstrated by our robots allow them to be controlled at progressively higher levels of abstraction. Such systems are not perfect; human intervention is frequently required when unanticipated circumstances develop, or an object beyond the capability of the perception systems must be evaluated [11].

The requirement for intermittent intervention suggests that our interfaces should be capable of both high-level (task-based) and low-level (direct teleoperation) control of remote robots, as well as the capacity to easily switch between these modes, as necessitated by the situation [11]. The goal of a single operator controlling many robots further indicates the need for a system which allows an individual to alert the operator when help is needed. If there are too many robots for the operator to attend to at once, there is a clear possibility a robot may sit idle, waiting for assistance, for an extended period of time, without such a notification system [11].

Many of the obstacles of controlling remote robots are similar to the challenges of controlling characters in video games. Real-time strategy games require controlling many tens, or hundreds, of heterogeneous units at once. First-and third-person games involve the detailed direct control of a single character. We believe that the interfaces

utilized for these games make ideal candidates for interfaces for mobile robot control [11].

Our motivation for drawing from video games is straightforward. Video games with easy-to-use, effective interfaces will be played more often, and they will sell better than games with poor interfaces. Almost four decades of market forces have refined interfaces for many genres of games; this has resulted in systems that are intuitive, easy to use, and effective. Furthermore, since many people play these games, they are already familiar with the interfaces. We hypothesize that these facts will make robot control interfaces based on computer game interfaces highly effective tools [11].

We present RIDE, the Robot Interactive Display Environment, a control interface for robots that draws heavily on computer game interfaces for inspiration. RIDE combines aspects from a number of computer game genres, allowing the operator to switch between direct and supervisory control, as the situation requires. Unlike existing interfaces in literature, RIDE allows both the user and the robot to negotiate the level of autonomy. RIDE achieves this negotiable level of autonomy by integrating several types of interfaces: a supervisory control mode and a direct control mode. The supervisory mode borrows display and control elements from real-time strategy games to allow task-oriented control of many robots. The direct mode borrows visual elements from racing games for a more fine-grained control of a single robot. Additional details of RIDE's design—including the transition between the two modes—is described in Section 3.1.

To evaluate the effectiveness of the RIDE interface, specifically the notification system, we conducted a formal user study. Our experiment asked subjects to perform a search task using two simulated robots in a small house. The experiment had two conditions: one with notifications displayed, and one with notifications hidden. We instrumented the application to record the user's behavior and timing information. The results of this data, along with the pre-experiment and post-experiment questionnaires, are presented in Section 4.2. In addition to discussing video game interfaces and presenting RIDE, we present the results of our initial user studies with the interface, which suggests it is well suited to search tasks with multiple robots.

Chapter 2

Human-Computer Interaction

Human-Computer Interaction (HCI) is the study of how humans interface with technology. This includes traditional interfaces, such as graphical user interfaces controlled with a mouse and keyboard, as well as alternative interfaces such as the accelerometers in the Nintendo Wii™ controller or the motion tracking system in the Microsoft Kinect™.

Despite the advances in efficacy and popularity of non-traditional devices, this paper will be limited to standard devices found on most computer systems, e.g., a monitor, mouse, and keyboard. In Section 5.2 we will briefly highlight how alternative interfaces would enhance our proposed interface.

2.1 Video Game Interfaces

Originally created as an application of computer graphics for use in the entertainment industry, video game development has matured into an independent industry in its own right. The popularity of video games has grown significantly throughout the past decade. The Entertainment Software Association estimates 68% of American households now play computer or video games [19]. This large demographic represents a pool of users already versed in exploring 3D virtual worlds interactively. In these interactions, users are not only perceiving the virtual world through simulated senses, but users are also performing actions and controlling their presence in the virtual world through input devices, eg., a mouse and keyboard.

2.1.1 Video Games in Research

Due to their common ancestry with human-computer interaction, many of the principles of video game design can be applied to the field of human-robot interaction (HRI), which will be discussed further in Section 2.2. There are several published applications of video games within the fields of artificial intelligence and machine learning. We will explore two applications that leverage human resources to complement artificial intelligence.

Von Ahn concluded that video games can be an effective tool for generating training data sets for machine learning [22]. Von Ahn developed special video games he labeled "games with a purpose." Such games provide a means to solve problems often considered trivial for humans, yet simultaneously considered quite challenging for computers. The *ESP Game*, Von Ahn's first game, uses human computation to classify images that computer algorithms are unable to recognize. In these games, the interaction of the human and the computer has changed to include the human in the computation of data. This is an interesting inversion of traditional autonomy, where the human is given tasks by the computer's artificial intelligence [22].

Grollman, in his dissertation work, applied a video game interface with HRI principles to capture training data for robotic user interfaces [6]. His application, RGame, recorded user's actions as they controlled a robotic dog and taught it to play soccer. Grollman created an AI model for shooting and defending tasks, which utilized the aggregate of data collected from RGame. Similar to Von Ahn's research, the RGame interface allowed the computer request information from the user to help it solve a problem [6].

2.1.2 First-Person & Third-Person Games

First-person games involve the direct control of a single character by the player. The game world is rendered from the viewpoint of that character, often with additional game information overlaid upon it.

The majority of the interface is occupied with a rendering of the world from the character's perspective. The player can only see what the character can see, and the viewpoint is fixed. Additional game information is overlaid in the manner of a heads-up display.

The challenge of first-person games is to build a situational awareness of the unseen elements of the surrounding environment. While this limited perspective often leads to exciting and engaging gameplay, it is a design choice that explicitly limits situational awareness.

Third-person games are similar, except the viewpoint is typically located above and behind the character. The viewport in third-person games follows the character as she moves through the world. Third-person games allow the player to see the character in the world and yields a greater sense of context. However, the viewpoint is still fixed, forcing the player to reconstruct unseen parts of the world in the player's mind.

Figure 2.1 shows the user interface for *Halo 3*, a recent first-person game [3]. The bottom left shows a local map, with the locations of other players and adversaries, while character and weapon status is displayed along the top.

2.1.3 Real-Time Strategy Games

Real-time strategy (RTS) games focus on resource management and combat between a small number of players (human and computer) where each controls large numbers of heterogeneous units. These units represent troops, vehicles, buildings, and similar resources.

Players give orders to their units at the task level, instructing them to, for example, engage in combat, harvest some natural resource, or move to a particular location. Units then carry out these orders autonomously, reporting back when they are done or when some unforeseen circumstances are encountered. Orders are typically given using a graphical interface, where displayed units can be selected and then assigned a task.

Figure 2.1: Bungie's *Halo 3*: A first-person interface

Figure 2.2 shows the user interface from Microsoft's *Age of Empires III* RTS, a representative and extremely popular example of the genre [14]. The interface is dominated by the main display, which shows an iconic, isometric view of the world. The viewpoint of the display can be moved to show different parts of the game world. The display shows different types of units (cannons, cavalry, and infantry), terrain (water, trees, and plains), buildings (gates, farms, and roads), and game information (selected units, waypoints). The player selects units, individually or in groups, with mouse and keyboard commands. Once selected, these units can be assigned a specific task from the task list.

The task list appears in the bottom right of the interface and displays icons for all available tasks, based on the currently-selected units. Some tasks, such as movement, require more information, which is provided by clicking on the main display with the mouse.

Figure 2.2: Microsofts's *Age of Empires III*: A real-time strategy interface

Information about the currently-selected units is displayed to the left of the task list. In the figure, a single heavy cannon unit has been selected, and information on its allegiance, player name, health, weaponry, and armor is shown.

A map of the entire game world, called the minimap, is displayed in the bottom left corner of the interface. This map shows the locations of all units, colored according to the controlling player, along with basic terrain type. Overlaid on the map is a representation of the field-of-view in the main display (white rectangle in Figure 2.2). In Figure 2.2, only the areas previously explored by the player are displayed in the minimap.

Clicking anywhere on the displayed part of the minimap moves the viewpoint in the main display to focus at that point. This option provides a quick way of moving the main display viewpoint around the (potentially large) virtual world. The minimap also displays primitive notifications; when a unit needs attention, a colored dot representing the unit flashes. A button, to the side of the minimap, centers the viewpoint of the main display on a unit currently without an assigned task, if one exists.

The interface in Figure 2.2 highlights notifications in two locations. Large notifications, in the center of the screen, provide urgent information to the user about the current objective. The larger notification is displayed on the screen for a short time before it is automatically hidden. Smaller notifications on the left side of the screen provide less urgent, actionable information to the user. The smaller notifications persist until dismissed by the user. When clicked the smaller notifications focus the camera on location in this case, a new piece of artillery.

Finally, a menu bar stretches across the top of the display. This displays current status in the top right (current quantities of resources), and it gives access to less-frequently-used functions, such as help screens and configuration dialogs.

2.2 Human-Robot Interaction

Human-Robot Interaction (HRI) focuses on the interface between humans and robots. In their survey paper on HRI, Goodrich and Schultz introduce five attributes, shown in Figure 2.3, that define the interactions between humans and robots [5].

- Level of behavior and autonomy
- Nature of information exchange
- Structure of the team
- Adaptation, learning, and training of people and the robot
- Shape of the task

Figure 2.3: Five attributes of Human-Robot Interaction

Of the five attributes of HRI, we will focus on autonomy and information exchange. Autonomy is defined as the extent of actions a robot may take independently. "Sliding autonomy," the idea that autonomy may vary depending on context, is introduced in further detail in Section 2.2.1. Information exchange describes the data that flows between a robot and a human. Like sliding autonomy, we introduce the idea that

information exchange may vary based on context. Section 2.2.2 introduces our refinements to the existing work in this field [5].

2.2.1 Autonomy

The relationship between a robot and a human, as provided through the interface, can be viewed or conceptualized along the lines of a continuum or a scale of autonomy. Teleoperation represents one end of the spectrum, where the robot exhibits no qualities of autonomous behavior. On the other hand, a fully autonomous robot, acting independent of human input, represents the other extreme of autonomy. A simple scale of autonomy can be found in Figure 2.4 [5].

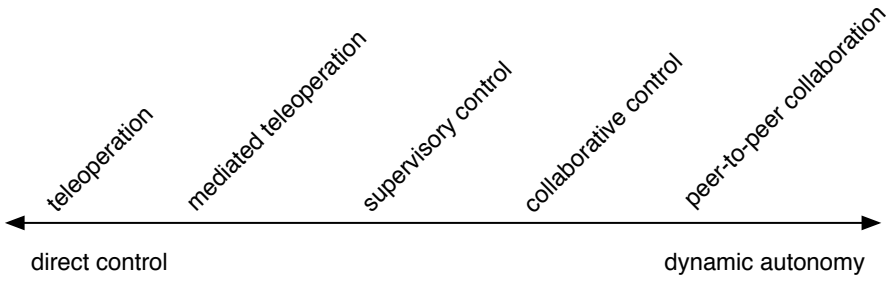


Figure 2.4: Scale of Robot Autonomy

Robots, as they presently exist, have a fairly limited set of autonomous actions available. As a result, most current interfaces are artificially conservative in their autonomy due to limitations in artificial intelligence. As research into artificial intelligence and machine learning grows, a much more diverse set of autonomous behaviors will inevitably become available. As user interface designers, we are tasked to create extensible interfaces to support future behaviors.

The idea of sliding autonomy allows a human, and a robot, to change the level of autonomy as needed. For illustration purposes, consider a robot that explores a warehouse. If the robot planned a poor navigation path, a user could manually specify the waypoints of an optimal path. If the robot is unable to complete the specified path due to an obstruction in the path, the robot could ask the user to directly teleoperate around the obstacle. Once free of the obstruction, the user would

input a new set of goal coordinates and allow the autonomous navigation system to resume control. When combined with machine learning, sliding autonomy provides the robot an excellent platform for interactive learning from demonstration. The idea of interactive learning from demonstration is already well established [22] [6].

2.2.2 Information Exchange

The nature of information exchange defines the flow of data between the human and the robot. What information is provided, how it is represented, and when it is communicated are all properties of the interface design. This encompasses low-level communication of navigation and sensor data as well as high-level commands sent from the human [5].

The purpose of visualization interfaces is to represent the one-way exchange of information from the robot to humans. This data typically includes location information, laser or sonar readings, battery readings, accelerometer readings, and map data. Original interfaces were written to be accessible programatically. These interfaces were slowly adapted to display sensor data graphically, but the resulting interfaces were disjointed and required operator training. Examples of these 2D user interfaces can be seen in Figure 2.5 [1].

In 2007, Nielsen introduced an ecological user interface with the goal of combining sensor data into a single, integrated display. Nielsen designed his interface for effective teleoperation, the technique of controlling a robot remotely. In his study, Nielsen compared an integrated 3D user interface to a simple 2D user interface for several environment-searching tasks. The 3D user interface displayed combined sensor data through a single viewport, while the 2D user interface displayed the same data through individual elements. The study found the 3D interface decreased collisions and decreased task completion time [16].

Nielsen attributes the success of the 3D interface to improved situational awareness. The observed effect of situational awareness and context on interface effectiveness is congruous with the findings of other researchers [16].

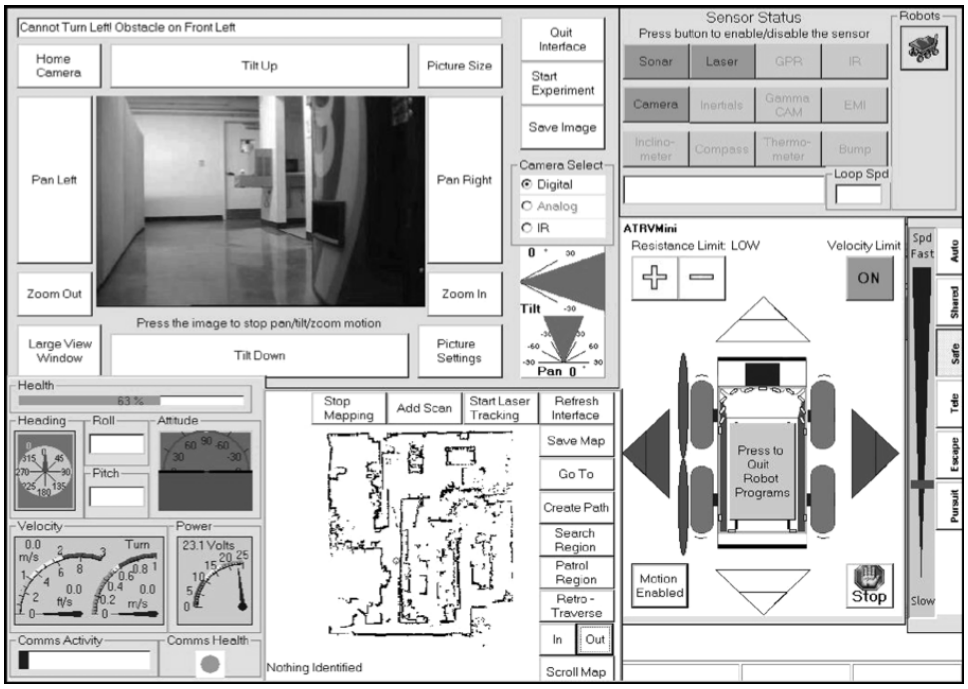


Figure 2.5: Existing Human-Robot Interface by Bruemmer et al

A limitation of Nielsen's interface is that its focus on teleoperation leaves the robot without autonomy. While teleoperation may be optimal for single robot environments it does not allow for concurrent robot control. This limitation requires a human operator for every active robot. We believe that providing an ecological interface common to multiple robots allows for more efficient use of human resources.

Chapter 3

Implementation

We present RIDE, the Robot Interactive Display Environment, as an example of a multi-mode interface. While a number of control interfaces either similar to, or draw direct inspiration from, RTS games [2] [10] [13] [21] or from first or third-person games [2] [9] [12], we are not aware of any systems which provide as many RTS interface features as RIDE. We are also not aware of any existing interfaces that allow the user to move between task-level and direct control in the way that RIDE allows. We should also note that the work presented in this paper builds directly on a previous prototype version of RIDE [20]. We discuss the user interface in full detail in Section 3.1.

We use a robotics middleware package, ROS, to abstract details of robotics software from the RIDE user interface. We briefly introduce key concepts of ROS, the Robot Operating System, in Section 3.2. In Section 3.3, we describe the system level architecture of RIDE. The source code for RIDE is freely available at the Washington University ROS repository, `http://wu-ros-pkg.sourceforge.net`.

3.1 User Interface

RIDE provides multiple interface modes, each specialized for different levels of autonomy. In our design phase, we decided to separate the user interface into two segments: the supervisor mode and the direct mode. The "supervisor mode" relies on the robot to perform actions autonomously and report the information back to the human, as appropriate. The "direct mode" allows the user to directly teleoperate a robot. By

including both control styles, the user is given flexibility to use the most appropriate control mode for a given situation.

In addition to switching the control schemes between modes, we also changed the fidelity of information exchanged. We found that by providing streaming video for each robot, it was not only taxing on the network bandwidth, but also wearing on the human's perception. The movement from the video would draw the users eye to several places on the screen making effective management difficult. To combat this problem, we reduced the number of available sensors in the supervisor mode. When a user wished to inspect a region with higher fidelity, he could switch into the direct mode and enable all sensors for the current robot.

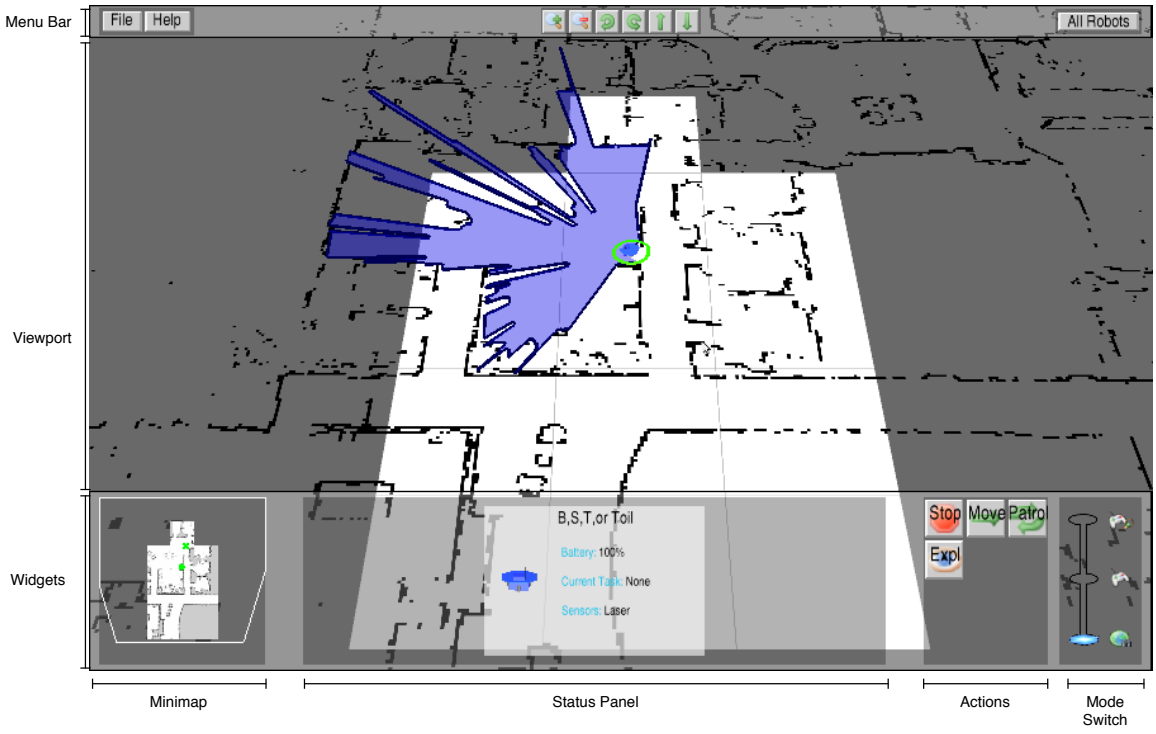


Figure 3.1: RIDE User Interface: Supervisory Mode

The screenshot figure in Figure 3.1 shows the RIDE supervisory control mode with annotated descriptions of the user interface elements. The viewport is the main portion of the display. The viewport changes camera perspective based on the control mode; it is either locked behind the robot in the direct control mode or user operated

in the supervisory interface. We took care to add alpha transparency to all user interface elements to prevent obstructing the viewport.

The menu bar, at the top of the screen, remains static between the two control modes. The various menus allow for customization of the user-interface on a global and per-robot basis. These settings can be saved, and restored in future sessions. There are also menu bar buttons to control the camera viewpoint and zoom level, toggle the sensor information displayed, and to list all currently known robots.

The widget panel is displayed at the bottom of the screen. The widget panel always contains a left, center, and right widget. A widget to change the display mode is always locked to the far right of the widget bar. Table 3.1 describes the widgets displayed based on the state of the interface.

Mode	**Left Widget**	**Center Widget**	**Right Widget**
Direct Control	minimap	Information Panel	Proximity Panel
Supervisory Control (Single Robot)	minimap	Status Panel	Action Panel
Supervisory Control (Multiple Robots)	minimap	Group Panel	Action Panel

Table 3.1: RIDE User Interface Widgets

3.1.1 Supervisory Control

As mentioned in Section 2.1.3, RTS games employ a top-down view of the world to control many units. Real time strategy games exemplify the qualities we desire for a supervisory interface. Real-time strategy games support the task-based control of heterogenous unit types; each unit type has separate abilities, strengths, and weaknesses. We wanted a way to visualize this information in the user interface without cluttering the main viewport during normal use. We developed two user interface widgets to represent the abilities, strengths, and weaknesses of each robot visually.

The first element, known as the "information panel," can be seen in Figure 3.1. When a robot is selected, the information panel shows a listing of the robot's name, type,

battery status, current task, and a list of sensors. An iconic rendering of the robot is displayed to visually represent the model of the robot.

The second panel developed for the supervisory mode was the "action panel." The action panel displays a list of buttons representing available actions for the selected robot. This list of actions is populated from what actions the robot self-reports it can perform. The implementation details of action auto-discovery is described in Section 3.3. Clicking on a button prompts for any additional arguments, such as asking the user to click on a position; it then sends the task instruction to the robot. In our development, we found that visualizing a confirmation helped improve clarity of what instructions were given to the robot. As an example, when the user instructs a robot to move to a specific destination, the user interface will blink the target icon, twice, on the destination.

Grouping Robot Teams

In our study of RTS games, we found that players organized units into heterogenous groups such that the individual units complemented each other's strengths and weaknesses. This required an interface that allowed a user to select multiple robots, at the same time, and then control the group effectively.

When a user wished to select a group of robots, instead of a single left click, the user left click and dragged a rectangle around the robots she wished to select. Much like single unit selection, a detailed information panel displays relevant data about the robot. However, instead of showing detailed information on a single robot, the "information panel" will be updated to show basic information on the group as a whole. The information panel displays a 3D mockup for each robot currently selected and the name of the robot. The user may move the mouse on top of the 3D mockup to obtain additional information about the robot.

In addition to the information panel, the "action panel" also has a slight change in behavior. The action panel selects the intersection of a set of actions available to each robot. RTS games provided a base set of actions that all controllable units must be able to perform: "Stop" and "Move." We require the same two actions be defined for all robots in RIDE. The advantage of requiring the move command for all robots is

that we could utilize a shortcut found in most RTS games. When a robot is selected – instead of clicking on the move button and then clicking on a destination, the user need only right click on the destination location.

3.1.2 Direct Control

The direct control mode sets up a high fidelity of information exchanged between the human and the robot as the user teleoperates the robot. Due to the large amount of prior work in the field of interfaces for teleoperation, we searched for video game design elements that would complement the existing interfaces. While there are several styles of video games that we could draw from for inspiration, we decided to base our direct control mode on "open environment games."

While not a genre in its own right, open environment games, or sandbox games, define a style of gameplay. Rather than the story following a predetermined linear path, the player explores a world where the story unfolds around her. For some, this style of gameplay is heralded as the future of games. Setting aside the narrative aspect of open environment games, the design elements that they employ have direct applicability to robot interfaces.

The interfaces of open environment games are usually fairly minimal; the preference is to leave most of the user interface available for camera display. The few user interface elements that do exist are typically semi-transparent, which allows the user to see through UI components to the map. We applied this principle to all user interface components in RIDE by applying a small amount of alpha transparency. You can see an example of the transparency looking at the floor behind the user interface elements, at the bottom of the screen, in Figure 3.1.

In most open environment games, a third person camera follows behind the player's avatar. While controls exist that allow the user to manually rotate the camera, most gameplay occurs with default camera positioning. This camera control style is the inspiration behind our third-person direct control mode. The user uses the arrow keys, on the keyboard, to move the robot in a given direction and uses the mouse to rotate the camera as needed.

A common component of most open environment games is a minimap which allows users to orient themselves, with relation to the "world." In video games, important characters, landmarks, and other notable locations may be displayed in the minimap. We added a minimap to the lower right hand corner of the RIDE user interface. The minimap is a component common to both the direct and supervisory control interfaces. We feel strongly that the minimap's location on screen should remain in the same location for both modes.

3.1.3 Tasks and Notifications

We adapted several elements from Nielsen's ecological user interface in our design of RIDE. We arranged the user interface to display inside a single viewport, to avoid splitting the user's attention. We also procedurally constructed a 3D world using the map and sensor data. This rich user experience allows to the user visually perceive the state of the world, as the robot perceives it. Finally, we introduced new concepts to improve upon the weaknesses of the existing interfaces: a task system and a notification system [16].

We introduced an asynchronous task system, which allows the human to directly command robots through a set of predetermined tasks. A robot may run only one single task at a time, and receipt of a new task replaces the previous task. The task system was written generically to allow each robot to provide actions appropriate to its hardware and software. Additional details on the implementation of the task system can be found in Section 3.1.1.

To complement the task system, a notification system was also introduced. The notification system enables the robot to communicate with the human. The design of the notification system enables the robot to select the importance of the message, and it permits the human to filter messages by importance. This gives the human operator the ability to ignore routine informational messages, while still receiving urgent messages. An example notification is shown in Figure 3.2. The example shows the robot self-reporting it has discovered a box, it's goal in this search and rescue scenario.

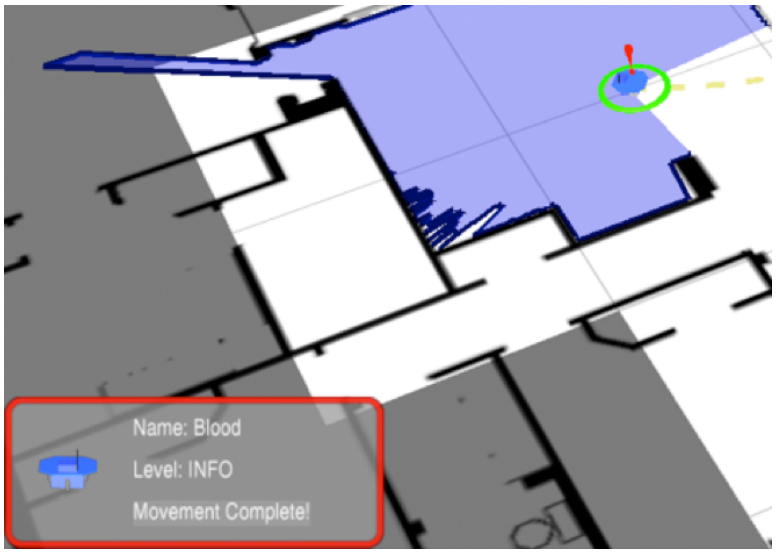


Figure 3.2: RIDE Notification Example

3.2 Robot Operating System

The RIDE interface uses the ROS robotics middleware framework to abstract common components of robotics software. ROS breaks a large robotic system into many discrete computational *nodes*. Because the nodes for a single robot operate as distinct processes, potentially across multiple platforms, ROS provides communication middleware. The communication between nodes is brokered through a special `roscore` node. This `roscore` node maintains a list of active nodes, topics, services, and configuration values [18]. We describe the publish/subscribe *topic* system and the synchronous *service* system in Section 3.2.1. Finally, we discuss `tf`, ROS's built-in kinematic tree in Section 3.2.2.

3.2.1 Topics and Services

ROS provides a framework where computational *nodes* communicate via *topics* and *services*. A topic is a named broadcast endpoint that supports multiple readers and writers. Nodes that write to a topic are known as "publishers." A publisher announces it is publishing to one or more topics to the `roscore` node. Nodes that wish to read

from a topic are known as "subscribers." While similar in design to the Common Object Request Broker Architecture (CORBA), the `roscore` node only provides a list of publishers to a subscriber node; the subscriber communicates directly with the publisher. In general, nodes either publish sensor data, transform and republish data, or control actuators from published data. A simple example of a ROS digraph can be seen in Figure 3.3. While topics are well suited for periodic broadcast messages, the "best effort" delivery make topics unsuitable for command data [18].

The *services* system within ROS addresses the need for "reliable" message delivery. A service is a named endpoint with one *service provider* and many *service clients*. Similar to topic subscribers and publishers, service providers and clients announce their intentions to the `roscore` node. Unlike broadcast topics, messages pass between the service provider and client through a unicast stream.

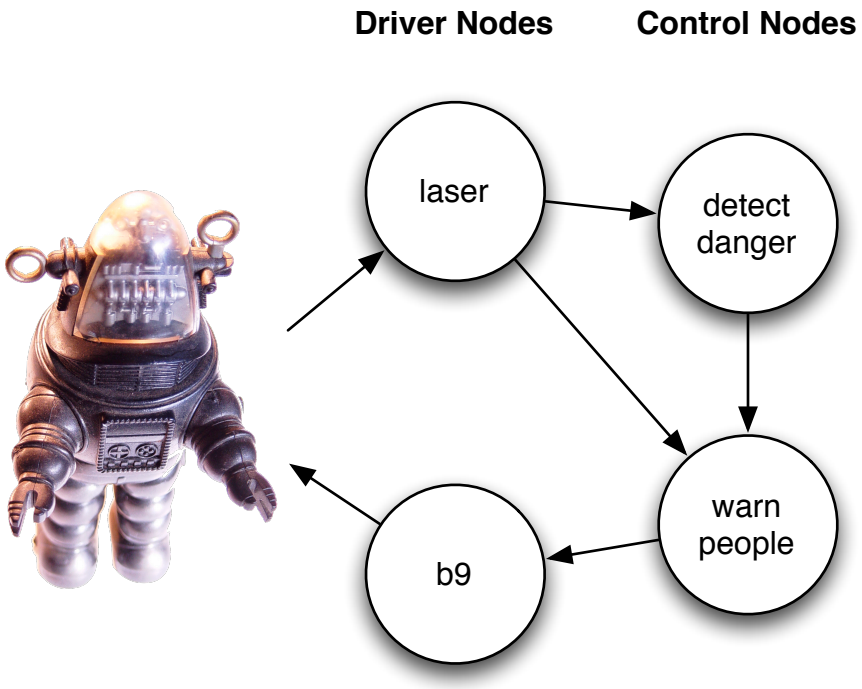


Figure 3.3: Example ROS nodes and topics

3.2.2 Kinematic Tree

A large part of the complexity of robotic systems is transforming coordinate frames in a timely manner. ROS offers a special topic located at `tf` and provides a cross-language library, also called "tf", to ease coordinate transformation tasks.

ROS builds a kinematic tree using data published to the `tf` topic. Coordinate frames contain a timestamp, an object's name, the parent coordinate frame and the quaternion offset between the two. The tf package allows a developer to request a quaternion offset between two coordinate frames.

In order to maintain a correct representation, the library automatically discards stale coordinate frame data. This requires objects to publish their location at a specified frequency. When all nodes are published at an acceptable rate the full kinematic tree is accessible. A sample kinematic tree for a robot with a stero-vision rig mounted on-top of a pan-tilt unit can be seen in Figure 3.4.

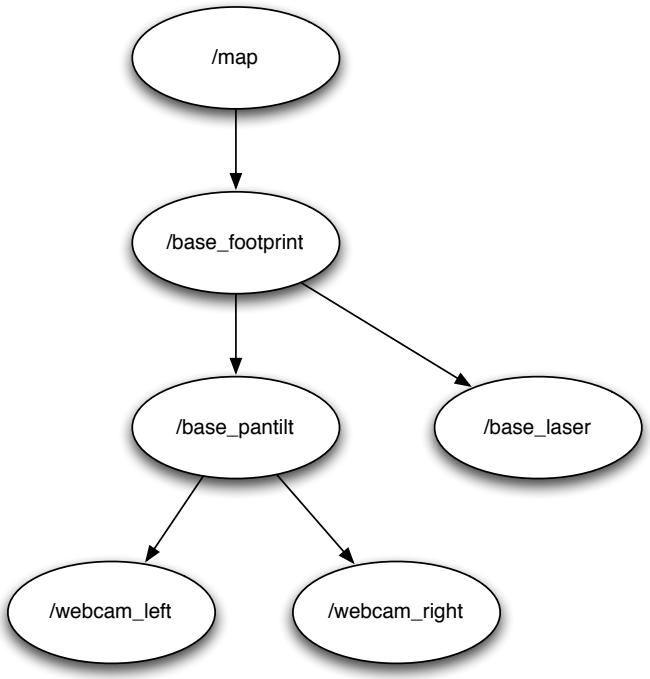


Figure 3.4: Sample Kinematic Tree

3.3 RIDE Architecture

Originally designed for a single robot system, ROS is not well suited to control multi-robot teams. This conflicts with RIDE's design to control large numbers of robots. We decided to maintain the ROS architecture by maintaining a `roscore` and set of nodes on each robot. We define a collection `roscore` and supporting nodes as a "realm." We developed a utility, `rosmultimaster`, to allow a single node to connect to multiple realms. This design allows robots to operate independently and gives RIDE the freedom to connect to robot realms as needed.

Each robot realm runs a RIDE node (`ride_agent` in figure 3.5), which publishes information about the robot, including the tasks that it can perform, the sensors that it has, diagnostic information, and notifications. The RIDE interface uses this information to subscribe to relevant sensor, diagnostic, and notification topics from the robot.

Figure 3.5 shows the set of ROS nodes for a simulated robot. All sensor information (`base_scan`, `odom`) originates from the simulator node (`state_bridge`). Similarly, all movement commands (`cmd_vel`) are consumed by `state_bridge`. On a real robot, there would be separate nodes for each of the sensors, and for the movement controller.

RIDE also establishes direct, service connections with the nodes responsible for performing the tasks that can be allocated to the robot. In Figure 3.5, `GoToNode` is responsible for movements tasks given to the robot. The `ride_agent` node is also capable of sending movement commands directly to the motion controller (or simulator), to stop the robot when needed.

When robots connect to the RIDE interface, the tasks that they can perform are recorded, and this information is used to populate the task list when the robots are subsequently selected. Currently, the set of tasks that a robot can perform is assumed to be static, and drawn from a fixed set of known tasks. This assumption allows us to assign an appropriate icon for each task, and to know the parameters that each task requires (which require additional modal interactions with the interface). Similarly, we assume that the sensors available to our robots are drawn from a known set. This

lets us be sure that we have appropriate visualizations for each possible sensor. We discuss plans to relax these assumptions in future work (see Section 5.2).

3.4 Panda3D

RIDE uses the popular 3D rendering engine Panda3D to display the virtual environment to the user. Like most rendering engines, Panda3D relies on a scene graph for rendering objects onto the screen. We dynamically constructed the 3D environment based on data from the mapping realm and each robot's representation of the world. Due to the common mapping realm, mentioned in Section 3.3, all robots share a common coordinate frame. This allows the RIDE UI to represent a one-to-one mapping of objects from each robot's kinematic tree to objects in Panda3D's scene graph. For example, the robot given in the Figure 3.4 would render a robot equipped with a laser rangefinder and a two web cams mounted to a pan-tilt unit.

Figure 3.5: Publish/Subscriber graph in RIDE Simulation Realm

Chapter 4

User Study

We conducted a formal user study to evaluate the effectiveness of the RIDE interface and to test the validity of our thesis, specifically the notification system. We designed an hour-long user study with the goal of testing the asynchronous task and notification system. Our experiment had two-conditions; in one condition the notification system was displayed, and in the other condition, the notification systems were not enabled. An example notification is shown in Figure 3.2.

The user study was advertised via flyers posted publicly on the Washington University Campus. Our user pool consisted of 22 adult participants, 6 female and 16 male ($M = 23.5, SD = 5.45$). The median age was 21.

Using two simulated robots in a small house, subjects were asked to perform a search task. Subjects were directed to locate three boxes in the house, as quickly as possible. The subjects were told the boxes were not shown on the map and that the boxes would be detected by the laser range-finder sensor. To confirm that the subjects had found the box, they were asked to point it out to the experimenter, once the box had been found. Subjects were informed they could use any of the features of the interface they wanted. In all runs of the experiment, the starting positions of the robots, and box locations, were the same. The simulated robots modeled Videre Design Erratic ERAs with a Hokuyo URG laser range finder.

Each subject was individually run through the user study. A user study session consisted of a pre-experiment questionnaire, a short set of practice sessions, two search and rescue experiments, and a post-experiment questionnaire. We created two

separate experiment conditions, one without notifications and one with notifications, and created a program to execute both in a random order during the user study.

4.1 Experimental Design

The primary goal of our user studies was to determine if the RIDE user interface was an effective tool for control groups of robots. Our secondary goal was to test the individual interface elements unique to RIDE and to determine if they enhanced the user experience. We examined prior work to find activities that a group of robots could effectively accomplish, if controlled properly, and would not require additional background experience of the subjects. We decided upon the "search and rescue" modality, as, generally, it is an easily understood activity that can be accomplished quickly, through effective coordination of robots.

As discussed, we designed a scenario where the robots were used to find boxes hidden throughout a house. The boxes would not be visible on the on-screen map directly, but they would display through the sensors on the robots. The house can be viewed, without boxes, in Figure 4.1.

The robots were configured to simulate Erratic ERA-MOBI robots. The simulation engine, as described in Section 3.3, was programmed to emulate a laser sensor and an odometry sensor. These two sensors allowed the robot to detect the boxes, walls, and other obstacles, through the laser readings or by picking up a stalled engine through the odometry sensor.

We designed three separate runs for a user study session: a training run, a run with notifications disabled, and a run with notifications enabled. We elected to assume that subjects would require a small amount of training before they were allowed to control robots. In order to meet this assumption, we provided each subject with the same five-minute practice run to ensure a common training set.

In the following subsections I will describe, in order, the components of a user study session. This research protocol was approved by the Washington University Institutional Review Board (IRB). The full documentation submitted to the IRB is documented in Appendix A.

4.1.1 Pre-Experiment

At the beginning of each user study, we obtained informed consent, from each subject, to participate in research. The informed consent paperwork may be found in Appendix A. After obtaining informed consent, subjects were asked to complete a short pre-experiment questionnaire, to record the subject's background information, such as demographic information, experience with computers, video games and identification of which video games the subject's played regularly, as well as the subject's robotic experience. Ultimately, the goal of the questionnaire was to obtain useful information for correlation in data analysis.

After completion of the questionnaire, subjects were introduced to the RIDE interface and given a short lecture on robotics. The target audience for RIDE required a basic understanding of the robots they were to be controlling. The lecture covered basic concepts including robot movement (quasi-holonomic vs. holonomic) and laser range finders.

After the scripted lecture on robotics, we asked the subjects to narrate their thought processes out loud, a practice called "speak aloud" within the HCI community. We asked subjects to play a round of Minesweeper, while speaking their thoughts out loud, to give them practice and become comfortable speaking out-loud. The pre-experiment preparation concluded, after approximately five minutes, when the researcher was confident in the subject's ability to describe her thought process.

4.1.2 Training Run

The training run took place inside the virtual house pictured in Figure 4.1. Unlike the experimental runs, there were no boxes present within the house during the training

run. Instead, the subject was asked to demonstrate basic control over the interface by accomplishing a series of tasks. Once the user completed the training activities, he/she was given an opportunity to continue using the user interface until the user was comfortable proceeding with the experiments.

4.1.3 RIDE UI Experiments

The user study program randomly chose either the notification or the non-notification experiment. In both experiments, the subjects were told of four boxes hidden within the house. We asked the subjects to find the boxes quickly. Once a user believed they had found the box, they were instructed to circle the box with their mouse pointer. This information was used in conjunction with other data in post-processing, to determine if the user correctly identified each box.

During the experiment, the researcher was unable to offer any advice or answer any questions. The researcher took hand-written notes on each experiment with an emphasis on recording information related to the user's thought process. The automated user study application is set to record keystrokes, mouse events, application state and the screen of the participant.

4.1.4 Post-Experiment

We asked participants a series of structured and semi-structured questions to conclude the user study. The post-experiment questionnaire is found in Appendix A.

We also conducted a brief semi-structured interview after the post-experiment questionnaire. These questions were based on remarkable events noted by the researcher during the experiments. The semi-structured interview results were not included in formal analysis of RIDE, rather they were aimed to provide insight for continued application development.

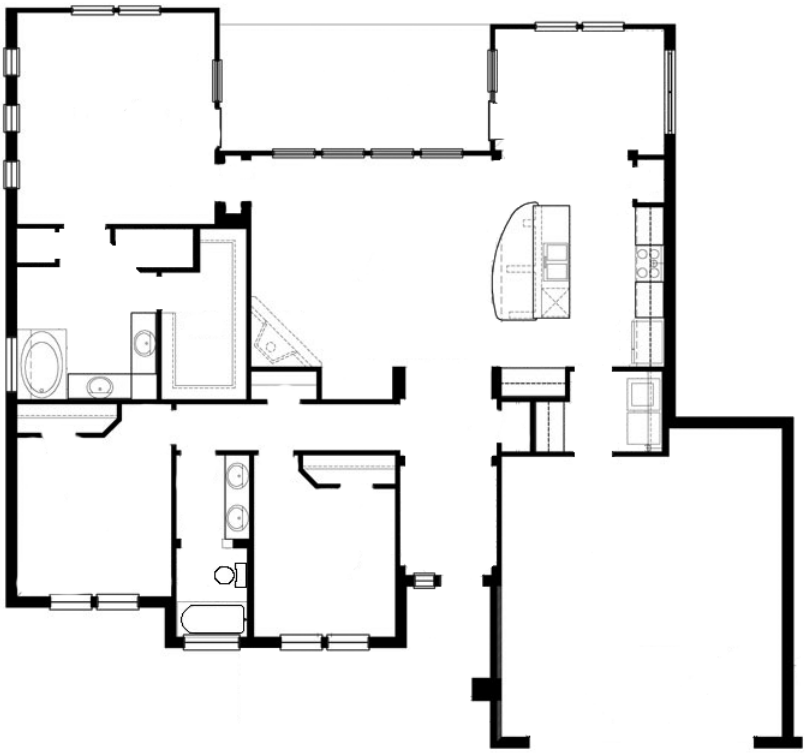


Figure 4.1: User Testing Environment

4.2 Results

In this section, we present the results of our user study. The subjective ratings in the pre-experiment and post-experiment questionnaires were measured using a 7-point Likert scale. Unless noted, in the following analysis we present the results of running a between-participants analysis of variance (ANOVA) using completion time, total neglect time or total idle time as the dependent variables. All times are presented in seconds. The total neglect time is the sum of the neglect times for each robot, and consequently, can be greater than the completion time for the task. Similarly, the total idle time is the sum of the idle times for the two robots. Tabular results report the mean, standard deviation, of times, the F-statistic from the ANOVA (for $F(1, 20)$), and the significance level.

4.2.1 Effects of Prior Experience

In general, we found that prior experience with video games or with controlling a robot affected a subject's performance on the search task. The single factor that showed the greatest influence on performance was video game use (Table 4.2.1). Subjects who did not regularly play video games took almost twice as long, on average, to complete the search task than subjects who did regularly play video games. An even more marked difference was seen in total neglect and idle times. Non-gamers had, on average, approximately three times longer neglect and idle times, compared to regular gamers.

	VG	**NVG**	F	p
Completion Time	244.67 (142.57)	550.43 (178.03)	18.798	< 0.001
Neglect Time	181.27 (160.54)	585.57 (218.85)	24.073	< 0.001
Idle Time	238.13 (190.06)	720.00 (319.07)	19.850	< 0.001

Table 4.1: Effects of regular video game use (VG) vs. no regular video game use (NVG).

Surprisingly, experience playing RTS games did not have a significant effect on completion time, total neglect time, or total idle time. However, first-person game experience was mildly significant for completion time (see Table 4.2.1). Subjects who

played first-person games completed the search task more quickly than those that did not play first-person games. Similar results, but with more significance, were seen for both total neglect time and total idle time.

	FP	**NFP**	F	p
Completion Time	264.82 (150.42)	419.09 (237.44)	3.3138	$= 0.0837$
Neglect Time	195.45 (170.41)	424.36 (291.50)	5.0555	< 0.05
Idle Time	255.09 (202.24)	527.81 (375.06)	4.5062	< 0.05

Table 4.2: Effects of regular first-person video game play (FP) vs. no regular first-person video game play (NFP).

Prior experience controlling a robot was also significant. Subjects with previous experience controlling a robot had lower completion times than subjects with no prior experience (Table 4.2.1). Similarly total neglect time and total idle time were both significantly lower for subjects with previous experience controlling a robot than for those without.

	RE	**NE**	F	p
Completion Time	236.33 (170.27)	415.08 (207.99)	4.5248	< 0.05
Neglect Time	163.33 (178.53)	411.38 (265.54)	5.9435	< 0.05
Idle Time	220.22 (222.86)	510.00 (339.22)	5.0228	< 0.05

Table 4.3: Effects of prior experience controlling a robot (RE) vs. no prior experience controlling a robot (NE).

Finally, the percentage of total time spent in supervisory mode was somewhat dependent on prior experience of first-person games. Subjects with prior experience of first person games spent more time in supervisory mode than the other two modes ($M = 89.86\%, SD = 15.20$) than those subjects with no experience of first-person games ($M = 72.60\%, SD = 23.28$), $F(1, 20) = 4.2348, p = 0.05287$.

4.2.2 Use of Supervisory Mode

Subjects spent, on average, more than 69% of their time in supervisory mode ($t = 2.4944, p < 0.01$), and less than 0.3% of their time in first-person mode ($t = -2.747, p <$

0.01). Completion time, total neglect time, and total idle time were significantly affected by the percentage of total time spent in supervisory mode. The median time percentage of time spent by subjects in supervisory was 94.19%. Subjects that spent more than this median percentage of time in supervisory mode completed the search task, on average, twice as fast as subjects who spent less than median time in supervisory mode (Table 4.2.2). The effects on total neglect time and total idle time were similar.

	GM	**LM**	F	p
Completion Time	197.17 (79.80)	515.70 (181.79)	30.119	< 0.001
Neglect Time	124.17 (89.09)	532.80 (218.64)	35.198	< 0.001
Idle Time	174.17 (111.90)	652.20 (305.51)	25.482	< 0.001

Table 4.4: Effects of spending greater than median time in supervisory mode (GM) vs. spending less than median time in supervisory mode (LM).

4.2.3 Effects of Notifications

Subjects rated the task as easier to perform with notifications ($M = 1.70, SD = 0.94$) than without notifications ($M = 2.67, SD = 0.98$), $F(1, 20) = 5.4319, p < 0.05$. Subjects also rated themselves as needing less help with notifications enabled ($M = 1.50, SD = 0.71$) than with notifications disabled ($M = 2.50, SD = 1.09$), $F(1, 20) = 6.2338, p < 0.05$.

Subjects' ratings of how easy it was to control the robot were also mildly affected by the use of notifications. Subjects rated the robot as easier to control with notifications enabled ($M = 1.80, SD = 0.92$) than with notifications disabled ($M = 2.58, SD = 0.90$), $F(1, 20) = 4.0528, p = 0.05775$.

Although notifications had no significant effect on completion time, total neglect time, or total idle time across all subjects, a significant effect was noticed for subjects without prior robot control experience. For subjects with no prior experience controlling robots, the time taken to complete the search task was significantly less with notifications enabled (Table 4.2.3). Total neglect time, and total idle time were similarly dramatically reduced in this group with notifications enabled.

	NE	**ND**	F	p
Completion Time	301.57 (149.13)	547.50 (195.07)	6.6401	< 0.05
Neglect Time	269.21 (204.21)	576.50 (241.72)	6.1615	< 0.05
Idle Time	319.86 (205.03)	731.83 (340.67)	7.2453	< 0.05

Table 4.5: Effects of enabling notifications (NE) vs. disabling notifications (ND).

Chapter 5

Conclusion

We propose applying techniques and design elements from video games to address weaknesses in existing human-robot interfaces. We submitted RIDE, an interface for controlling heterogeneous robot teams using design elements from video games. We introduced the concept of "sliding autonomy" to allow the human operator and the robot to vary the degree of robot autonomy dependent on the situation. We presented an overview of the implemented system, RIDE, and using the ROS middleware framework. Finally we present the results of preliminary user studies showing, among other results, the effectiveness of our notification system.

5.1 Discussion

The results presented in Section 4.2 show, unsurprisingly, that subjects who play video games or have prior experience controlling robots are better at performing the search task (ie. have lower completion, total neglect, and total idle times) with the RIDE interface than those who do not regularly play video games. While experience with RTS games seems to have no significant effect on the how well subjects perform the task, experience with first-person games clearly demonstrates an effect, decreasing all three times. First-person game experience caused subjects to use the first-person interface less than subjects who did not play first-person games, suggesting experienced first-person gamers recognize limitations of the interface.

All subjects, regardless of experience, showed a preference towards the supervisory mode interface spending over 69% of their time, on average, in this mode. The

first-person mode was only used by a single subject during non-training runs. This leads us to believe limitations in our current implementation of the first-person are preventing adoption. Currently, the map is rendered as a texture on the ground plane in all modes, making it hard to see and correctly interpret at acute viewing angles. We hypothesize extruding the map into 3D space, effectively creating "walls" coming out of the floor, would provide a first-person user a much better sense of positional awareness with respect to the map. The idea of extruding the map to create 3D walls is similar to an interface by Bruemmer et al [2].

The percentage of time spent in supervisory mode dramatically affects the times, with higher percentages correlating with much faster completion, and much reduced neglect and idle times. This suggests that an RTS interface is more appropriate than first- or third-person interfaces for this type of search task.

Notifications have a similarly beneficial effect, making the task subjectively easier to perform, and causing the subjects to ask for help less frequently. Surprisingly, across our whole population, notifications did not have significant effect on completion time, total neglect time, or total idle time.

However, the use of notifications did have a significant effect on subjects with no prior robot control experience, lowering completion time, neglect time, and idle time dramatically. We attribute this difference in effect to a flaw in our experimental design. We believe that the two-robot search task was not taxing enough for those who had controlled robots before. We hypothesize that these advanced subjects were able to manually monitor both robots at the same time, actively anticipating failures, making the notification system redundant. More inexperienced users, on the other hand, might not notice a robot in need of help when attending to the other robot, leading to increased total neglect and idle times. In this situation, notifications serve a critical function.

In conclusion, we believe that mixed-mode video-game-based interfaces, such as RIDE, offer an effective way to control large numbers of mostly-autonomous mobile robots. Our initial studies suggest that an active notification system allows inexperienced users to more efficiently control teams of robots, reducing neglect and idle times. We believe that this result will also apply to more experienced users in situations where they are sufficiently overloaded, or where unexpected events happen frequently.

5.2 Future Work

Observing the user interface usage, during the user studies, revealed several weak points in our implementation. We believe that small enhancements to RIDE will greatly benefit the end user experience.

In Section 2.2.2 we introduced Nielsen's prior work designing ecological interfaces for robotics. While the paper was a large influence on our design work, we feel our interface would benefit from additional visual cues. In their proposed UI, Nielsen extruded the known elements from the coverage map vertically to create an approximate three dimensional representation [16]. In our original design, we rejected this design element as we believed it misrepresented the third dimension to the user. However, during user studies, the users often mentioned it was difficult to position the camera for teleoperation while keeping the map visible.

During our user studies we also discovered that many users, especially users without prior robotics experience, found the robots to behave in a counterintuitive manner. As an example, the path planning algorithm we used in our study is weighted in a way that often chooses a longer, more circuitous route to avoid obstacles, than a more direct route that passes near obstacles. We believe that exposing more behavioral information will reduce the cognitive load on users. Future versions of the RIDE application would benefit from adapting the prior work in visualizing autonomous navigation systems to the field of robotics.

5.3 Future User Studies

While successful, the design of the preliminary user study introduced several mitigating factors that prevented a definitive acceptance of our hypothesis. We feel our hypothesis can be clearly demonstrated with a revised experimental design. We identify and propose solutions to address several major problems with our experimental design.

In our preliminary user study, our intent was to provide a common set of knowledge in terms of robotics and video game experience. Unfortunately, by providing this

training we made the analysis between experience groups much more difficult. As a result, we have widened our target audience to include users without robotics experience. In future studies, we plan to limit our training to include only speak-aloud practice to avoid introducing any potential biases.

In addition to the problems in our pre-experiment preparation, we have identified several ways to improve upon the experimental design itself. The purpose of our experiments was to prove applying video game principles to HRI interfaces will improve usability. In order to prove our hypothesis, we used notification display as an independent variable and recorded completion time as a measure of usability. While this is a valid approach, we would like to re-focus our experiment to directly test our mixed-mode interface by measuring cognitive load.

In order to evaluate the sliding autonomy approach, we will run a randomized three-condition within subject experiment. Instead of using notifications we will use the control modes as our independent variable. This gives us three conditions to test: direct only control, supervisory only control, and blended control. This would allow us to directly compare the given modes and record metrics on the blended control interfaces, such as time spent in each mode and the number of mode switches.

As mentioned above, while we can correlate successful task completion with improved usability, we would like to utilize a more direct metric. Researchers at the United States National Aeronautics and Space Administration (NASA) developed a metric for describing a human's workload while accomplishing a task. The metric, known as the "Task Load Index," is computed from a user's perceived workload on six sub-scales: mental demand, physical demand, temporal demand, performance, effort, and frustration [8]. Our revised user study would ask subjects to rate their workload on each of the Task Load Index's six sub-scales after each experiment. At the conclusion of the user study, we would ask users to answer the fifteen questions to determine the weighting of the sub-scales when computing the task load index [7].

Given the context of testing the control modes using the NASA Task Load Index, we decided to modify the subjects' tasks to more accurately represent our use cases. These new - more challenging - tasks would require a larger simulated world, more robots, and a different scenario. A weakness in the preliminary user study's scenario, is that the robots were able to complete their task immediately. This model does not

match a use case in the real world, where most actions will require a non-zero amount of time to complete. Such a study, we believe, will show the benefits of mixed-mode interfaces across the whole population.

Appendix A

Institutional Review Board Documentation

Included Documentation:

- Consent for Participation in Research Activities (2 pgs)
- Pre-Experiment Questionnaire (1 pg)
- Post-Experiment Questionnaire (2 pgs)

Washington University in St. Louis

CONSENT FOR PARTICIPATION IN RESEARCH ACTIVITIES

Title of Project:	**Game-Based Robot Control Interfaces**	HRPO Approval Number:	

1. a) You are invited to participate in a research study conducted by Professor William Smart. The overall purpose of this research is evaluate control interfaces for robots that are in a remote location, and to determine how easy it is to get them to perform a set of tasks. During this study you will be asked to drive a mobile robot that is either located in Waterville, Maine, or is in a computer simulation. You will be asked to perform a series of simple tasks with the robot, such as finding a particular object, or driving to a specified location. You will control the robot using an interface that is similar to that of a real-time strategy computer game.

   b) The amount of time required for your participation will be approximately one hour, during which you can take short breaks if you want to. You will receive $10 for your time.

2. No known risks are associated with this research other than the potential for mild boredom or fatigue.

3. There are no direct benefits associated with your participation in this research other than the potential enhancement of scientific knowledge.

4. Your participation is entirely voluntary and you may choose not to participate in this study or withdraw your consent at any time. You will not be penalized in any way should you choose not to participate or withdraw.

5. Sometimes there are alternatives to participating in research. Certain studies, such as those that involve a therapy or intervention, are examples of when alternatives might be available. Because this study does not involve an intervention or treatment of any kind, no alternatives are offered.

6. We will do everything we can to protect your privacy. As part of this effort, your identity will not be revealed in any publication that may result from this study. In rare instances, a researcher's study must undergo an audit or program evaluation by Washington University or an external oversight agency (such as the Office for Human Research Protection). This may result in the disclosure of your data as well as any other information collected by the researcher. If this were to occur, such information would only be used to determine whether the researcher conducted this study properly and adequately protected your rights as a human participant. Importantly, any and all audits would maintain the confidentiality of any information reviewed by their office(s).

7. a) If you have any questions or concerns regarding this study, please contact Professor Smart, at (314) 935-4749.

b) If you wish to talk with someone else because you have questions about your rights as a research participant or because you feel that you have been harmed in any way by your participation in this research, please call Dr. Philip Ludbrook, Executive Chair of Washington University's Human Research Protection Office, at 3l4-633-7400 or 1-800- 438-0445.

I have read this consent form and have been given a chance to ask questions. I agree to participate in the research study described above titled "Game-Based Robot Control Interfaces". I will receive a signed copy of this form for my records.

_____ _____
Signature of Research Participant Date

_____
Printed Name of Research Participant

_____ _____
Signature of person obtaining consent Date

_____
Printed Name of person obtaining consent

This form is valid only if the Human Research Protection Office's current stamp of approval is shown below.

Subject ID: _____

Game-Based Robot Control Interfaces

Pre-Experiment Questionnaire

Age: Gender: Major/Occupation:

How many hours per week do you spend working on a computer?
__ None __ 1 to 5 __ 6 to 10 __ 11 to 15 __ 16 to 20 __ more than 20

How many hours per week do you spend playing computer or video games?
__ None __ 1 to 5 __ 6 to 10 __ 11 to 15 __ 16 to 20 __ more than 20

What type for games do you play regularly?
__ First-person shooter (eg Halo)
__ Adventure (eg King's Quest)
__ MMORPG (eg World of Warcraft)
__ Strategy (eg Age of Empires)
__ Music (eg Guitar Hero)
__ Card games (eg solitaire)
__ Virtual presence (eg Second Life)
__ Platform games (eg Super Mario)
__ Role-playing (eg Diablo)
__ Sports (eg Madden NFL)
__ Simulation (eg The Sims)
__ Arcade (eg Pacman)
__ Board games (eg chess)
__ Other (please specify)

List the three games that you play the most:

What systems do you regularly play games on?
__ Playstation __ Xbox __ Ninendo Wii
__ Nintendo (non-Wii) __ Computer (PC/Mac) __ Other (please specify)

Do you regularly play computer of video games with other people?
__ No, I usually play single-player games
__ Yes, I usually play with a group of people who are in the same room
__ Yes, I usually play with a group of people over the Internet

Do you have a driver's license? __ Yes __ No

Have you ever explored a virtual environment using a computer?
__ No
__ Yes (please describe briefly, eg "Second Life", or "architectural walk-through")

Have you ever controlled, operated, or driven a real robot?
__ No
__ Yes (please describe briefly)

Subject ID: _____

Game-Based Robot Control Interfaces

Post-Experiment Questionnaire

How easy was it to control the robot?

1	2	3	4	5	6	7
very easy	easy	somewhat easy	neutral	somewhat difficult	difficult	very difficult

Did the robot behave in the way that you expected it to?

1	2	3	4	5	6	7
exactly as expected	mostly as expected	somewhat as expected	neutral	somewhat unexpectedly	mostly unexpectedly	completely unexpectedly

How easy was it to perform the tasks specified in the experiment?

1	2	3	4	5	6	7
very easy	easy	somewhat easy	neutral	somewhat difficult	difficult	very difficult

Was the robot an effective tool for performing the experiment?

1	2	3	4	5	6	7
very effective	effective	somewhat effective	neutral	somewhat ineffective	ineffective	very ineffective

How does using a robot to perform the experiment compare to performing it yourself, in person?

1	2	3	4	5	6	7
much easier	easier	somewhat easier	about the same	somewhat more difficult	more difficult	much more difficult

Was the interface easy to use?

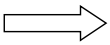
1	2	3	4	5	6	7
very easy	easy	somewhat easy	neutral	somewhat difficult	difficult	very difficult

Was the interface intuitive to use?

1	2	3	4	5	6	7
very intuitive	intuitive	somewhat intuitive	neutral	somewhat unintuitive	unintuitive	very unintuitive

Did you have to ask for help while performing the experiment?

more questions on other side



Subject ID: _____

1	2	3	4	5	6	7
never	sometimes	once or twice	a few times	often	very often	all the time

Do you have any comments or any suggestions on how to improve the interface?

References

[1] D. Bruemmer, D. Few, R. Boring, J. Marble, M. Walton, and C. Nielsen. Shared understanding for collaborative control. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(4):494 – 504, july 2005.

[2] D. Bruemmer, D. Few, M. Walton, R. Boring, J. Marble, and C. Nielsen. Turn off the television: Real-world robotic exploration experiments with a virtual 3-d display. In *Proc. HICSS*, 2005.

[3] Bungie. Halo 3, 2007.

[4] S. Cousins. An open platform for robotics research. Cox Distinguished Lecture Series, 2009.

[5] M. Goodrich and A. Schultz. Human-robot interaction: a survey. *Found. Trends Hum.-Comput. Interact.*, 1(3):203–275, 2007.

[6] D. Grollman. *Teaching Old Dogs New Tricks: Incremental Multimap Regression for Interactive Robot Learning from Demonstration.* PhD thesis, Brown University, May 2010.

[7] S. Hart. Nasa-task load index (nasa-tlx); 20 years later. *Human Factors and Ergonomics Society Annual Meeting Proceedings*, 50:904–908(5), 2006.

[8] S. Hart and L. Stavenland. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In P. A. Hancock and N. Meshkati, editors, *Human Mental Workload*, chapter 7, pages 139–183. Elsevier, 1988.

[9] C. Humphrey, C. Henk, G. Sewell, B. Williams, and J. Adams. Assessing the scalability of a multiple robot interface. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, HRI '07, pages 239–246, New York, NY, USA, 2007. ACM.

[10] M. Kadous, R. Sheh, and C. Sammut. Controlling heterogeneous semi-autonomous rescue robot teams. In *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, volume 4, pages 3204 –3209, oct. 2006.

[11] E. Karulf, M. Strother, P. Dunton, and W. Smart. Ride: mixed-mode control for mobile robot teams. In *Proceedings of the 6th international conference on Human-robot interaction*, HRI '11, pages 161–162, New York, NY, USA, 2011. ACM.

[12] B. Maxwell, N. Ward, , and F. Heckel. A human-robot interface for urban search and rescue. In *Proceedings of the AAAI 2003 Mobile Robot Competition Workshop*, 2003.

[13] J. Mclurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau, and B. Schmidt. Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots.

[14] Microsoft. Age of empires 3, 2005.

[15] C. Nielsen and M. Goodrich. Comparing the usefulness of video and map information in navigation tasks. In *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 95–101, New York, NY, USA, 2006. ACM.

[16] C. Nielsen, M. Goodrich, and R. Ricks. Ecological interfaces for improving mobile robot teleoperation. *Robotics, IEEE Transactions on*, 23(5):927 –941, oct. 2007.

[17] R. Pausch, R. Gold, T. Skelly, and D. Thiel. What hci designers can learn from video game designers. In *CHI '94: Conference companion on Human factors in computing systems*, pages 177–178, New York, NY, USA, 1994. ACM.

[18] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

[19] S. Siwek. Video games in the 21st century, 2007.

[20] W. Smart and T. Blakely. A supervisory control interface for large mobile robot groups. In *Proceedings of the American Nuclear Society 12th Robotics and Remote Systems for Hazardous Environments Topical Meeting*, pages 457–463, 2008.

[21] S. Tejada, A. Cristina, P. Goodwyne, E. Normand, R. O'Hara, , and S. Tarapore. Virtual synergy: A human-robot interface fro urban search and rescue. In *Proceedings of the AAAI 2003 Mobile Robot Competition Workshop*, 2003.

[22] L. von Ahn. Games with a purpose. *Computer*, 39:92–94, 2006.

Vita

Erik Karulf

Date of Birth	July 19, 1986
Place of Birth	Dayton, Ohio, USA
Degrees	M.S. Computer Science, May 2011 B.S. Computer Science, December 2009
Professional Societies	Association for Computing Machinery
Publications	Erik Karulf, Marshall Strother, Parker Dunton, and William D. Smart. RIDE: mixed-mode control for mobile robot teams. In *Proceedings of the 6th international conference on Human-robot interaction*, HRI '11, pages 161–162, New York, NY, USA, 2011. ACM.

May 2011

Mixed-Mode Control of Robot Teams, Karulf, M.S. 2011