Report Number: WUCSE-2012-3

2012

RIDE: A Mixed-Mode Control Interface for Mobile Robot Teams

Erik Karulf, Marshall Strother, Parker Dunton, and William D. Smart

There is a growing need for robot control interfaces that allow a single user to effectively control a large number of mostly-autonomous robots. The challenges in controlling such a collection of robots are very similar to the challenges of controlling characters in some genres of video games. In this paper, we argue that interfaces based on elements from computer video games are effective tools for the control of large robot teams. We present RIDE, the Robot Interactive Display Environment, an example of such an interface, and give the results of initial user studies with the interface, which lend support... Read complete abstract on page 2.

RIDE: A Mixed-Mode Control Interface for Mobile Robot Teams

Erik Karulf, Marshall Strother, Parker Dunton, and William D. Smart

Complete Abstract:

There is a growing need for robot control interfaces that allow a single user to effectively control a large number of mostly-autonomous robots. The challenges in controlling such a collection of robots are very similar to the challenges of controlling characters in some genres of video games. In this paper, we argue that interfaces based on elements from computer video games are effective tools for the control of large robot teams. We present RIDE, the Robot Interactive Display Environment, an example of such an interface, and give the results of initial user studies with the interface, which lend support to our claim.

2012-3

RIDE: A Mixed-Mode Control Interface for Mobile Robot Teams

Authors: Erik Karulf, Marshall Strother, Parker Dunton, and William D. Smart

Corresponding Author: wds@cse.wustl.edu

Abstract: There is a growing need for robot control interfaces that allow a single user to effectively control a large number of mostly-autonomous robots. The challenges in controlling such a collection of robots are very similar to the challenges of controlling characters in some genres of video games. In this paper, we argue that interfaces based on elements from computer video games are effective tools for the control of large robot teams. We present RIDE, the Robot Interactive Display Environment, an example of such an interface, and give the results of initial user studies with the interface, which lend support to our claim.

Type of Report: Other

RIDE: A Mixed-Mode Control Interface for Mobile Robot Teams

Erik Karulf, Marshall Strother, Parker Dunton, and William D. Smart
Department of Computer Science and Engineering
Washington University in St. Louis
{eak2,mas1,pbd1,wds}@cse.wustl.edu

Abstract— **There is a growing need for robot control interfaces that allow a single user to effectively control a large number of mostly-autonomous robots. The challenges in controlling such a collection of robots are very similar to the challenges of controlling characters in some genres of video games. In this paper, we argue that interfaces based on elements from computer video games are effective tools for the control of large robot teams. We present RIDE, the Robot Interactive Display Environment, an example of such an interface, and give the results of initial user studies with the interface, which lend support to our claim.**

I. Introduction

There is a growing need for robot control interfaces that allow a single user to effectively control more than one remote robot. The increasing levels of autonomy demonstrated by our robots allow them to be controlled at higher and higher levels of abstraction. However, these systems are not perfect, and human intervention is sometimes required when unexpected circumstances arise, or an object beyond the capability of the perception systems must be analyzed.

This requirement for occasional intervention suggests that our interfaces should be capable of both high-level (task-based) and low-level (direct teleoperation) control of remote robots, with the ability to switch easily between these modes, as required by the situation. The goal of a single operator controlling many robots also suggests that a system that allows an individual to alert the operator when help is needed would be useful. If there are too many robots for the operator to attend to at once, there is a distinct possibility that a robot may sit idle, waiting for help, for a long time without such a notification system.

Many of the problems of controlling remote robots are similar to the problems of controlling characters in video games. Real-time strategy games involve 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 used for these games make ideal candidates for interfaces for mobile robot control.

Our motivation for drawing from video games is simple. Video games with easy-to-use, effective interfaces will be played more often, and will sell better than games with poor interfaces. Almost four decades of market forces have refined interfaces for many genres of games, resulting in systems that are intuitive, easy to use, and effective. Additionally, 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.

In this paper we present RIDE, the Robot Interactive Display Environment, a control interface for robots that draws heavily on computer game interfaces for inspiration. It combines aspects from a number of computer game genres, allowing the operator to switch between direct and supervisory control as the situation warrants. In addition to discussing video game interfaces and presenting RIDE, we give the results of our initial user studies with the interface, which suggests that it is well-suited to search tasks with multiple robots.

II. Computer Game Interfaces

A. Real-Time Strategy Games

Real-time strategy (RTS) games[1] are typically focused on resource management and combat between a small number of players (human and computer) who each control large numbers of heterogeneous units. These units represent troops, vehicles, buildings, and similar resources. The goal of these games is to achieve some pre-specified victory condition before the other players. These victory conditions are often specified in terms of accumulating a certain amount of some resource, occupying some location on the map, annihilating other players' units, or some combination of these.

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 1 shows the user interface from Microsoft's Age of Empires II [9] RTS, a representative and extremely popular example of the genre. 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

This work was supported by the National Science Foundation, under awar d OCI 0753340.

[1]There is an unfortunate overloading of terms here. In the context of computer games, "real-time" means that all of the players move their units at once, rather than taking turns. It has nothing to do with the technical meaning of "real-time" in the robotics and controls literatures.

Fig. 1. Microsoft's Age of Empires II real-time strategy user interface (from [9]).

Fig. 2. Halo 3 first-person user interface (from [7])

various types of units (farmers, infantry, cavalry, buildings, large weapons), terrain (water, trees, plains, farms), and game information (selected units, waypoints). The player can select 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 left 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.

Information about the currently-selected units is displayed to the left of the task list. In th figure, a single light cavalry 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 right 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 view in the main display (white rectangle). In the figure, only the areas previously explored by the player are displayed in the minimap. The world is of finite size in this game, and the minimap is fixed, both in size and viewpoint.

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

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

1) Gameplay: A key feature of RTS games is that there are too many units for the player to control directly. Although the individual units have some amount of autonomy, they occasionally need help and re-tasking from the player. Part of the challenge of RTS games is to ensure that as many units as possible are engaged in tasks that help achieve the victory condition. This means that the player has to switch attention between different (groups of) units frequently, regain situational awareness quickly, and deal with the circumstances that necessitated the intervention.

B. First- and Third-Person Games

First-person games involve the direct control of a single character by each player. The game world is rendered from the viewpoint of that character, often with additional game information overlaid on it. Third-person games are similar, except that the viewpoint is typically above and behind the character, and moves with it. Third-person games allow the player to see the character in the world, and give more of a sense of context.

Figure 2 shows the user interface for Halo 3 [8], a recent first-person game. The majority of the interface is occupied with a rendering of the world from the character's perspective. Additional game information is overlaid on this in the manner of a heads-up display. 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. The player can only see what the character can see, and the viewpoint is fixed.

1) Gameplay: First-person games involve the direct control of a character with a limited field of view. Part of the challenge of these games is to build up a situational awareness of what is going on around the character, but cannot be seen. While this often leads to exciting and engaging games, it is arguably not a design choice that makes it easy to maintain situational awareness.

Third-person games display the character embedded in the local world, and give the player some more local context.

However, the viewpoint is still fixed, forcing the player to reconstruct unseen parts of the world in her head.

III. User Interface

In this section, we describe the various elements of the RIDE interface, and how they relate to the video game interfaces discussed in the previous section. RIDE has three interface modes: supervisory (corresponding to an RTS interface), third-person, and first-person.

A. Supervisory View

The RIDE supervisory interface is shown in figure 3. We consider this to be the main interface mode, and the one in which users will spend the most time when controlling teams of robots. The main part of the interface shows an isometric view of the world, showing the robots, their sensor information, the map (if there is one), and additional status information. In the figure, two robots are shown, along with visualizations of their laser range-finders. Notice that the laser visualization does not exactly correspond to the displayed map; RIDE simply displays the sensor information reported to it, and does not attempt to reconcile information from different sources. Areas of the world that have been explored are tiled in white, to give the user a sense of the current coverage. A grid is overlaid on the tiles, to give a sense of scale.

The user can move the viewpoint around using the keyboard and mouse, to focus on the part of the world that is currently relevant. To maintain a global perspective, a display of the entire world is provided in the lower left of the interface. If a map is given, it is also displayed here. As more of the world is explored, this mini-map is scaled so that it continues to show the whole world. Robot positions are displayed on the mini-map, with currently-selected robots being colored green. Clicking on the mini-map causes the main display to jump to the corresponding location. The viewing frustum of the main display is also shown in the mini-map for additional context (white line towards the bottom left of the mini-map in the figure).

Robots in the main display can be selected individually or in groups using the mouse and keyboard. Selected robots are highlighted with a green circle (the leftmost robot in the figure has been selected). When a robot is selected, information about it appears in the information panel, located at the bottom of the interface in the center. In the figure, the selected robot is called "Blood", is an Erratic, has not reported its battery status, has no current task, and has only a laser range-finder. A small iconic representation of the robot is also shown. These information items are illustrative; in principle, any information that the robot decides to publish can be displayed here.

When one or more robots are selected, the tasks that they are capable of are displayed in the task panel, in the bottom right of the interface. When more than one robot is selected, the intersection of the tasks that they are capable of are displayed. Clicking on a task button assigns that task to all selected robots, over-riding any currently assigned tasks.

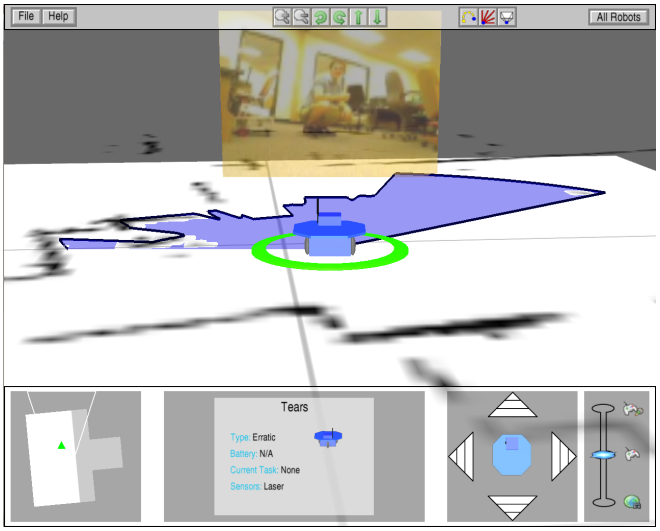

Fig. 4. The RIDE third-person interface mode.

Some tasks require further location information, which is entered by clicking on the main display of the interface. The move task, for example, requires an additional single click to specify a goal location. The patrol task, on the other hand, requires a sequence of clicks, specifying waypoints to be followed. The final waypoint is specified by a double-click. Again, the tasks shown in the interface are illustrative; robots report the tasks that they are capable of performing to the interface, and the additional parameters that these tasks require (see section IV).

A menu bar at the top gives access to configuration dialogs that allow the user to select colors for various elements in the display, the sensor information that is displayed (globally, or on a per-robot basis), and other settings. 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.

Additional sensor information displayed in the interface includes robot paths and notification status (both active for the leftmost robot in the figure). Robots may post notifications when situations requiring user intervention occur (see section III-D). These are displayed as a red exclamation mark above the robot, and a corresponding text box on the left side of the display.

In the bottom right of the interface, a slider allows the user to select between supervisory, first-, and third-person modes.

B. Third-Person View

The RIDE third-person interface is shown in figure 4. We consider this to be a secondary interface mode, used when direct control of a single robot is needed, to extricate it from situations that it cannot deal with autonomously. The main display of this mode shows the world from a viewpoint above and behind the robot. Unlike supervisory mode, the viewpoint moves with the selected robot. The user controls the robot directly using the keyboard, mouse

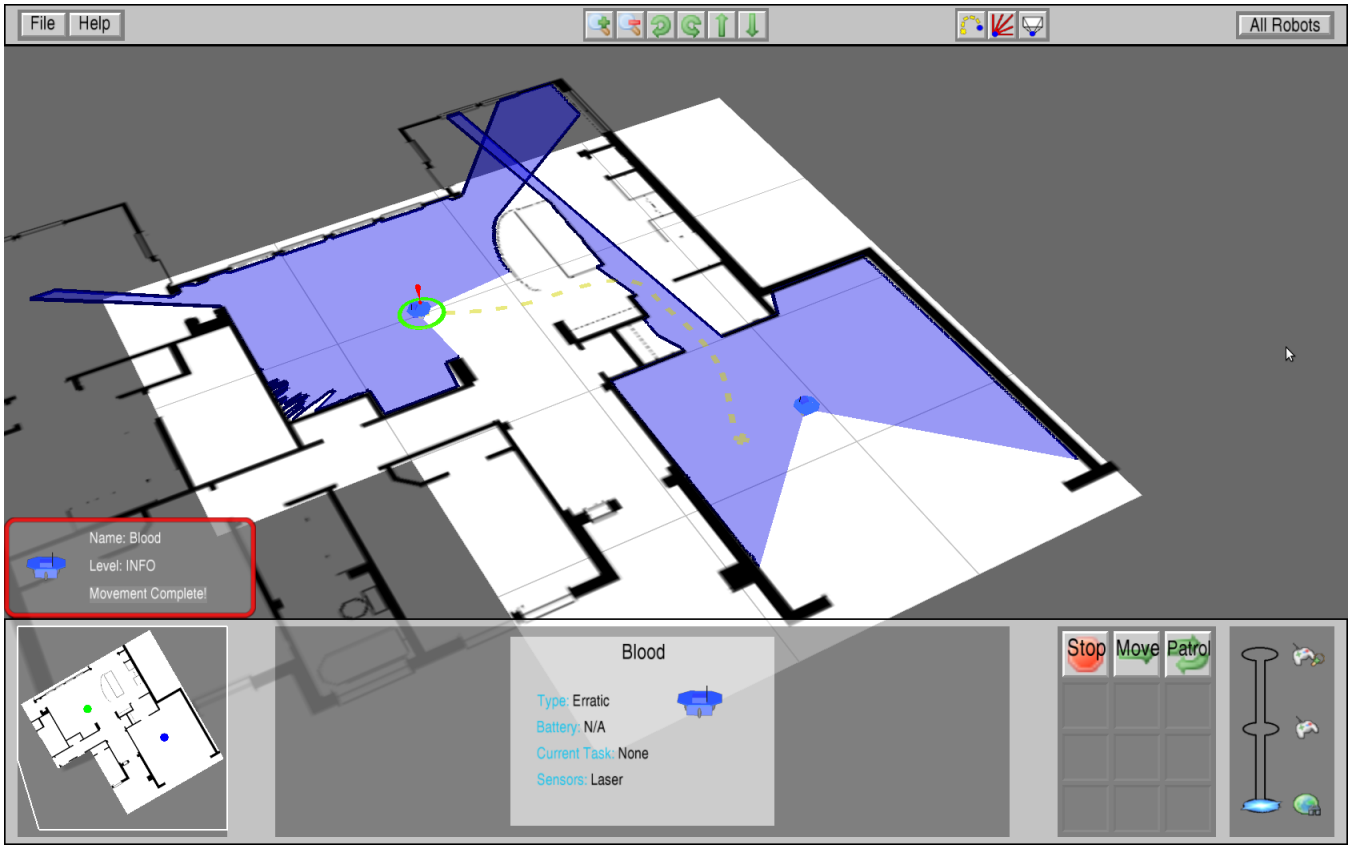

Fig. 3. The RIDE supervisory control interface.

and joystick. Third-person view is entered from supervisory mode by toggling the slider in the lower right of the display when a single robot is selected, or by double-clicking on a robot. The user returns to supervisory mode by hitting the "escape" key.

In third-person mode, live camera images may be displayed in front of the robot, to help the user see things that are not represented in the sensor visualizations. This is illustrated in figure 4. The image is displayed as if on a projector at a (configurable) fixed distance from the robot.

Other than the camera viewpoint, the information displayed in this mode is the same as that displayed in supervisory mode. The other elements of the interface also remain unchanged, with the exception of the task display. Since third-person mode implies direct control of a single robot, the task display is replaced with additional sensor visualizations for that robot. In the figure, we show a plan-view of the robot, with indicators around it to show the proximity of obstacles.

C. First-Person View

The RIDE first-person interface is identical to the third-person interface, except that the camera viewpoint is located at a fixed position on the robot. If the robot has a video camera, the viewpoint is typically coincident with the camera axis. This allows the user to see the world "as the robot sees it". If the camera information is displayed in this mode, it occupies the whole of the main interface display.

D. Notifications

An important feature that cuts across all three interface modes is the notification system. This allows a robot controlled by RIDE to notify the operator of events that require attention, such as the completion of a task or a situation that the robot needs assistance with. Notifications are displayed in a priority queue on the left of the interface (as shown in figure 5), where their order is determined by their importance. Currently, notifications have four importance levels: informational, warning, error, and fatal.

Notifications are a way of bringing the user's attention to a situation that is outside of his current focus. For example, a robot might have completed an assigned task, or might have encountered a situation that it cannot deal with autonomously. In these cases, we would like the user to respond to the event as quickly as possible, reducing the amount of time that the robot is sitting idle. A robot may post notifications to RIDE at any time. RIDE then displays a red exclamation mark above the robot, and adds the notification to a priority queue, ordered by severity. Notification boxes are displayed at the left of the interface, and contain the name of the robot, the level of the notification, and a brief description (supplied by the robot). Clicking on a notification box selects the corresponding robot, and moves the display viewpoint to focus on it.

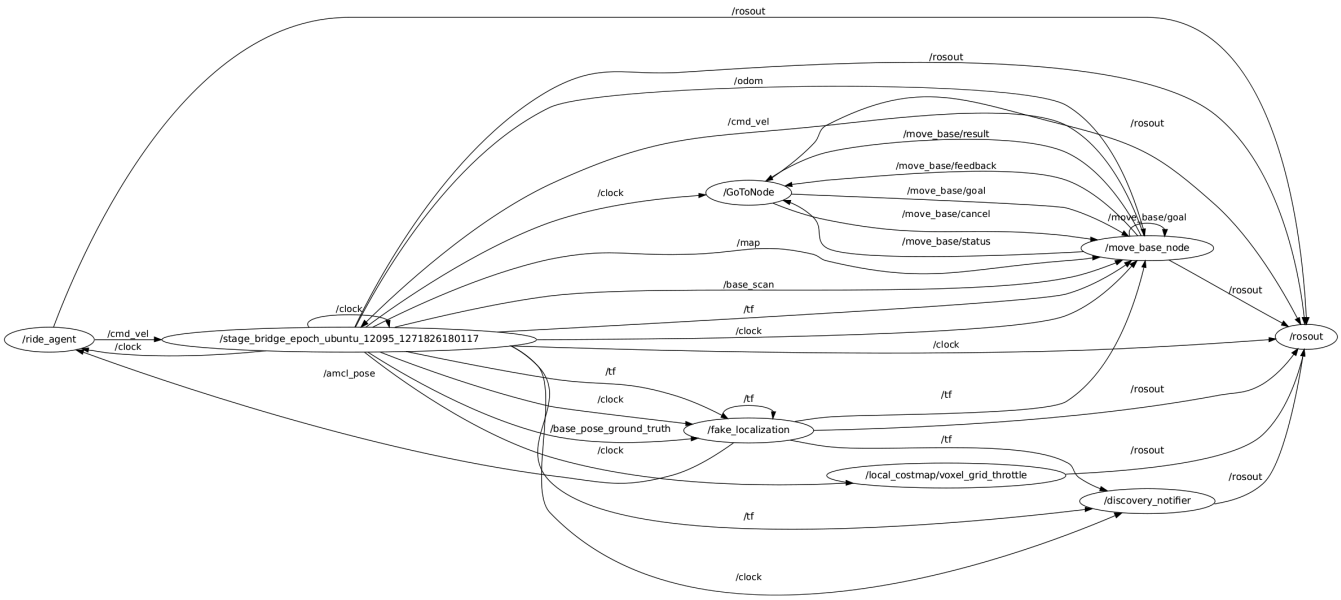

Fig. 6. The minimal set of ROS nodes running on each robot controlled by RIDE, for a simulated robot.

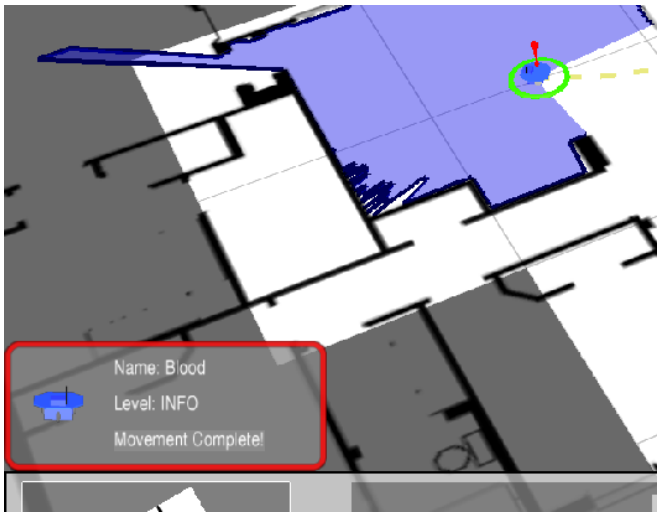

Fig. 5. Detail of the RIDE notification system.

IV. Implementation

RIDE is integrated with the ROS robotics middleware framework [12]. ROS is publish/subscribe framework, where computational *nodes* advertise data on *topics*. Nodes generally publish data from sensors, take published data and send it to the robots actuators, or transform data from other nodes for republishing. The network of ROS nodes and topics for a robot controlled by RIDE is shown in figure 6. Note that neither the RIDE interface itself nor the topics it subscribes and publishes to are shown in the figure for the sake of clarity.

Each robot runs a RIDE node (`ride_agent` in figure 6), 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. Once established, these are peer-to-peer connections with the nodes that publish the information, and do not go through the RIDE node. Figure 6 shows the set of ROS nodes for a simulated robot. All sensor information (`base_scan`, `odom`) comes from the simulator node (`state_bridge`) and, similarly, all movement commands (`cmd_vel`) are sent there. On a real robot, there would be separate nodes for each of the sensors, and for the movement controller.

RIDE also establishes direct connections with the nodes responsible for performing the tasks that can be allocated to the robot. In figure 6, this is `GoToNode` which is responsible for all movements tasks on 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 plan to relax these assumptions in future implementations (see section VIII).

RIDE is freely available at the Washington University ROS repository, `http://wu-ros-pkg.sourceforge.net`.

V. Related Work

We are not the first to propose an RTS-style interface for robot control. Jones and Snyder [3] describe a system that is very similar in spirit to ours, although it is designed for the control of a small number of free-flying space robots. This interface is most similar to the main display in RIDE, but it lacks other features, such as sensor visualization, the information panel, the minimap, and notifications.

Parasuraman, Galster, and Miller describe a task-level control interface called Playbook [11], and evaluate it's effectiveness on a task where subjects controlled six simulated unmanned vehicle under a range of conditions. Their interface had a 2d representation of the world, and limited sensor visualizations.

Nielsen and Goodrich [10] compared a control interface that is very similar to the main display of RIDE to one that presented a direct video feed from the robot alongside a 2d map. They concluded that the third-person display, with the video displayed as if on a projection screen in front of the robot, allowed for easier control of the robot. Our use of this technique for video display in third-person mode is motivated by the results of this work.

Richter and Drury [13] surveyed a number of video game interfaces, and characterized the various recurring elements within them. They proposed a Video-Game Based Framework (VGBF) for characterizing HRI interfaces by the input and output devices and methods, and by their input classifications.

A number of control interfaces either similar to, or draw direct inspiration from, RTS games [4], [6], [15], [1] or from first- or third-person games [5], [2], [1]. However, 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 [14].

VI. User Study

To evaluate the effectiveness of the RIDE interface, and of the notification system in particular, we conducted a formal user study. 22 participants (6 female, 16 male) from the Washington University community participated in the study. The mean age of the participants was 23.5 (*SD* = 5.60), and the median was 21.

Our experiment had two conditions, one in which the notification system was enabled, and one in which it was not. Subjects were asked to perform a search task, using two simulated robots in a small house. Subjects were instructed to locate three boxes in the house as quickly as possible, that the boxes were not shown on the map, and that they would be detected by the robot's laser range-finder sensor. Once a box had been found, they were asked to point it out to the experimenter, to verify that they had found it. Subjects were also told that they could use any of the features of the interface that they wanted to. The box locations and starting positions of the robots were the same in all runs of the experiment. The simulated robots modeled Videre Design Erratic ERAs with a Hokuyo URG laser range-finder.

Subjects were asked to fill in a pre-experiment questionnaire,to record their demographic information, experience with computers and video games, which video games they play regularly, and experience controlling a robot. After filling in the questionnaire, subjects were introduced to the RIDE interface, and allowed approximately five minutes of training time to familiarize themselves with it.

After the training period, the interface was restarted, and the subject was asked to perform the task in a randomly-assigned condition. For each subject, we recorded the total amount of time needed to compete the task (completion time), the total time that the robots were not moving and not selected in the interface (total neglect time), and the amount of time the robots spend not moving (total idle time). We also recorded all interface events (button pushes, selections, etc) for each subject.

On completing the search task, each participant was then asked to perform a second search task in the other condition. The data from the second task were not used in this paper.

After completing both search tasks, participants were asked to fill in a post-experiment questionnaire, recording their subjective impressions of the interface on a 7-point Likert scale.

VII. Results

In this section, we present the results of our user study. All times are presented in seconds. The total neglect time is the sum of the neglect times for each robot and, thus, can be greater than the completion time for the task. Similarly, the total idle time is the sum of idle times for the two robots.

Unless otherwise specified, in the following analysis we present the results of running a between-participants analysis of variance (ANOVA), using completion time (CT), total neglect time (NT), or total idle time (IT) as a dependent variable. Tabular results report the mean (standard deviation) of times, the F-statistic from the ANOVA (for $F(1, 20)$), and the significance level. Subjective ratings were measured on a 7-point Likert scale.

A. 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 I). 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.

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 II). Subjects

	VG	NG	F	p
CT	244.67 (142.57)	550.43 (178.03)	18.798	¡ 0.001
NT	181.27 (160.54)	585.57 (218.85)	24.073	¡ 0.001
IT	238.13 (190.06)	720.00 (319.07)	19.850	¡ 0.001

TABLE I

EFFECTS OF REGULAR VIDEO GAME USE (VG) VS. NO REGULAR VIDEO GAME USE (NG).

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
CT	264.82 (150.42)	419.09 (237.44)	3.3138	= 0.0837
NT	195.45 (170.41)	424.36 (291.50)	5.0555	¡ 0.05
IT	255.09 (202.24)	527.81 (375.06)	4.5062	¡ 0.05

TABLE II

EFFECTS OF REGULAR FIRST-PERSON VIDEO GAME PLAY (FP) VS. NO REGULAR FIRST-PERSON VIDEO GAME PLAY.

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 III). 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
CT	236.33 (170.27)	415.08 (207.99)	4.5248	¡ 0.05
NT	163.33 (178.53)	411.38 (265.54)	5.9435	¡ 0.05
IT	220.22 (222.86)	510.00 (339.22)	5.0228	¡ 0.05

TABLE III

EFFECTS OF PRIOR EXPERIENCE CONTROLLING A ROBOT (RE) VS. NO PRIOR EXPERIENCE CONTROLLING A ROBOT (NE)

Finally, the percentage of total time spend 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), $F(1,20)=4.2348$, $p=0.05287$.

B. 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 spend in supervisory mode. The median time percentage of time spent by subjects in supervisory was 94.19%. Subjects that spend 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 IV). The effects on total neglect time and total idle time were similar.

	GM	LM	F	p
CT	197.17 (79.80)	515.70 (181.79)	30.119	¡ 0.001
NT	124.17 (89.09)	532.80 (218.64)	35.198	¡ 0.001
IT	174.17 (111.90)	652.20 (305.51)	25.482	¡ 0.001

TABLE IV

EFFECTS OF SPENDING GREATER THAN MEDIAN TIME IN SUPERVISORY MODE (GM) VS. SPENDING LESS THAN MEDIAN TIME IN SUPERVISORY MODE (LM).

C. 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 V). Total neglect time, and total idle time were similarly dramatically reduced in this group with notifications enabled.

	NE	ND	F	p
CT	301.57 (149.13)	547.50 (195.07)	6.6401	¡ 0.05
NT	269.21 (204.21)	576.50 (241.72)	6.1615	¡ 0.05
IT	319.86 (205.03)	731.83 (340.67)	7.2453	¡ 0.05

TABLE V

EFFECTS OF ENABLING NOTIFICATIONS (NE) VS. DISABLING NOTIFICATIONS (ND).

VIII. DISCUSSION

The results presented in the previous section 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. However, playing RTS games seems to have no significant effect on the how well subjects perform the task. Experience with first-person games, on the other hand, does have an effect, decreasing all three times. However, this first-person game experience caused subjects to use the first-person interface *less* than subjects who did not play first-person games. While this seems somewhat counter-intuitive at first, we hypothesize that, because of their prior familiarity with this type of interface, they recognized it as less appropriate for the task that the supervisory interface.

Subjects preferred to use the supervisory mode interface, spending over 69% of their time, on average, in this mode. First-person mode was only used by a single subject during non-training runs. This might be caused by a current limitation of our interface, however. Currently, the map is rendered as a texture on the ground plane, making it hard to see and correctly interpret while in first-person view, because of the acute viewing angle. We hypothesize that, if the map were rendered in 3d, with the "walls" coming out of the floor (similar to the interface in Bruemmer *et al.* [1]), the operator would have a better sense of where the robot was with respect to the map, and the first-person interface would be used more.

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 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 and who, presumably, understood something of how they worked. We hypothesize that these subjects were able to manually monitor both robots at the same time, waiting for the expected failures and interventions, making the notification system redundant. More naïve 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.

Our future plans call for another study with a larger world and more robots. Such a study, we believe, will show the benefits of notifications across the whole population when even experienced users cannot realistically monitor all robots at once.

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 naïve users to more efficiently control teams of robots, reducing neglect and idle times, and 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.

REFERENCES

[1] D. J. Bruemmer, D. A. Few, M. Walton, R. L. Boring, J. L. Marble, C. W. Nielsen, and J. Garner. "turn off the television!": Real-world robotic exploration experiments with a virtual 3-d display. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, 2005.

[2] C. M. Humphrey, C. Henk, G. Sewell, B. M. Williams, and J. A. Adams. Assesing the scalability of a multiple robot interface. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, pages 239–246, Arlington, VA, 2007.

[3] H. Jones and M. Snyder. Supervisory control of multiple robots base on a real-time strategy game interaction paradigm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, pages 383–388, 2003.

[4] M. W. Kadous, S. R. K.-M., and C. Sammut. Controlling heterogeneous semi-autonomous rescie robot teams. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 3204–3209, 2006.

[5] B. A. Maxwell, N. Ward, and F. Heckel. A human-robot interface for urban search and rescue. In W. D. Smart and M. Bugajska, editors, *Proceedings of the AAAI 2003 Mobile Robot Competition Workshop*, 2003. AAAI Technical Report WS-03-01.

[6] J. McLurkin, J. Smith, J. Frankel, J. Sotkowitz, D. Blau, and B. Schmidt. Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In *Proceedings of the AAAI Spring Symposium "To Boldly Go Where No Human-Robot Team Has Gone Before"*, 2006.

[7] Mechanical R. Halo 3 review. `http://mechanicalr.wordpress.com/2009/08/10/halo-3-review/`.

[8] Microsoft Game Studios. Halo 3. `http://halo.xbox.com/en-us/intel/titles/halo3`.

[9] Microsoft Games. Age of Empires II: Age of Kings. `http://www.microsoft.com/games/age2`.

[10] C. W. Nielsen and M. A. Goodrich. Comparing the usefulness of video and map information in navigation tasks. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, pages 95–101, Salt lake City, UT, 2006.

[11] R. Parasuraman, S. Gastler, and C. Miller. Human control of multiple robots in the roboflag simulation environment. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, pages 3232–3237, 2003.

[12] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: An open-source robto operating system. In *ICRA 2009 Workshop on Open Source Software in Robotics*, Kobe, Japan, 2009.

[13] J. Richter and J. L. Drury. A video game-based framework for analyzing human-robot interaction: Characterizing interface design in real-time interactive multimedia applications. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, pages 266–273, Salt lake City, UT, 2006.

[14] W. D. Smart and T. M. 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.

[15] 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 W. D. Smart and M. Bugajska, editors, *Proceedings of the AAAI 2003 Mobile Robot Competition Workshop*, 2003. AAAI Technical Report WS-03-01.