Game-Based Design of Human-Robot Interfaces for Urban Search and Rescue

ABSTRACT

Robot Urban Search and Rescue is a new task in robotics that is challenging not only robot hardware, but also human-robot interfaces. It is a real-world challenge in an unstructured environment that requires cooperation between a remote robot and a human operator. The study of human-robot interfaces is still in its infancy, and most of the urban search & rescue systems field so far use an ad hoc approach to user interface design. This paper presents an overview of current thinking on interface design for urban search & rescue, presents a design and evaluation of a user interface based on a video game interface, and then makes the case that video games offer a valid road map for future development of both robot capabilities and user interfaces. A comparative analysis shows that the goals and methods used in both video games and urban search and rescue are similar, if not identical. Therefore, we can leverage the empirical success of video game interfaces in developing current and future user interfaces for urban search and rescue.

Keywords

Robot-human interface, urban search and rescue, computer game interfaces.

ACM Classification Keywords

H.5 Information Interfaces and Presentation, H.5.2 User Interfaces, interaction styles, prototyping

INTRODUCTION

The Urban Search and Rescue [USR] event at the American Association for Artificial Intelligence [AAAI] Robot Competition challenges robot teams to find victims in a simulated urban environment. In order to be successful at this task, the robot must answer three questions:

1. Where should it search?
2. How should it get there?
3. How does it identify victims?

This event has been part of the AAAI Robot Competition and Exhibition since the summer of 2000. In the past three competitions, the most successful teams have answered these questions by providing a human operator with as much information as possible about the robot's situation. Then the human--not the robot--attempts to answer the relevant questions. Currently, the second question, making decisions about global and/or local navigation, is the only area in which multiple teams have successfully given the robots some significant autonomy.

Whatever the robot's capabilities, effective user interfaces [UI] are critical in this task. The UI must be centered around providing the human operator sufficient information to make correct decisions about future actions of the robot at the required level of decision-making. This constitutes human robot awareness, as defined by Drury, Scholtz and Yanco [10].

Because of existing robot capabilities, the decisions required are currently primarily low-level, and they focus on either moving the robot to new locations, or moving a camera on the robot to look at a point of interest. As robot capabilities improve, the level of decision-making should become more high-level. For example, a robot might be told to explore an area, or go to a point in an area already explored without any further oversight until it achieves the goal.

In our four years of experience with the USR task, it has become apparent that the UI is critical to success in the task. Thus, the focus of our 2003 USR entry was to develop an effective, responsive user interface that permitted the operator to use a range of autonomy in navigation, and provide the greatest flexibility in the use of the robot's hardware.

This paper presents an overview of the concepts in USR UIs, presents our UI design for the 2003 AAAI USR Competition, and offers a reflective analysis on how USR interfaces ought to be constructed for now, and in the future as robot capabilities continue to improve.

Inspiration and Awareness

For our design we turned to the computer game industry for inspiration. The genre of computer games called *first-person shooters* [FPS] give players control over low-level decisions and permit them to quickly assess their environment and make time-critical decisions [8][17]. Using a single robot to explore a USR arena and find victims is similar in

task and information content to searching a maze or building for enemies.

Using a computer game as inspiration for a user interface for real tasks is not new [2][6]. Unlike many other attempts at using game interfaces, however, USR has all the elements of a good game, and the goals are virtually identical. Following the heuristics laid down by Malone, USR has a specifically defined goal, the outcome is uncertain, the task is emotionally appealing, and the physics of the game are those of real life, so the user understands them intuitively [11]. The element of the interface that we need to design appropriately is providing the optimal level of informational complexity, introducing, or compressing information content as needed.

Unlike computer games, however, in the USR task we are dealing with real-world communications and real robots on the other end of the screen. Thus, the interface cannot ignore response time--which is determined by hardware and environment--or use capabilities that do not yet exist on the robots. Nor can it ignore the fact that robots break down, and go in and out of communications range. The latter, in particular, is going to be built into future USR competition rules. None of the existing USR UIs, however, deal explicitly with the issue of extended communications outages.

The literature specifically on the USR robotics task is still small, as the competition has only existed since AAAI 2000. Based on the experience of teams who have undertaken the task so far, there are certain factors and capabilities of UIs that appear to be crucial to success in the task.

In order to start quantifying these factors, Drury, Scholtz, and Yanco undertook an analysis of three USR UIs in the 2002 competition and developed a vocabulary for analyzing human-robot interfaces [HRI] [10]. The key concept they defined is HRI awareness.

> *HRI awareness (base case): Given one human and one robot working on a task together, HRI awareness is the understanding that the human has of the location, activities, status, and surroundings of the robot, and the knowledge that the robot has of the human's commands necessary to direct its activities and the constraints under which it must operate.*

They further defined subcategories that are all the possible directed pairings of the agents in the situation (humans and robots). For the UIs in the study, errors on the USR course were due to failures in at least one subcategory of HRI awareness [10].

Note that the concept of HRI awareness also matches the flow of information in first-person shooter computer games where a human is directing the actions of an avatar. For first-time users, for example, failure to succeed is often due to lack of HRI awareness about the status of the avatar (health, ammunition, location). As users get used to the UI, however, their assessment of avatar status becomes habitual and failures will be due to judgements in game play, rather than failures of HRI awareness. Note that, if this were not the case, users wouldn't enjoy the game because it would be difficult to enter the "Flow" state due to the design of the UI [13].

USR User Interfaces

The USR UIs developed so far meet the requirement of HRI awareness in different ways. Since the human is generally making most of the decisions, human-robot awareness--the awareness the human has of the robot--appears to be the most critical, since failures in human-robot awareness lead to judgement errors for the system as a whole.

To understand human-robot failures, and thus improve design, we believe it is important to further subdivide the concept into two categories: awareness of the robot's surroundings, and awareness of the robot's status. *Surroundings awareness* is knowledge of important local landmarks in the environment and knowledge about how the robot can move safely. Another way to think about surroundings awareness is how accurately the user can describe the robot's local environment. *Status awareness* is knowledge about the state of the robot itself, such as its power level, orientation, attitude, or the position of its camera relative to the front of the robot.

The primary method of obtaining surroundings awareness in current USR UIs, and FPS games, is through a video feed. Because most robots do not possess significant autonomy, the most crucial of the UI attributes is the video lag time and bandwidth. Of the two factors, lag time is more important, as long lag times lead to instability in the control loop, which occurs when decisions are made and sent to the robot based on old information.

As an example of this, in 2002 one competitor switched from using an integrated graphical user interface to using primitive text-based debugging commands to control their robot simply because it reduced the video lag time [4][10].

A positive example is the INEEL interface, which uses specialized hardware compression to manage the video signal, enabling both high bandwidth and low lag time [1]. The INEEL interface is currently considered the best interface for the USR task, not only because of its integrated nature and fast video feed, but also because it uses heavy-duty hardware capable of traversing most of the USR arena [14].

The INEEL interface is not game-like, however, and suffers from several deficiencies: the screen is too busy, the video is not central on the interface, peripheral vision is not used effectively, and there is no use of sound. In the 2002 competition the interface was actually a hindrance, and the clutter

Figure 1: a Magellan Pro robot entering the USR arena.

on the UI caused the driver to ignore the fact that the robot's camera was not facing forward while driving [1][10].

A second method of obtaining surroundings awareness is through the use of a dynamically created map. Several entries have used real-time mapping to improve surroundings awareness [10][9]. The quality of the real time maps to date, however, is such that they are not generally used as the primary means of surroundings awareness. Instead, they function largely as a historical record of the robot's path. The one exception is the MITRE interface, which makes the map window the primary method of surroundings awareness [9]. The best maps so far were generated by the Georgia Tech team in 2002, but these were generated off-line at the end of the run [7][14].

A USR DESIGN: HARDWARE AND SOFTWARE

Our robots in the USR competition were two iRobot/RWI Magellan Pro's with onboard 450MHz Pentium III computers running Linux. Figure 1 shows one of the robots entering the USR arena. The Magellans come with 16 sensor panels, each containing a bump sensor, an IR sensor, and a sonar. In addition, each Magellan has a Canon VC-C4 pan-tilt-zoom [PTZ] camera mounted on top, near the front, with a pan range of -100˚ to 100˚, a tilt range of -30˚ to 90˚, and a 16X optical zoom. The camera video is connected to a bt848-based PCI framegrabber, and it receives pan-tilt-zoom commands via the serial port.

In order to obtain sound from the robot's environment, we mounted a wireless microphone on the robot and piped it to headphones worn by the operator. This enabled us to take advantage of sounds emitted by victims or other devices in the USR arena.

The robots are wheeled, and do not have a high clearance, so they are only able to enter the yellow area of the arena. Thus, the focus of the interface is on making the operator more productive, and enabling the robots to traverse the area faster.

Each robot has the ability to make relative motions autonomously with obstacle avoidance. We also have basic mapping software that uses the sonar/infrared sensor readings to build maps, and a vision module for controlling the PTZ camera. All communication with the robot and between modules is via the Inter-Process Communication [IPC] package [15].

Interface Design

As noted above, we took our inspiration for the UI from first-person shooter video games. The most successful FPS games have very simple interfaces: the main portion of the interface is dedicated to a large viewing area that contains the game world. This provides the primary source of surroundings awareness. Small status areas around the perimeter are allocated for information regarding the player's health and equipment, which constitutes status awareness. Most games also have a radar mini-map showing the direction to enemies or other players, as well as a separate screen containing a map of the current game level. In FPS games, these are most useful for strategic decision-making rather than immediate time-critical actions.

Our robot interface integrates three sources of information: a video stream from the robot's PTZ camera, a "radar" view visualizing the range sensor data (sonar and infrared, combined) from the robot, and a manually updated map view showing the robot's internal map of the navigated area. Together, these three windows, shown in Figure 2, bring the robot's view of the world to the remote operator and provide both surroundings and status awareness. Note that the primary status information is in the radar view, which includes not only current sensor readings, but also indicators for the pan and tilt of the camera.

Unlike other USR interfaces, ours is not intended to be primarily mouse-based, hence we do not, by default, show buttons for controlling the robot. Instead, most control is done via the computer's keyboard, much like an FPS. The standard WASD directional pad is preferred by many right-handed gamers, because it puts most of the basic controls within one key of the fingers of the left hand, and frees up the right hand for using the mouse.

Unfortunately, "mouselook", where moving the mouse rotates the player's head in the game world, was not something that could be implemented with the robot's existing PTZ camera implementation. Because of this limitation, we use a second d-pad control for the camera using UHJK on a QWERTY keyboard. With the operator's left and right hands resting on the keyboard, the vast majority of the interface controls are a single keypress away. The operator can use the fast-twitch muscles in their hand to react as quickly

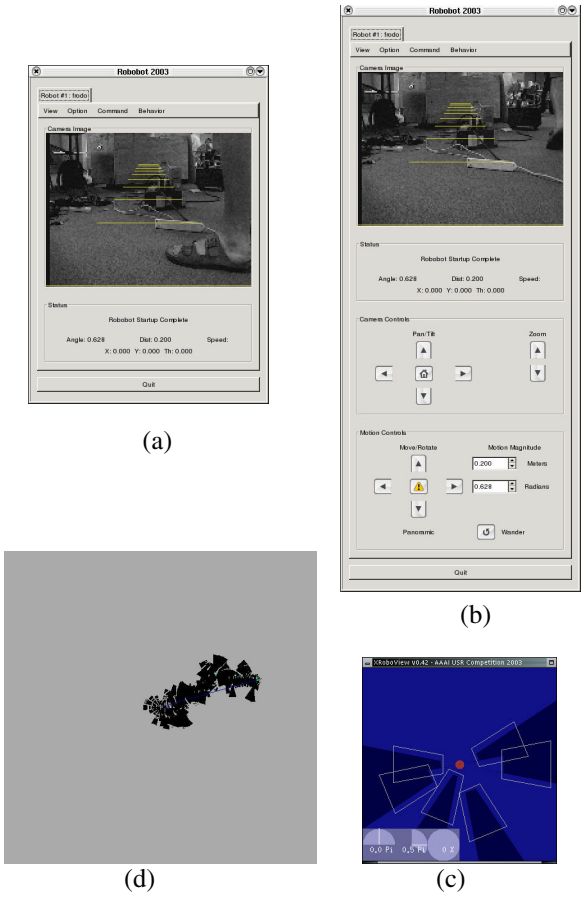

Figure 2: (a) Standard video and navigation interface screen, (b) expanded screen with button controls, (c) map screen showing a map built by the robot in USR run 3, (d) sensor and camera status display.

as possible to changing conditions in the robot's environment.

The camera's PTZ controls implemented a small step angle (around $\pi/8$ radians) for tilting and panning using the keyboard. This was important so that the operator did not lose surroundings awareness. To speed up a pan across a large area, however, the interface also implemented *jump-to* options for moving the camera to the limit of its PTZ ranges, as well as for quickly resetting the camera to a straight ahead view.

The video feed from the robot is enhanced slightly by the addition of ground-plane bars. Based on the robot's configuration and camera tilt angle, these bars appear in the image as though they were laying on the ground at 0.5m intervals. This assists the operator in determining distances, which are difficult to assess otherwise from a single video image.

Interest Points

The mouse does play one important role in interacting with the video image received from the robot: it allows the operator to set and manipulate points of interest, including landmarks and victim locations, which are then stored in the robot's map. The operator's own human visual system provides the brain behind what the robot is seeing.

The operator can set an interest point on the ground by clicking on the image. The robot will use its current position and camera orientation data to calculate the location of the selected point in the world by assuming that the click occurred somewhere on the ground plane, similar to the method of drawing the ground plane bars. An operator can click multiple times on the same landmark over time to correct the orientation of the robot. This ability to orient on local landmarks significantly reduced errors in localization, and improved the accuracy of the robot's maps.

Landmark selection also allowed the addition of some interface features specifically for the USR event. For example, after the user selects a point, the system can save images of the interest point, or query the user for other data that should be associated with it.

In the USR competition, creating a victim interest point brings up a form giving the user the opportunity to fill out the information required for each victim as part of the competition. Figure 3 shows an example victim dialog box. Having the victim data sheet be part of the UI saved the operator time compared to filling out a paper form by hand. It also enabled further feature development.

In particular, the interface generates a web page on the fly using the victim data collected from the operator. In a real USR situation, this page could be served over a wireless network from the operator's computer to rescue workers with properly equipped PDAs. The rescuers would then have instant access to the victim's status, an image of the victim and the surrounding structure, a map to the victim with a robot-navigable path highlighted, and images of any navigation points set along that path. This kind of information access is highly useful to human rescue workers. Returning this data also fulfills the robot's role as a scout into a potentially dangerous situation.

RESULTS AND EVALUATION

The competition provided a number of insights into the strengths and weaknesses of the user interface. Some of these were a result of design choices, while others were a result of hardware or software limitations on the robot itself.

One of the design choices, for example, was to make the default screen use the keyboard exclusively for low level control of the robot. For an experienced user, the improvement gained by using only the keyboard exclusively is valuable and saves time relative to a mouse. Unfortunately, having a large number of controls accessible only by potentially hard-to-remember keystrokes dramatically increases the interface learning curve. An operator who has trained on the interface will have no problems, and a new operator who

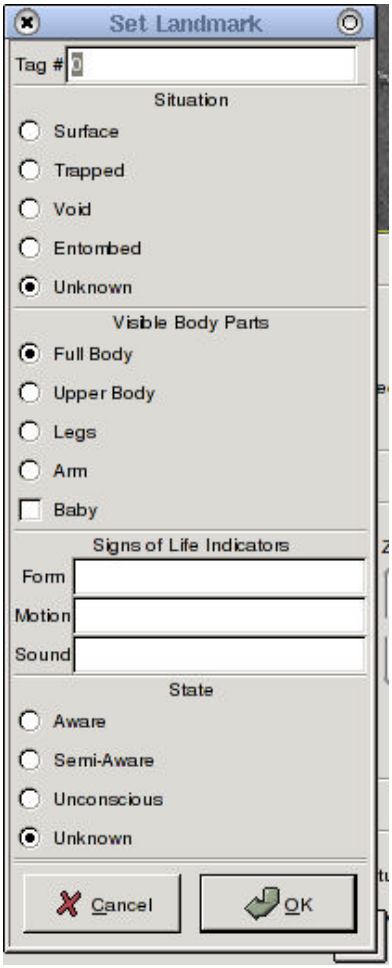

Figure 3: Dialog window for entering victim data.

has experience playing FPS games with similar key layouts will be able to adjust quickly.

New operators with little or no gaming experience, on the other hand, may find the interface too obscured. During the time that they would spend learning the keyboard layout they would be effectively useless in a real operational setting, due to very slow response times and an increased incidence of mistaken keypresses that could endanger the robot through inadvertent motion.

This problem became obvious when a new operator with effectively no gaming experience attempted to use the interface. Even after a brief training period, and making the mouse-based controls visible, the operator was barely able to get the robot to move, let alone use it to search for victims. When considering the trade-off between efficiency for a trained user and usability for a wide range of inexperienced users, the increased efficiency granted by the keyboard interface is more important in the field. However, the mouse-based interface is still necessary if there is a need for naive users to manage the robots. Training a new user would not require a significant investment in time or resources, but enough to acclimate the user to the interface.

A limitation of the underlying robot system was that the autonomous navigation was too jerky to permit effective monitoring of long forward motions by the operator. Thus, although the magnitude of the robot's forward motion or rotation could be set to an arbitrary value before sending a move or rotate command, we found that small rotations or motions worked best. Asking the robot to travel several meters in a straight line using its own obstacle avoidance simply did not work well in the enclosed spaces with noisy sensors. This is clearly an area for future improvement in the underlying navigation system.

Making large rotations (greater than an eighth of a turn, or about 45˚) seriously exacerbated the motion errors, and also tended to disorient the operator when the video blurred and lagged. Using short forward motions (less than 0.5 meters) and small angles (around $\pi/10$ radians) allowed the operator to nudge the robot in the desired direction without worrying as much about the robot's own behaviors suddenly kicking in to avoid an obstacle. Smaller motions also made far more accurate maps, which made the victim data web pages more useful.

Both of these issues with the underlying system affected surroundings awareness because the motions, particularly rotations, were either too fast or too large to be followed using the video feed. Thus, the operator would lose a sense of orientation and have to spend time making small rotations to regain surroundings awareness.

In our attempts to optimize surroundings awareness, however, we short-changed some of the status awareness displays, which were located on a separate window from the video feed. As a result, there were several occasions where the operator was driving the robot around with the camera off-center, so the operator's perception of forward did not match the actual orientation of the robot.

This type of human error slows down progress considerably, since the operator is constantly fighting the robot's correct behavior. When using veto mode (when the operator completely overrides the robot's obstacle avoidance behaviors), the operator must be very careful to orient themselves and the robot properly. Otherwise, they stand a chance of damaging the robot or getting it trapped. As this occurred during one of the runs, we modified the interface to automatically reset the camera to forward if the user initiate the veto mode control.

Usually the loss of status awareness was caused by the operator ignoring one of the status displays, like the camera orientation or sensor range information. These losses in status awareness inevitably led to the loss of accurate surroundings awareness because the operator's frame of reference did not match that of the robot.

In summary, the system worked well for an operator with gaming experience, especially once the operator adapted to using short motions and rotations to best match the underlying robot capabilities. The one serious flaw in the UI was not making the status displays more prominent, or notifying the user of obvious mistakes such as driving forward with the camera pointing sideways.

In terms of quantitative evaluation, our team scored more points in the yellow part of the arena than any other team, largely because of the speed with which the operator could traverse the area and find more victims. We came in second, overall, because our robots could not traverse the more difficult sections where each victim found is worth 50 or 100% more than victims found in the yellow section.

A ROAD MAP FOR USR INTERFACES

Using a FPS as the inspiration for our interface worked well both qualitatively and quantitatively. As noted in the introduction, USR possesses most of the same characteristics as a successful computer game, including increasing difficulty due to different levels of the arena and more complex victim placement by the judges [11].

Failures in the interface occurred, not because we mimicked FPS interfaces, but because we did not sufficiently integrate the displays for both surroundings awareness and status awareness, as is done in FPS games. Thus, more closely following the overall layout and iterative refinement process of video game production would likely have reduced interface failures [3].

While robot capabilities are such that the human operator must provide low level decision-making to a robot or be responsible for victim identification, the FPS interface is most appropriate. It gives the user the most intuitive feel for the robot's situation, optimizing the decision-making ability of the operator. Per unit of robot time, this is, arguably, the most effective method of solving the task [5][12].

As robot capabilities improve and they gain more autonomy, however, there will be a push to have more robots exploring the arena, preferably with no increase in the number of operators. Reasons for using multiple robots include redundancy, faster recovery at forks in the arena, and faster coverage of the arena overall. Besides enabling the management of more robots per operator, robots will need more autonomy in the future in order to deal with reduced communications bandwidth and communications blackouts, which will be part of future USR contests.

While Crandall and Goodrich argue that the effectiveness per unit of robot time will decrease when the robots take on more autonomy, the decrease in individual robot effectiveness can be overcome by the increase in the number of robots [5][12]. Also, loss of communication or reduction in bandwidth immobilizes a tele-operated robot, while a more autonomous robot can continue to explore.

In order to effectively manage multiple robots, however, the look and feel of the UIs must change. A FPS design is effective for managing a single avatar, but it is difficult to manage two or more avatars with a FPS interface. In an FPS interface the context building--and the relationship between avatars--is left to the operator, who must juggle multiple contexts, and maintain them using discontinuous cues (you can't always be looking at both video feeds).

There is a class of video games that does provide a guide to interfaces for managing multiple semi-autonomous units: *real-time strategy* [RTS] games [16]. For RTS games, the primary mode of providing surroundings awareness is through a central map, similar to the MITRE interface [9].

The map contains iconic representations of all units in the area, as well as representations of walls, landmarks, and other interest points. Notification of events, unit status, or the flagging of units that require management is provided through multiple modalities, including sound, visual markers attached to the units, and changes in the iconic representations [16].

In a typical RTS game, users are limited by their mental ability to manage multiple units, but good players can effectively manage up to 200 units in real time against another human opponent. Using this type of interface, we ought to be able to manage a half-dozen robots, provided they are capable of fulfilling their required roles.

The key to enabling a RTS interface is to enable robust autonomy on the robots. In particular, they will need the following.

1. Accurate localization and mapping.
2. Autonomous navigation and searching
3. Robot-robot sharing of information.
4. Ability to identify relevant visual features.

The first item enables the use of a single map as the primary channel for surroundings awareness. The second item enables point-to-point control--tell the robot to go to a location--and the third item permits all the robots to make decisions about navigation based on the most recent information. Finally, the fourth item permits the placement of icons in the map and notification of the operator of items of interest. Without the combination of robot capabilities, it is difficult to see how the RTS scheme can be realized in a manner that will permit effective control of multiple robots.

CONCLUSION

It is our belief that the eventual solution to the USR task will be a combination of effective user interfaces and robust autonomy on the part of the robot in solving the pieces of the USR task. In other words, it is important to have a human in the loop, but the human should be processing and answering the most difficult questions encountered in the search, not providing input about basic search patterns or

navigation except in difficult circumstances. It is similar to the difference between the role of the captain and the role of the pilot on a large ship; the robot should be able to take over the role of the pilot.

Through our experience with USR UI design, we have found that a UI design based on an appropriate game design works well. Unlike many other attempts at UI design based on games, the USR task follows virtually all of the heuristic guidelines for a good game, and the needs of the USR UI closely track those of a video game.

As robot capabilities improve, we expect to see a transition in USR UIs from a video-based FPS style interface, to a map-based RTS style interface to permit management of multiple robots. To move to an RTS style interface before the robots achieve a minimum level of robust autonomy, however, will reduce the effectiveness of the human-robot team on the USR task. Thus, the interface must be appropriate for the robot capabilities. Currently, the FPS style interface provides the most effective UI design as measured by quantitative and qualitative success in the USR task.

ACKNOWLEDGEMENTS

This work was supported in part by NSF IIS-0308186, and the American Association for Artificial Intelligence.

REFERENCES

[1] D. Bruemmer, D. Dudenhoeffer, J. Marble, "Dynamic-Autonomy for Urban Search and Rescue", In *Proc. of AAAI Mobile Robot Competition & Exhibition Workshop*, Technical Report WS-02-18, AAAI Press, pp. 33-37.

[2] D. Chao, "Doom as an interface for process management", in *Proc. of Conf. on Human Factors and Computing Systems*, pp. 152 - 157, 2001.

[3] C. Clanton, "An interpreted demonstration of computer game design", in *Proc. of Conf. on Human Factors and Computing Systems*, pp. 1 - 2, 1998.

[4] J. Corder, O. Hsu, A. Stout, B. Maxwell, "A Modular Software Architecture for Heterogeneous Robot Tasks", In *Proc. of AAAI Mobile Robot Competition & Exhibition Workshop*, Technical Report WS-02-18, AAAI Press, pp. 18-23.

[5] J. W. Crandall and M. A. Goodrich, "Measuring the Intelligence of a Robot and Its Interface", to appear in *Proc. of PERMIS 2003*, September 2003.

[6] M. Christoffel, B. Schmitt, "Accessing Digital Libraries as Easy as a Game", in *Proc. of the 2nd Int'l Workshop on Visual Interfaces for Digital Libraries*, July 2002, Springer LNCS 2539 (extended version).

[7] F. Dallaert, T. Balch, M. Kaess, R. Ravichandran, F. Alegre, M. Berhault, R. McGuire, E. Merrill, L. Moshkina, D. Walker, "The Georgia Tech Yellow jackets: A Marsupial Team for Urban Search and Rescue", In *Proc. of AAAI Mobile Robot Competition & Exhibition Workshop*, Technical Report WS-02-18, AAAI Press, pp. 44-52.

[8] *Doom*. Id Software, Mesquite TX, 1993.

[9] J. Drury, L. D. Riek, A. D. Christiansen, Z. T. Eyler-Walker, A. J. Maggi, and D. B. Smith, "Evaluating Human-Robot Interaction in a Search-and-Rescue Context", to appear in *Proc. of PERMIS 2003*, September 2003.

[10] J. Drury, J. Scholtz, and H. A. Yanco, "Awareness in Human-Robot Interactions", to appear in *Proc. of the IEEE Conf. on Systems, Man and Cybernetics*, October 2003.

[11] T. W. Malone, "Heuristics for Designing Enjoyable User Interfaces: Lessons from Computer Games", In *Proc. of Conf. on Human Factors and Computing Systems*, pp 63 - 68, 1982.

[12] D. R. Olsen, Jr., and M. A. Goodrich, "Metrics For Evaluating Human-Robot Interactions", to appear in *Proc. of PERMIS 2003*, September 2003.

[13] R. Pausch, R. Gold, T. Skelly, D. Thiel, "What HCI designers can learn from video game designers", In *Proc. of Conf. on Human Factors and Computing Systems*, pp. 177 - 178, 1994.

[14] R. Rogers, "Assessment of the AAAI USAR Robotics Competition", In *Proc. of AAAI Mobile Robot Competition & Exhibition Workshop*, Technical Report WS-02-18, AAAI Press, pp. 32.

[15] R. Simmons and D. James, *Inter-Process Communication: A Reference Manual*, Carnegie Mellon University, March 2001.

[16] *Starcraft*, Blizzard Entertainment Inc, 1998.

[17] *Unreal Tournament*, Epic Games Inc, 1999.