2014

Human-robot Interaction For Multi-robot Systems

Bennie Lewis
University of Central Florida

HUMAN ROBOT INTERACTION FOR MULTI-ROBOT SYSTEMS

by

BENNIE G. LEWIS JR.
M.S. University of Central Florida, 2008

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2014

Major Professor: Gita Sukthankar

ABSTRACT

Designing an effective human-robot interaction paradigm is particularly important for complex tasks such as multi-robot manipulation that require the human and robot to work together in a tightly coupled fashion. Although increasing the number of robots can expand the area that the robots can cover within a bounded period of time, a poor human-robot interface will ultimately compromise the performance of the team of robots. However, introducing a human operator to the team of robots, does not automatically improve performance due to the difficulty of teleoperating mobile robots with manipulators. The human operator's concentration is divided not only among multiple robots but also between controlling each robot's base and arm. This complexity substantially increases the potential neglect time, since the operator's inability to effectively attend to each robot during a critical phase of the task leads to a significant degradation in task performance.

There are several proven paradigms for increasing the efficacy of human-robot interaction: 1) multimodal interfaces in which the user controls the robots using voice and gesture; 2) configurable interfaces which allow the user to create new commands by demonstrating them; 3) adaptive interfaces which reduce the operator's workload as necessary through increasing robot autonomy. This dissertation presents an evaluation of the relative benefits of different types of user interfaces for multi-robot systems composed of robots with wheeled bases and three degree of freedom arms. It describes a design for constructing low-cost multi-robot manipulation systems from off the shelf parts.

User expertise was measured along three axes (navigation, manipulation, and coordination), and participants who performed above threshold on two out of three dimensions on a calibration task were rated as expert. Our experiments reveal that the relative expertise of the user was the key determinant of the best performing interface paradigm for that user, indicating that good user mod-

eling is essential for designing a human-robot interaction system that will be used for an extended period of time. The contributions of the dissertation include: 1) a model for detecting operator distraction from robot motion trajectories; 2) adjustable autonomy paradigms for reducing operator workload; 3) a method for creating coordinated multi-robot behaviors from demonstrations with a single robot; 4) a user modeling approach for identifying expert-novice differences from short teleoperation traces.

To my mother for making this all possible

ACKNOWLEDGMENTS

It has been an amazing and challenging journey to get to this stage. There have been many road blocks and barriers along the way that I was able to conquer with the help and support from some very special people. With that being said, I first off would like to thank my wife Tawanna Lewis for her understanding, love and support throughout this entire experience. I would like to thank Bulent Tastan for his help and for collaborating with me on multiple publications. I want to give thanks to members of the IAL Research Lab past and present, particularly Fahad Shah and Xi Wang for all their help and support over the years. A special thanks goes to my advisor Gita Sukthankar for all of her direction, advice, and support through this entire process. I also would like the thank her for always being there to provide guidance and for not only being a wonderful advisor but also a great person.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Human-agent-robot teams fill an important niche in robotics since they can accomplish tasks that robots cannot complete autonomously, forming a team unit that is greater than the sum of the parts. Ideally the human users focus on the difficult cognitive and perceptual tasks, the robots manage the planning and execution of repetitive physical tasks, while the agents handle the most cumbersome information processing tasks. A well-designed human-robot interface is vital for the success of teleoperating a multi robot team. Teleoperating multiple robots significantly increases the complexity of the human's cognitive task, since the operator's concentration is divided among multiple robots. The objective of a fully-autonomous multi-robot team for real-world tasks remains elusive; this research focuses on agent-based approaches that partially automate the task and complement the human operator's control. During cognitively demanding tasks, the human operator's attention is typically focused on controlling a single robot; in such periods, the other robots in the team are under-utilized.

Multi-robot manipulation, where two or more robots cooperatively grasp and move objects, is extremely challenging due to the coordination requirements. In our delivery task, the human driver commands a pair of robots to move stacks of objects from starting piles to the goal location in a lightly cluttered environment. Each robot is composed of a non-holonomic wheeled base and 3-DOF arms for lifting objects. A contribution of the dissertation is our low-cost multi-robot manipulation system design. Some of the objects are large enough to require two robots to lift, necessitating tight coordination between the robots. The human operator's role is to align the robots, grasp the objects, and oversee the delivery process. The task was created to be simple to learn, but challenging to do within the allotted time. It models many types of real-world robotic usage scenarios, including delivering supplies to urban search and rescue victims, explosive ordinance disposal, robotic butler chores, and multi-robot office delivery systems. Unfortunately, introducing

a human operator does not necessarily ameliorate performance due to the complexity of teleoperating mobile robots with high degrees of freedom. The human operator's attention is divided not only among multiple robots but also between controlling a robot arm and its mobile base. This complexity substantially increases the potential neglect time, since the operator's inability to effectively attend to each robot during a critical phase of the task leads to a significant degradation in task performance. A common solution is to decrease the time period of autonomous operation and increase the amount of user intervention, but in cases where the task is complicated and the user's workload is already high, this approach threatens to degrade the overall system performance.

In summary, human-robot interfaces for multi-robot manipulation face the following challenges:

- Concentration is divided among multiple robots.
- Fully-autonomous multiple robot teams for real-world applications are unreliable.
- The coordination demands of multi-robot manipulation are high.

There is a rich body of pre-existing work on reducing operator workload in human-robot interaction [47, 24]. Multimodal interfaces [48] that replace finicky keyboard/mouse commands with speech and gesture can improve the operator experience and reduce the time required to perform individual actions. Adaptive interfaces that intelligently increase the robots' autonomy can be highly effective on multi-robot tasks [63], particularly when they are coupled with good user modeling mechanisms [19, 39]. Learning from demonstration (LfD) has been successfully used in many robotics applications to create policies based on a set of demonstrations provided by the user [4]. Embedding LfD within a configurable user interface empowers the user to create customized primitives that the system can execute autonomously, enabling the automation of frequently repeated sections of the task. However, it may be difficult for inexperienced users to demonstrate complicated sections of the tasks, where help is desirable, effectively enough to be automated using

learning from demonstration.

The thesis of this dissertation is: *Machine learning is a useful tool for detecting user distraction and classifying teleoperation style.* This dissertation demonstrates how various machine learning classifiers can be used to create both configurable and adaptive user interfaces and evaluates the relative benefits of different interfaces for multi-robot manipulation. User expertise was measured along three axes (navigation, manipulation, and coordination), and users who performed above threshold on two out of three dimensions on a calibration task were rated as expert. Our experiments reveal that the relative expertise of the user was the key determinant of the best performing interface paradigm for that user, indicating that good user modeling is essential for designing a human-robot interaction system meant to be used for an extended period of time.

The contributions of the dissertation include: 1) a low-cost design for constructing multi-robot manipulation systems from off the shelf parts; 2) a model for detecting operator distraction from robot motion trajectories; 3) adjustable autonomy paradigms for reducing operator workload; 4) a method for creating coordinated multi-robot behaviors from demonstrations with a single robot; 5) a user modeling approach for identifying expert-novice differences from short teleoperation traces.

Reader's Guide

This section provides a roadmap to the different sections of the dissertation.

Chapter 2 - Surveys related work in human robot interaction, multi-robot manipulation, learning from demonstration, user interfaces, and applications in those domains.

Chapter 3 - Describes the experimental testbeds that were used throughout this dissertation.

Chapter 4 - Describes a simple user interface for a simulated USAR environment using the GSAR

simulator.

Chapter 5 - Discusses two possible user interfaces for multi-robot control suitable for human teleoperation of a Robocup Rescue team.

Chapter 6 - Discusses a CoOperator interface for assisting the human operator in managing, controlling, and navigating multiple robotic agents through the disaster scenario.

Chapter 7 - Describes a mixed-initiative interface that provides the user with two important new cooperative functions: 1) locate ally, which unites the two robots in one location, and 2) mirror mode in which the second robot simultaneously executes a modified version of the commands that the user has issued to the actively controlled robot.

Chapter 8 - Describes a configurable interface for learning combined manipulation, coordination, and navigational tasks in a team setting.

Chapter 9 - Describes an adaptive interface that learns a model of expert-novice differences for the various aspects of the teleoperation task.

Chapter 10 - Evaluates an adaptive interface that adjusts the robot's autonomy based on the user's expertise vs. a configurable interface that allows the user to create macros for autonomous behaviors using a simple learning from demonstration paradigm.

Chapter 11 - Concludes this dissertation and presents an agenda for future work.

Appendix Technical Description - Presents additional technical details about the IAI interface and the HU-IE robot.

Appendix IRB - IRB approval for the user studies.

CHAPTER 2: RELATED WORK

Human Robot Interaction

Human-robot interaction design falls at the confluence of several research areas including autonomous systems, human factors, intelligent user interfaces and task analysis. Not only can software and hardware failures sabotage performance but mismatches between the interaction paradigm and the task or between the users and the system are equally problematic. Tools such as goal-directed task analysis or cognitive work analysis can help the prepared designer customize the interface design for a particular task [19, 38]. However, it can be harder to predict which type of user interface is likely to appeal to the population of potential users without extensive user interface studies.

Much of the work in human-robot interaction has centered on having the robots do more when the human user is unavailable using approaches such as cognitive workload modeling [19, 38] or adjustable autonomy [61, 57]. However fundamentally, synthetic and biological entities have very different capabilities that need to be respected during task division. Collaborative control [20] intelligently utilizes these differences by leveraging the user to perform perception and cognition tasks, rather than merely involving the user in the planning and command.

An alternative approach, is to view constructing human-agent-robot teams as a process of coactive design, which was first introduced by Johnson et al. [29]. Coactive design concentrates on understanding the interdependence of joint activity and carries the expectation that human and robot will function in close and continuous interaction. Often how this interaction occurs is pre-determined by the designer; however, in our system the user interface is configurable, allowing the user's understanding of the task guide the periods of interdependence. To do this we allow the user to

identify common subtasks before the actual task execution. Our intelligent user interface analyzes these sections of the task and creates a task abstraction for these activities. This task abstraction is then used 1) to build macros using a simple programming by example method and 2) to inform the system's understanding of the user's autonomy preferences.

In a sense, this can be seen as a simple form of social learning between humans and robots [10]; the human teaches a single robot what to do, but during execution, the robot must account for the actions of the second robot and the user when performing the macro. To do this, the robot maintains a mental model of its robot teammates and modifies the macro to be useful in a team setting. Socially-guided exploration has been utilized in robot learning systems, but in that case the human partner provides social scaffolding during the learning process to guide the robot's actions during the learning of a non-cooperative task [9]. Imbuing the robot with social skills such as turn-taking, intention recognition, and emotion models, can yield high dividends in improved human-robot interaction and increased trust in automation [8]. This can be very important in manipulation tasks that involve the human sharing the robot workspace and handing objects directly to the robot (e.g., [18, 42]).

Operator neglect was identified as an important factor by Crandall et al. [14] who used an analysis of neglect and interaction time to predict the performance of a team of robots controlled by a single human. Wang and Lewis [63] theorize that in multi-robot control problems where tasks and robots are largely independent the operator sequentially neglects robots until their performance deteriorates sufficiently to require new operator input. This leads to poor performance in tasks with higher coordination demands, such as when the robots have differing sensing capabilities. Introducing a teamwork proxy [64] that can enable the robots to coordinate among themselves was shown to successfully increase robot autonomy and decrease demands on the human operator. Operator neglect can also be detected using hidden state estimation techniques [19, 37] and compensated for by the robots. Chen and Barnes's study on multi-robot supervisory control indicates that individual differ-

ences in spatial ability, attentional control and video gaming can affect the quality of human-robot interaction [12].

Another option is to shift workload from the operator to the robots by increasing the robots' autonomy. The robot's autonomy can be adjusted intelligently by encoding the user's interruption preferences as rewards, allowing the robot to make optimal use of the human's time [57]. Muszynski and Behnke [45] presented an interface for personal service robots that allowed the users to select the appropriate level of autonomy based on their perceived workload. Adjustable autonomy, having the robots alter their level of autonomy in a situationally-dependent manner, has been used successfully in human-robot teams [57, 61]. In this paradigm, the robots reason about the tradeoffs between disturbing the human user vs. the risk of task errors. Here, rather than focusing on the user's interruption threshold or distraction level, autonomy is adjusted based on the user's capability to perform different aspects of the task. Multiple robots substantially increase the complexity of the operator's task because attention must be continually shifted among robots. Lewis and Scerri [40] presented an experiment on the use of multi agent teamwork proxies to control robots performing a search and rescue task. Increasing robot autonomy allows robots to be neglected for longer periods of time making it possible for a single operator to control more robots.

Conventional models of multi robot control presuppose that there are independent robots and independent tasks; this allows an additive model in which the operator controls robots sequentially neglecting each until its performance deteriorates sufficiently to require new operator input. Wang and Lewis [63] introduced a model of coordination demand that extends the neglect tolerance model to situations in which robots must cooperate to perform dependent tasks. For multi-robot tasks, reasoning about teamwork decisions, such as role allocation and communication timing, can be as important for the success of the task as taskwork decisions. The Multiagent Adjustable Autonomy Framework (MAAF) was designed specifically for facilitating teamwork in large mixed human-robot teams [22]. In this dissertation, I focus on improving human robot interactions (HRI)

by implementing a decision making intelligent co-operator user interface that manages robotics agents that are not being controlled by the human operator at the time. Fan and Yen [19] demonstrate the use of Hidden Markov Models to track humans' cognitive capacity. In their model, the cognitive load is divided into five states from negligibly loaded to overly loaded in which the transitions in between states show the changes in the cognitive load of the human subject while cooperating with an agent in a human-agent team.

Improved visualization can improve the operator's situational awareness and reduce the amount of time required to make decisions [52]. Shirakura and Morita [60] introduced a method by which the operator gains a sense of the environment surrounding the robot using an eye tracking system. The line of sight of the robot is matched to the operator's line of sight, which is detected using gaze tracking. The authors were able to demonstrate reduced eye fatigue when viewing 3D images and to simplify the process of robot control for the operator. The effort required by the user to communicate with the robot can be minimized through the use of voice, gesture, or EEG controls [25]. Gesture-based interfaces, in particular, may be more intuitive for manipulation and remotely controlling robot arms [13]. Pieska et al. developed a gesture-based system for assisting inexperienced operators at industrial robot programming tasks; they note that their interface was useful for users with varying levels of robot programming expertise [49].

Multi-Robot Manipulation

Khatib et al. [32] defines the basic manipulation capabilities needed by robots to operate in human-populated environments as being: 1) integrated mobility and manipulation; 2) multi-robot coordination; 3) human-robot interaction; 4) collision-free path planning. Our work focuses on the use of effective human-robot interaction and multi-robot coordination to address deficiencies in the robots' sensing. Note that the human interacts with the robots exclusively through a gamepad

controller and graphical user interface rather than cooperatively grasping the object along with the robots as was done in [27].

Haptics can be a valuable tool for the human-robot interaction of manipulation tasks, allowing the operator a more immersive telepresence. Boompion and Sudsang [7] presented a distributed formation control algorithm for a robot team moving in a obstacle filled workspace. The algorithm models the robots as a set of spring forces and a potential field. The advantage of the algorithm is that it works under limited sensory information and requires no communication between the robot team. The authors created an intuitive human robot interaction system by means of a data glove. Their interface allows the human operator to control the group formation parameter with only a hand gesture.

Okada and Beuran [46] proposed a motion planning method based on the Probabilistic Roadmap (RPM) algorithm. Each robot on the team is equipped with a motion planner in order to avoid collisions in real time. The authors constructed an experimental platform based on a large-scale network test bed. By using an virtual environment manager and support software the authors were are able to simulate a large-scale team of autonomous networked mobile robot systems. The experimental results validated the usefulness of collaborative motion planning, which resulted in reaching the destination faster with less frequent re-planning. Fua and Lim [23] proposed a graph structure for defining agent neighborhood relations to support the agent flocking framework. Their Inter-Roam-Space movement algorithm directs the movement of individual agents, allowing robot teams to adapt to unforeseen circumstances, while still making use of available time information for preliminary scheduling and planning. In contrast, our user interface has a single autonomous agent with different modes that can be either executed autonomously or requested by the user; in this work we demonstrate that injecting these extra modes into the human-robot interaction significantly improves task performance and user satisfaction. Rosenthal et al. [55] propose the notion of a symbiotic human-robot relationship where the user can help the robot overcome its limitations.

Similarly, we rely on the human to address some of our robot's deficiencies, particularly in perception. However, our work differs in that the human is not physically present but rather that robot control moves seamlessly between user and the agent, enabling multiple robots to cooperate with a human in-the-loop that drives the system's high-level goals.

Srinivasa et al. [62] have developed HERB, an autonomous mobile manipulator that performs common household tasks. HERB can search for objects, navigate disorderly indoor scenes, perform vision-based object recognition, and execute grasp planning tasks in cluttered environments. Although lifting heavy objects is beyond the capabilities of several small robots due to center of gravity considerations, tasks such as clearing household clutter can benefit from the combined efforts of multiple robots. By giving every robot manipulation capabilities, small tasks such as flipping switches or opening doors can be performed in parallel.

Learning from Demonstration

In this dissertation, I evaluate the performance of a configurable interface that allows each user to customize the behavior of the system. One method to achieve this personalization is to allow the user to program the system directly, using learning from demonstration. Learning from demonstration (LfD) is one approach for simplifying the user interaction paradigm that is a promising alternative to classical engineering approaches. The user simply shows the robot what is desired by teleoperating the robot, and the robot attempts to reproduce the demonstration in a novel situation. Learning from demonstration has been used successfully to learn manipulation primitives [15, 34], and even watching human action strategies for dual-arm manipulation tasks has proven to be useful in robot controller design [33]. Alternatively, increased personalization can be achieved by allowing the user to modify the retargetting function used to map the user teleoperation sequences onto the robot [16].

Programming by example has been integrated into an assortment of demonstrational user interface systems (see [41] for an overview). In a straightforward method, macros are recorded and replayed at the user's command without modification. To generalize the macros to alternate situations, machine learning methods such as supervised or inverse reinforcement learning can be used to learn an abstraction over features or rewards. Within robotics systems, learning by demonstration [4] or apprenticeship learning [1] has been principally used as a method to learn robotic controllers for high dimensional action spaces or to bootstrap reinforcement learning. Our work differs from conventional learning from demonstration in that the user remains continually involved in system control during macro execution. The task work is learned by the user through demonstration, and the teamwork coordination model is preprogrammed. The users can express their autonomy preferences through designating sections of the task work to be automated and can opt to either accept or reject the learned macro if the initial demonstration does not match their expectations of the learned system.

Schneider and Ertel [59] introduced a new Gaussian process regression model that clusters the input space into smaller subsets. They demonstrated how Gaussian processes can be used to encode a robotic task and presented an experiment on a physical robot arm. Our user interface includes a learning by demonstration component; instead of using Gaussian processes to cluster the input space, we assume that there are a small set of likely cases and ask the user to verify the correctness of each learned macro. There has been other work in learning team tasks by demonstration for urban search and rescue that relied on spatio-temporal clustering to segment robot behaviors [43]. Unlike our method, their system requires cooperative demonstrations to learn the team behaviors and no attention was paid to user acceptance aspects of the problem.

Dang and Allen [15] proposed a technique to decompose a demonstrated task into sequential manipulation tasks and construct a task descriptor. One goal of this research was to create a database of knowledge of how to manipulate ordinary objects. Our taskwork abstraction is similar, but can

also be extended to the multi-robot manipulation problem. Grollman and Jenkins [26] note that many of the current robot learning algorithms require the existence of basic behaviors that can be combined to perform the desired task. On the other hand, robots that exist in the world for long timeframes and learn numerous tasks over their lifetime could exhaust this basis set and need to acquire new behaviors. To address this issue, Grollman and Jenkins learning paradigm that is capable of learning both low-level motion primitives and high-level tasks from interactive demonstration. They use nonparametric regression within this framework to learn a complete robot soccer player in stages, first by teaching the robot how to walk, then seek, and acquire the ball.

User Interfaces

There are many different ways to structure a RAP (Robots, Agents, People) system. For instance, Kawamura et al. [30] propose an agent-based architecture in which different aspects of the user interface are tasked to separate agents. An alternative approach is to base the human-robot interaction on the same principles that govern effective human-robot teamwork. Hoffman and Breazeal [28] introduce a collaborative interaction architecture to support turn-taking, mutual support, and joint grounding for settings where a human user works alongside an autonomous humanoid robot in a shared workspace. Equipping each of the robots with a separate teamwork proxy that can enable the robots to coordinate among themselves was shown to successfully increase robot autonomy and decrease demands on the human operator in a USAR task [63].

Sato and Kon [56] present a navigation interface for robots that offers convenient interaction with multiple robots at the same time. In the proposed interface, the system displays the environment map surrounding the robots on the monitor and the operator can graphically input the target position for multiple robots. Using the touch-pen device, the operator can group multiple robots easily, and grouped robots can move to the target position using a formation control algorithm.

The guiding principle behind the first two approaches is the reduction of operator effort through good user design. In particular, 3D user interfaces can provide a more natural metaphor for interactions with the physical world. Ricks, Nielsen, and Goodrich [52] present an ecological interface paradigm that fuses video, map, and robot pose information into a 3-D mixed-reality display. Results from their user studies show that the 3-D interface improves robot control, robustness in the attendance of delay, awareness of the camera orientation with respect to the robot, and the ability to perform search tasks while navigating the robot.

Earlier work in this area has studied how an interface can be adapted to the user's profiles and preferences. For example, Kawamura et al. [31] developed an agent-based architecture for an adaptive human-robot interface, and Ahmad et al. [2] have done work on adaptive user interfaces in educational systems. Adaptive intelligent tutoring systems modify the performance of the ITS in response to a model of the learner's abilities [65]. However unlike adaptive intelligent tutoring systems, our user interface models but does not attempt to improve the user's teleoperation skills.

Application Domains

The Robocup Rescue League [54] was introduced to provide testbeds, datasets, and a reference problem for researchers interested in studying urban search and rescue. USAR teams are evaluated using a scoring metric that rewards teams for their ability to locate victims and accurately map the environment. A successful USAR team must overcome many research challenges including traversing complicated terrain, identifying victims in perceptually challenging situations, and creating an effective human-robot interface. Current competition rules state that only one person is allowed at the operator station at any time, regardless of the number of robots on the team. There are a number of possible approaches to improving the performance of a USAR team including adjusting the autonomy level of the individual robotic agents [58] and optimizing the robotic agent

task allocation.

Augmenting the robots with manipulation capabilities dramatically increases the number of potential usage cases for a human-agent-robot team. For instance, a number of USAR (Urban Search and Rescue) systems have been demonstrated that can map buildings and locate victims in areas of poor visibility and rough terrain [63]. Adding manipulation to the robots could enable them to move rubble, drop items, and provide rudimentary medical assistance to the victims. Effective human-robot interaction is an important part of the challenge of building urban rescue systems since full autonomy is often infeasible. The Robocup Rescue competition has recently been extended to award points for manipulation tasks.

Another Robocup competition, Robocup@Home [53] which aims to develop domestic service robots, also includes manipulation of household objects such as doors, kitchen utensils, and glasses. A set of standardized tests is utilized to evaluate the robot's abilities and performance in a realistic home environment setting. Our scenarios are designed to simulate the problem of clearing clutter on the floor of a household environment and depositing it into a collection area. We only work with non-breakable items so the system is tolerant to failed pickups.

CHAPTER 3: EXPERIMENTAL DOMAIN

This section describes the platforms, simulators and development tools that were used in our research on human robot interaction.

The Gridworld Search and Rescue Simulator

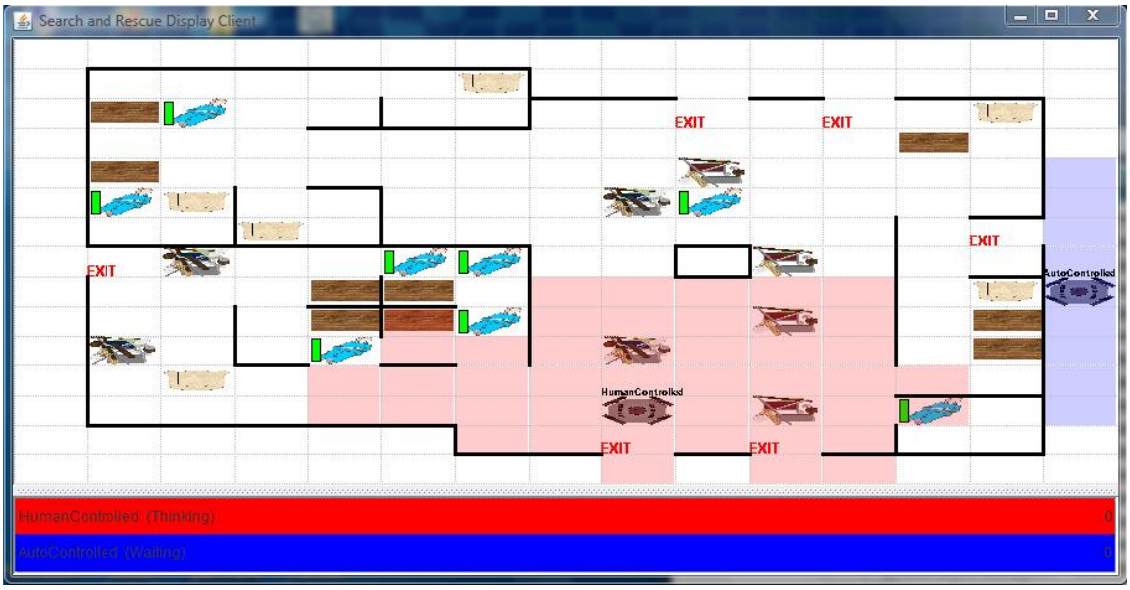

Figure 3.1: Gridworld Search and Rescue (GSAR) Simulator.

The first simulation we used for evaluating our user interfaces was the Gridworld Search and Rescue (GSAR), developed by Eric Eaton [17]. The simulator is suitable for educational use and allows students to develop an intelligent agent to solve the search and rescue problems from a high-level perspective. It uses a networked client-server framework that enables students to run their intelligent agent on local computers while interacting with the remote simulation server. With the exception of an agent carrying an object, each cell in the grid can be occupied by only one object at

a time. Cell coordinates remain fixed throughout the duration of the simulation, providing absolute locations for the agents positioning system. For simplicity, the building does not contain any doors that require opening, and objects cannot move through walls and sensors cannot penetrate walls. The agents' knowledge is updated online, offering the ability to keep the gridworld map secret until simulation. For a more difficult challenge the initial knowledge can be disabled.

The simulation models disaster victims that need to be rescued by the robots. Their probability of movement is based on their health status. Victims may perish during the course of the simulation as determined by the model, or they may be dead from the beginning. The rescue robot is equipped with sensors that provide high-level perception. Long-range object recognition and localization sensors cover a rectangular area around the agent. Short-range medical diagnostic sensors must be activated by the agent to provide information on the victim in the adjacent cell. Self-feedback sensors provide information on the robot itself. The disaster-severity parameter of the simulation controls the severity of injuries among the victims. For each live victim delivered the agent is credited one point. Dead victims or ones found wandering to the exit are not worth any points.

Microsoft Robotics Studio (MSRS)

The Microsoft Robotics Studio (MSRS) is a development environment tool for robot control and simulation [44]. It is aimed at academic, hobbyist, and commercial developers and handles a wide variety of robot hardware. The developer can use any of the programming languages that are supported by the .NET framework such as Visual C#, Visual C++, and Visual Basic .NET. MSRS comes with a graphical programming language called Visual Programming Language (VPL). VPL offers beginning developers and programmers a visual interface that allows them to use drag-and-drop techniques to build robotic programs. Some programs written with VPL can be converted into Visual C# applications. Microsoft Robotic Studio includes support for packages to add other

services on the Studio. Those currently available include a Soccer Simulation and a Sumo Competition by Microsoft, and a community-developed Maze Simulator. The Visual Simulation Environment (VSE) tool that is included with MSRS offers programmers a way to get involved with robotics development without any robotics hardware. VSE is powered by the NVIDIA PhysX engine, that allows the VSE to render advance physics graphics comparable to graphics used in video and computer games today. VSE can be useful for developers wanting to prototype their robotics project. Instead of having to spend a lot of time designing, building, and configuring robots, developers can test their code before they buy expensive equipment. The downside to using the VSE tool is that it not entirely realistic. In a real world environment robots encounter unforeseen obstacles and have to react in ways their developer never expected. The VSE is a useful tool for the beginning stages of a project, but only the use of an actual robot will yield real world results. VSE works by managing entities which can be represent any physical object, such as a robot, a wall, a camera, and even sun light. Services are used to move the robots around and gather data from sensors. Orchestration services are used to coordinate a group of services. Graphic scenes can be rendered in one of four modes visual, physics, wireframe, or a combination of the three modes.

Services are the essential building blocks for robotic applications created with Microsoft Robotics Developer Studio [44]. Every web-based service contains the code required to carry out one or more functions, such as reading data from a single sensor or transferring an output signal to an actuator. The service can also be used to communicate with an additional service or external software. A robotics application consists of several services that work together to accomplish a common task operating the robot. MSRDS allows twenty services to run on a single host. Microsoft's Decentralized Software Services (DSS) is a NETbased runtime environment that sits on top of the Concurrency and Coordination Runtime (CCR). Decentralized Software Services provides a small state-oriented service model, which combines the concept of representational state transfer (REST) with a system level approach. DSS services are exposed as resources that are

available both programmatically and for UI management. A crucial design goal of DSS is to couple performance with simplicity and robustness; this makes DSS for the most part suited for creating applications as compositions of services regardless of whether these services are running within the same node or across the network. DSS uses the Decentralized Software Services Protocol and HTTP as the foundation for interacting with services. DSSP is a SOAP based protocol that provides a fresh symmetric state transfer application model, with support for state manipulation and an event model determined by state changes. The DSS runtime provides a hosting environment with built-in support for service composition, security, monitoring, and logging both within a single node and across the network. Services can be written either in Visual Studio or using Microsoft's Visual Programming Language. Additionally, the DSS Manifest Editor provides a graphical environment for running DSS applications on a single node or across the network.

MSDRS can be used to simulate a wide range of robotic hardware. This research features a simulation version of the Pioneer 3DX Figure 3.3, with bumper sensors, a webcam and a SICK Laser Range Finder. The Pioneer 3DX is a differential drive robot which can be modeled using the MSRDS service called SimulatedDifferentialDrive that implements the abstract Drive contract. The Drive contract has operations to position the left and right wheel speeds individually, alternate the entire drive by a particular angle, and to drive a specified distance. The drive service sends notification when its state changes; events include changes in wheel speed and drive enabled status. The user can configure a Pioneer 3DX with a front and rear bumper ring consisting of multiple contact sensors using the SimulatedBumper service that implements the abstract Contact Sensor contract. The service sends a notification when a bumper is pressed or released. The SICK Laser Range Finder is simulated using a service called SimulatedLRF that sends a notification for each scan.

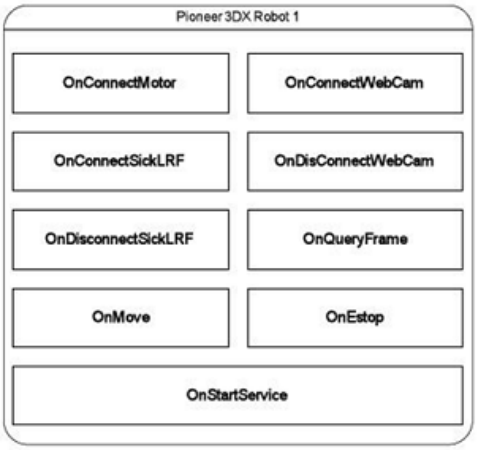

Figure 3.2: Services used to operate the Pioneer 3DX Robot.

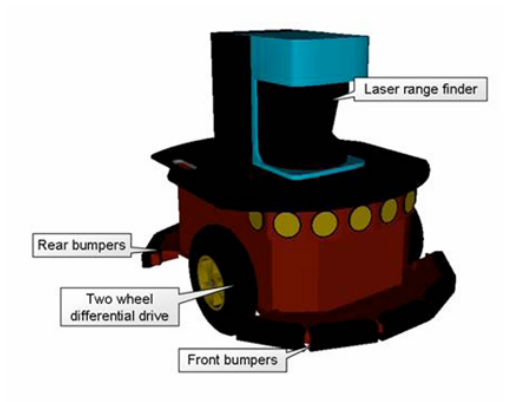

Figure 3.3: Simulated Pioneer 3DX robot [44].

Robotic Search and Rescue Simulation Toolkit (RSARSim)

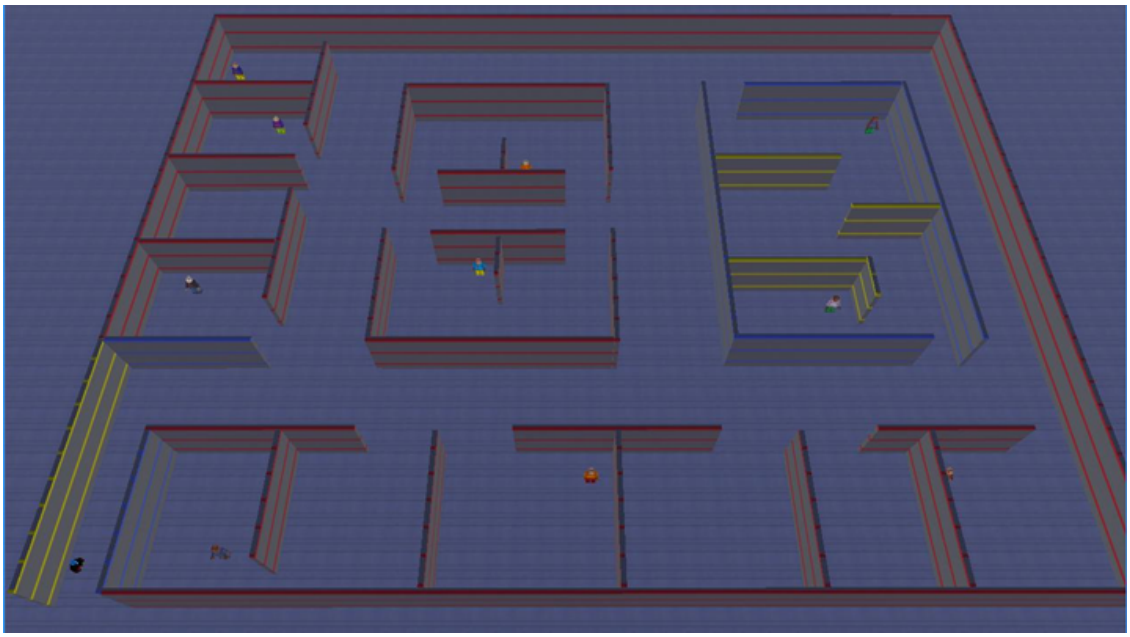

Figure 3.4: An search and rescue scenario simulated using RSARSim. The user is controlling a single Pioneer robot.

Since none of the existing simulation environments satisfied all of our experimental needs, we created a new environment, RSARSim. The Robotic Search and Rescue Simulation Toolkit 1.0 (RSARSim 1.0) was designed to be used in conjunction with MSRDS to further research and development in Human Robot Interaction (HRI) paradigms for the search and rescue domain. RSARSim (implemented using the Visual Simulation Environment) makes it easy for developers to implement their algorithms and evaluate their techniques on human subjects without a multi-robot system. RSARSim 1.0 is fully compatible with all versions of MSRDS including the free express edition and can be downloaded at http://ial.eecs.ucf.edu/Projects/RSARSim. Additionally this simulator can serve a prototyping and practice environment for teams competing in the physical Robocup Rescue competition, since code developed in MSRDS can be made to be fully compatible with the physical robot hardware. 3D models and entities for the rescue level (shown in

Figure 1) are specified in a Visual C# file, RescueLevel.cs; this file is called by RobotInterface.cs file to start the simulated environment in our toolkit. The environment is divided into different color zones similar to the RoboCup Rescue levels. The walls in the RSARSim have colored lines that correspond with the autonomy/mobility zones used in the Robocup Rescue competition: yellow, orange, and red. For instance, the yellow zone requires that the robots search this area fully autonomously; failure to do so will result in no points when a victim is found, but the victim will still count as being found. The red and orange zones pose mobility challenges, but allow users to teleoperate the robots. In our environment, walls with blue lines on them can be used to denote an upcoming zone change. This blue zone gives the operator time to switch from teleoperation to autonomous. Victims are represented using person shaped models distributed to the locations specified in the RescueLevel.cs file.

Home and Urban Intelligent Explorer (HU-IE) First Generation

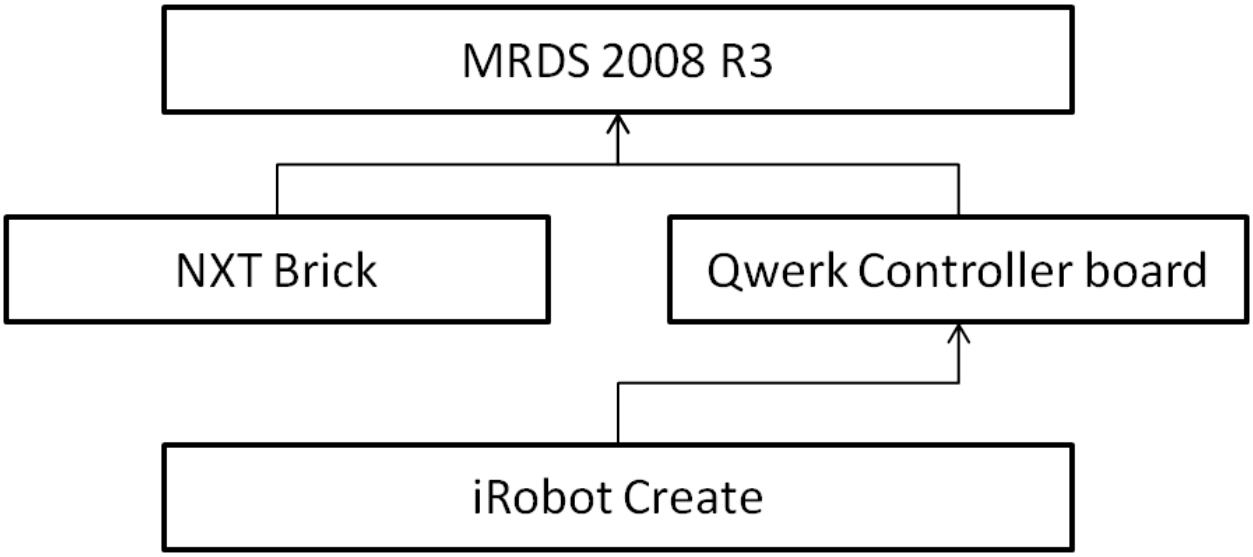

Figure 3.5: Overview of HU-IE Robot System hardware components.

The Home and Urban Intelligent Explorer (HU-IE)1.0, features a mobile base attached to an arm and gripper (Figure 3.6). It is designed to be able to retrieve light objects in a household environment with either carpets or hard floors. We constructed our robot using three components: iRobot Create, Charmed Labs' Qwerk board [50], the arm from the NXT 2.0 Robotics Kit, and a Logitech Communicate STX Webcam.

The robot base consists of the following components:

Actuator: The iRobot Create has a differential drive that allows left and right wheel speeds to be independently specified.

IR Sensor: There is one IR sensor on the front left of the robot that can be used to detect walls and other robots. Objects do not always register on the IR sensor and can only be reliably detected by the operator using the camera.

Bump Sensor: The Create has a left and right bump sensor that trigger during physical collisions.

Cliff Sensor: We use the cliff sensor under the base of the robot to detect whether the robot has been lifted up and moved (e.g., between trials). In a household environment, it would be used to detect proximity to staircases.

Webcam: A camera mounted on the robot arm presents a first-person perspective to the user during teleoperation. The user can also access the feed from a ceiling camera to obtain an overhead view of the workspace and both robots.

Qwerk: The Qwerk board is a 200 MHz ARM9 RISC processor with MMU and hardware floating point units running Linux 2.6. For our purposes it functions a relay, forwarding sensor information from the Create sensors and webcam to the user interface.

The arm on the HU-IE robot was created using the LEGO NXT Robotic Kit. It is 1.2 feet long and extends 8 inches in front of the robot. The arm is actuated using three motors, can rotate 360° around the robot base and has an operating range of -45°–90° in elevation. At the end of the arm is a four tong claw with rubber grips capable of grasping objects sized for a human hand. An NXT intelligent brick, containing a 32-bit ARM7 microprocessor, functions as the brain of the robotic arm, connecting all the actuators together. Commands from the user interface are sent directly to the arm via Bluetooth, bypassing the Qwerk board. The webcam is mounted on the arm to enable the operator to view the object from the arm's perspective.

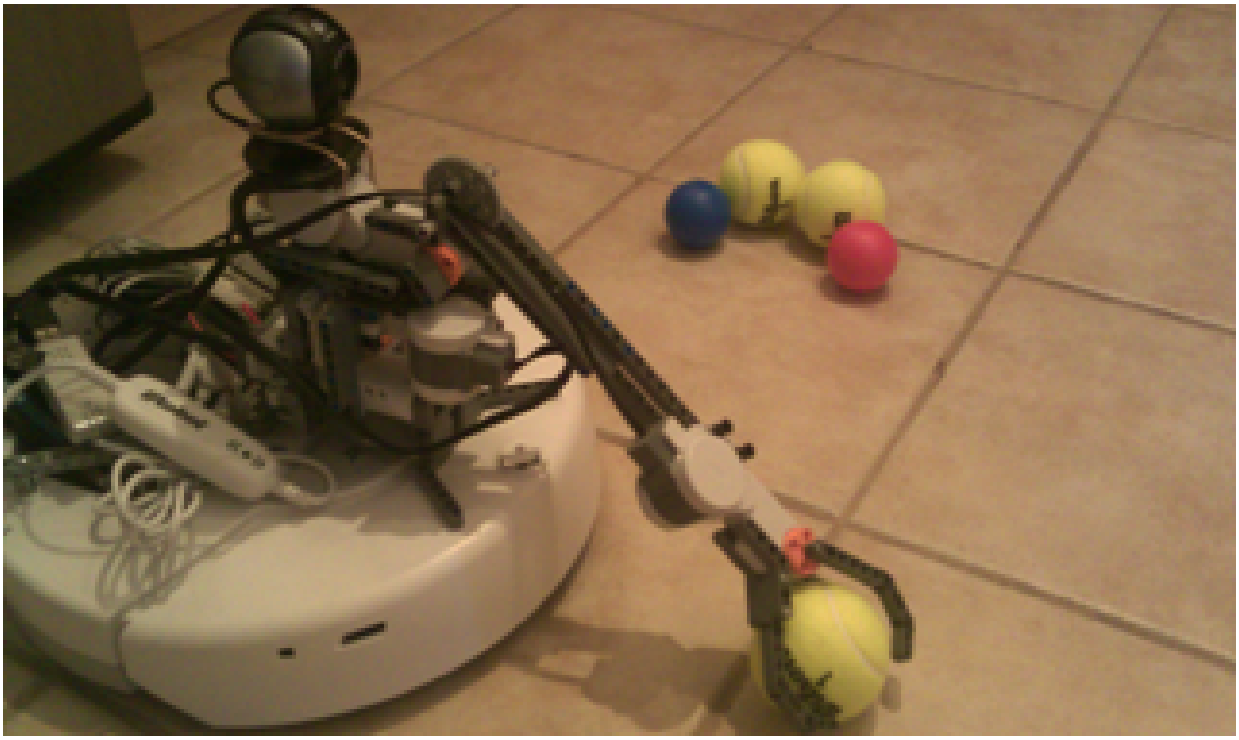

Figure 3.6: HU-IE combines a mobile base (3 DOF) with a robot arm (2 DOF) equipped with a gripper. This enables HU-IE to navigate indoor environments and pick up small objects. The user can wirelessly teleoperate the robot using a webcam.

Home and Urban Intelligent Explorer (HU-IE) Second Generation

Due to the lack of sensors, processing power, and the obsolescence of the Qwerk board, we decided to upgrade the HU-IE 1.0. The Home and Urban Intelligent Explorer (HU IE) 2.0 includes the following components: an iRobot Create, Acer Aspire One netbook, the NXT 2.0 Robotics kit, a Logitech Communicate STX webcam, Turtlebot shelves, and Tetrix Robotics parts. The total cost per robot is around US $1000. Figure 3.7 shows the robot architecture.

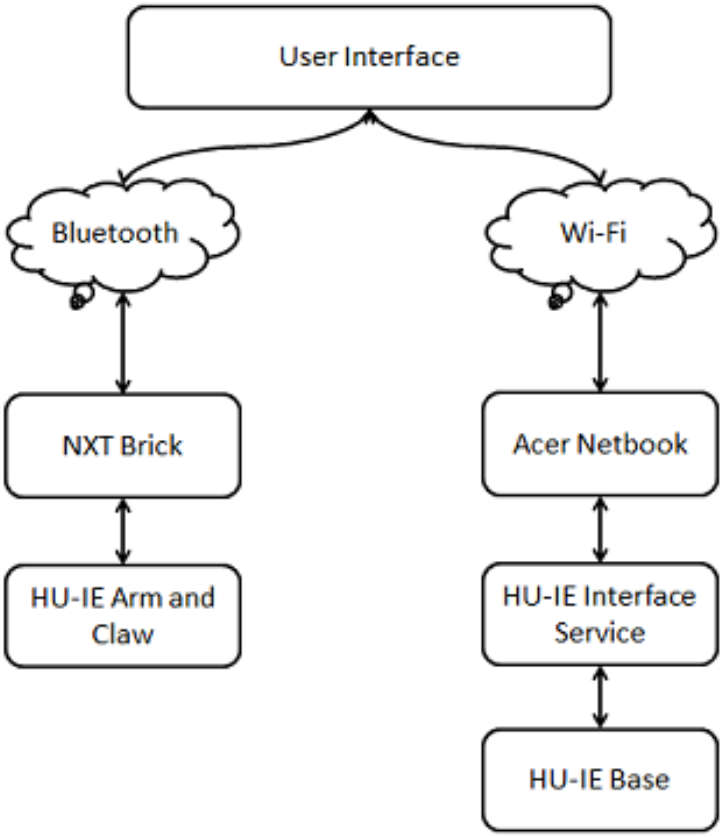

Figure 3.7: Connections between the HU-IE robot hardware components.

In addition to the internal Create sensors, we added an ultrasonic sensor mounted on the claw of the robot to determine the distance between the claw and the pickup object along with an accelerometer to measure the arm angle. An Acer netbook (Intel Atom 1.6 GHz processor with Windows 7) functions as a relay forwarding forwarding sensor information from the Create sensors and webcam to the user interface.

The arm on the HU-IE robot was created using the LEGO NXT Robotic Kit. It is 1.2 feet long and extends 8 inches in front of the robot. The arm is actuated using three motors and has an operating range of -45° to 90° in elevation. At the end of the arm is a four tong claw with rubber grips capable of grasping objects sized for a human hand. Textrix Robotic Metal parts are used to bolt the arm to

the iRobot Create and serve as the rigid structure of the arm. A NXT intelligent brick, containing a 32-bit ARM7 microprocessor, is used to control the arm and communicate with all the sensors and actuators. Commands from the user interface are sent directly to the arm via Bluetooth, bypassing the Acer netbook.

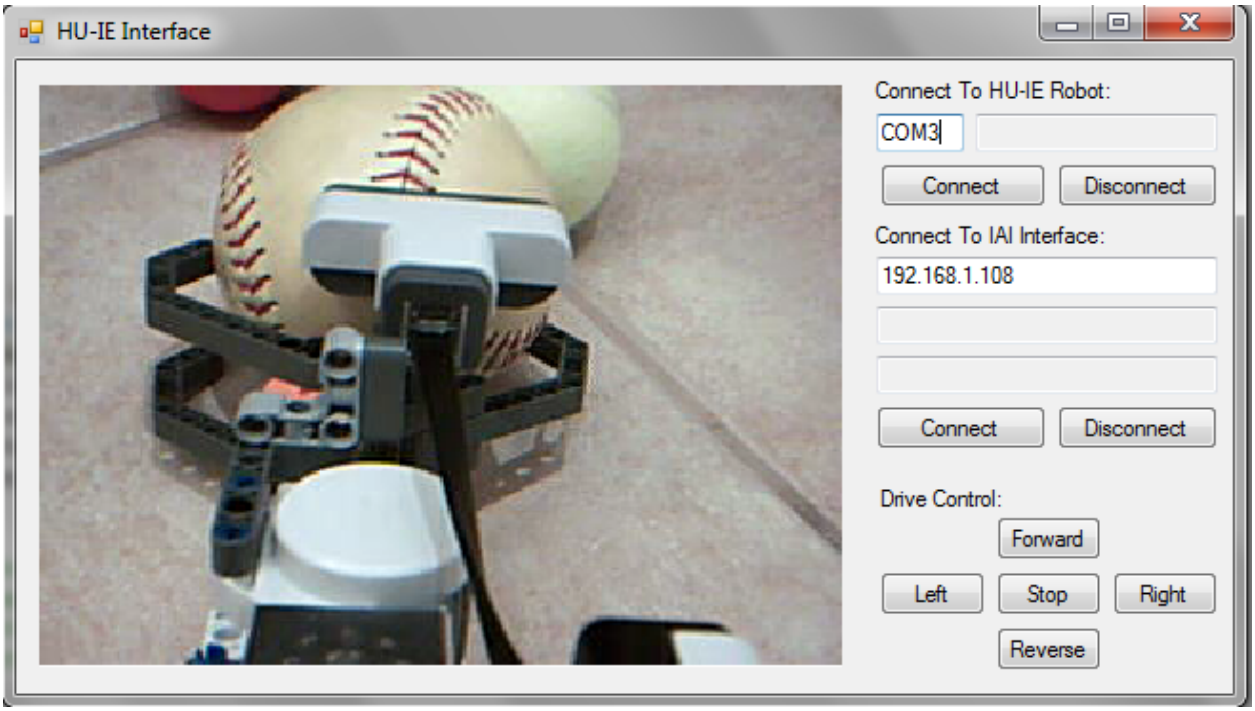

Figure 3.8: The user views the object being manipulated through a webcam mounted on the robot's arm.

The robots' workspace is monitored using a separately mounted Microsoft Kinect sensor. The Kinect provides RGB-D data directly to the user interface which uses it to track and display the location of the objects in the area. The position of the robots, based on the internal Create odometry, is marked on an occupancy grid and verified with the Kinect sensor. A modified blob detection technique is used to detect the other objects in the environment.

Figure 3.9: Two robots cooperate to lift an object under the direction of the human operator. In the multi-robot manipulation task, the robots must lift and deliver a series of objects of different sizes to the goal location.

Home and Urban Intelligent Explorer (HU-IE) Third Generation

Due to the increasing amount of manipulation that was needed for task completion we upgraded the Home and Urban Intelligent Explorer (HU-IE) to include two arms on each robot, in contrast to the HU-IE 2.0 which only had one arm per robot.

Our robot was constructed using the following commercially available components: an iRobot Create, Acer Aspire One netbook, two NXT 2.0 Robotics kits, a Logitech Communicate STX webcam, Turtlebot shelves, and Tetrix Robotics parts. The HU-IE 3.0 has similar parts has the HU-IE 2.0, and a complete design for our system is available at: `http://ial.eecs.ucf.`

edu/HUIE.php. Having two arms gives the users more choices on how to parallelize the task, but makes the teleoperation problem slightly more difficult. Even with two arms, some objects are still too large to be carried by a single robot.

Figure 3.10: The multi-robot delivery task requires the operator to control the pair of robots to move objects to a goal location. Objects too large to be lifted by a single robot (such as the box shown in the photo) must be delivered by the team of robots driving in tandem under the direction of the human operator.

CHAPTER 4: A SIMPLE USAR USER INTERFACE

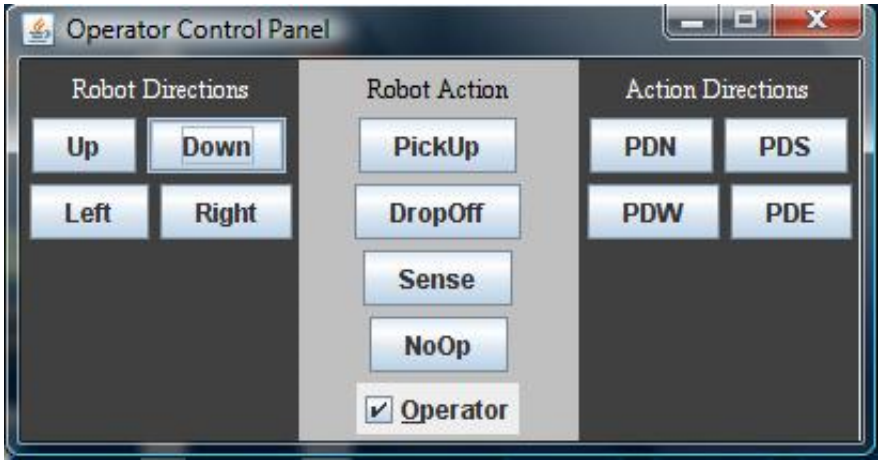

Figure 4.1: Operator Control Panel

Good human-robot collaboration for search and rescue is a must for the success of the application. Controlling multiple robot agents increases the difficulty for the human operator. The attention of the human-operator has to be split between multiple robots making the risk of a robot remaining idle greater then if the human was only controlling one robot. This section describes two approaches to help the human operator control multiple robot agents better; by removing human-operator dependence from the robots and creating a better user interface for the human-operator to control multiple robots.

We developed a coordination method for autonomous agents and designed a user friendly GUI control system. The robotic agents' task is to find and rescue victims autonomously while the operator is not controlling them. The human operator can perform the following actions while controlling any of the robotic agents: 1) pickup and drop-off victims or objects, 2) navigate and sense for victims, and 3) switch between each robotic agent. This project utilized the GSAR simulator which models the environmental information of injured victims being trapped in an

urban disaster. The information generated by the GSAR simulator consists of data that is loaded with the Grid world map (.gw file). The simulation allows the user to load as many created agents as needed into the simulator and execute search and rescue scenarios in which the user must rescue the victims within a specified timeframe. Each robotic agent has a GUI control panel that is used by the human operator. The human operator had to type in commands from the console command line to control an agent. The operator/robot control panels makes it easy to control the agent with just a click of the mouse. Since the GSAR simulator is a turn based simulator the operator/robot control panels wait for the human operator to perform its action before the next time step. If the operator doesn't want to perform anything for that specific time step the operator can just press the No operation button. When the human operator wants to switch between the operator/robot control panels, the human operator simply unchecks the operator check box on either control panel and can resume control by rechecking the check box. The operator can only control one robotic agent at a time.

While the user is not teleoperating a Robotic agent via the operator/robot control panels, the auto agent mode is activated. This method can perform all the operations that the human can perform automatically except for switching from operator to auto. While the auto agent mode is initialized, it randomly moves the robotic agent around the map from one empty grid to another in search of victims. It makes moves based on the finite state machine such that if the surrounding grids are empty and it has not been to that spot before it moves to that spot. If there is a victim in a grid that is next to the robotic agent and it is not carrying anything at that time the robotic agent will perform a pick up action on the victim. Once a victim is picked up the robotic agent will start randomly moving around until it finds an exit. While in auto agent mode, if the robotic agent has a victim and is next to an exit it will perform the drop off action. If the robotic agent is trapped and cannot perform a move it will perform the no operation function till the next time step. To evaluate the operator/robot control panels and auto agent mode we performed a informal study that consisted

of six users teleoperating the operator/robot control panels and using the auto agent mode in the GSAR simulator environment.

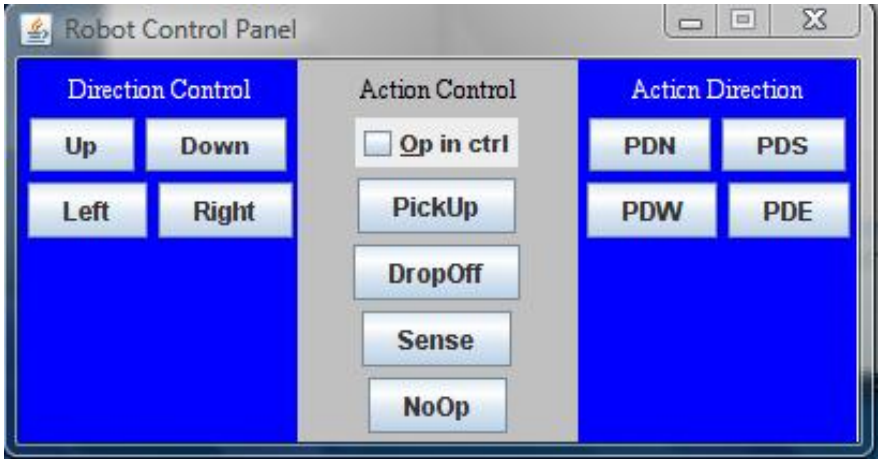

Figure 4.2: Robot Control Panel

In our study we established the dilemma that human operators have when trying to operate multiple robots simultaneously and discovered the simplest resolution for the human operator to control multiple robots in a trouble-free and cost effective manner. In our experiment we combined two approaches, a control panel GUI for better control of the robot and coordination automation that will allow the robots to roam the map and find victims automatically without the use of the human operator.

CHAPTER 5: MULTI-ROBOT COORDINATION IN A COMPLEX USAR SCENARIO

Controlling multiple robotic agents increases the complexity of the human operator's control task since the human operator's attention has to be divided between several robots. Without a good user interface, adding more robots to the system will not necessarily increase the area that the robots can cover within a bounded period of time; unmanned robots need to be able to explore the environment effectively when the operator's attention is elsewhere. Using RSARSim, we developed and evaluated two possible user interfaces for controlling a Robocup Rescue team—one using an Xbox 360 gamepad controller and the other based on a keyboard control paradigm.

Having the capability to prototype human-robot interfaces, experiment with different control paradigms, and train an operator to rapidly find victims with a specific user interface is a valuable resource for Robocup Rescue developers. To do this we use the simulation toolkit that we have implemented, RSARSim (Robotic Search and Rescue Simulation) for simulating multi-robot systems and evaluating user interfaces. RSARSim was designed to be used with Microsoft Robotics Developer Studio 2008, a flexible and powerful Windows-based environment that allows developers to create robotics applications for a variety of hardware platforms. Although there are other publicly available USAR simulation systems, ours is the only one currently available for Microsoft Robotics Developer Studio. We describe the implementation and usage of RSARSim for evaluating human-robot interfaces in an urban search and rescue domain.

Using RSARSim, we were able to prototype two possible user interfaces for multi-robot control suitable for human teleoperation of a Robocup Rescue team. In the simpler user interface, the user controls the team of four simulated Pioneer robots using the standard first-person shooter controls on a keyboard. In the second condition, the human operator navigates the robot through

the RSARSim environment via a gamepad base approach using an Xbox 360 controller. The system implementation has sixteen individual developed Microsoft Robotics Developer Studio services that it uses.

OnStartService: establishes the startup connection.

OnLoad and OnClose: used to handle the loading and closing of the rescue robot control panel.

OnChangeJoyStick: used to handle the connection of Xbox 360 controller.

OnConnectMotor: used to handle the connection of the drive function.

OnMove and OnEstop : manages the start and stop of the motor of the four Pioneer 3DX robots.

OnConnectSickLRF and OnDisConnectSickLRF: manages the connection and disconnection of the laser range finder of the four Pioneer 3DX robots.

OnConnectWebCam: manages the webcam connection for the four Pioneer 3DX robots.

OndisconnectWebcam: handles the disconnection of all cameras and the OnQueryFrame service executes the frame updates for individual cameras.

Since the predefined webcam service was designed for a single robot, we developed a different webcam service for RSARSim, that inherits from the MSRDS predefined webcam service. Our webcam service gets the latest frame and uploads the image to the corresponding image box on the rescue robot control panel. Since each robot has its own camera, our webcam service generates four services to handle all the cameras. When the webcam service is called, it automatically determines the correct service for the specified robot.

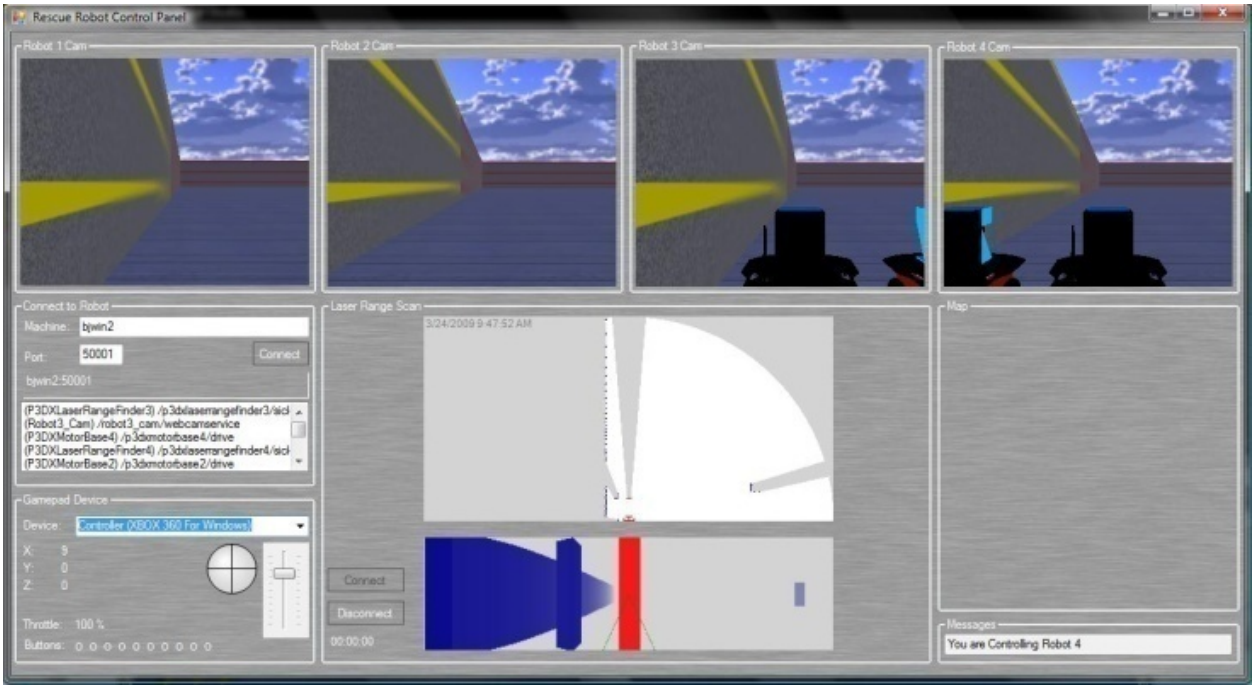

Figure 5.1: Robot Rescue Control Panel.

Our game controller service also inherits some traits from the predefined controller service included with MSRDS. The rescue robot control panel, the button configuration for the gamepad, and the keyboard controller are specified in the DriveControl.cs file. The layout of the gamepad was based on the layout of the Xbox 360 Controller. To connect the rescue robot control panel to VSE, the operator must press the start button on the controller. The four colored buttons each represent a robot. If pressed, the button allows the human to teleoperate that robot, and the control panel displays the corresponding sensor information. To drive that robot the operator presses the RB button and steers the robot using the left analog stick; pressing the select button disconnects the laser range finder service. To control the robot using the keyboard, the user selects the robot drive service using the list box on the rescue robot control panel and drives the robot using the W,A,S,D keys.

We evaluated an initial group of human subjects on their ability to locate victims while teleoperating 4 simulated Pioneer 3DX robots using two different low-level teleoperation paradigms: 1) a keyboard based user interface and 2) the Xbox 360 gamepad controller. Preliminary results indicate that the gamepad controller is preferred by the majority of subjects and results in a higher number of victims found. Using RSARSim, we were able to rapidly evaluate the comparative effectiveness of two different HRI paradigms for multi-robot control.

CHAPTER 6: LEARNING MODELS OF OPERATOR DISTRACTION FOR HUMAN-ROBOT INTERFACES

Identifying when the operator is distracted and mismanaging a particular robot is a non-trivial task since many observable features such as eye movement and mouse motion relate to the users general cognitive state and are difficult to associate to a particular robot. It is possible to calculate task progress based on observing the robots and recommend alternate allocations, but in cases where solving the robot task allocation problem is difficult, incorrect re-allocations can jeopardize an already good solution identified by the user. Since the user interface is most valuable in the cases where automatically calculating the task allocation is difficult, it is counterproductive to eliminate unique solutions suggested by the user. The key contribution of this work is to automatically infer from motion trajectories which robots in the team are being neglected by the user and to allocate those robots effectively.

We are able to learn and infer when the operator is distracted from a particular robot using a hidden Markov model embedded into an agent-based user interface (the CoOperator). During a short training period, the user interface artificially distracts the users attention from the robots while collecting on the robots trajectory which is then used to learn parameters for the model.

The CoOperator interface is designed to assist the human operator in managing, controlling, and navigating multiple robotic agents through the disaster scenario. The physical interface to the system consists of an Xbox 360 gamepad, from which the operator can select a robot in the team and give it commands. To enable the human to simultaneously move multiple robots, the standard teleoperation interface allows each robot to be placed in a low-level wander mode, where it continues moving in the specified direction while avoiding obstacles.

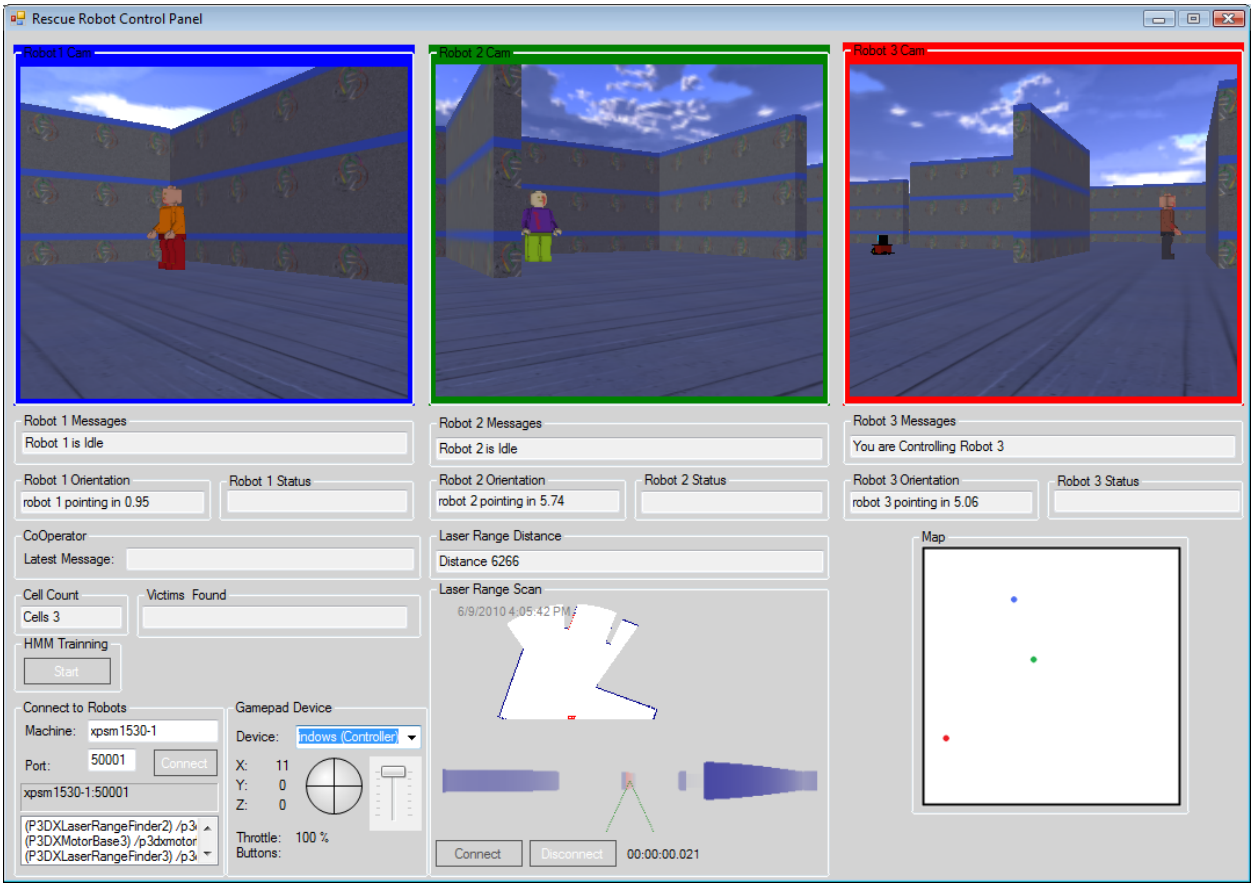

Figure 6.1: The CoOperator agent-based interface for multi-robot teleoperation. The human operator controls a team of three robots, each of which sends a first-person perspective of the environment from its onboard camera. The CoOperator agent infers operator distraction and identifies whether a robot is not being effectively managed. These under-utilized robots are guided along a path that complements the human's explicit teleoperation.

CoOperator augments this basic teleoperation interface in several important respects. First, it monitors each of the robots in the team and determines which ones are suffering from operator neglect. This is non-trivial since a robot in wander mode receives no explicit instructions from the operator yet can be effectively exploring the environment. We employ a Hidden Markov Model to infer operator neglect from the robot's observable state. Second, the CoOperator agent assumes control of robots while they are unattended. Specifically, it overlays a higher-level goal on the wander

behavior to ensure that the robot explores a series of waypoints in its assigned region of the disaster zone. This is done by mapping the immediate environment and employing A* to find efficient routes between waypoints. Third, it transparently and responsively cedes control to the human operator whenever the user sends the robot an explicit teleoperation command. This means that the CoOperator agent is not in conflict with the human's higher-level goals. One of the CoOperator agent's primary goals is to determine whether a given robot is currently under active human supervision (as opposed to explicit teleoperation). This is more subtle than simply observing whether the robot is moving because operators typically place robots in wander mode as they switch cognitive contexts between robots.

We employ a Hidden Markov Model (HMM) [51] to infer, based on observable state, whether it is likely that the given robot could be a good candidate for CoOperator control. A similar inference approach was successfully employed to infer cognitive load in human-agent teams [19]; however our observed features and user task are quite different. We use a straightforward model for the operator's (hidden) cognitive state, as it applies to a particular robot. The three states are: (1) **A**ctive, referring to a robot that is under active teleoperation control; (2) **M**onitored, for a robot that is moving without explicit control, but that is acting in a manner consistent with the user's higher-level goals; (3) **I**nactive, for a robot that is not currently being monitored by the human operator. The transition matrix between these states is illustrated in Figure 6.2.

We quantize the observable robot state into three discrete symbols that characterize the instantaneous activity of the robot into three levels: low, medium and high, with emission probabilities summarized in Table 6.1. The intuition is that an unattended robot should exhibit a lower level of activity over a period of time. We sample each robot's activity approximately once a second. At the start of the scenario, we initialize all of the robots in the team to the **I** state. Inference is performed over observation sequences of length 10, corresponding to a 10-second observation window. The CoOperator agent applies a standard Viterbi search to identify the most common sequence of hid-

den states to explain the observation sequence and determines that a robot is neglected if its current hidden state is estimated to be **I**nactive.

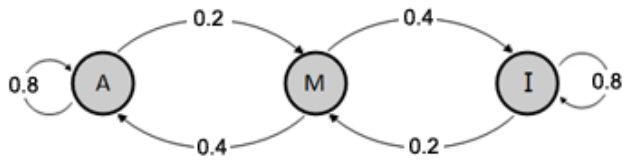

Figure 6.2: HMM states and transition probabilities.

Table 6.1: Observation probabilities matrix.

	Low Activity	Medium Activity	High Activity
Active	0.05	0.3	0.65
Monitored	0.3	0.4	0.3
Inactive	0.65	0.3	0.05

As outlined above, the CoOperator agent moves an unattended robot to various waypoints in its assigned regions in the disaster scenario and initiates a directed exploration in the local neighborhood of each waypoint. At the start of the scenario, the region is roughly partitioned among the robots in the team, based solely on gross geometric characteristics (i.e., without detailed map information). Given this partition, each robot independently generates a set of waypoints that cover its assigned space. The basic idea is to exhaustively explore the region near each waypoint and then move efficiently to the closest unexplored waypoint.

The local search in the neighborhood of each waypoint is a simple outwards spiral that looks for victims while avoiding obstacles. This is a predictable search pattern that the human operator can supervise with minimal attention using peripheral vision. Once the region around the waypoint has been swept clear, the robot plans a path to the next waypoint.

For path planning, the CoOperator agent employs a standard discretized A* search from its current location to each of the unexplored waypoints. Unexplored waypoints in the robot's assigned zone are given priority over those assigned to other robots. The robot then follows the efficient route to the next waypoint without specifically searching for victims; any victims that are fortuitously encountered en route are tagged.

The CoOperator agent's current state (whether it is traveling between waypoints or executing a search) is discreetly shown on the user interface so that the operator can determine, at a glance, whether to resume explicit control of a given robot.

Our goal was to study the benefits of employing an agent-based assistant for improving multi-robot teleoperation. The baseline system, denoted as *manual*, consisted of the standard teleoperation setup described earlier: three Pioneer robots operating in a simulated disaster scenario, controlled by a single human operator using an Xbox 360 gamepad. In the baseline condition, the operator could place the robots in wander mode and thus get better utilization and coverage from two robots while the third was under explicit control.

The augmented system, denoted as *CoOperator*, augmented the *manual* configuration with an agent per robot that would infer whether the robot was being effectively utilized by the operator. Inactive robots would be directed to travel to unexplored waypoints and perform a spiral search strategy.

Four indoor scenarios, consisting of different topological configurations of connected rooms (see Figure 6.3) were employed in our user study. These were presented to users in a random order.

Each participant in the study was processed using the following protocol. Participant information (such as prior familiarity with game interfaces) was collected at the start of the experiment via a pre-questionnaire. A training session explaining how to use the interface was given. While in

training, the participants followed instructions and practiced each of the available operations to become familiar with the navigation controls and the rescue robot control panel. After training, each participant was assigned to find victims in four different scenarios (Office1, School, Hotel, and Office2), presented in a random order.

The CoOperator agents were enabled on two of these four runs, and the operator was restricted to manual mode on the other two runs. The participant had 15 minutes on each scenario to locate the victims. Paid participants were recruited from a university to participate in the experiment. At the end on the experiment the users completed a post questionnaire to elaborate on any problems that they encountered and to explain their strategy.

We measured performance using two metrics: (1) the time taken to find all of the victims in each scenario, bounded by 15 minutes; (2) the area covered by the robot team during a given scenario, as measured in terms of unique discretized cells. We present our findings in the next section.

Table 6.2 summarizes the demographics of our user study. Half of the participants had limited experience with game controller interfaces.

Table 6.2: Demographics and experience level of the user-study participants.

Participant Demographics and Experience					
Age	Gender		PC Game Skill Level		
23–27	Male	Female	Expert	Inter.	Beginner
8	6	2	3	1	4

Figure 6.4 shows a scatter plot of the time taken to find all of the victims in a given scenario (metric 1). We see that in the majority of runs, the agent-assisted CoOperator condition accelerates the robot team's progress. We can attribute this to the fact that the robots controlled by the CoOperator agents continue to search with minimal human supervision while the inactive robots in the manual

condition (whether stationary or wandering) are less effective.

We analyzed the data further to see whether the benefit of the CoOperator agents changes with operator experience on the task. Figures 6.6 and 6.7 show histograms showing the fraction of users for whom Manual and CoOperator led to faster rescue times. In general, users were able to complete the task faster on their second run; however, the fraction of users for whom CoOperator helped remained the same. An important difference between the initial and subsequent runs was that operators had a greater number of user-initiated accidents (Pioneer robot flipped over) in their initial run.

We also analyzed our data according to the second metric, map coverage (see Figure 6.8). Here we see that the number of map cells explored by the robot team in each of the different scenarios correlates less well with the experimental condition. In other words, a robot team employing manual+wander teleoperation is able to explore approximately the same area as one that has been augmented with CoOperator. The scenario length of 15 minutes was therefore sufficient (in terms of coverage) for either strategy. The difference is that the CoOperator agents direct the inactive robots to efficiently explore the space in a directed manner, resulting in a faster rescue of victims (as seen in Figure 6.4).

Our final results summarize the user satisfaction reported by participants in our post-questionnaire study (see Figure 6.9). We see that an overwhelming fraction (90%) of the participants expressed a clear preference for the agent-assisted (CoOperator) mode of teleoperation. 10% of the participants stated that CoOperator made no difference. None of the participants felt a negative impact from using CoOperator in these tasks. This validates our belief that allowing the inactive robots to operate effectively under minimal operator supervision is highly valued.

On our post questionnaire we asked the subjects about their usage of the CoOperator and their strategies for finding victims. Here are the comments ordered by subject number.

1. “I partition map area by robots and separated them to maximize the search area; also it compensated for multiple visits per victim by manually adjusting the robots path. I usually left 1 or 2 robots on CoOperator”.

2. “I go through the maze and try to remember which victim I already found. If the other robots were on CoOperator then if they saw a victim I would switch to that robot, find the victim, and then switch back to the original robot”.

3. “I would let the CoOperator handle two robots and go around the map the third robot in the opposite direction as the other robots”.

4. “I would let the CoOperator handle part of the maze while I search the other part”.

5. “In CoOperator mode I would travel in a circle and I would control a robot in one section and leave it there as a marker that I found all the victims in that section”.

6. “At first I relied on the CoOperator and my own memory of which victims I had tagged then later I tried to use the robots as landmarks, so I could avoid retagging and revisiting the same area over and over”.

7. “Send two robots around the perimeter of the map and one down the middle”.

8. “I control one robot and let the other to the CoOperator”.

Overwhelmingly the comments indicate that the subjects were adapting their own search strategies to effectively make use of the CoOperator and that the users had a general trust in the CoOperator’s ability to locate victims. On the post-questionnaire, 90% of the participants expressed a preference for the CoOperator condition.

RSARSim was used to develop and evaluate the CoOperator, an agent assistant for multi-robot teleoperation. The CoOperator explicitly infers the operator’s neglect level and increases the au-

tonomy of the robots. In our user study, the majority of the participants were able to locate victims more rapidly using the CoOperator and relied on it as part of their search strategy.

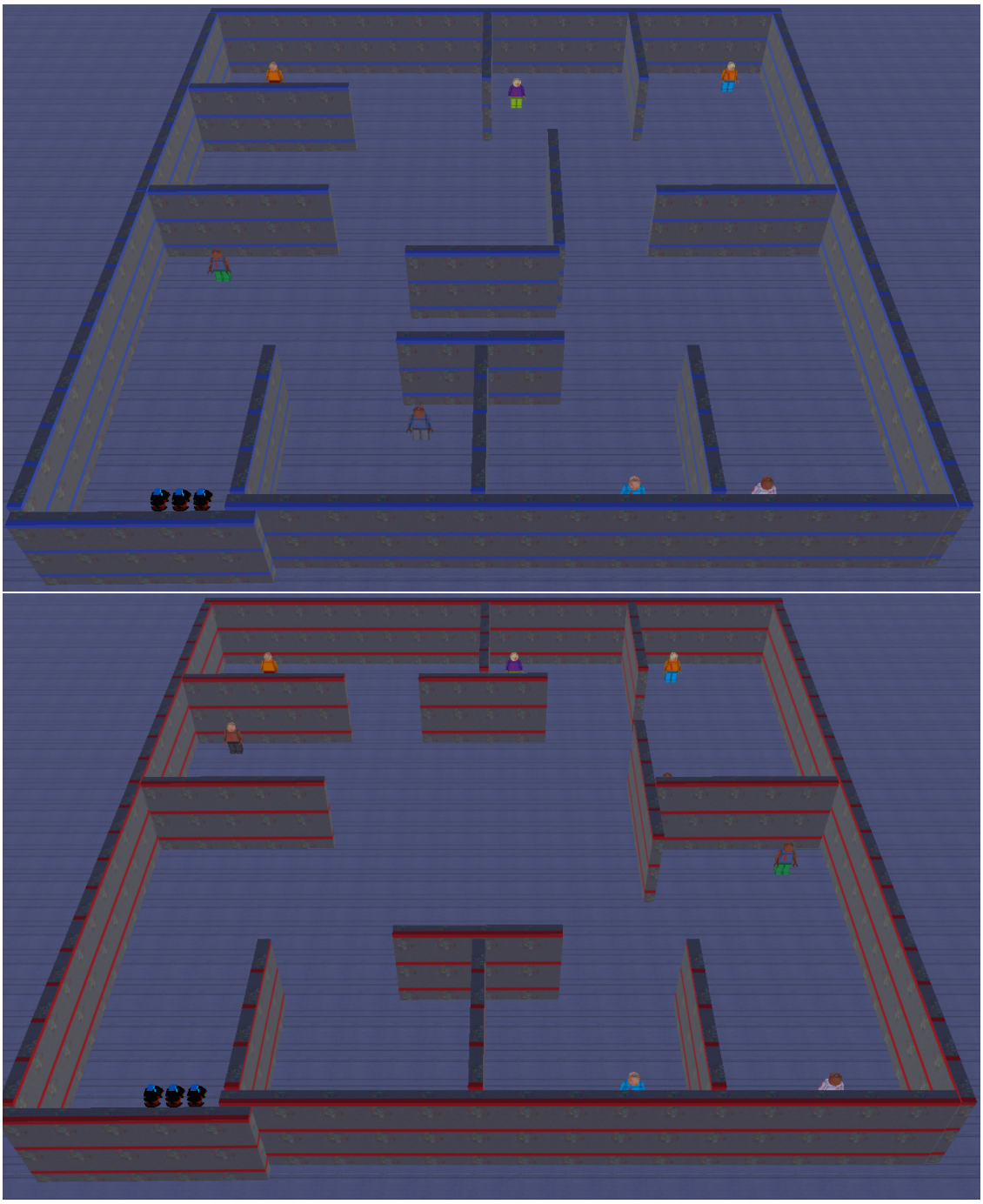

Figure 6.3: Overhead view of two scenarios in RSARsim. A team of three Pioneer 3DX robots is visible in the bottom left of each view and several victims that need to be rescued are scattered through the various rooms. Operators do not have access to this view and control the robots through a first-person perspective.

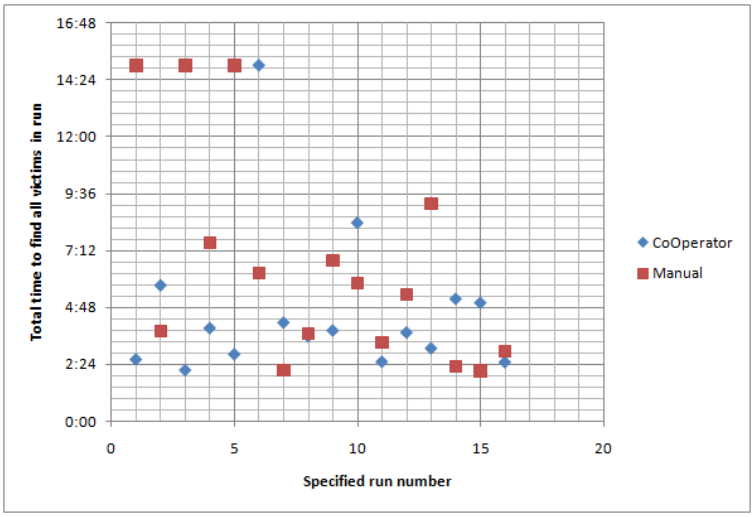

Figure 6.4: Scatter plot showing the time taken to find all of the victims in each scenario under the two experimental conditions: manual teleoperation (denoted as *manual*) and with the agent-assisted system (denoted as *CoOperator*). In the majority of scenarios, CoOperator significantly reduced the time needed to find the victims.

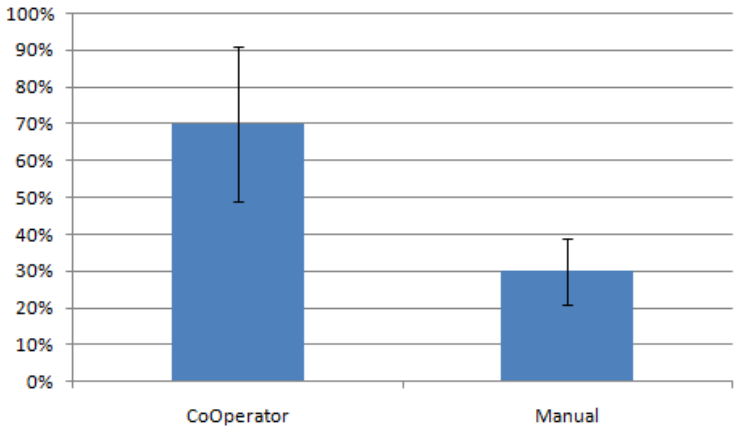

Figure 6.5: Fraction of users for whom each experimental condition led to a faster rescue (aggregated over all runs). Clearly, for a majority of users, the agent-assisted interface leads to faster rescues.

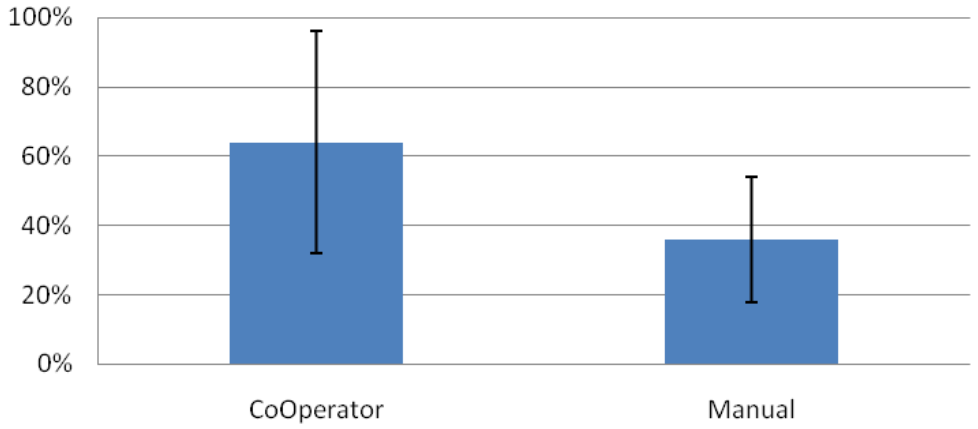

Figure 6.6: Fraction of users for whom each experimental condition led to a faster rescue (initial experience with system).

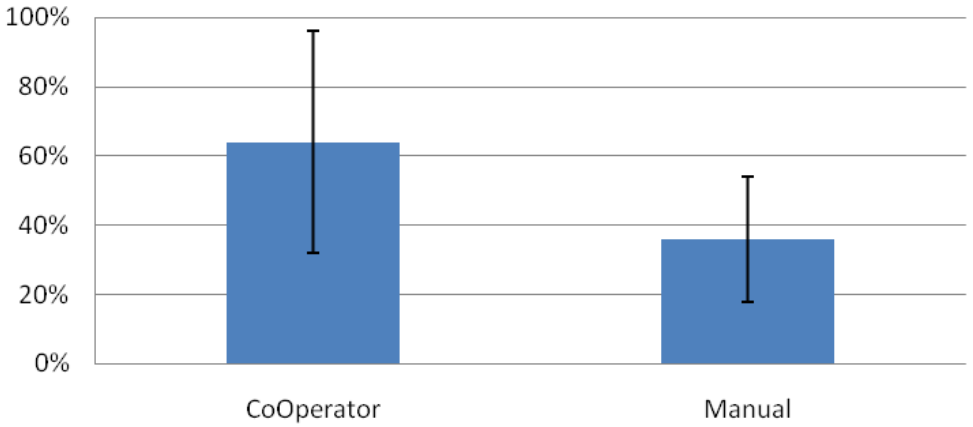

Figure 6.7: Fraction of users for whom each experimental condition led to a faster rescue (subsequent experience with system).

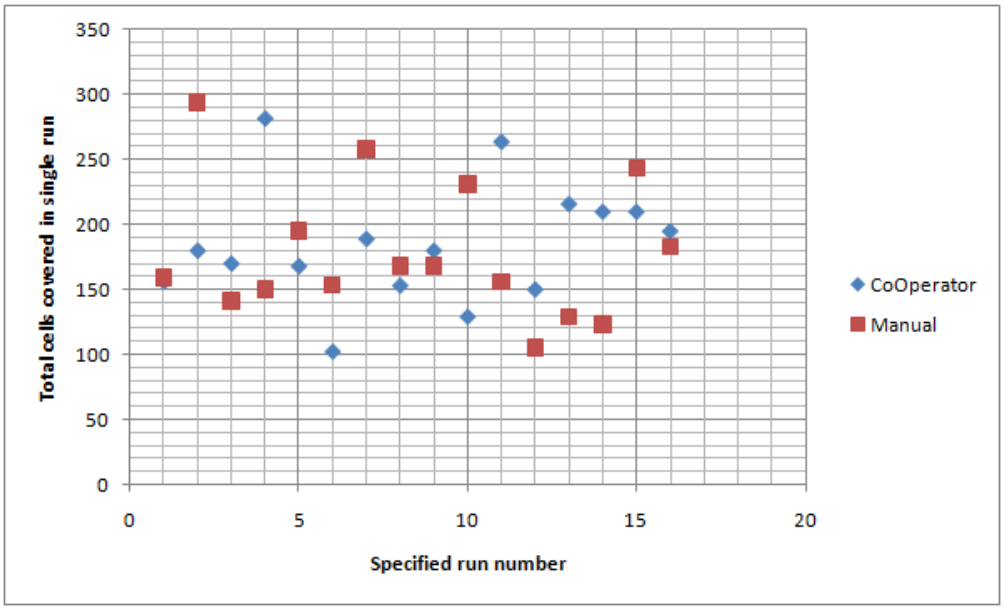

Figure 6.8: Scatter plot showing the area coverage by the robot team during a single run, under the two experimental conditions. We note that teleoperation with CoOperator does not necessarily cover a greater portion of the disaster area, even though it significantly reduces the time needed to find all of the victims.

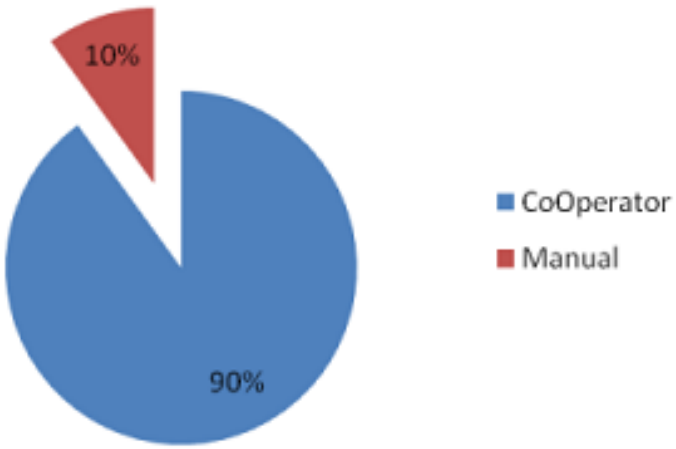

Figure 6.9: An overwhelming fraction of participants expressed a clear preference for using the agent-assisted (CoOperator) mode of teleoperation.

CHAPTER 7: ASSISTING USERS WITH MULTI-ROBOT MANIPULATION

In our multi-robot manipulation task, the user directs a team of two mobile robots to lift objects using an arm and gripper for transport to a goal location. The environment contains a heterogeneous selection of objects, some of which can be transported by a single robot and others that require both robots to lift. Figure 7.1 shows a picture of the team of robots cooperatively moving an object that cannot be carried by a single robot.

Our mixed-initiative interface provides the user with two important new cooperative functions: 1) autonomous positioning of the second robot (**locate ally**), and 2) a **mirror** mode in which the second robot simultaneously executes a modified version of the commands that the user has issued to the actively controlled robot. When the user requests help to move a large object, these cooperative functions enable the robot to autonomously move to the appropriate location, cooperatively lift the object and drive in tandem to the goal. The locate ally and mirror modes are created through intermittently propagating the user's command history across the robots. The unmanaged robot follows a simple learning-by-demonstration paradigm where it attempts to cooperate with the teleoperated robot, based on the user's current and previous commands.

The user views the environment and interacts with the robot team through our user interface (Figure 7.2), which was designed to minimize teleoperation workload. The operator issues controls to both robots through an Xbox 360 gamepad, using a button to switch between robots. Prior work on human-robot interfaces indicates that gamepad interfaces are generally preferred over keyboards/mice for mobile robot teleoperation [38].

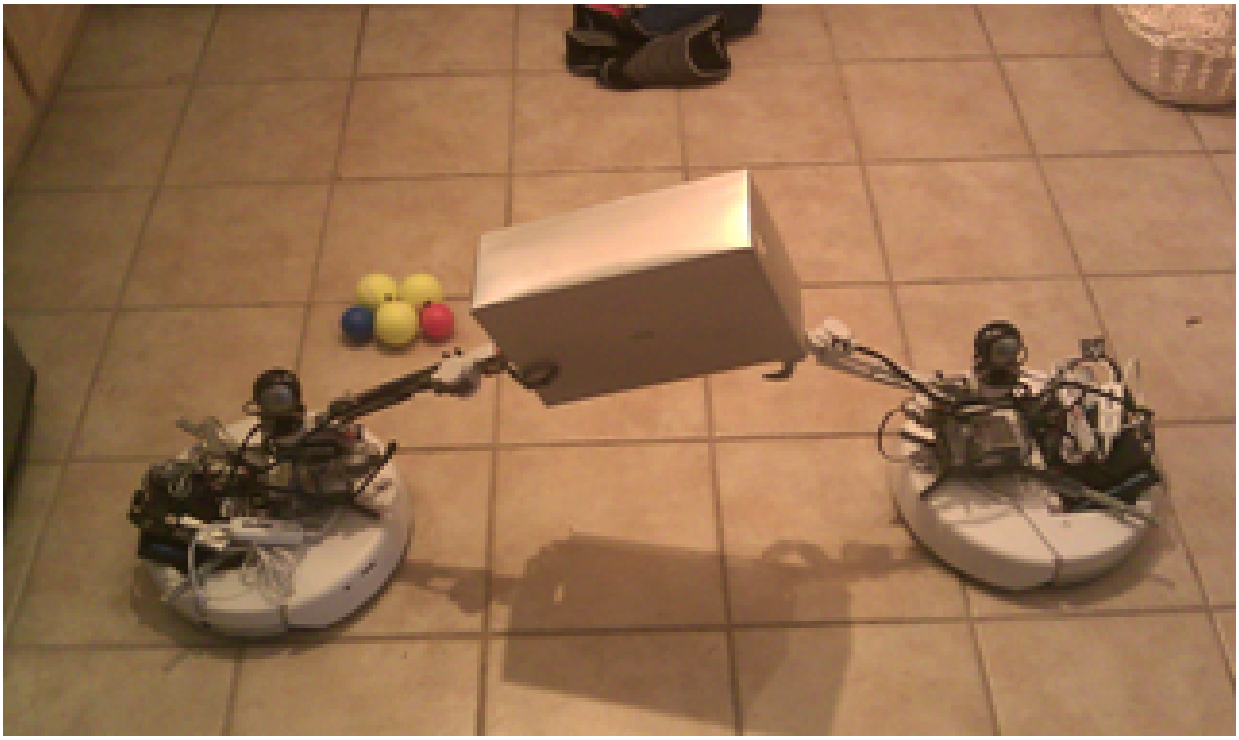

Figure 7.1: Two HU-IE robots cooperate to lift an object. One robot is teleoperated while the other moves autonomously to mirror the user's intentions. The user can seamlessly switch robots during such maneuvers.

The basic user interface requires the user to fully teleoperate both robots. In this dissertation, we present and evaluate the Intelligent Agent Interface (IAI) which adjusts its autonomy based on the user's workload. In addition to automatically identifying user distraction, the IAI leverages prior commands that the user has issued to one robot to determine a course of action of the second robot. To enable the human to simultaneously control multiple robots, the interface allows robots to be placed in a **search** mode, where the robot continues moving in the specified direction, while hunting for objects and avoiding obstacles. IAI monitors each of the robots and identifies robots that are ignored by the operator through measuring time latencies. It then assumes control of the unattended robot and cedes control if the user sends the robot an explicit teleoperation command.

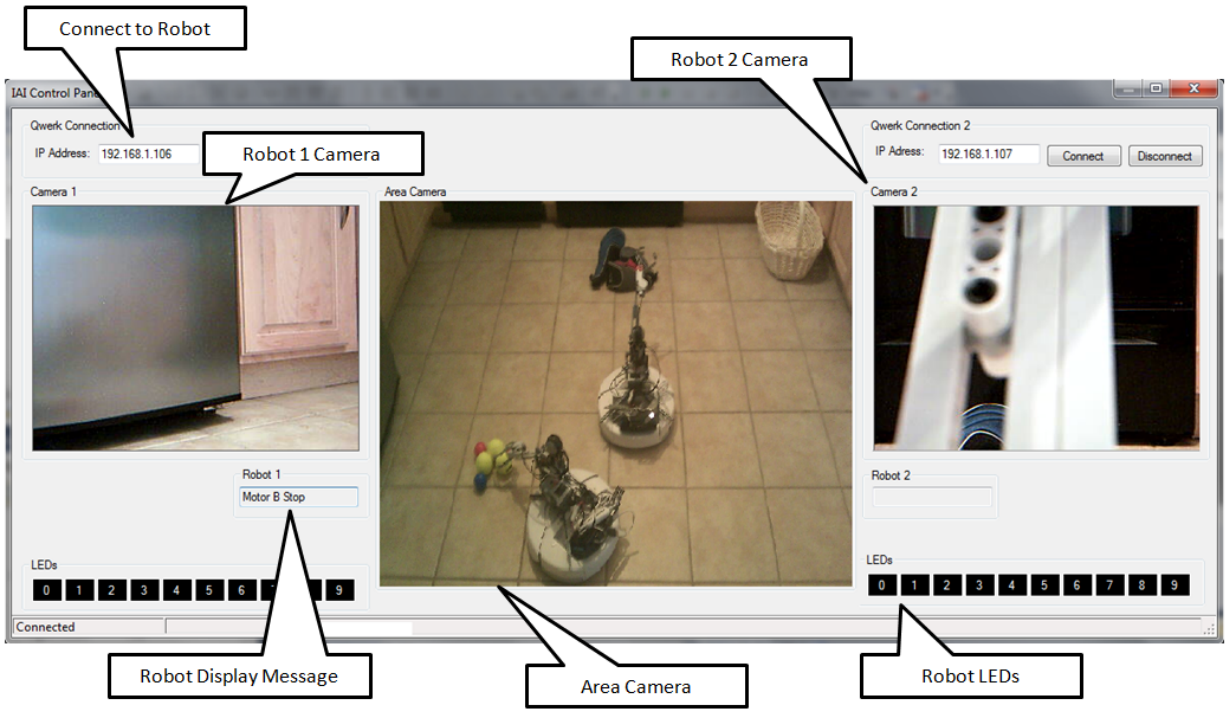

Figure 7.2: The intelligent agent interface is designed to enable the user to seamlessly switch teleoperation across multiple robots. The IAI supports a cooperative mode where the agent supports the user's active robot by mirroring its intentions.

The IAI provides the user with two important new cooperative functions: 1) autonomous positioning of the second robot (**locate ally**), and 2) a **mirror** mode. In these modes, the unmanaged robot attempts to learn through demonstration the intent of the user by monitoring the command history of the teleoperated robot. The autonomous robot uses a simple set of translation rules to modify the command sequence. In the **locate ally** mode, the robot positions itself on the side adjacent to the teleoperated robot, closest to the nearest pile of objects. For the **mirror** mode, it simply executes the same set of commands, corrected for differences in the robots' orientation.

Robots have the following modes of operation:

Search: the robots wander the area searching for objects.

Help: a robot enters this mode if the human operator calls for help using the gamepad or when the teleoperated robot is near an object too large to be moved by an individual robot.

Pickup: the robot detects an object and requests that the human teleoperate the arm.

Transport: the robot transports an object held by the gripper to the goal.

Locate Ally: the unmanaged robot autonomously moves to a position near the teleoperated robot based on command history.

Mirror: the robot mimics the commands executed by the teleoperated robot to simultaneously lift an object and transport it to the goal location.

In a typical usage scenario, the IAI moves the unattended robot around the environment in search of objects to be moved (clutter). At the start of the mission, the region is roughly partitioned into two areas of responsibility for exploration. Given this partition, each robot independently searches its assigned space. The robot's current state is displayed on the user interface for the benefit of the human operator.

When the user needs help manipulating an awkward object, the second robot can be called using the gamepad controller. The **Help** function can also be automatically activated by the IAI system, based on the other robot's proximity to large objects. Once in the **Help** mode, the robot executes the **Locate Ally** behavior. IAI maintains a history of both robots' navigational movements and uses dead reckoning to determine the teleoperated robot's position.[1] Each HU-IE robot has a cliff sensor, which when activated indicates that a robot has either been forcibly moved. If that occurs, the IAI system notifies the user to reorient the robot by driving it to its initial starting position. If the user is not actively soliciting help, the unmanaged robot typically moves into the **Search** mode;

[1] In indoor environments, the Create robot base only experiences a small slippage so the robot's position estimates are accurate to within a few cms.

once the robot detects an object, it notifies the user that it needs help with manipulation. After the object has been lifted by the user, the robot transports it to the goal. The aim of the IAI system is to smoothly transition between the unmanaged robot *rendering* help to the user and *asking* for help with the manipulation section of the task.

The human operator can also opt to put the other robot into **Mirror** mode. In this mode, the unmanaged robot intercepts the commands given to the teleoperated robot and attempts to duplicate them in its own frame of reference. This mode is essential for reducing the workload of the operator during cooperative manipulation, when two robots are required to lift the object. By combining the **Help**, **Locate Ally**, and **Mirror** modes, the robot can autonomously detect when its help is needed, move to the correct position and copy the teleoperated robot's actions with minimal intervention from the operator.

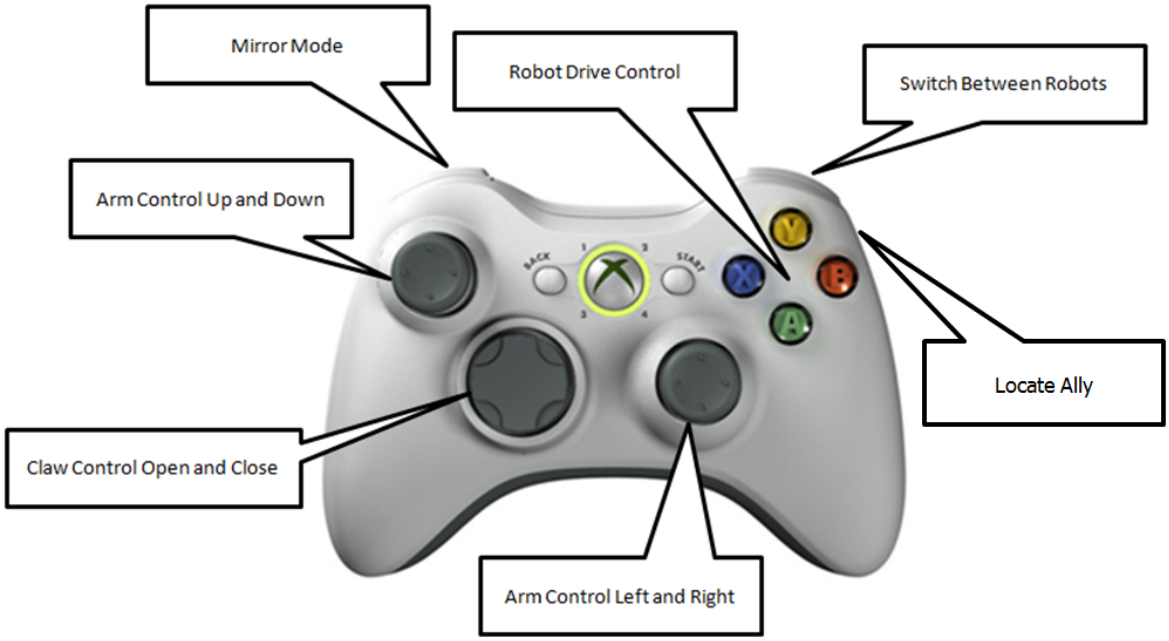

Figure 7.3: The physical interface to the IAI is through an Xbox 360 gamepad from which the operator can select a robot and send it teleoperation commands.

The operator controls the robots using an Xbox 360 Gamepad controller (Figure 7.3) as follows. The trigger buttons on the Xbox 360 controller are used to toggle teleoperation between the two robots and to activate the **mirror** mode in the unmanaged robot. The **A**, **B**, **X** and **Y** buttons are used to drive the mobile base. The right button halts the actively managed robot. The left and right analog sticks control the elevation and azimuth, respectively, of the robot arm. The claw grip is controlled by the D-pad on the Xbox 360 controller.

Our experiments are designed to evaluate the performance of the IAI vs. manual teleoperation on a variety of measures, including speed of task completion, number of object drops, and user satisfaction. Three indoor scenarios plus a training scenario were employed in our user study. The user was asked to execute each of the scenarios twice, once using our Intelligent Agent Interface and the other using the basic teleoperation functionality. The scenarios were always presented in ascending order of difficulty, with a randomized ordering of the user interface condition. Participants were allotted 10 minutes of practice time and 15 minutes for each of the scenarios. The scenarios are as follows:

Training: Each participant was given a ten minute training session during which they were able to familiarize themselves with the teleoperation controls and the IAI system. The human operator was allowed to practice picking up objects and transporting them to the goal location.

Scenario 1: For the first task, the participant had to use the two robots to search the area and transport small objects (movable by a single robot) to the appropriate goal. The environment contained three piles with five objects. The first pile had large round objects, the second pile contained oddly-shaped objects, and the third pile contained small round objects.

Scenario 2: In the second scenario, the participant had to locate and retrieve awkward objects that required both robots to simultaneously lift and carry. These objects were grouped in three sections, each containing 3 awkward objects.

Scenario 3: The final mission consisted of a mixture of objects as shown in Figure 7.4: the first section contained five (small and large) objects; the second had oddly-shaped objects; and the third contained awkward objects that required bimanual manipulation.

The baseline system, designated as manual operation, consisted of a standard teleoperation setup where the human operator controls all aspects of the robot's motion using the Xbox 360 controller for the three scenarios. The user interface is only used to display camera viewpoints, and the robots never attempt to act autonomously. In our proposed approach, the user has access to the additional commands, **help** and **mirror** through the controller. The IAI automatically detects lack of robot activity and triggers the **search** mode to hunt for objects with the unmanaged robot. When an object too large for a single robot is detected, the IAI system autonomously positions the robot and waits for the user to activate the mirror.

We measured task performance using two metrics: (1) the time required to complete the task (bounded by 15 minutes); (2) the number of times objects were dropped during the scenario. We present our findings in the next section.

Table 7.1 summarizes the demographics of our user study. The majority of the users had limited prior experience with game controller interfaces. Figure 7.5 presents a scatter plot of the time taken by each user to complete each scenario under the two experimental conditions, pure teleoperation (denoted "Manual mode") and IAI. We see that in the majority of runs, IAI significantly accelerates the human operator's progress. We attribute this to the fact that the robot controlled by the IAI continues to assist the human-operator while the teleoperation condition forces the user to inefficiently multi-task between robots.

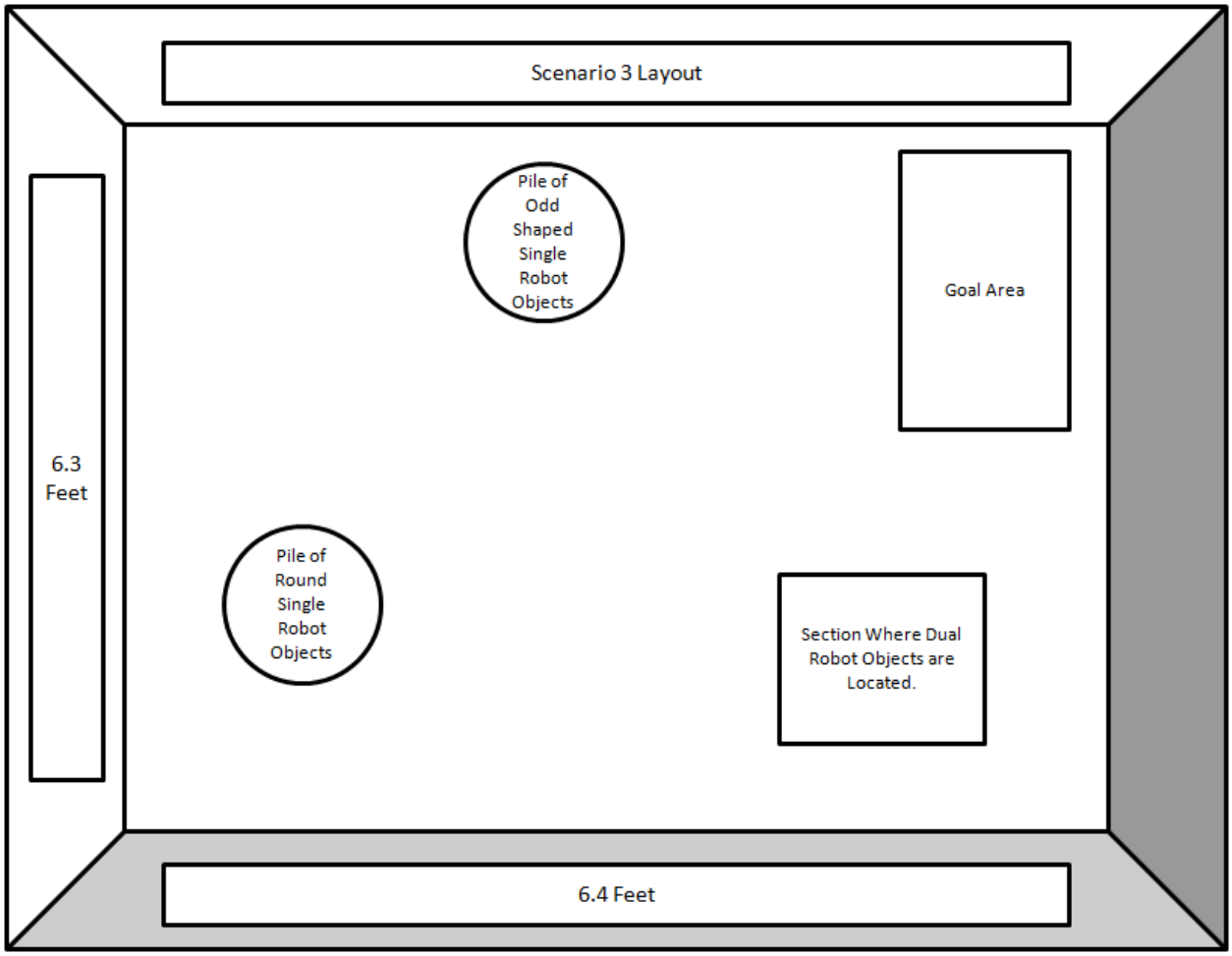

Figure 7.4: Scenario 3 Layout: the two robots operate in an area of $6.3' \times 6.4'$ and move objects from various piles to the goal area. Objects differ in difficulty and demand different strategies from the robot-user team.

Table 7.1: Demographics and experience level of the user-study participants.

Age	Gender		PC Game Skill Level		
20–28	Male	Female	Expert	Inter.	Beginner
20	10	10	4	6	10

We confirm that the improvements in completion time reported by the majority of users are statistically significant under a Student's one-tailed t-test $p < 0.05$. In Scenario 1, users reported statistically significant improvements using IAI (M = 361.1, SD = 63.2) in time completion in over the manual interface (M = 411.8, SD = 81.9), $p = 0.017$. Scenario 2 also reported statistically significant improvements for users using IAI (M = 356.6, SD = 56.0) over the manual interface (M = 407.2, SD = 59.4) for time completion $p = 0.003$. It was confirmed as well in Scenario 3 that IAI (M = 538.5, SD = 157.3) showed statistically significant improvements for users in time completion over the manual interface (M = 627.8, SD = 158.6), $p = 0.004$.

Figure 7.6 shows the number of failed pickup attempts by the user in each scenario, both under IAI and manual conditions. We see that the number of average drops is lower with IAI in all three scenarios. However, the improvement in Scenario 1 for IAI (M = 5.75, SD = 1.55) vs. the Manual interface (M = 6.20, SD = 1.64) did not show a statistically significant on the Student's t-test $p < 0.05$. As for Scenario 2, users did report an statistically significant improvements using IAI (M = 3.30, SD = 1.21) in fewer dropped objects over the manual interface (M = 4.70, SD = 1.55), $p = 0.002$. It was also confirmed in Scenario 3 that the IAI Interface (M = 5.00, SD = 2.22) showed statistically significant improvements for users in fewer dropped objects over the manual interface (M = 6.80, SD = 3.31), $p = 0.026$. In general, IAI results in fewer drops because the mirror mode enables the user and agent to coordinate grasping and movement, whereas in manual mode the user risks dropping the object as a robot is moved during two-handed manipulation task.

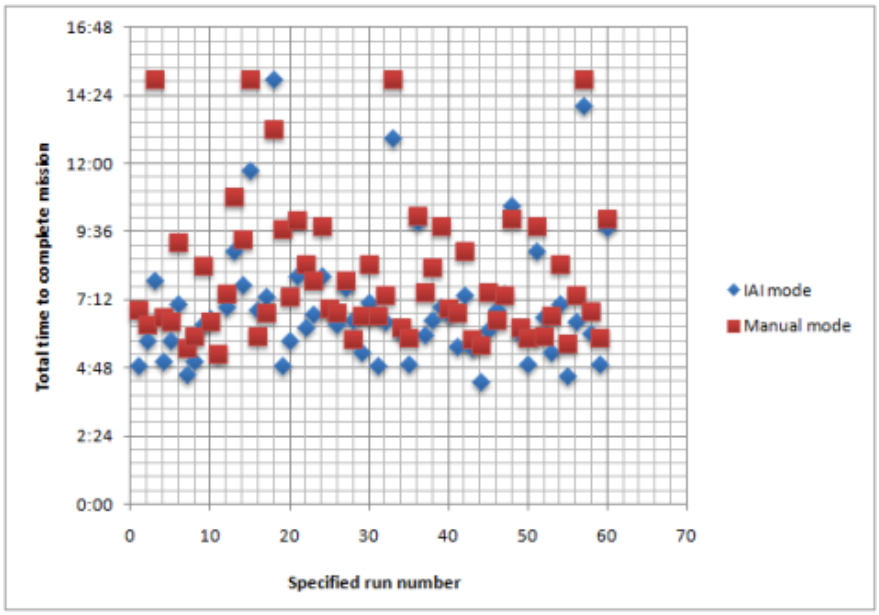

Figure 7.5: Scatterplot showing the time required the complete the task in each scenario under the two experimental conditions: IAI (adjustably autonomous) and Manual (pure teleoperation). We observe that in the majority of scenarios, IAI reduces the time required to complete the task.

Our user post-questionnaire study indicated a strong preference (90%) for IAI over the manual teleoperation mode; the remaining 10% expressed no preference between the two conditions. As part of our post-task questionnaire we asked subjects about their strategies for collecting objects. The comments indicate that the users were definitely relying upon IAI for either 1) autonomously transporting objects or 2) coordinated object pickup. On the post-questionnaire, 90% of the participants expressed a preference for IAI when queried about their preferences and all of the participants gave IAI high ratings (Figure 7.7).

From our own observations, we noted that many of the users spent the entire time controlling a single robot and exclusively relied upon IAI for controlling the second robot's navigation. When the IAI system notified the user to pick up an object, most users switched to that robot, used the arm to lift the object, and then chose to immediately return to controlling the original robot. At

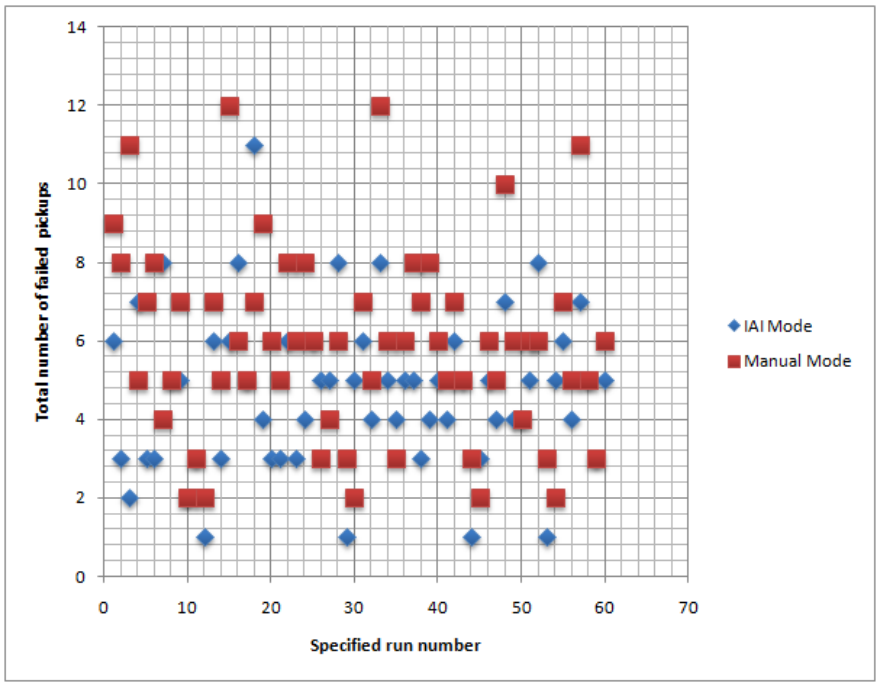

Figure 7.6: Scatterplot showing the number of failed pickup attempts for each user in each scenario.

that point, IAI would autonomously deliver the object to the goal. Some users chose to call for assistance on each pile of objects, instead of assigning robots to different piles. Many participants experienced some initial difficulty during the training period and first scenario learning how to lift objects with the arm. By the second scenario, most users learned the knack of controlling the arm, resulting in fewer object drops. Users experienced more problems in manual mode while lifting the larger objects as compared to IAI.

Adding manipulation capabilities to a robot team widens its scope of usage tremendously at the cost of increasing the complexity of the planning problem. By offloading the manipulation aspects of the task to the human operator, we can tackle more complicated tasks without adding additional sensors to the robot. In this chapter, we demonstrate and evaluate an effective human-robot interface paradigm for multi-robot manipulation tasks. Rather than increasing the workload of

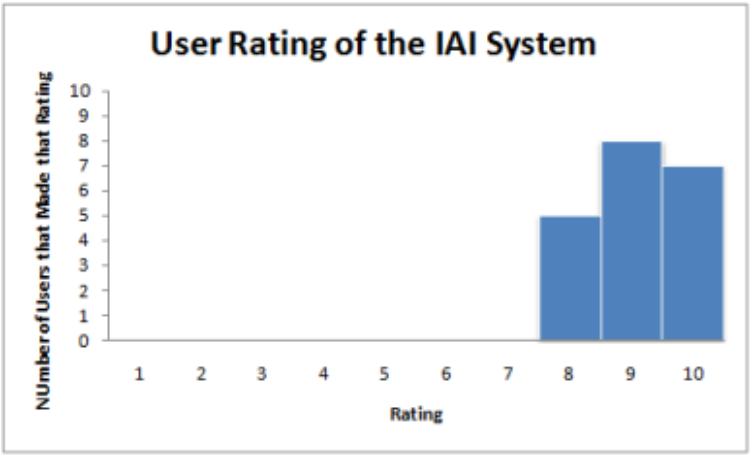

Figure 7.7: Histogram of user ratings of the IAI user interface on post-task questionnaires. Users were overwhelmingly positive.

the human user, we propose an alternate approach in which the robots leverage information from commands that the user is executing to decide their future course of action. We illustrate how this approach can be used to create cooperative behaviors such as mirroring and locate ally; together the robots can coordinate to lift and transport items that are too awkward to be manipulated by a single robot. In the user study, our mixed-initiative user interface (IAI) shows statistically-significant improvements in the time required to perform foraging scenarios and the number of dropped items. Users were able to master the interface quickly and reported a high amount of user satisfaction.

CHAPTER 8: CONFIGURABLE HUMAN-ROBOT INTERACTION FOR MULTI-ROBOT MANIPULATION

We address the general question of structuring the human-agent-robot interactions—how to design an interface that utilizes the humans' perceptual and cognitive abilities without frustrating the user? Our belief is that the system must respect the humans' individual differences, and give the users the flexibility to identify which task components should be performed autonomously. To do this, we introduce a macro acquisition paradigm for learning combined manipulation/driving tasks in a team setting.

In our multi-robot manipulation task, the user directs a team of two HU-IE robots to lift objects using an arm and gripper for transport to a goal location. The environment contains a selection of two different objects, some of which can be transported by a single robot and others that require both robots to lift.

The user views the environment and interacts with the HU-IE robot team through our configurable user interface (IAI: Intelligent Agent Interface). A simple agent is embedded within the user interface to support teamwork by managing information propagation between the team members; it governs the information that gets sent to the robots and displayed on the user interface. Additionally it contains a macro acquisition system that allows the user to identify four key subtasks which are abstracted and used to create robot behaviors which the user can deploy during task execution. All commands to the robots are issued through an Xbox 360 gamepad, using a button to switch between robots.

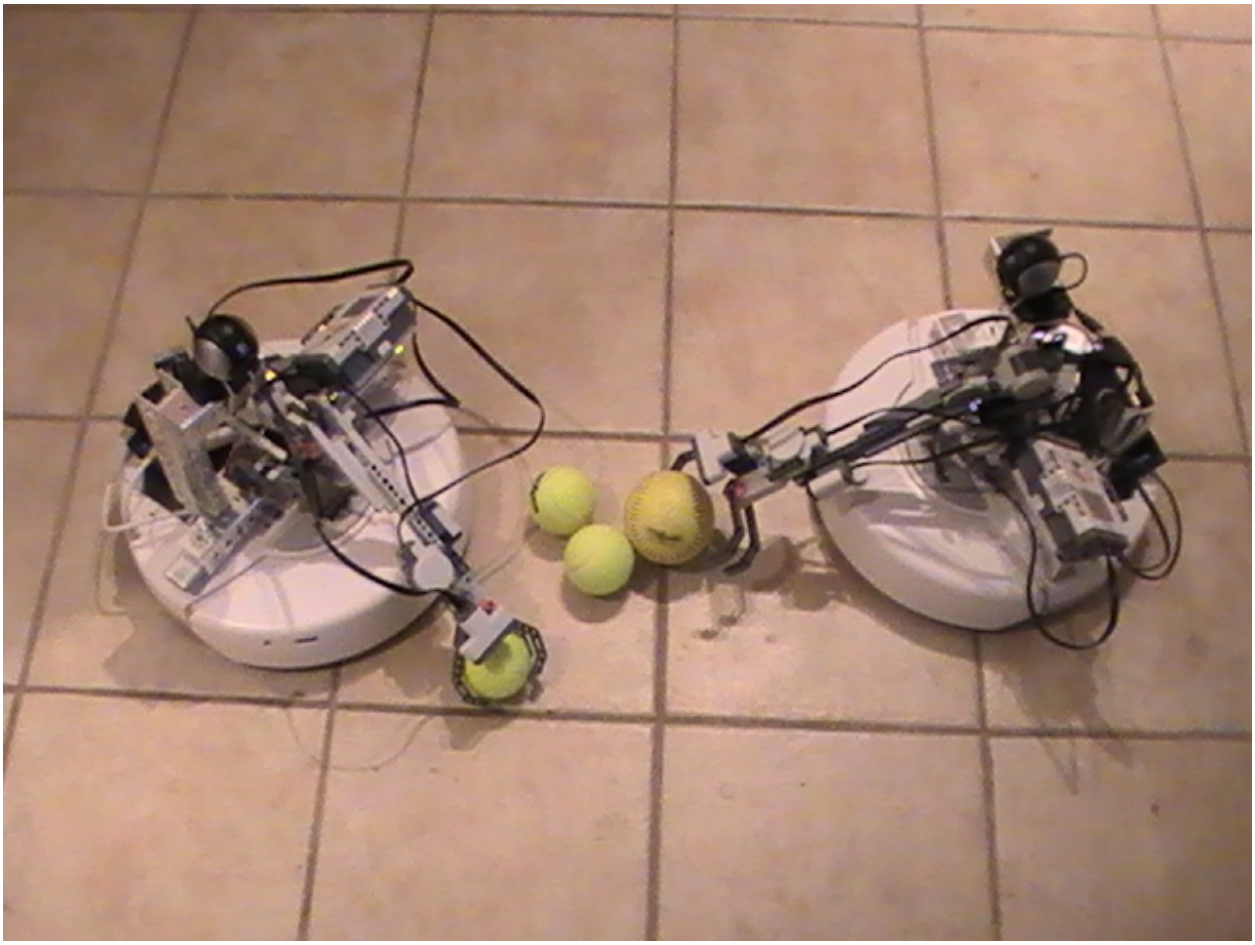

Figure 8.1: Two HU-IE robots cooperating together to clear the environment of objects and deposit them in the goal location.

The basic user interface provides the user with two explicitly cooperative functions: 1) autonomous positioning of the second robot (**locate ally**), and 2) a **mirror** in which the second robot simultaneously executes a modified version of the commands that the user has issued to the actively controlled robot.

When the user requests help to move a large object, these cooperative functions enable the robot to autonomously move to the appropriate location, cooperatively lift the object, and drive in tandem to the goal. Robots have the following built-in modes of operation:

Search: the robots wander the area searching for objects.

Help: a robot enters this mode if the human operator calls for help using the gamepad or when the teleoperated robot is near an object too large to be moved by an individual robot.

Pickup: the robot detects an object and requests that the human teleoperate the arm.

Transport: the robot transports an object held by the gripper to the goal. Path planning is performed using A*.

Locate Ally: the unmanaged robot autonomously moves to a position near the teleoperated robot.

Mirror: the robot mimics the commands executed by the teleoperated robot. This is used to simultaneously lift an object and transport it to the goal location.

Macro: This allows the user to designate a section of the task to be logged for analysis.

In this section we present and evaluate the configurable section of the user interface.

The operator controls the robots using an Xbox 360 Gamepad controller (Figure 8.3) as follows. The trigger buttons on the Xbox 360 controller are used to toggle between the two robots and to activate the **mirror** mode in the unmanaged robot. The **A**,**B**,**X**,**Y** buttons are used to drive the mobile base. The right button halts the actively managed robot. The left and right analog sticks control the elevation and azimuth, respectively, of the robot arm. The claw grip is controlled by the D-pad on the Xbox 360 controller. To execute a previously acquired macro the user must press and hold the back button and then press one of the **A**,**B**,**X**,**Y** buttons.

The most important aspect of the user interface is that it empowers the user to designate sections of the task for the robots to execute autonomously. The user might specify sections for multiple reasons: 1) they occur frequently 2) are tedious to perform 3) need to occur while the user is busy

with other tasks, such as teleoperating the second robot. To make the process of macro acquisition simpler, the user initially performs a demonstration by teleoperating a single robot. However, during the task, the macro is automatically generalized to account for the execution state of the second robot. The macro can also be propagated across both robots by invoking the **Mirror** mode, without additional examples (Figure 8.4).

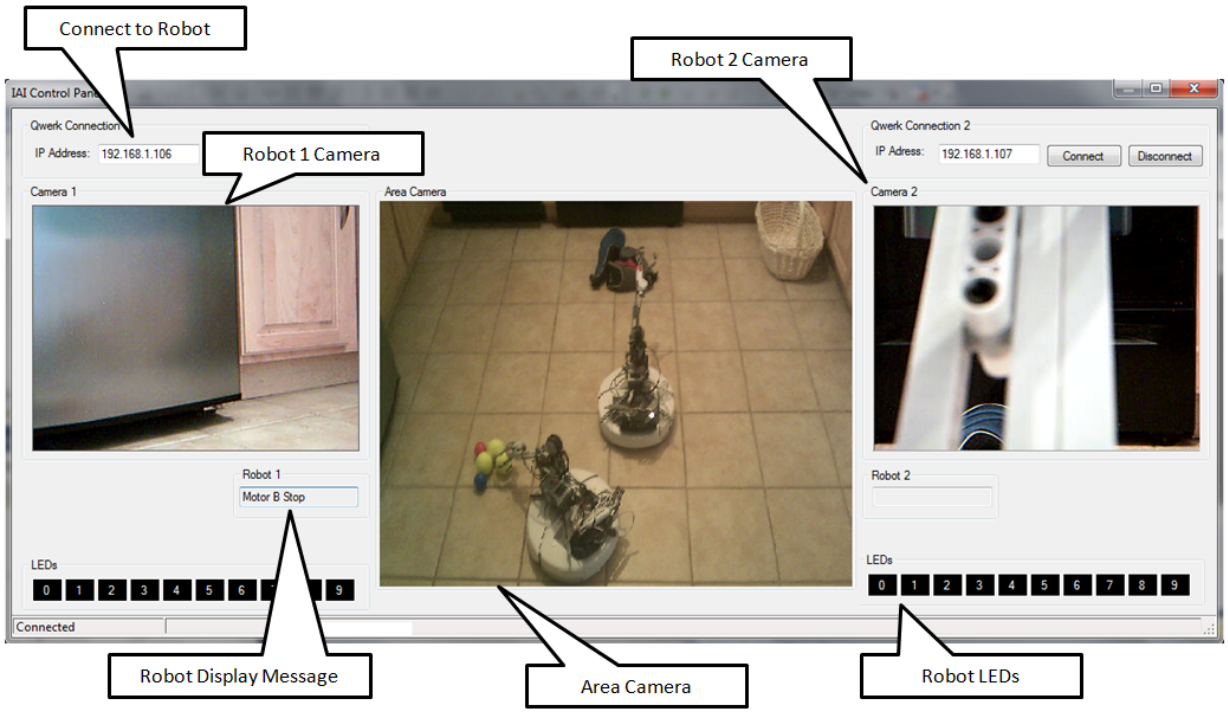

Figure 8.2: The user interface is designed to enable the user to seamlessly switch teleoperation between multiple robots. The IAI supports a cooperative mode where the agent supports the user's active robot by mirroring its planned actions. The user views the environment through an overhead camera and the robots' webcams.

During the macro acquisition phase, the robot's state space trajectory is recorded, paying special attention to the initial and final states of the trajectory. The state includes the following features in absolute coordinates: drive start/end position, arm start/end, claw open/closed (Figure 8.5). Additionally, the status of all of the key sensor systems (cliff, wall, and bumper sensors) is logged. The

agent also notes the current location of known movable objects in the environment and whether the user is teleoperating the second robot. The state space trajectory is then used to create an abstract workflow of the task which can be combined with the teamwork model and the path planner to generalize to new situations. To build the workflow, the state space trajectory is separated into drive, arm, and claw segments. Adjacent drive and arm segments are merged to form one long segment (Figure 8.6). The terminal position of the robot is retained in both absolute coordinates and also the relative position to the nearest object or robot.

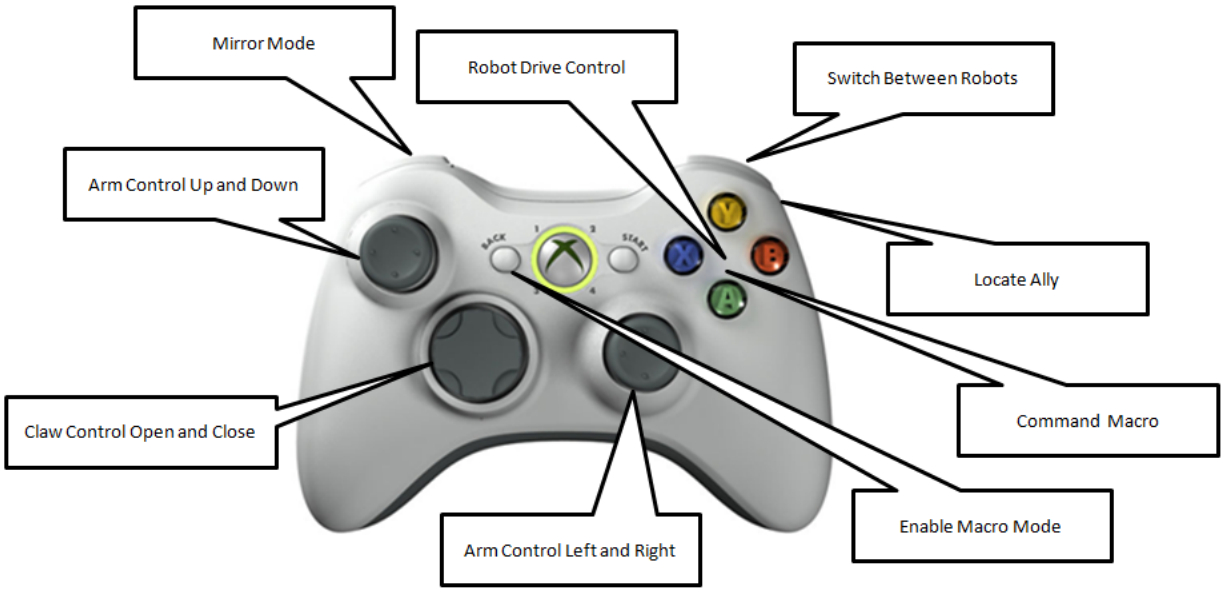

Figure 8.3: The physical interface to the IAI is through a Xbox 360 gamepad from which the operator can select a robot, send it explicit teleoperation commands, utilize built-in autonomous functions, and create macros.

After the macro acquisition phase, there is an acceptance phase during which the operator is given a chance to verify the macros' performance. When the human operator is satisfied that the macro was performed correctly then the macro is accepted and mapped to one of the Xbox 360 **A**,**B**,**X**,**Y**

buttons. During the acceptance phase, the macro is evaluated in multiple locations on the map and with the HU-IE robot arm at different angles.

Figure 8.4: Example of the execution of a recorded macro using mirror mode. Even though the macro was initially created using a single robot demonstration, it is generalized for coordinated action.

Figure 8.5: State representation of a recorded macro

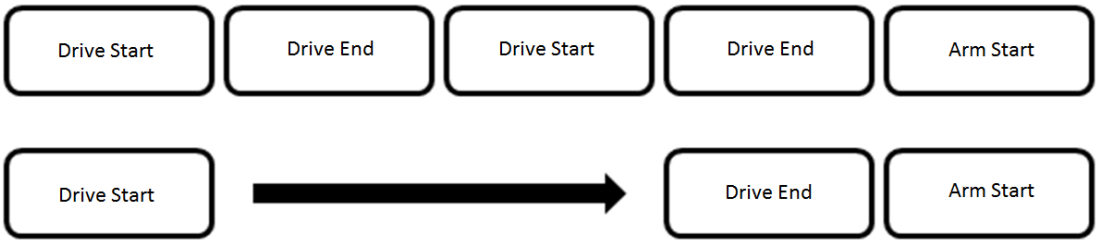

Figure 8.6: If the demonstration contains multiple short segments, the abstract task representation is created through merging superfluous segments.

If the macro representation was not accepted by the human operator, the system attempts to modify the macro using a set of taskwork rules. For instance, during the initial phase, it is assumed that the terminal positions are of key importance and that the robot should use the path planner to return to the same absolute position. In the second demonstration, the system used the recorded sensor date to identify the most salient object located near the terminal position and return the robot to

that area. If an object is dropped during the acceptance phase, it is assumed that the drop is the principal reason for the macro non-acceptance and the macro is repeated using the same abstraction but with minor modifications to its positioning relative to the object using the ultrasonic sensor. For simplicity of user interaction, macro acquisition is done by teleoperating a single robot but during actual task execution many of the macros are actually executed in mirror mode, using the pre-programmed teamwork model. One of the most common macros developed by both expert and novice users was a macro for driving the robot to the goal (Figure 8.7).

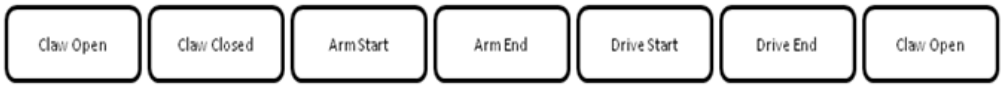

Figure 8.7: Example abstract task representation for driving to the goal

Our experiments were designed to evaluate the performance and usability of the configurable interface on a variety of measures. The users were asked to clear objects from a cluttered household environment and transport them to a goal area using two robots guided by the configurable user interface. In total, the users interacted with the system for an hour and a half under the following conditions:

Training: Each participant was given a ten minute training session during which they were able to familiarize themselves with the teleoperation controls and the autonomous built-in modes. Subjects were encouraged to practice picking up objects and transporting them to a goal location.

Macro Acquisition: Each participant was allotted forty minutes to create four macros and map them to appropriate buttons. During the macro acquisition phase, the subjects principally interacted with a single robot. After creating each macro, they described the macro on a worksheet.

Scenario 1: For the first task, the participant had to use the two HU-IE robots to search the area and transport small objects (movable by a single robot) to the appropriate goal. The environment contained three piles with five round shaped objects (shown in Figure 8.8).

Scenario 2: For the second task, the participants had to use the two HU-IE robots to search the area and transport awkward objects that required bimanual manipulation to the appropriate goal (shown in Figure 8.9). This scenario contained three piles with large objects (boxes), arranged in a similar layout to Scenario 1. This was the hardest condition and was always presented last.

Detailed logs were collected of the user's entire interaction with the system, and the users were asked to complete pre and post test questionnaires. In total, twenty participants completed the user study, and Table 8.1 summarizes the demographics of the user group.

Table 8.1: Demographics and experience level of the user-study participants

Age	Gender		PC Game Skill Level		
20–28	Male	Female	Expert	Mid	Beginner
20	10	10	4	6	10

The main purpose was to evaluate the benefits of the configurable human-robot interface and answer the following questions:

1. what macros did users create and how were they used?
2. were there differences in the macro usage patterns in single vs. bimanual manipulation?
3. did the users prefer the macros to the build-in system functions? We also performed a post hoc within-user comparison of the configurable user interface vs. a non-configurable user interface designed for an earlier study [3].

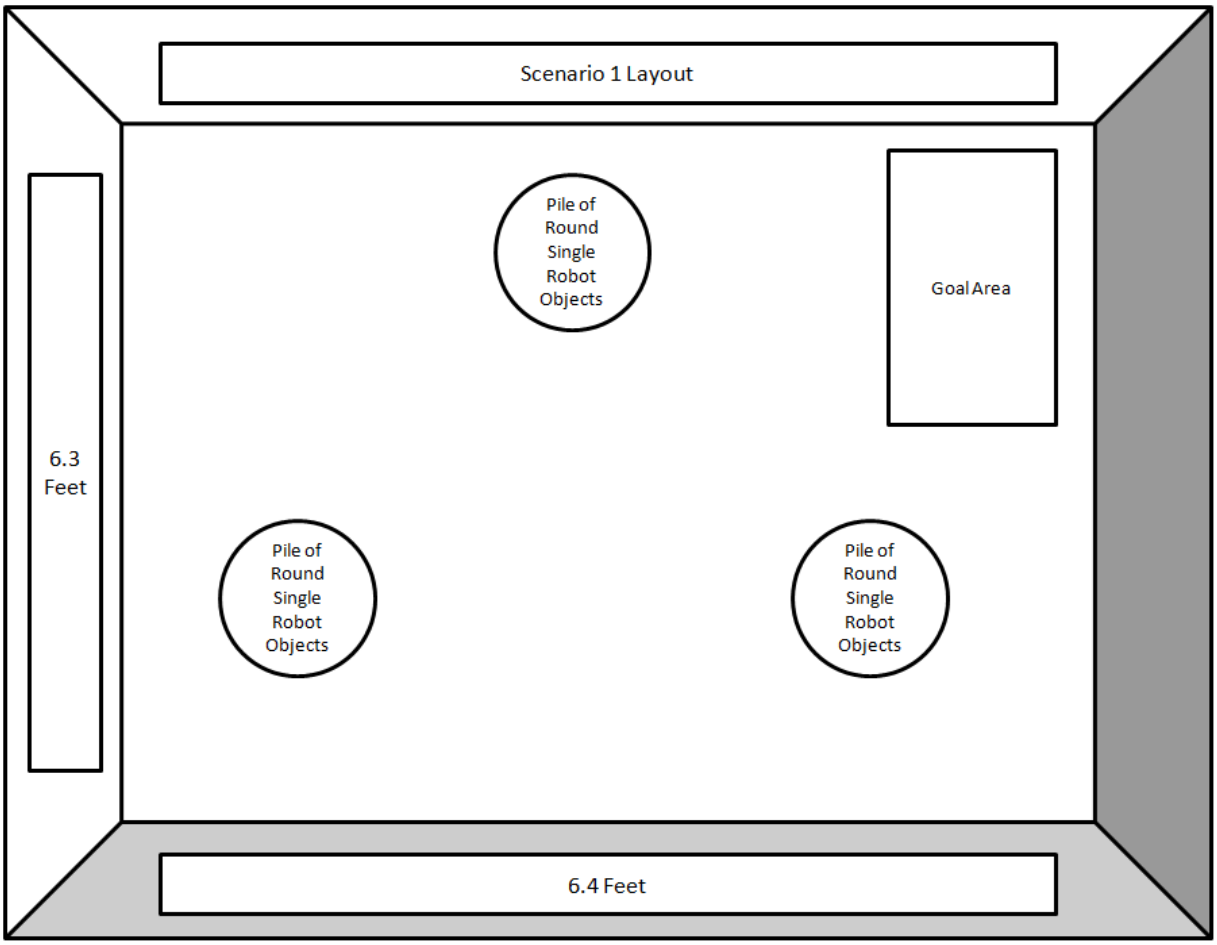

Figure 8.8: Scenario 1 Layout: the two robots operate in an area of $6.3' \times 6.4'$ and move small objects of different shapes from all piles to the goal area. This scenario is highly parallelizable if the users create the correct type of macros.

The macros created by users varied in length and complexity, with a general trend that game skill correlated with shorter macros and longer periods of user teleoperation. Figure 8.10 shows an example of the macro usage pattern for an expert user performing Scenario 1. This can be contrasted with the pattern of novice macro usage (Figure 8.11) that shows a heavier reliance on macros. Overall, we found it encouraging that the configurable aspects of the user interface were more heavily used by novice users.

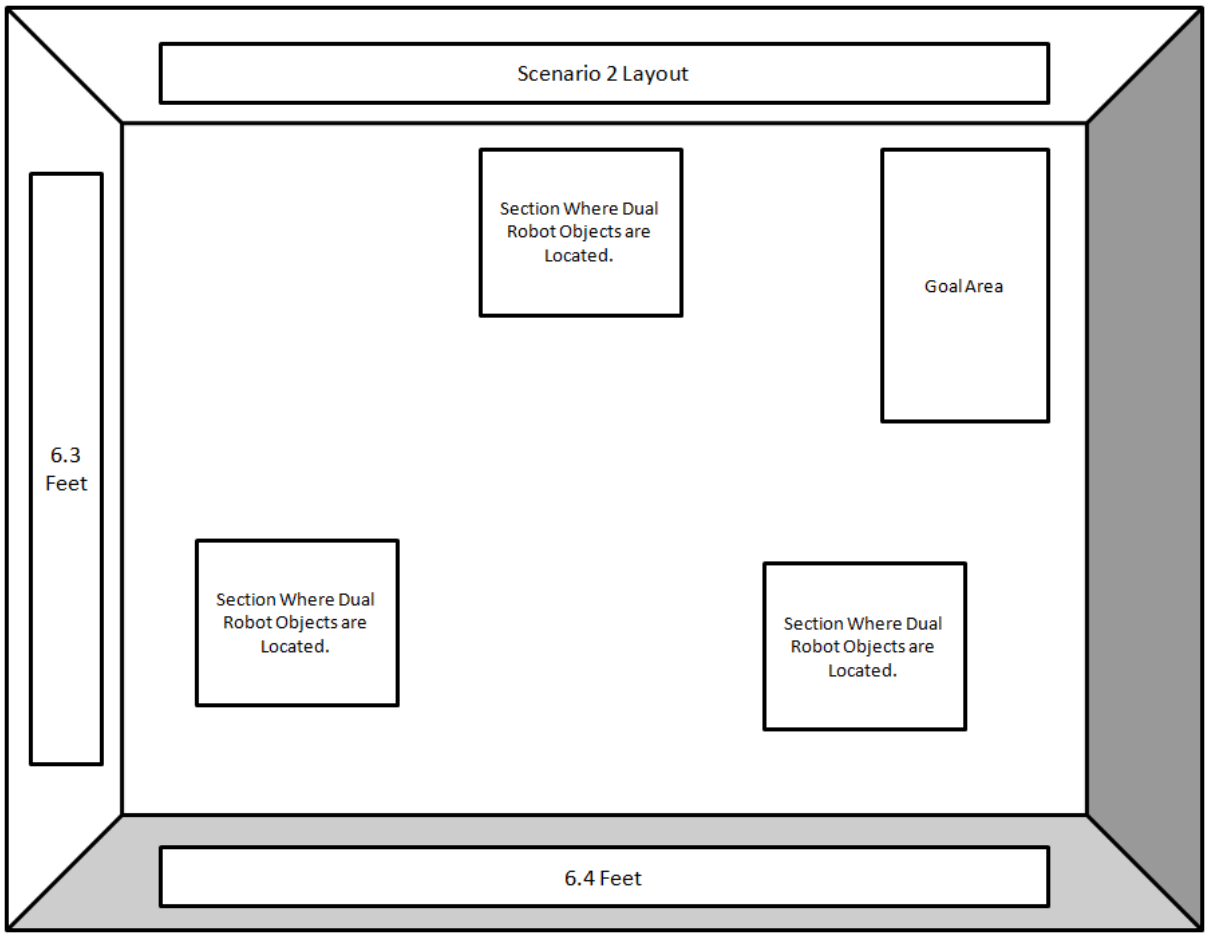

Figure 8.9: Scenario 2 Layout: the two robots operate in an area of $6.3' \times 6.4'$ and move objects from various piles to the goal area. Some of the objects are large enough to require two robots and hence demand different strategies from the robot-user team.

Pick up and delivery macros were very common, with the most frequently occurring macro being one for delivering objects to the goal. Interestingly, the execution of this macro was similar to the the built-in mode (**Transport**), but users consistently trusted their own macro and preferred to use it instead. It hints at the possibility that the process of creating their own macro made the system less opaque and more predictable to the user. From observation, we noted that the users created macros to help them with parts of the task that they struggled on during training; for instance, users

who experienced more failed pickups would often focus on creating a good object pick up macro.

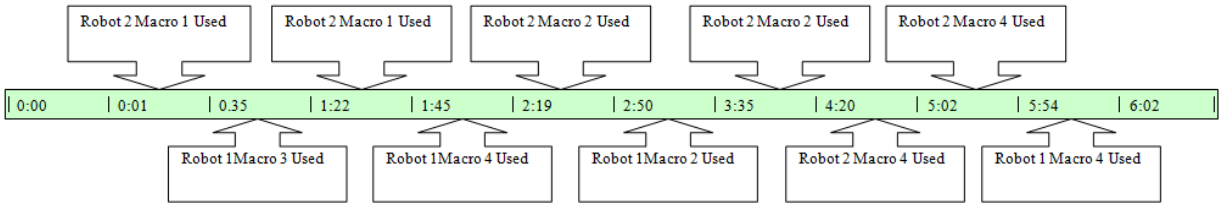

Figure 8.10: Timeline showing macro usage by an expert user in Scenario 1. Ten macros were used in total during the fifteen minute period. The set of macros included: 1) drive to pile, lift object, and deliver to goal 2) lift object and deliver to goal 3) lift object 4) deliver to goal.

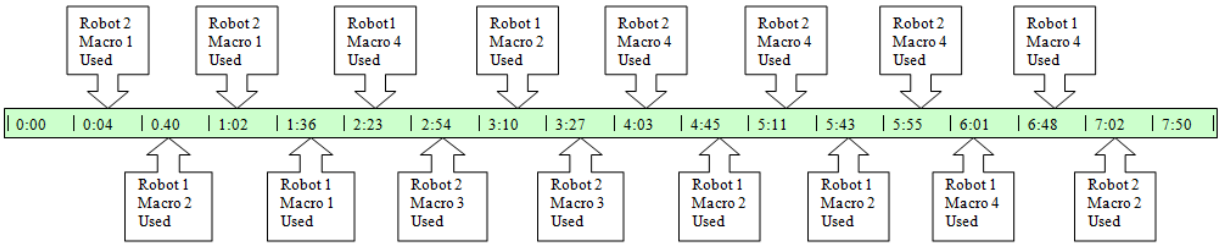

Figure 8.11: Timeline showing macro usage by a novice user in Scenario 1. Sixteen macros were used in total during the fifteen minute period. The set of macros included: 1) drive to pile and lift object 2) lift object and deliver to goal 3) lift object 4) deliver to goal.

Many participants experienced some initial difficulty during the training period and first scenario in learning how to lift objects with the arm. By the second scenario, most users learned the knack of controlling the arm, resulting in fewer object drops. Users experienced more problems when using macros to pick up large objects that required bimanual manipulation and tightly synchronized action from both robots. This is reflected in the overall time required to complete both scenarios; unsurprisingly users require significantly more time to complete Scenario 2 than Scenario 1 (Figure 8.12).

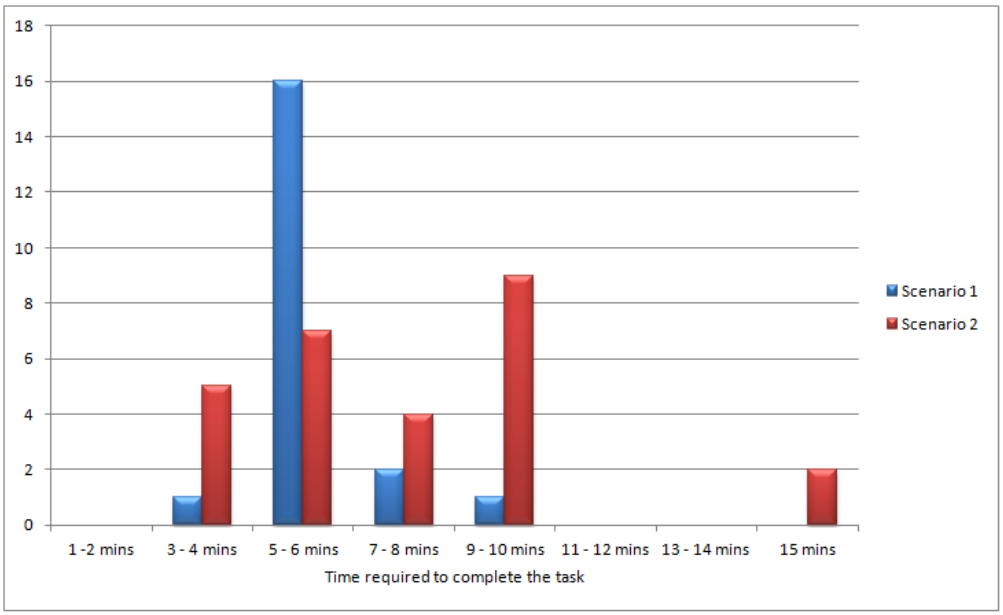

Figure 8.12: Histogram showing the time required the complete Scenario 1 and 2. Most users were able to complete the Scenario 1 task in one third of the allotted time. Note that there is more variance in the time required to complete the coordinated manipulation task (Scenario 2), and two users were not able to complete it in the allotted time.

During the experiments, we observed that users who used between 5-10 macro commands performed the task faster than the users who relied more on macros or were constantly teleoperating the robot. Overall macro usage for both scenarios is shown in Figure 8.13. In a post hoc comparison to users from a previous study who used a non-configurable version of the same user interface, macros appeared to confer a slight time advantage (Figure 8.15). The most significant results were in the user rankings of the interface which enthusiastically (70%) preferred the configurable user interface; overall, the interface scored high ratings in the post-questionnaire user ratings (Figure 8.17).

Adding a configurable user interface to a human-agent-robot team empowers the human operator to structure his/her user experience by expressing task-specific preferences for the amount of interdependence vs. autonomy between human and robot. This is consistent with the coactive design

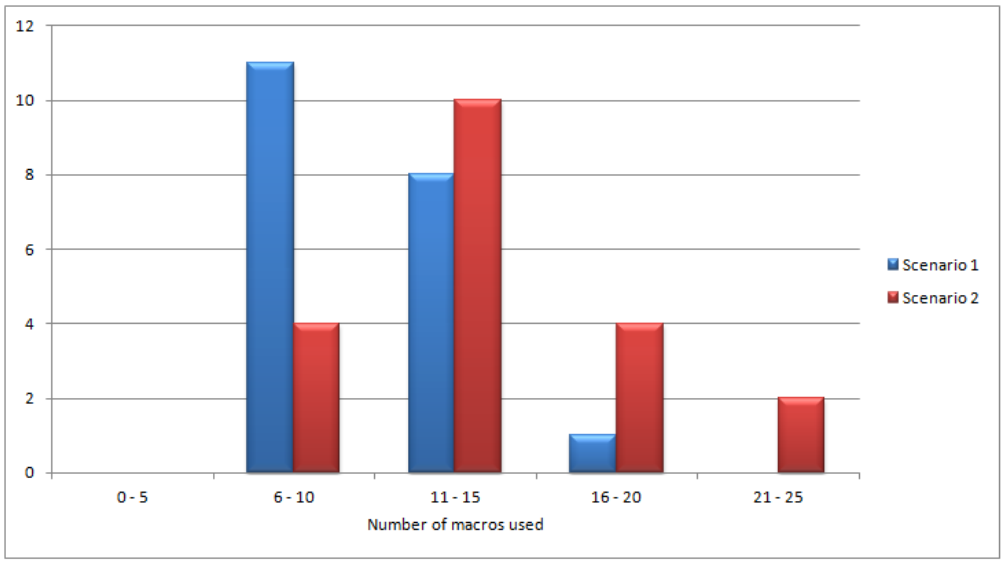

Figure 8.13: Histogram showing the macro usage by scenario. Bimanual manipulation (Scenario 2) was more macro intensive.

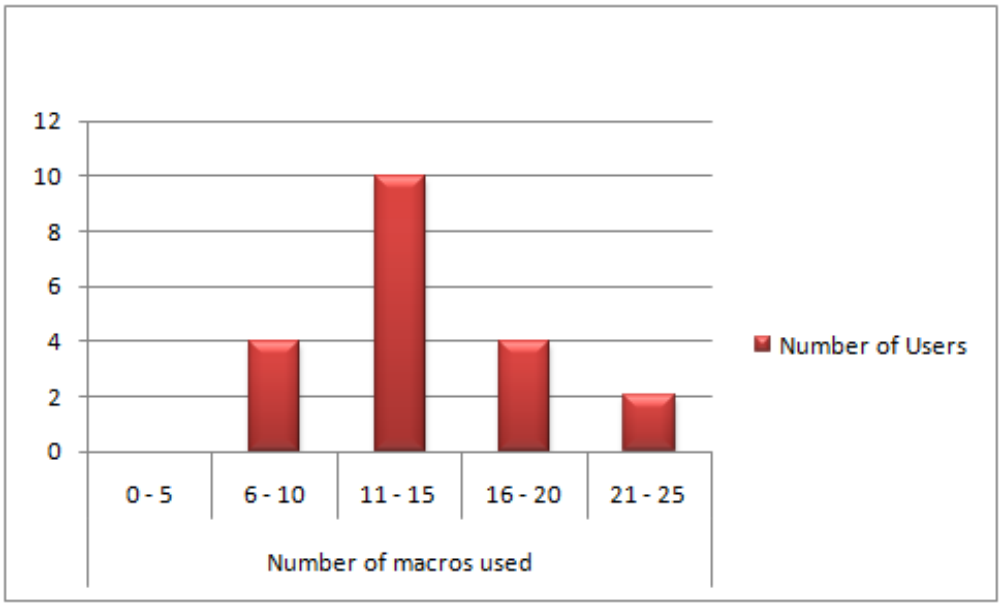

Figure 8.14: Histogram showing the macro usage in Scenario 2 (bimanual manipulation)

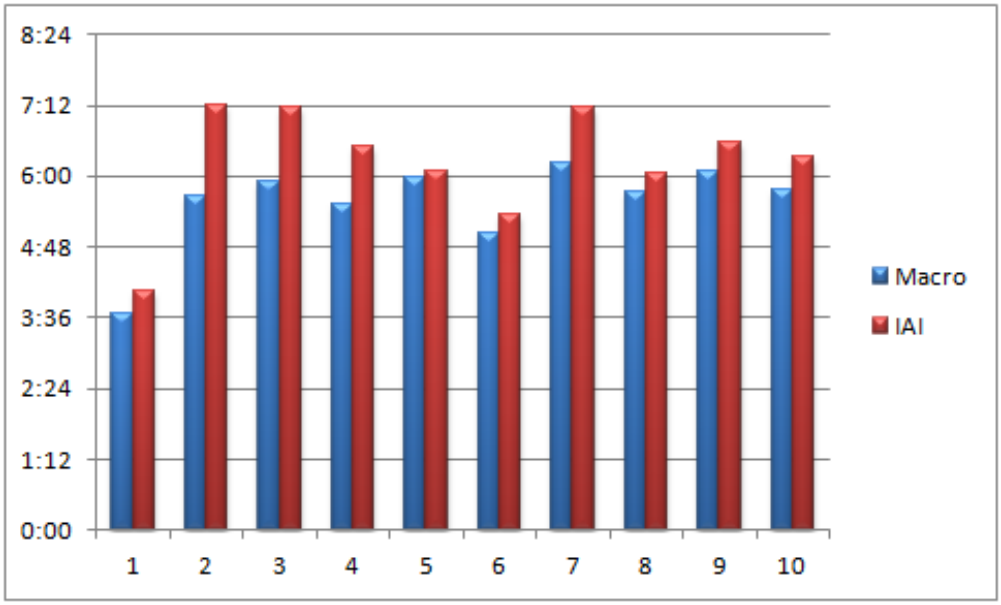

Figure 8.15: Post hoc analysis comparing the configurable and non-configurable user interface. The y axis shows time required to complete the scenario and the x axis the subject number. The configurable user interface appears to confer a slight time advantage.

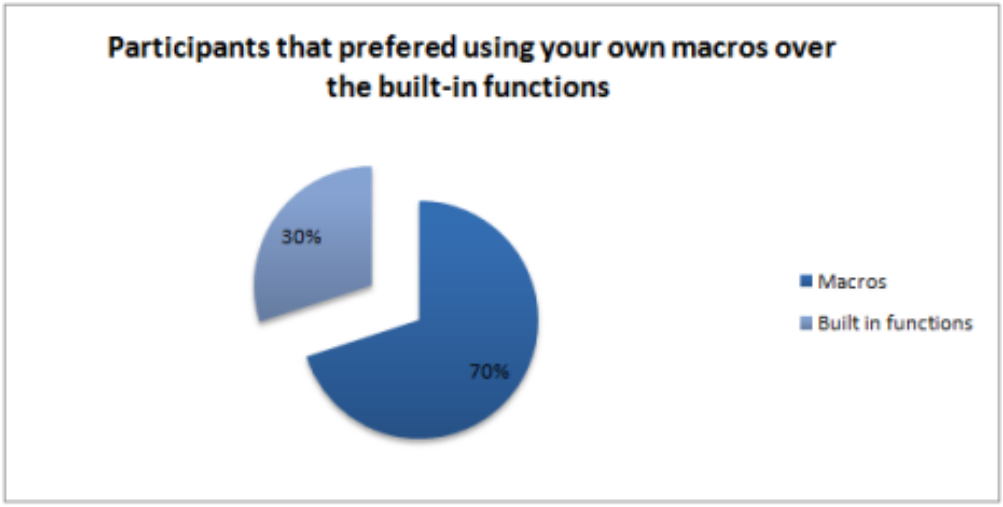

Figure 8.16: 70% of participants expressed a clear preference for agent-assisted mode of teleoperation; the remaining 30% expressed no preference between the two modes.

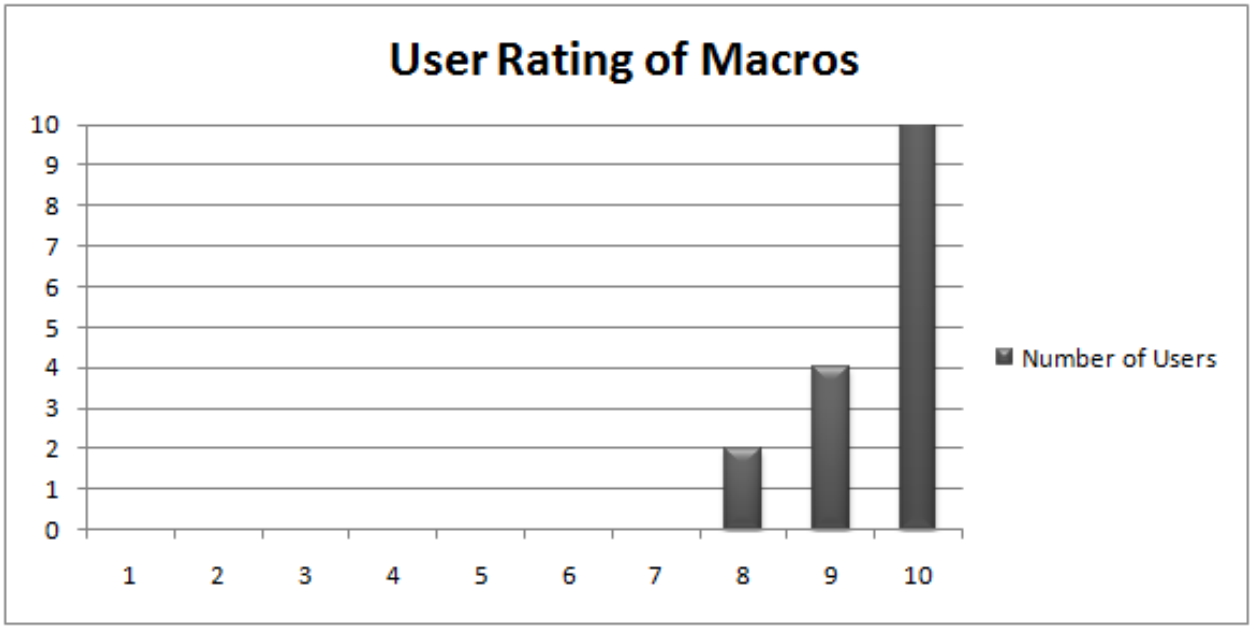

Figure 8.17: Histogram of user ratings of the configurable user interface on post-task questionnaires

model for human-agent-robot systems. Here, we address the problem of multi-robot manipulation in unstructured environments with limited sensors, which is a relatively new and challenging problem which utilizes the capabilities of all team members (human, agent, and robot) to achieve complicated bimanual pickups. Users expressed a significant preference for the configurable autonomy of macros over the built-in autonomous functions, and gave the user interface high overall ratings.

CHAPTER 9: ADAPTING TO EXPERT-NOVICE DIFFERENCES IN HUMAN-ROBOT INTERACTION

This chapter describes our adaptive user interface for adjusting the autonomy of the robots based on the operator's skill level on three separate axes of competence. We present a paradigm for learning a model of the user's competences from a short example teleoperation trace. In our multi-robot manipulation task, the human operator coordinates a team of two mobile robots to lift objects using an arm and gripper for transport to the goal location. The household environment contains an assortment of small and large objects, some of which can be transported by a single robot and others that require both robots to lift. Figure 9.1 shows a picture of the team of robots cooperatively moving an object that cannot be carried by a single robot. This cooperative pickup task is an important component of many potential applications of multi-robot systems, including cooperative assembly [18], home service robot teams [35], urban search and rescue [11], and patient recovery robot teams.

To examine this problem of multi-robot manipulation, we used the Home and Urban Intelligent Explorer (HU-IE) system that is designed to be proficient at picking up light objects in a household environment with either carpets or hard floors. The user views the environment and interacts with the robot team through our user interface running on a separate computer (Figure 9.2). We evaluate an adaptive version of the user interface that learns a model of expert-novice differences for the various aspects of the teleoperation task vs. a non-adaptive version. The baseline user interface provides the user with a mirror mode for simultaneously controlling both the robots in which the second robot simultaneously executes a modified version of the commands that the user has issued to the actively controlled robot. This enables the robots to cooperatively lift objects and drive in tandem to the delivery location.

Figure 9.1: Two robots cooperate to lift an object under the direction of the human operator. In the multi-robot manipulation task, the robots must lift and deliver a series of objects of different sizes to the goal location.

The operator controls the robots using an Xbox 360 Gamepad controller as follows. The trigger buttons on the Xbox 360 controller are used to toggle between the two robots and to activate the mirror mode in the unmanaged robot. The **A**,**B**,**X**,**Y** buttons are used to drive the mobile base. The right button halts the actively managed robot. The left and right analog sticks control the elevation and azimuth, respectively, of the robot arm. The claw grip is controlled by the D-pad on the Xbox 360 controller.

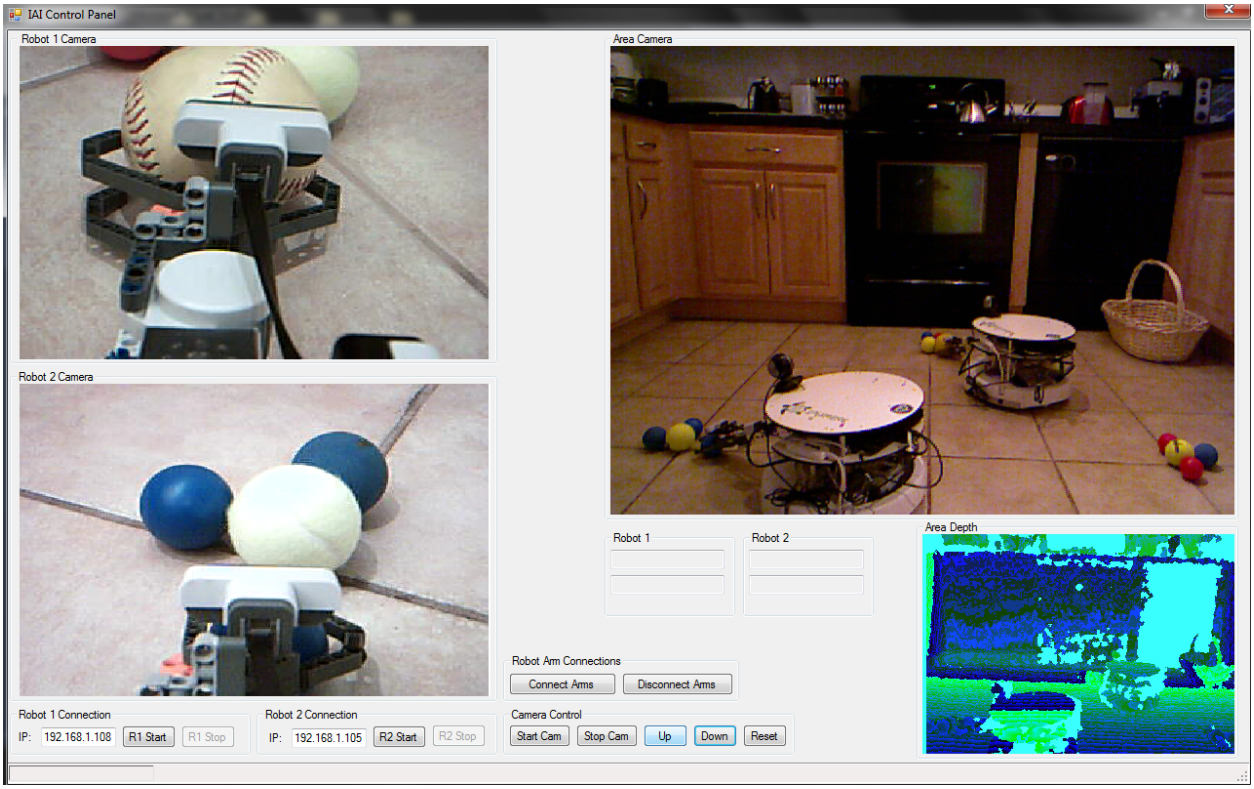

Figure 9.2: The user interface simultaneously provides the operator with an overhead view of the scene through a separately mounted camera (top right), a depth map of the scene from the Kinect (bottom right), and the webcam perspective from the two robotic arms (left).

Layered on top of the basic user interface is an adaptive interface component that adjusts the robots' autonomy based on a learned model of the user's teleoperation competence. An assessment of the user's teleoperation performance is performed offline and loaded into the adaptive interface component (Figure 9.3). The adaptive section of the user interface is structured as a multi-agent system containing the following elements:

Attribute Component: Imports the attribute report generated offline describing the human operator's competence on the three task axes of navigation, manipulation, and cooperation.

Operator Interface Agent: Adjusts the commands passed to the robots based on the user model.

HU-IE Interface Agent: Handles interactions with the robots.

Human Input Component: Handles interactions with the the human operator.

Status Component: Gathers and updates the status information from the robots to be displayed on the user interface.

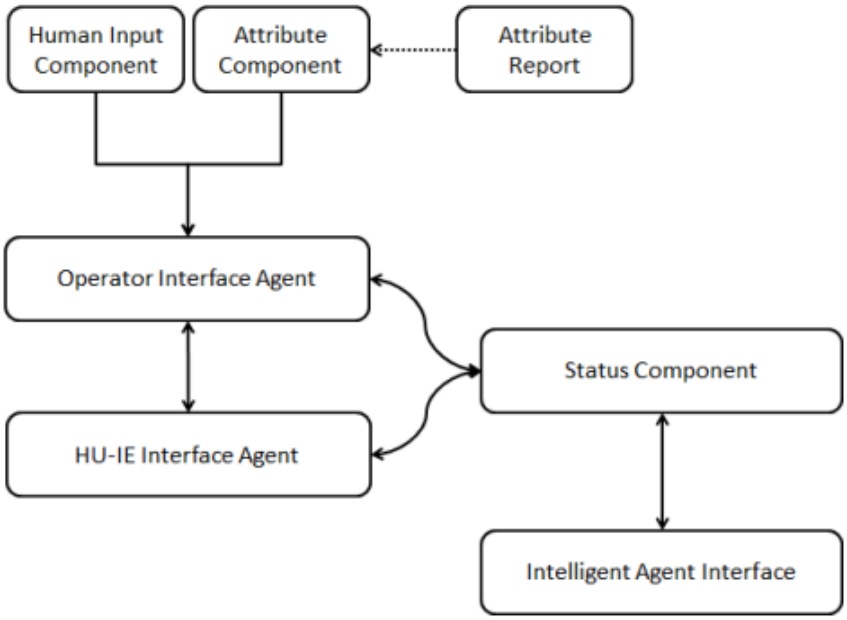

Figure 9.3: Overview of the Adaptive Interface Component.

All adjustable autonomy decisions occur within the Operator Interface Agent, which takes the offline attribute report describing the human operator's competence on the three task axes and modifies the teleoperation commands sent to the robots. In general, the lower the human operator's skill level, the more the agent filters the commands that are passed to the robots.

To construct a model of expert-novice differences in teleoperation performance, we collected example teleoperation sequences from twelve users and clustered the data using a semi-supervised

version of k-means. The goal of this process was to learn a model of user competence on the three axes of navigation, manipulation, and collaboration. We selected these three axes as being both an accurate representative of our previous experiences with users and well-suited to inform adjustable autonomy decisions for the multi-robot manipulation task.

To model navigation proficiency we extracted the following features from the raw trace: 1) task completion time; 2) number of seconds the robots spent moving in each cardinal direction; 3) number of seconds robots were halted; 4) number of times the user reversed driving direction. For classifying manipulation competence, the features used were: 1) task completion time 2) number of backward and right-left robot movements 3) number of seconds the arm spent at high, mid, and low elevations 4) number of claw command switches. Backward and right-left movements were particularly significant since they were rarely used by expert users who were able to drive forward and lift the item in one smooth motion, without reverses and changes of direction. The features for classifying robot coordination include the same features used for manipulation plus the percentage of time the user controlled both robots.

We observed the performance of the users on a simplified teleoperation task and rated them as being either confident or not confident on an axis of performance. The results of k-means clustering with $k = 2$ and a Euclidean distance measure proved to be a good fit for our data. The accuracy on separating the training data set was 100% for the navigation axis, 91% for the manipulation axis, and 83% for the coordination axis.

Based on the learned model of expert-novice differences on the three axes of teleoperation proficiency, the adaptive version of the user interfaces selectively modifies the autonomy of the robots. Users who are less confident on the navigation axis receive more help during sections of the task that involve driving the robots. Two additional functions are invoked:

Auto goal return: When a human operator has successfully picked up an object, based on the ultrasonic sensor readings and robot arm accelerometer, the Operator Interface Agent commands the robot to drive the object to the goal area. The A* algorithm is used to find the shortest path to the goal, while avoiding obstacles marked in the occupancy grid.

Nearest object seeking: Once an object is delivered to the goal, the Operator Interface Agent detects the nearest object and starts driving the robot in that direction.

Any time that the robot is under autonomous operation, the human operator can retake control of the HU-IE robot by canceling the drive command. For novice human operators only, the system will reactivate the drive command during robot idle times. If the user is classified as being confident at navigation, the system does not reactivate the drive command.

For users that are classified as less confident at the manipulation sections of the task, the adaptive user interface autonomously adjusts the arm and the claw to help the user using the functions:

Auto arm adjustment: The robot arm needs to be at a certain angle relative to the target object for a successful grasp and lift. Based on arm accelerometer sensor data and Kinect object detection, the adaptive user interface attempts to calculate the angle required for a successful pickup and adjusts the arm accordingly when an object is within a certain radius of the robot. The Operator Interface Agent observes the incoming commands, adds the required adjustments to the end of the command string, and displays it to the user before sending it to the robot.

Auto claw adjustment: If the ultrasonic sensor indicates that the grasp will not be successful. the mobile base and claw are autonomously adjusted to improve the grasp.

Note that even though it is possible to autonomously calculate reasonable base, arm, and claw po-

sitions for grasping objects an expert human user can still outperform fully autonomous operation. Users who perform poorly on the coordination axis are experiencing difficulty in maneuvering the robots together and performing object lifts with both arms simultaneously. The adaptive user interface attempts to adjust the arm, claw, and base of both robots when they are within close proximity of the same pickup object using the auto arm adjustment and auto claw adjustment functions. This behavior is also triggered if the arms of both robots are not positioned evenly.

Our experiments were designed to evaluate the human operators' ability to complete a set of indoor multi-robot manipulation scenarios under both the adaptive and non-adaptive version of the user interface. 20 users (8 male, 12 female) between the ages of 20 and 35 participated in the study. Before the user interface evaluation scenarios, all users were given 10 minutes of practice time and asked to complete three skill assessment tasks designed to measure their teleoperation performance on the axes of navigation, manipulation, and cooperation. Several of the subjects had prior experience playing Xbox games, but none of them had previous robotics experience.

Teleoperating Assessment Task 1: Each participant was allotted ten minutes to navigate a single robot through an obstacle course; the results of this task were used to classify the user's navigation skill.

Teleoperating Assessment Task 2: Each participant was allotted ten minutes to lift a single small object; the results of this task were used to classify the user's manipulation skill.

Teleoperating Assessment Task 3: Each participant was allotted ten minutes to lift a large box (shown in Figure 9.1)); the results of this task were used to classify the user's cooperation skill.

Scenario 1: For the first scenario, the participant had to use the two robots to search the area and transport small objects (movable by a single robot) to the goal basket within 15 minutes.

The environment contained three piles with five round shaped objects (shown in the left and center panels of Figure 9.4). The participant performed this scenario twice in randomized order, once with the adaptive interface and once with the baseline version.

Scenario 2: For the second task, the participants had to use the two HU-IE robots to search the area and transport awkward objects that required bimanual manipulation to the goal basket within 15 minutes. There were three piles with bimanual objects in this scenario (shown in the right panel of Figure 9.4). The participant performed this scenario twice in randomized order, once with the adaptive interface and once with the baseline version.

Figure 9.4: The two robots operate within a $6.3' \times 6.4'$ household area and move objects from various piles to the goal area. Scenario 1 (left, middle) contains piles of small objects that can be moved with a single robot, whereas Scenario 2 (right) contains objects that require bimanual manipulation.

We compare the performance of the adaptive vs. the non-adaptive version of the user interface. Figure 9.5 presents a comparison of the times required for each participant to complete Scenario 1 (small objects) and Scenario 2 (bimanual manipulation) under both experimental conditions. We confirm that the improvements in completion time is statistically significant under a paired two-tailed t-test $p < 0.01$. In Scenario 1, users reported statistically significant improvements using the adaptive interface (M = 340.9, SD = 86.8) in time completion in over the non-adaptive interface (M = 408.0, SD = 79.8), $p = 0.015$. Scenario 2 also confirmed an statistically significant

improvements for users using the adaptive interface (M = 600.7, SD = 117.7) over the non-adaptive interface (M = 716.4, SD = 150.7) for time completion $p = 0.010$.

Figure 9.6 presents a comparison of the object drops by each participant in Scenario 1 (small objects) and Scenario 2 (bimanual manipulation) under both experimental conditions. We confirm that the reductions in dropped objects is statistically significant under a paired two-tailed t-test $p < 0.01$. In Scenario 1, users reported statistically significant improvements using the adaptive interface (M = 1.10, SD = 1.65) in fewer dropped objects over the non-adaptive interface (M = 3.25, SD = 1.74), $p = 0.00027$. As for Scenario 2, users did report an statistically significant improvements using IAI (M = 3.85, SD = 2.58) in fewer dropped objects over the manual interface (M = 7.85, SD = 3.15), $p = 0.00009$.

The figures show that for all of the participants (other than subject #10) the adaptive component improves the human operator's performance measured by both task completion time and reductions in dropped objects. Our post-questionnaire indicated that 90% of the users had a strong preference for adaptive vs. the non-adaptive version of the user interface, and the remaining 10% expressed no preference between the two conditions.

Table 9.1 shows the results of the user modeling component of the system. The classifier learned from previous teleoperation traces identified half of the users as being expert at navigation and manipulation, and slightly fewer as being experts at the cooperative sections of the task. Figure 9.7 shows the relative distribution of commands issued by experts vs. novices using the non-adaptive version of the interface.

Several interesting facts emerge: 1) novices more frequently issue *stop* commands for the robot base, whereas experts more frequently use *forward*; 2) novices open the claw more often than expert users, probably following object drops; 3) experts more regularly issue the *up* command to the robot arm, whereas novices more frequently stop the arm in its trajectory. The classifier is able

to utilize these differences in command distribution to accurately learn a model of expert/novice differences along the three teleoperation axes. In most cases, the users' self-reported level of confidence on each axis agreed with the classifier. However, we believe that relying strictly on self-reports of expertise in undesirable, particularly in situations where the users' have greater external motivation to claim expertise.

This chapter demonstrates the utility of our adaptive user interface that adjusts the robots autonomy based on expert-novice differences. A user model of teleoperation competence on three axes of performance (navigation, manipulation, and coordination) is learned from short example tasks. The adaptive user interface modifies the robots autonomy in a task specific way, based on the operators skill level. In our user study, the proposed user shows statistically significant improvements in reducing the task completion times and dropped objects.

Table 9.1: Performance level on axes of teleoperation according to both classifier and self-report

Axis	# Expert	# Novice	Self-report agreement
navigation	10	10	90%
manipulation	10	10	85%
cooperation	8	12	75%

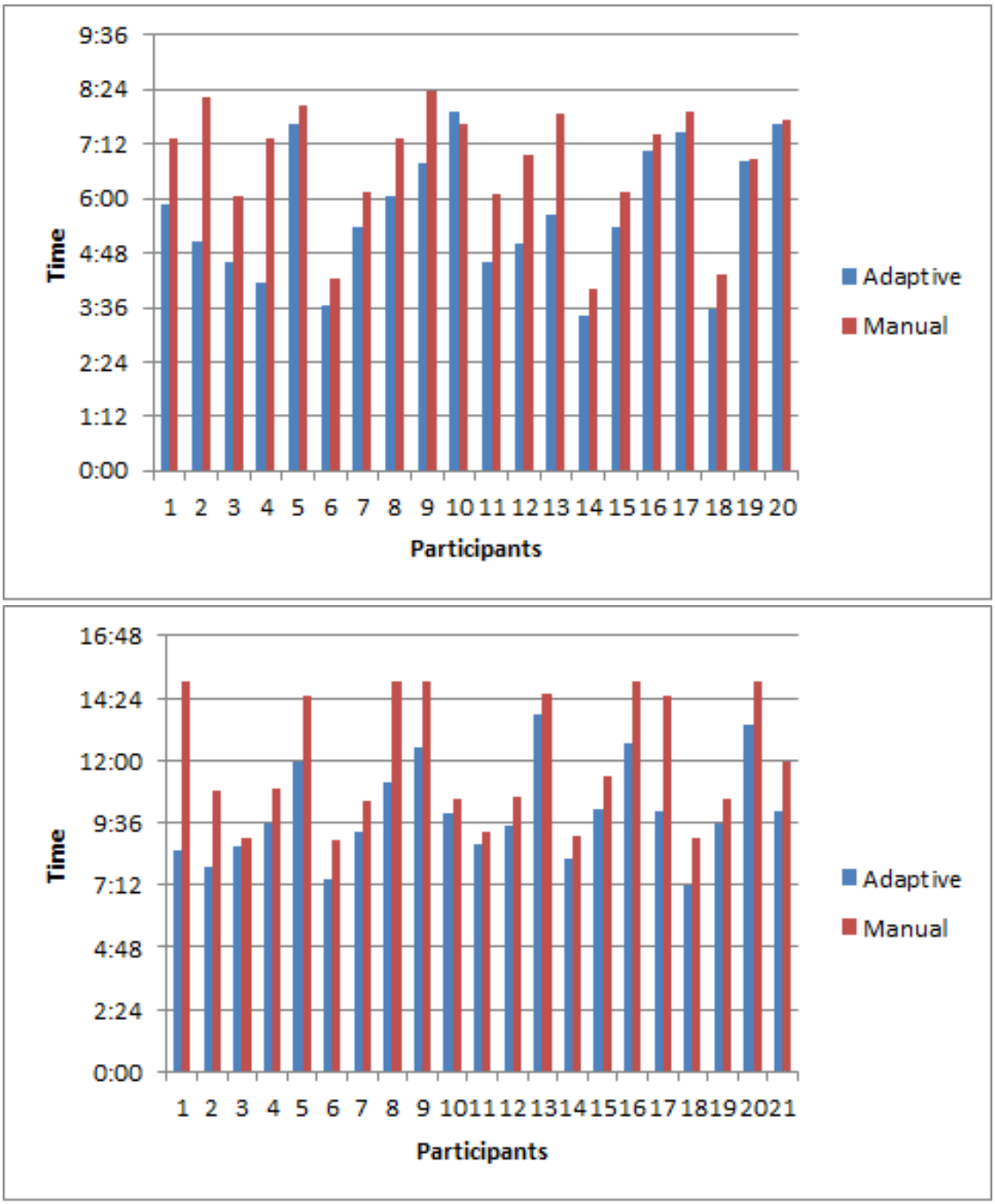

Figure 9.5: Time to complete Scenario 1 (left) and Scenario 2 (right) in minutes for each subject (x-axis). All of the participants (except subject #10) experience time improvements with the adaptive version of the user interface.

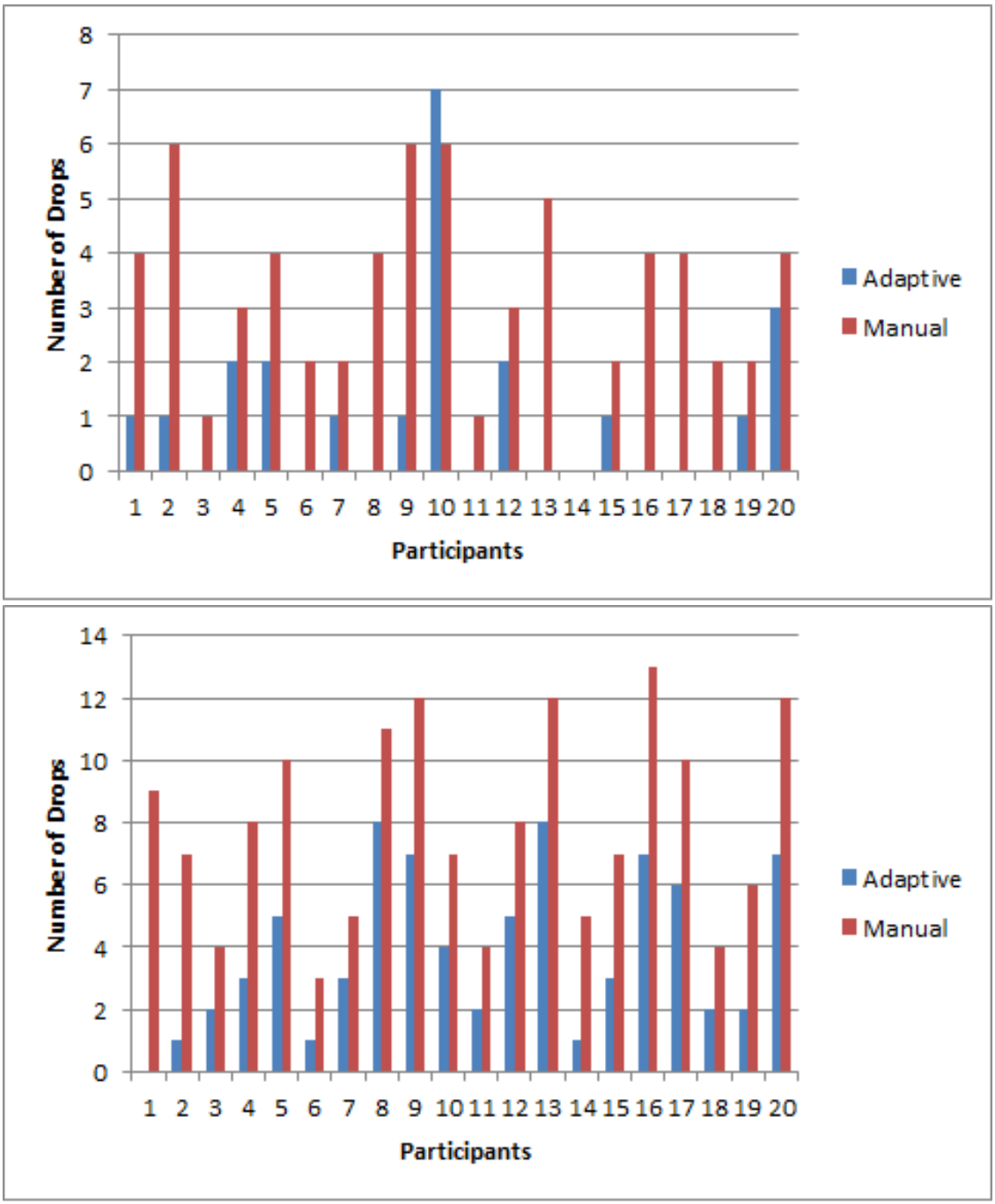

Figure 9.6: Number of objects dropped by each subject (x-axis) in Scenario 1 (left) and Scenario 2 (right). All of the participants (except subject #10) experience reductions in dropped objects with the adaptive version of the user interface.

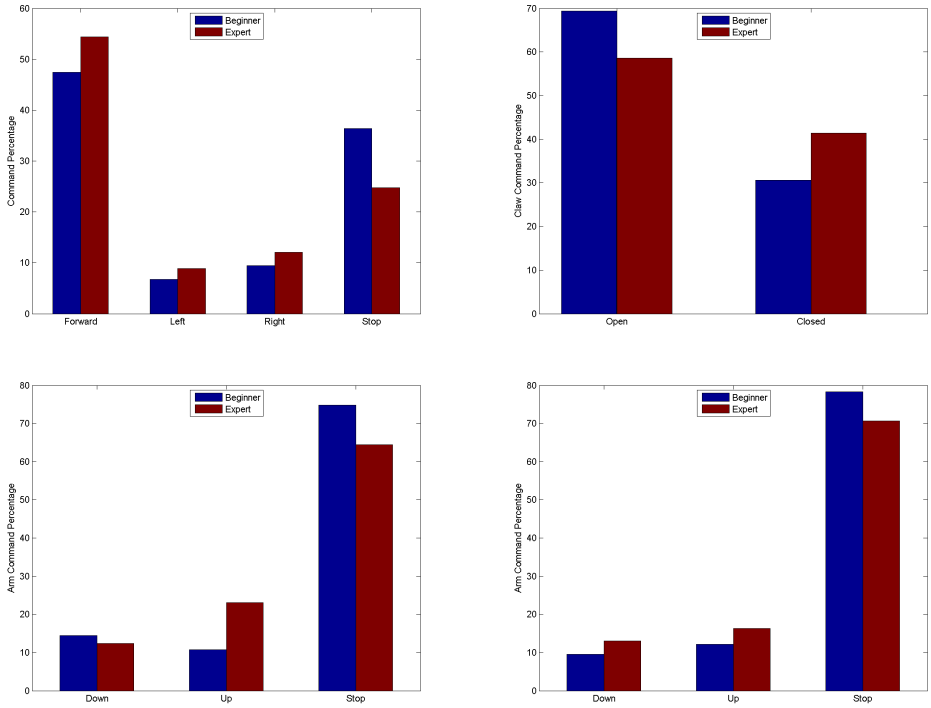

Figure 9.7: Expert/novice differences in frequency of command utilization for navigation, manipulation, and collaboration. Beginners (blue) utilize the stop command more frequently than experts (red) both when driving the robot base or moving the arm. They open the claw more frequently than experts who require fewer attempts to lift objects. In contrast, experts issue the close claw and forward drive commands more frequently than the beginners.

CHAPTER 10: INTERACTION EFFECTS BETWEEN USER EXPERTISE AND INTERFACE PERFORMANCE

We evaluated the existence of interaction effects between the expertise of the user and their performance with a specific user interface (adaptive vs. configurable) on a multi-robot manipulation task. Users controlled the movement of the robots with a combination of Nintendo Wii commands and arm gestures. Both styles of user interfaces were equally effective at reducing dropped objects and task completion time, but our experiments revealed a strong interaction between the expertise of the user, as measured on a set of calibration tasks, and their performance using a particular class of user interface. Only experts were able to effectively leverage the configurable interface, while novice users performed better with the adaptive interface. This suggests that using a real-time user modeling system to activate or deactivate user interface functions could be an effective way to support long-term users whose expertise is constantly improving. For instance, Francois et al. demonstrated a system that classified changing human-robot tactile interaction styles in real-time [21].

We evaluated an adaptive version of the interface that adjusts the robot's autonomy based on the user's expertise [39] vs. a configurable version that allows the user to create macros for autonomous behaviors using a simple learning from demonstration paradigm [36]. The operator interacts with the robots through a combination of arm gestures, tracked using a Microsoft Kinect sensor, plus gamepad controls executed with a Nintendo Wii remote. To facilitate tandem operation, the system supports special *mirror* modes in which the teleoperation commands are copied by the unmanaged robot [35]. Single mirror mode allows the user to simultaneously control both arms on a single HU-IE robot. Dual mirror mode is used for simultaneously controlling both the robots including the arms; it is particularly useful for instructing the robots to cooperatively lift and deliver objects.

Figure 10.1: The multi-robot delivery task requires the operator to control the pair of robots to move objects to a goal location. Objects too large to be lifted by a single robot (such as the box shown in the photo) must be delivered by the team of robots driving in tandem under the direction of the human operator.

The Wii remote controller is configured as follows. Buttons **One** and **Two** are used to toggle between the two robots and to activate the mirror mode in the unmanaged robot. The **+** and **-** buttons are used to switch between controlling a single robot arm and activating the mirror mode for the unmanaged arm on the robot currently being controlled by the user. The **B** button (while pressed) enables arm motion tracking, allowing the user to control the HU-IE robotic arms, and the claw can be toggled open and close using the **A** button. The mobile base is driven with the **D Pad**. In the configurable version of the interface, the **Home** button activates the macro mode, and the **D Pad** is then used to select the macro to execute.

The user interface runs on a separate desktop computer, connected to two Microsoft Kinect sensors. The first Kinect sensor is oriented toward the user and tracks the user's arm positions in order

to control the elevation of the robot arms. The Kinect SDK provides access to the user's skeletal tracking information for twenty joint track states every frame. Since we are only concerned with the user's arm gestures, we extract the **Right/Left Hand**, **Right/Left Wrist**, **Right/Left Elbow**, and **Right/Left Shoulder** track states. We then extrapolate the data and map it to the active arm on the HU-IE robot. Since the human's arm has more degrees of freedom and a greater operating range, we only take the Y axis position data into consideration and ignore anything outside the operating range of the robotic arm. This allows the user to comfortably move their arm up and down without strain while controlling the corresponding arm on the active HU-IE robot. The system monitors the location of the HU-IE robots and delivery objects using a second Kinect sensor, oriented toward the workspace. The user interface utilizes a grid-based localization technique to track the robot positions and an A* path planner to select paths for the robots under autonomous operation. Figure 10.3 shows the user's view of the interface.

The adaptive interface (described in more detail in Chapter 9) adjusts the robots' autonomy based on the user's teleoperation skill level, providing less autonomy to expert users and more autonomy to novices. User expertise is assessed using a set of calibration tasks, designed to measure the user's ability to drive the robots, manipulate objects, and control both robots simultaneously; semi-supervised k-means is then used to group the users as being expert or novice on the three separate dimensions of teleoperation proficiency. Based on this initial expertise assessment, the adaptive interface selectively adjusts the autonomy of the robots. Users who were rated as novices on the navigation dimension receive more help during sections of the task that involve driving the robots. We implemented two autonomous navigation behaviors which the adaptive interface invokes when the user starts driving: 1) auto goal return and 2) nearest object seeking. Any time that a HU-IE robot is under autonomous operation, the user can retake control of the HU-IE robot by canceling the drive command. For novice users, the system will reactivate the drive command during robot idle times. If the user is classified as being expert at navigation, the system does not reactivate the

drive command.

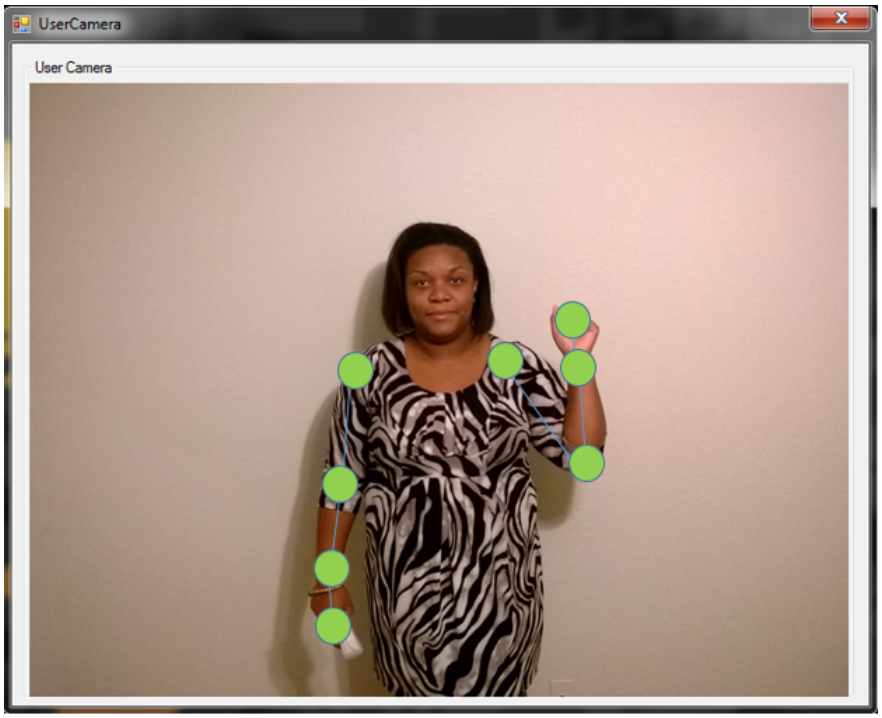

Figure 10.2: A panel on the interface displays the camera's view of the user. The user gestures up and down to modify the elevation of the robotic arms. Joint tracking information is obtained from a Kinect sensor; only the joints marked in green are used to control the arms.

For users who are classified as novices at manipulation, the adaptive interface autonomously adjusts the arm and the claw to help the user when the robots are near objects. Based on arm accelerometer sensor data and Kinect object detection, the adaptive interface attempts to calculate the angle required for a successful pickup and adjusts the arm accordingly when an object is within a certain radius of the robot. The adaptive interface observes the incoming commands, adds the required adjustments to the end of the command string, and displays it to the user before sending it to the robot. If the ultrasonic sensor indicates that the grasp will not be successful, the adaptive interface autonomously adjusts the claw and mobile base of the HU-IE robot to improve the grasp. Note that even though it is possible to autonomously calculate reasonable base, arm, and claw po-

sitions for grasping objects, an expert user can still outperform fully autonomous operation. Users who perform poorly on the coordination calibration task receive assistance when the robots are within close proximity of the same pickup object; the adaptive interface attempts to adjust the arm, claw, and base of both robots to improve the alignment. This behavior is also triggered if the arms of both robots are not positioned evenly.

Figure 10.3: The user interface simultaneously provides the operator with an overhead view of the scene through a separately mounted camera (right) and the webcam perspective from the two robotic arms carrying the cameras (left).

The configurable interface (described in more detail in in Chapter 8) enables the user to elect sections of the task for the robots to execute autonomously. The user might specify sections for multiple reasons: 1) they occur repeatedly 2) are tiresome to complete 3) need to transpire while the user is occupied with other tasks, such as teleoperating the second robot. To make the process of macro acquisition simpler, the user initially performs a demonstration by teleoperating a single robot. However, during the task, the macro is automatically generalized to account for the execution state of the second robot. The macro can be propagated across both robots by invoking the mirror mode, without additional examples (Figure 10.4).

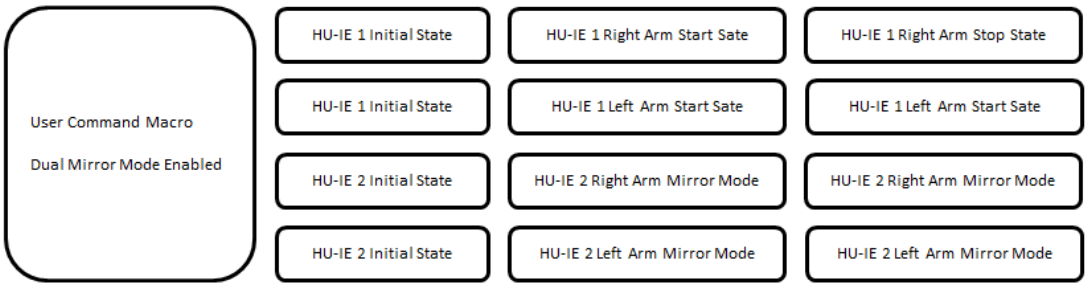

Figure 10.4: Example of the execution of a recorded macro using mirror mode. Even though the macro was initially created using a single robot demonstration, it is generalized for coordinated action.

During the configurable interface phase, the robot's state space trajectory is recorded, paying distinct attention to the initial and final states of the trajectory. The state includes the extracted features in absolute coordinates: drive start/end position, arm start/end, claw open/closed. Additionally, the status of all of the key on-board sensor systems (cliff, wall, and bumper sensors) are recorded. The configurable interface also tracks the present location of known movable objects in the environment and whether the user is teleoperating the second robot. The state space trajectory is then used to generate an abstract workflow of the task to generalize to new situations. To build the workflow, the state space trajectory is divided into drive, arm, and claw segments. Adjacent drive and arm segments are merged to form one long segment. The position of the robot is retained in both absolute coordinates, along with its relative position to the nearest object or robot. After the user records a macro, the macro is evaluated in multiple locations, with the HU-IE robot arms at altered angles. This user is then given a chance to verify the performance of a recorded macro. If the user is dissatisfied, the system presents a new variation of the macro according to a predefined list of macro generalization rules. If an object is dropped during the acceptance stage, it is assumed that the drop is the principal reason for the macro non-acceptance and the macro is repeated using the same abstraction but with minor modifications to its positioning relative to the object using the

ultrasonic sensor. When the user is satisfied, he/she accepts the macro and maps it to one of the Wii remote **D Pad** buttons.

To evaluate our research hypothesis that the users' expertise level on the calibration tasks is a good predictor of their performance with the different versions of the interface (adaptive vs. configurable), we recruited 40 users (18 male, 22 female) between the ages of 23 and 36 to use both interfaces. Experiments were conducted using the following procedure:

1. Each participant was trained in basic usage of both systems for 10 minutes. The systems were introduced to the user in randomized order.
2. Users then performed the three 10 minute calibration tasks used to assess their expertise on our dimensions of teleoperation proficiency (navigation, manipulation, and coordination). The tasks included 1) navigating one robot through a course 2) lifting different types of small objects 3) lifting large objects that require both robots to move. The machine learning classifier was used to categorize the user's performance on each of the three dimensions; the output of the classifier was then added to the configuration file of the adaptive interface.
3. Participants were then allowed to create macros using the configurable version of the interface. Each participant was allotted 30 minutes to create four macros and map them to the controller. During the macro acquisition phase, the subjects principally interacted with a single robot. After creating each macro, they described the macro on a worksheet.
4. Users were asked to complete Scenario 1 twice, using the two different interfaces. The conditions were randomly ordered, and subjects were allotted 15 minutes per condition In this scenario, the users had to use the two HU-IE robots to search an indoor area and transport small objects (movable by a single robot) to the goal. Objects were arranged in three piles (shown in the left and center panels of Figure 10.5). Two of the piles contained a mixture of irregularly shaped small objects, and the third pile was composed of boxes that required the robots to use both arms.

5. Users were asked to complete Scenario 2 twice, using the two different interfaces. The conditions were randomly ordered, and subjects were allotted 15 minutes per condition. This scenario stressed the user's ability to perform bimanual manipulation. The piles contained a combination of large objects, some that could be lifted with two arms on the same robot and some that required two robots to transport to the goal location.

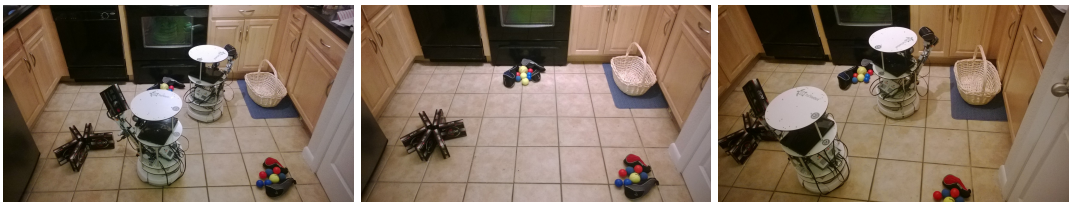

Figure 10.5: Users operated the two robots within a $6.3' \times 6.4'$ household area observed by one Kinect sensor and an overhead camera. The experimental scenarios required the subjects to use the robots to transport objects from the three piles, to the goal basket in the right corner.

The execution traces from the calibration tasks performed at the beginning of the experiment were used to classify user expertise on three teleoperation dimensions (navigation, manipulation, cooperation) using a semi-supervised k-means classifier (k=2, novice/expert) seeded with data from a pilot study. For the purposes of this experiment, participants who were rated by the classifier as being expert in two dimensions were grouped into the expert category. Users were also asked to assess their own performance after the initial training period. There was a high agreement between the subject's self-report of their skill on the various tasks and the classifier's assessment. Note that there is a strong similarity between our user interface controls for driving the robots and common gaming interfaces, and the number of users expert at navigation was higher than for the other two dimensions (manipulation and cooperation).

Table 10.1: Classifier Assessment of User Teleoperation Expertise

Dimension	# Expert	# Novice	Self-report agreement
navigation	24	16	95%
manipulation	17	23	85%
cooperation	15	25	90%

To evaluate user performance, we examined the task completion time and the number of object drops on the two different scenarios. For task completion time on Scenario 1, a two-way analysis of variance yielded a main effect for the user's expertise, F(1,64) = 37.2, $p < .01$, such that experts completed the scenario significantly faster than novices. The main effect of user interface type was non-significant, F(1,64) = 0.54. However, the interaction effect was significant, F(1,64) = 9.73, $p < .01$. Experts had a reduced task completion time using the configurable interface, whereas novices performed better with the adaptive one. A two-way analysis of variance yielded similar results on Scenario 2: a main effect for the user's expertise, F(1,64) = 23.9, $p < .01$, such that experts completed the scenario significantly faster than novices. The main effect of user interface type was non-significant, and the interaction effect was significant, F(1,64) = 12.1, $p < .01$. Experts had a reduced task completion time using the configurable interface, whereas novices performed better with the adaptive one.

We also manually counted the number of times the robots dropped objects in Scenario 1. In a two-way analysis of variance, neither main effect (expertise or interface type) resulted in significant differences, but the interaction effect was significant F(1,64) = 13.76, $p < .01$. Novices dropped fewer objects with the adaptive version of the interface, and experts dropped fewer objects with the configurable version. Scenario 2 followed the same pattern, with neither main effect being significant but with a significant interaction effect F(1,64) = 20.3, $p < .01$.

To better understand the users' experiences with the configurable interface, we measured the percentage of time that the robots operated autonomously (i.e. executed pre-programmed macro behaviors) out of the total scenario execution time. Figure 10.6 shows these results for both scenarios. The percentage of time that the robots were performing macro behaviors was comparable for novices and experts (no significant differences in the post-hoc comparison). In previous work [36], we had noted that differences may exist in the number of macros deployed and the length of macros created for more experienced vs. less experienced users, but apparently the amount of autonomy is comparable.

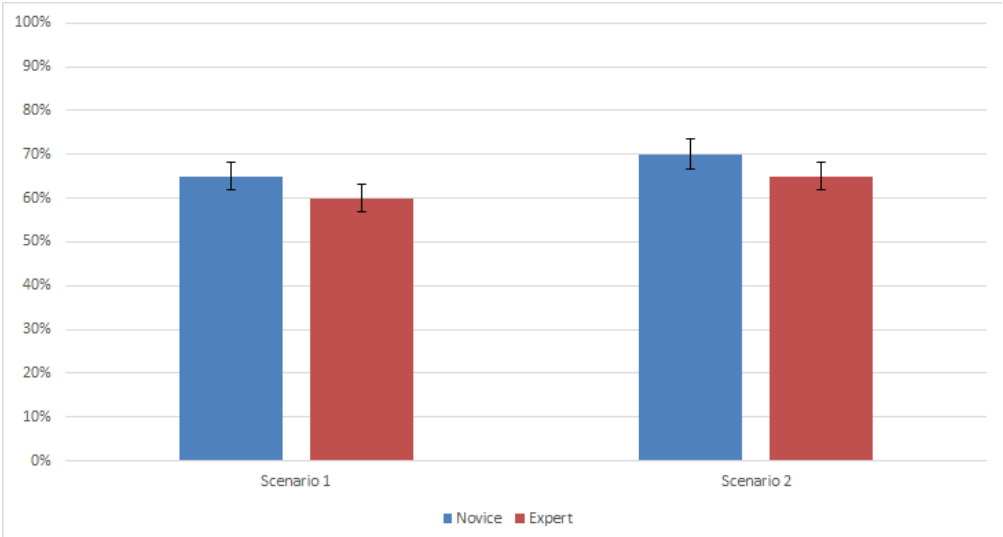

Figure 10.6: Autonomous operation (scenario %) using the configurable interface

On our post questionnaire we asked all participants about their usage strategies for both types of interfaces. Here are the comments from two novice and expert users.

1. **Expert** "For the configurable part I created longer macros that could help me complete the task faster, and just commanded both HU-IE robots. For the adaptive part I mainly focus on controlling one HU-IE robot, and only used the other when I needed to."

2. **Expert** ”For the adaptive section I was able to utilize the help I got with Navigation to travel the area and get the HU-IE robot in place for successful pick ups. For the configurable section i created macros for pick up and taking objects to the goal area to help me during the scenario.”

3. **Novice** ”Configurable: I created two pickup macros and two macros and drove to the piles. I made two pickup macros because I wanted to have more pickups map to buttons so if I pressed the wrong button it would still work. Adaptive: I used the help I got from the system to help me pick up objects and navigate them to the goal. with the help of the system I was able to complete the task pretty well and had less drops.”

4. **Novice** ”In the adaptive portion I was able to get help with getting the robot in place to pickup objects, and was able to use the help with controlling both robots at the same time. In the configurable portion I create simple short macros for pickup and navigation so I would not need to pickup objects myself.”

CHAPTER 11: CONCLUSION AND FUTURE WORK

Multi-robot manipulation tasks are difficult for a human-robot team to complete due to the complexity and responsibility that is required by the human user to manage and coordinate the robotic team. There is an elevated level of performance variability between a novice and an expert user's skill to teleoperate the robot team in a sufficiently tightly coupled manner to manipulate objects without dropping them. The ultimate success of the task relies on the skill level of the human operator to supervise and direct the robot team. Although the majority systems focus their efforts on finding connections between the robots and the human operator, less attention has been spent on the problem of identifying and adapting to the human operator's skill level.

This dissertation presented a user modeling technique for reliably predicting which group of users will perform well with a specific interface, adaptive vs. configurable. From a developer point of view it is essential to reduce "feature creep" —the addition of new unneeded functions that add complexity to the development process, without directly benefiting the user. Therefore, our aim is to create the simplest system that can still do a good job serving the user's needs for the specific task, while still accommodating the needs of a diverse user population. Not only can software and hardware failures sabotage performance but mismatches between the interaction paradigm and the task or between the users and the system are equally problematic. In a real-world usage scenario, the user's expertise, and hence their interaction style, is likely to change over time. We believe that activating or deactivating features in the interface is one way to accommodate the user's changing expertise.

Because this dissertation only scratches the surface of human robot interaction for multi-robot manipulation, there is ample future work to be explored. For instance, it would be valuable to the train classifier on a larger number of pilot users and extract more features from those runs.

Another extension would be to retrain the classifier in real time to handle fluctuations in the user's performance, due to experience or fatigue. In our current work, users that were experts in all three axes (Navigation, Manipulation, and Coordination) did not receive any autonomous help from the adaptive interface. However, in more difficult scenarios, experts may also need periodically help, and it would be useful to introduce a continuum of autonomy rather than treating autonomy as something that is either on or off. Further hardware expansions for the HU-IE robot are another realm to explore. Future work could include studying the performance of expert/novice users teleoperating manipulators with higher degrees of freedom.

APPENDIX: TECHNICAL DESCRIPTION

This appendix presents some additional details about the IAI interface and the HU-IE robot. The software for the IAI interface is publicly available at: `http://ial.eecs.ucf.edu/IAI_Package.zip`.

Robot Localization: The system manages and tracks the location of the HU-IE robots and objects in the area using a Microsoft Kinect sensor which provides RGB color and 3D depth imaging data to the user interface. The user interface uses a grid-based localization technique to keep track of all the robots. When the user interface is initialized, it creates an internal grid that is the same size as the physical area. Each in the grid is the size of an individual HU-IE robot. The display size of the grid is determined by the screen size of the Kinect sensor. The user interface has initial knowledge of the starting point of each robot on the grid and is able to store the sensor information from the Kinect in order to know where the robot has been and where it is currently located on the grid.

Kinect Tracking: The 3D depth imaging data is used to provide localization data for the robots and objects on the internal grid. The X and Z axes are extracted from the returned 3D depth data, and are used to determine where the robots and objects are located. Hence, the system maintains a simulated 3D world.

Compensate Tracking: The user interface keeps track of the robots' driving commands to know how long a robot has traveled. This tracking method compensates for any data latency between the Kinect sensor and the user interface. The variable (t) represents the amount of time for the wheels to rotate a full 360 degrees. It takes $5t$ to move from one grid cell to the next. The variable (t) can change depending on how much power is applied to the robots' drive motors. The same algorithm works for turns as well. The variable (r) is the amount of time it takes for the robot to turn one degree (d). The only difference is that the two wheels on the HU-IE robot turn in opposite

directions depending on if the robot needs to turn left or right. The robots' wheels are rubber based so there is little or no drifting in the small scenario area.

Path Planning: Since the user interface uses a grid to layout the area, A* is used for path planning. The A* algorithm uses a best-first search and finds a least-distance path from a given initial grid cell to one goal cell [5]. As A* traverses the grid, it follows a path of the lowest expected total distance, keeping a sorted priority queue of alternate path segments along the way. As opposed to traditional A* techniques, when a robot is on an autonomous course to a goal the user interface makes the cells in its path appear as occupied to the other robots; this is to prevent collisions. After the robot has reached the goal, the cells marked as occupied are released for use by the other robots.

Blob detection: A modified blob color detection technique is used to detect non skeletal objects using the Kinect sensor. A blob is an area of an image in which some properties are similar or dissimilar within a set assortment of values. All points in a blob can be considered to be comparable to each other [6]. The user interface uses common attributes found in the detected blob of an object, to distinguish between objects and robots. This technique considers the assumption that robots and objects are a specified color. The method uses a normalized color, and a similarity measure between current pixel vectors, registered color vectors, and segmentations. To detect an object or a robot, the method uses an image filtered by a predefined color model. After acquiring an image frame from the Kinect sensor, the function performs segmentation over the image. To detect an object region among the segmented candidate regions, the service uses an ellipse validation and two simple geometric constraints. The first constraint compares the preset colors of the robots and objects to a color blob on the image. The second constraint specifies that the object should be in the lower image plane.

Manipulation: Once an object is detected and a HU-IE robot is in range for a pickup, the HU-

IE's robotic arm is accurately lowered to the ground using information from the robotic arm's accelerometer sensor. There is also an ultrasonic sensor attached to the front of the HU-IE's claw to help guide the robot close enough to the object for a successful pickup. The following sensor information is sent from the HU-IE Robot for the IAI interface to calculate and attempt an autonomous pickup.

- Ultrasonic Sensor Information (attached to gripper)
- Accelerometer Sensor Information (attached to arm)
- Localization Information from the Kinect Sensor

The sensor information from the accelerometer sensor is used to position the arm at an angle (a) close enough to the ground for a successful pickup. The IAI maintains a list of predefined angles, known to be useful for object pickups. Once the arm angle is set to a predefined angle, it must now adjust the gripper for a successful pickup. All objects must also be in a predefined range for the ultrasonic sensor to detect the object and read that it is close enough for it to grasp. All objects and robots are tracked using the Kinect Sensor.

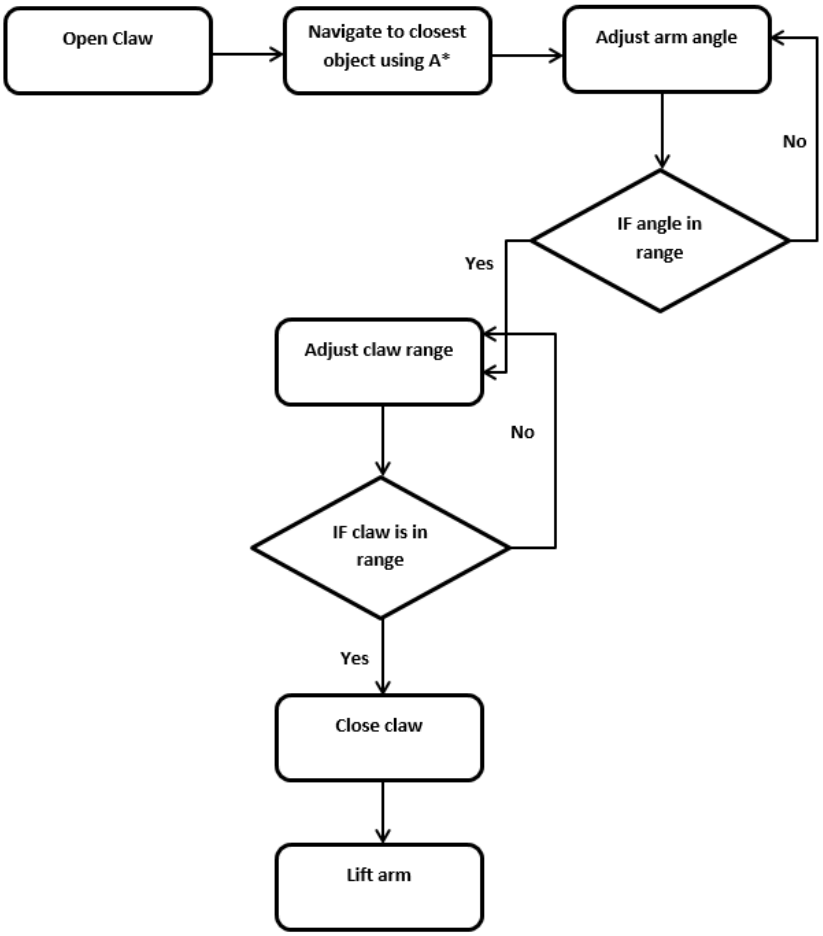

Figure 11.1: Block diagram describing how the system executes an manipulation task autonomously

APPENDIX: IRB

University of Central Florida

University of Central Florida Institutional Review Board
Office of Research & Commercialization
12201 Research Parkway, Suite 501
Orlando, Florida 32826-3246
Telephone: 407-823-2901 or 407-882-2276
www.research.ucf.edu/compliance/irb.html

Approval of Human Research

From: **UCF Institutional Review Board #1**
FWA00000351, IRB00001138

To: **Gita Reese Sukthankar**

Date: **September 30, 2010**

Dear Researcher:

On September 30, 2010, the IRB approved the following human participant research until 9/29/2011 inclusive:

Type of Review:	IRB Continuing Review Application Form
Project Title:	Psychological Models for Intent Inference
Investigator:	Gita Reese Sukthankar
IRB Number:	SBE-09-06439
Funding Agency:	University of Central Florida
Research ID:	1048486

The Continuing Review Application must be submitted 30days prior to the expiration date for studies that were previously expedited, and 60 days prior to the expiration date for research that was previously reviewed at a convened meeting. Do not make changes to the study (i.e., protocol, methodology, consent form, personnel, site, etc.) before obtaining IRB approval. A Modification Form **cannot** be used to extend the approval period of a study. All forms may be completed and submitted online at https://iris.research.ucf.edu .

If continuing review approval is not granted before the expiration date of 9/29/2011, approval of this research expires on that date. When you have completed your research, please submit a Study Closure request in iRIS so that IRB records will be accurate.

Use of the approved, stamped consent document(s) is required. The new form supersedes all previous versions, which are now invalid for further use. Only approved investigators (or other approved key study personnel) may solicit consent for research participation. Participants or their representatives must receive a copy of the consent form(s).

In the conduct of this research, you are responsible to follow the requirements of the Investigator Manual.

On behalf of Joseph Bielitzki, DVM, UCF IRB Chair, this letter is signed by:

Signature applied by Janice Turchin on 09/30/2010 04:29:57 PM EDT

Janice M Turchin

LIST OF REFERENCES

[1] P. Abbeel and A. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

[2] A.-R. Ahmad, O. Basir, and K. Hassanein. Adaptive user interfaces for intelligent e-learning: Issues and trends. In *The Fourth International Conference on Electronic Business (ICEB2004)*, pages 925–934, 2004.

[3] Anonymous, 2011.

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[5] A* search algorithm web page, 2010. `http://www.princeton.edu/~achaney/tmve/wiki100k/docs/A*_search_algorithm.html`.

[6] society of robots web page, 2011. `http://www.societyofrobots.com/programming_computer_vision_tutorial_pt3.shtml`.

[7] N. Boonpinon and A. Sudsang. Formation control for multi-robot teams using a data glove. In *RAM Robotics Automation and Mechatronics*, 2008.

[8] C. Breazeal, A. Brooks, D. Chilongo, J. Gray, A. Hoffman, C. Lee, J. Lieberman, and A. Lockered. Woorking collaboratively with humanoid robots. *ACM Computers in Entertainment*, 2(3), 2004.

[9] C. Breazeal and A. Thomaz. Learning from human teachers with socially guided exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[10] J. J. Bryson. Representations underlying social learning and cultural evolution. *Interaction Studies*, 10(1):77–100, March 2009.

[11] J. Casper and R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33(3):367–385, 2003.

[12] J. Chen and M. Barnes. Supervisory control of multiple robots: Effects of imperfect automation and individual differences. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 54:157–173, 2012.

[13] H. Cheng, Z. Sun, and P. Zhang. Imirok: Real-time imitative robotic arm control for home robot applications. In *IEEE International Conference on Pervasive Computing (Work in Progress)*, pages 360–363, 2011.

[14] J. Crandall, M. Goodrich, J. Olsen, D.R., and C. Nielsen. Validating human-robot interaction schemes in multitasking environments. *IEEE Transactions on Systems, Man and Cybernetics*, 35(4):438–449, 2005.

[15] H. Dang and P. Allen. Robot learning of everyday object manipulations via human demonstration. In *IEEE International Conference on Intelligent Robots and Systems*, 2010.

[16] A. Dragan, S. Srivastava, and K. Lee. Teleoperation with intelligent and customizable interfaces. *Journal of Human-Robot Interaction*, 2(2):33–79, 2013.

[17] E. Eaton. Gridworld search and rescue: A project framework for a course in artificial intelligence. In *In the Proceedings of the AAAI-08 AI Education Colloquium*, Chicago, IL, July 2008.

[18] A. Edsinger and C. Kemp. Human-robot interaction for cooperative manipulation: Handing objects to one another. In *The IEEE International Symposium on Robot and Human interactive Communication*, pages 1167–1172, 2007.

[19] X. Fan and J. Yen. Realistic cognitive load modeling for enhancing shared mental models in human-agent collaboration. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2007.

[20] T. Fong. Collaborative control: A robot-centric model for vehicle teleoperation. Technical report, Robotics Institute, Carnegie Mellon, 2001.

[21] D. Francois, D. Polani, and K. Dautenhahn. Toward socially adaptive robots: A novel method for real time recognition of human-robot interaction styles. In *IEEE International Conference on Humanoid Robots*, pages 353–359, 2008.

[22] A. Freedy, O. Sert, E. Freedy, M. Tambe, T. Gupta, W. Grayson, and P. Cabrera. Multiagent adjustable autonomy framework for multi-robot, multi-human teams. In *International Symposium on Collaborative Technologies and Systems*, pages 498–505, 2008.

[23] C. Fua, S. Ge, and K. Lim. Task allocation for multi-robot teams with self-organizing agent. *International Conference on Intelligent Robots and Automation*, 2006.

[24] M. Goodrich and A. Schultz. Human-robot interaction: a survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.

[25] J. Gorostiza, R. Barber, A. Khamis, M. Malfaz, R. Pacheco, R. Rivas, A. Corrales, E. Delgado, and M. Salichs. Multimodal human-robot interaction framework for a personal robot. In *IEEE International Symposium on Robot and Human Interactive Communication*, pages 39–44, 2008.

[26] D. Grollman and O. Jenkins. Learning robot soccer skills from demonstration, 2007.

[27] Y. Hirata, Y. Kume, Z.-D. Wang, and K. Kosuge. Decentralized control of multiple mobile manipulators based on virtual 3-D caster motion for handling an object in cooperation with a human. In *International Conference on Robotics and Automation*, 2003.

[28] G. Hoffman and C. Breazeal. Collaboration in human-robot teams. In *Proceedings of AIAA Intelligent Systems Technical Conference*, 2004.

[29] M. Johnson, J. Bradshaw, P. Feltovich, C. Jonker, B. van Riemsdijk, and M. Sierhuis. The fundamental principle of coactive design: Interdependence must shape autonomy. In M. D.

Vos, N. Fornara, J. Pitt, and G. Vouros, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems VI*, pages 172–191. Springer Berlin/Heidelberg, 2010.

[30] K. Kawamura, P. Nilas, K. Muguruma, J. Adams, and C. Zhou. An agent-based architecture for an adaptive human-robot interface. In *Hawaii International Conference on System Sciences*, 2002.

[31] K. Kawamura, P. Nilas, K. Muguruma, J. Adams, and C. Zhou. An agent-based architecture for an adaptive human-robot interface. *IEEE*, 2003.

[32] O. Khatib, K. Yokoi, O. Brock, K. Chang, and A. Casal. Robots in human environments basic autonomous capabilities. In *The International Journal of Robotics Research*, pages 684–696, 1999.

[33] S. Kim, D. Lee, S. Hong, Y. Oh, and S. Oh. From human motion analysis to whole body control of a dual arm robot for pick and place tasks. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1155–1161, 2013.

[34] L. Kunze, A. Haidu, and M. Beetz. Acquiring task models for imitation learning through games with a purpose. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 102–107, 2013.

[35] B. Lewis and G. Sukthankar. Two hands are better than one: Assisting users with multi-robot manipulation tasks. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2590–2595, San Francisco, CA, Sept 2011.

[36] B. Lewis and G. Sukthankar. Configurable human-robot interaction for multi-robot manipulation tasks. In *AAMAS Workshop on Autonomous Robots and Multi-robot Systems*, pages 51–70, Valencia, Spain, June 2012.

[37] B. Lewis, B. Tastan, and G. Sukthankar. Agent assistance for multi-robot control (extended abstract). In *Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1505–1506, Toronto, CA, May 2010.

[38] B. Lewis, B. Tastan, and G. Sukthankar. Improving multi-robot teleoperation by inferring operator distraction (extended abstract). In *Proceedings of International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2010.

[39] B. Lewis, B. Tastan, and G. Sukthankar. An adjustable autonomy paradigm for adapting to expert-novice differences. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1656–1662, TOYKYO, JAPAN, November 2013.

[40] M. Lewis, H. Wang, S. Chien, P. Scerri, P. Velagapudi, and K. Sycara. Teams organization and performance in multi-human/multi-robot teams. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2010.

[41] H. Lieberman. *Your Wish is My Command: Programming by Example*. Morgan Kaufmann, 2001.

[42] J. Mainprice and D. Berenson. Human-robot collaborative manipulation using early prediction of human motion. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 299–306, 2013.

[43] M. Martins and Y. Demiris. Learning multirobot joint action plans from simultaneous task execution demonstrations. In *Proceedings of International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, pages 931–938, 2010.

[44] S. Morgan. *Programming Microsoft Robotics Studio*. Microsoft Press, 2008.

[45] S. Muszynski, J. Stuckler, and S. Behnke. Adjustable autonomy for mobile teleoperation of personal service robots. In *Proceedings of the IEEE International Conference on Symposium on Robot and Human Interactive Communication*, 2012.

[46] T. Okada, R. Beuran, J. Nakata, Y. Tan, and Y. Shinoda. Collaborative motion planning of autonomous robots, 2010.

[47] R. Parasuraman, T. Sheridan, and C. Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics: Part A*, 30(3):286–295, 2000.

[48] D. Perzanowski, A. Schultz, W. Adams, E. Marsh, and M. Bugajska. Building a multimodal human-robot interface. *IEEE Intelligent Systems*, pages 16–21, January/February 2001.

[49] S. Pieska, J. Kaarela, and O. Saukko. Toward easier human-robot interaction to help inexperience operators in smes. In *IEEE International Conference on Cognitive Infocommunications*, pages 333–338, Dec 2012.

[50] Qwerk robot platform. `http://www.charmedlabs.com/`.

[51] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Readings in speech recognition*, pages 267–296. Morgan Kaufmann Publishers Inc., 1990.

[52] B. Ricks, C. Nielsen, and M. Goodrich. Ecological displays for robot interaction: a new perspective. In *Proceedings of Intelligent Robots and Systems*, 2004.

[53] Robot at home. `http://www.ai.rug.nl`.

[54] Robot rescue, 2009. `http://www.robocuprescue.org/`.

[55] S. Rosenthal, J. Biswas, and M. Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. In *Proceedings of International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2010.

[56] N. Sato, K. Kon, H. Fukushima, and F. Matsuno. Map-based navigation interface for multiple rescue robots. In *IEEE International Workshop on Safety, Security and Rescue Robotics*, pages 152–157, 2008.

[57] P. Scerri, D. Pynadath, and M. Tambe. Adjustable autonomy for the real world. In *Agent Autonomy*, pages 163–190. Kluwer, 2003.

[58] P. Scerri, K. Sycara, and M. Tambe. Adjustable autonomy in the context of coordination. In *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, 2004. Invited Paper.

[59] M. Schneider and W. Ertel. Robot learning by demonstration with local gaussian process regression. In *IEEE International Conference on Intelligent Robots and Systems*, 2010.

[60] N. Shirakura, M. Morita, and J. Takeno. Development of a human interface for remote-controlled robots using an eye-tracking system. In *IEEE International Conference on Mechatronics and Automation*, 2005.

[61] M. Sierhuis, J. M. Bradshaw, A. Acquisti, R. van Hoof, R. Jeffers, and A. Uszok. Human-agent teamwork and adjustable autonomy in practice. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2003.

[62] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Romea, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and J. Vandeweghe. Herb: a home exploring robotic butler. *Autonomous Robots*, 28(1):5–20, January 2010.

[63] J. Wang, M. Lewis, and P. Scerri. Cooperating robots for search and rescue. In *Proceedings of AAMAS Workshop on Agent Technology for Disaster Management*, 2006.

[64] J. Wang, H. Wang, M. Lewis, P. Scerri, P. Velagapudi, and K. Sycara. Experiments in coordination demand for multirobot systems. In *Proceedings of IEEE International Conference on Distributed Human-Machine Systems*, 2008.

[65] E. Wenger. Artificial intelligence and tutoring systems. *International Journal of Artificial Intelligence in Education*, 14:39–65, 2004.