HUMAN ROBOT INTERACTION FOR MULTI-ROBOT SYSTEMS

A Dissertation Proposal by

Bennie G. Lewis Jr.

11 October 2013

Submitted to the graduate faculty of the

Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements

for the Dissertation Proposal and

subsequent Ph.D. in Computer Engineering

Approved By:
Gita Sukthankar, Committee Chair
Joseph LaViola
Peter Hancock
Charles Hughes
Annie Wu

Outline

Abstract **ii**

1 Introduction **1**

2 Related Work **3**
2.1 Human Robot Interaction 3
2.2 Multi-Robot Manipulation 6
2.3 Learning by Demonstration 8
2.4 User Interfaces 9
2.5 Application Domains 11

3 Experimental Domain **13**
3.1 The Gridworld Search and Rescue Simulator 13
3.2 Microsoft Robotics Studio (MSRS) 14
3.3 Robotic Search and Rescue Simulation Toolkit (RSARSim) 17
3.4 Home and Urban Intelligent Explorer (HU-IE) First Generation 18
3.5 Home and Urban Intelligent Explorer (HU-IE) Second Generation 22

4 Completed Work **25**
4.1 Simple USAR User Interface. 25
4.2 Multi-Robot Coordination in a Complex USAR Scenario. 28
4.3 Learning Models of Operator Distraction for Human-Robot Interfaces 31
4.4 Assisting Users with Multi-Robot Manipulation 43
4.5 Configurable Human-Robot Interaction for Multi-Robot Manipulation 54
4.6 Adapting to Expert-Novice Differences in Human-Robot Interaction 70

5 Proposed Work **82**
5.1 Multi-Modal Interface and Natural Interfacing 82
5.2 Learning by Demonstration for Expert and Novice Users 82
5.3 Timeline 83
5.4 Publications 83

Abstract

The multi-robot manipulation task, where a team of robots cooperatively lift and move objects, is exceptionally difficult due to the synchronization between the robots. 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.

In this dissertation, I propose to develop an agent-based user interface for multi-robot manipulation interface that identifies and manages any single robot that is not currently being controlled by the human operator, as well a learning from demonstration technique with the robots learning from the pre recorded macro representation of the specified task. Propagating the user's macro commands from the actively-controlled robot to the neglected robot allows the neglected robot to leverage this control information and position itself effectively without direct human supervision.

The objective of robot learning from demonstration is to have a robot learn from watching a demonstration of the task to be performed. A policy is calculated based on the task learned from demonstrations in which the user verifies that the robot has executed the macro correctly in a new situation.

1 Introduction

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. Thus without a good user interface, simply adding more robots to the system does not necessarily expand the effective coverage region nor increase the rate at which the operator can search. 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. A key contribution of this work is to automatically detect which robots in the team are currently being ignored by the human operator and to manage those intelligently during such periods. Multi-robot manipulation, where two or more robots cooperatively grasp and move objects, is extremely challenging due to the coordination between the robots. 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. Multi-robot manipulation systems have options that are unavailable to normal robots. 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.

This dissertation proposes an alternate approach in which the agents and robots leverage information about what the user is doing and has recently done to decide their future course of action. I present an agent-based multi-robot manipulation interface that identifies and manages any single robot that is not currently being managed by the human operator, as well as a learning from demonstration technique with the robots learning from the pre recorded macro representation of the specified task. Our results demonstrate how this method can reduce task completion time and the number of dropped items over a fully teleoperated robotic system. Propagating the user's macro commands from the actively-controlled robot to the neglected robot allows the neglected robot to leverage this control information and position itself effectively without direct human supervision.

This dissertation will focus on three key research areas:

- Human robot interaction
- Multi robot manipulation
- Adjustable autonomy

In the next section, I provide an overview of related work in these areas.

2 Related Work

Table 1 provides a breakdown of the key research related to this dissertation.

Resarch Focus	Key Papers
Human Robot Interaction	[12, 23, 43, 39, 13, 18] [6, 5, 8, 45, 46] [26, 27, 42]
Multi-Robot Manipulation	[21, 16, 22] [27, 4, 31, 14, 9] [37, 44]
Learning by Demonstration	[41, 29, 9, 15] [28, 3, 1]
User Interfaces	[19, 17, 45, 38, 34, 20] [2, 47]
Application Domains	[36, 40, 45, 35] [27]

Table 1: Breakdown of Related Work

2.1 Human Robot Interaction

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 [12, 23] or adjustable autonomy [43, 39]. However fundamentally, synthetic and biological entities have very different capabilities that need to be respected during task division. Collaborative control [13] 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. [18]. 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 predetermined 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 [6]; 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 [5].

Operator neglect was identified as an important factor by Crandall et al. [8] 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 [45] 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 [46] 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 [12, 26] and compensated for by the robots.

Adjustable autonomy, having the robots alter their level of autonomy in a situationally-dependent manner, has been used successfully in human-robot teams [39, 43]. 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.

Lewis and Scerri [27] describe many applications of mobile robotics that require multiple robots. Multiple robots substantially increase the complexity of the operator's task because attention must be continually shifted among robots. One approach to increasing human capacity for control is to remove the independence among robots by allowing them to cooperate. Lewis and Scerri presented an experiment using multi agent teamwork proxies to help 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. Providing additional autonomy by enabling robots to cooperate among themselves extends automation to human control activities previously needed to coordinate the robots' events. Automating this task ought to decrease the demands on the human operator to the extent that attention is devoted to robot involved coordination with other robots. This coordination automation method allowed improvements in performance for control of larger teams.

Conventional models of multi robot control presupposes 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 [45] introduced a model of coordination demand and presented experiments that extend the neglect tolerance model to situations in which robots must cooperate to perform dependent tasks. The operators controlled two robot teams to perform a box pushing task under high cooperation demand, moderate demand and low demand conditions in the first experiment. Participants performed a search and rescue task requiring cooperation between robots creating maps and others carrying cameras. The authors' approach to measuring coordination demand was supported in both experiments. The second experiment illustrates the need for more sophisticated actions that can take into account strategies and outlines of actions as well as their durations.

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 [12] 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.

Shirakura and Morita [42] introduced a method by which the operator gains a sense of the environment surrounding the robot as a 3D space in the same manner as in normal human sight using an eye tracking system, This technique is based on the practice of detecting and tracking the line of sight of the operator and then changing the line of sight of the robot in conjunction with it. As a result of using this technique to realize a human robot interface the authors were able to demonstrate reduced eye fatigue when viewing 3D images and to simplify the process of robot control for the operator.

2.2 Multi-Robot Manipulation

Khatib et al. [21] 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 system 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 [16].

Numerous new applications require robots to work alongside people as capable members of human-robot teams. These include robots for homes and urban environments. Multi-robot systems possess benefits over single robot counterparts. [27] A team of robots can handle a wider range of responsibilities and accomplish many tasks more efficiently than a single robot. In addition, deploying a team of robots can provide robustness and fault tolerance.

Haptics can be a valuable tool for the human-robot interaction of manipulation tasks, allowing the operator a more immersive telepresence. Boompion and Sudsang [4] 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. The authors provided simulation results that confirmed effectiveness of their research approach.

Okada and Beuran [31] 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 with other robots or obstacles 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 [14] proposed an 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. The proposed method allowed robot teams to adapt to unforeseen circumstances, while at the same time, still making use of available time information for preliminary scheduling and planning. Realistic simulations were performed and verified the effectiveness of the approach.

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. [37] 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. [44] 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.

2.3 Learning by Demonstration

Learning from Demonstration is one method 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.

Schneider and Ertel [41] reviewed Gaussian processes and showed how these can be used to encode a robotic task. The authors introduce a new Gaussian process regression model that clusters the input space into smaller subsets. They demonstrated how these approaches fit into learning by demonstration and presented an experiment on a physical robot arm that shows how all these approaches interact. 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 [29]. 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 [9] 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 [15] 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, the authors exhibited a learning paradigm that is capable of learning both low-level motion primitives and high-level tasks built on top of them 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.

Programming by example has been integrated into an assortment of demonstrational user interface systems (see [28] 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 [3] 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 by demonstration this 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.

2.4 User Interfaces

There are many different ways to structure a RAP (Robots, Agent, People) system. For instance, Kawamura et al. [19] 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 [17] 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 [45].

Sato and Kon [38] present a navigation interface for multiple robots. The authors describe the development of an interface 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. A comparative experiment showed that the system is more efficient then the other approaches described in the paper.

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 [34] 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. [20] 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 [47]. However unlike adaptive intelligent tutoring systems, our user interface models but does not attempt to improve the user's teleoperation skills. We believe that the problem of attempting to train the users in addition to compensating for their weaknesses, is an interesting area for future work.

2.5 Application Domains

The Robocup Rescue League [36] 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 which includes; adjusting the autonomy level of the individual robotic agents [40], 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 [45]. 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 [35] 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.

Lewis and Scerri [27] developed a theory for human controlled robot teams describing how control varies across different task allocations. The authors' work focused on areas in which a team of robots perform mostly independent tasks. Their recent studies address the interaction between automation and organization of human teams in controlling large robot teams performing an Urban Search and Rescue task. The authors identified three subtasks: search, assistance, and navigation. Navigation was the main focus of their studies on automation because it involves fragile dependencies among the team of robots making it more complex. The authors focused on two possible ways to arrange operators that belonged to a shared pool from which operators service robots from the population as needed. The experiment evaluated two member teams of operators controlling teams of 12 robots and sharing control of 24 robots in the shared pool conditions using either waypoint control or autonomous path planning. The authors distinguished three categorized team tactics in the shared pool condition: joint control in which operators share full authority over robots, mixed control in which one operator takes primary control while the other acts as an assistant, and split control in which operators divide the robots with each controlling a sub team. Results of team organization favored operator teams who shared authority for the pool of robots.

3 Experimental Domain

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

3.1 The Gridworld Search and Rescue Simulator

The first simulation we used while tackling problems in HRI was the Gridworld Search and Rescue (GSAR). GSAR was developed by Eric Eaton [10]. 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. The developers are able to ignore the low-level details and focus on applying AI techniques. It uses a networked client-server framework to allow 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 which is indicated be vital signs. 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

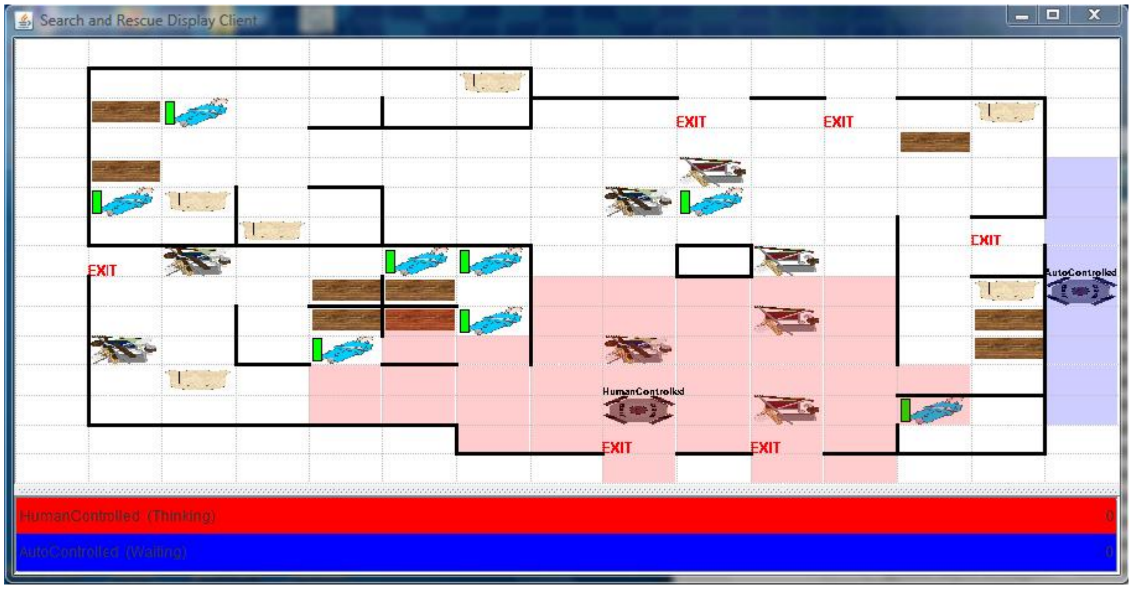


Figure 1: Gridworld Search and Rescue (GSAR) Simulator.

exit are not worth any points.

3.2 Microsoft Robotics Studio (MSRS)

The Microsoft Robotics Studio (MSRS) is a development environment tool for robot control and simulation [30]. It is aimed at academic, hobbyist, and commercial developers and handles a wide variety of robot hardware. The Visual Simulation Environment includes hardware acceleration, a services-oriented runtime and access to a robot's sensors and actuators through a .NET-based concurrent library implementation. MSRS runs on the .NET framework which allows the use of Microsoft Visual Studio to design and develop robot applications. 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. Microsoft XNA and DirectX 9 are used in VSE to do 3-D rendering. 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. With that being said 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. Visual mode is a representation of what the robot would look like in a real world environment.

Services are the essential building blocks for robotic applications created with Microsoft Robotics Developer Studio [30]. 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. Microsofts 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

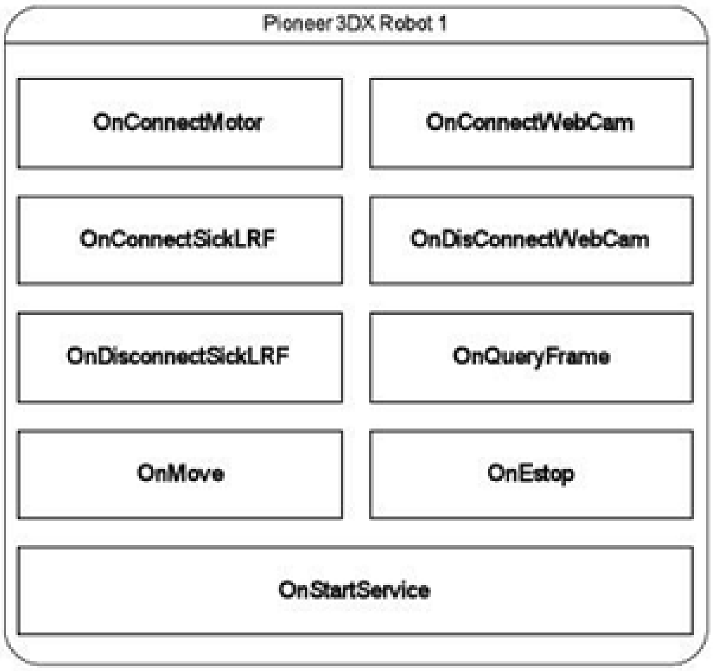


Figure 2: Services used to operate the Pioneer 3DX Robot.

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 Microsofts 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, 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

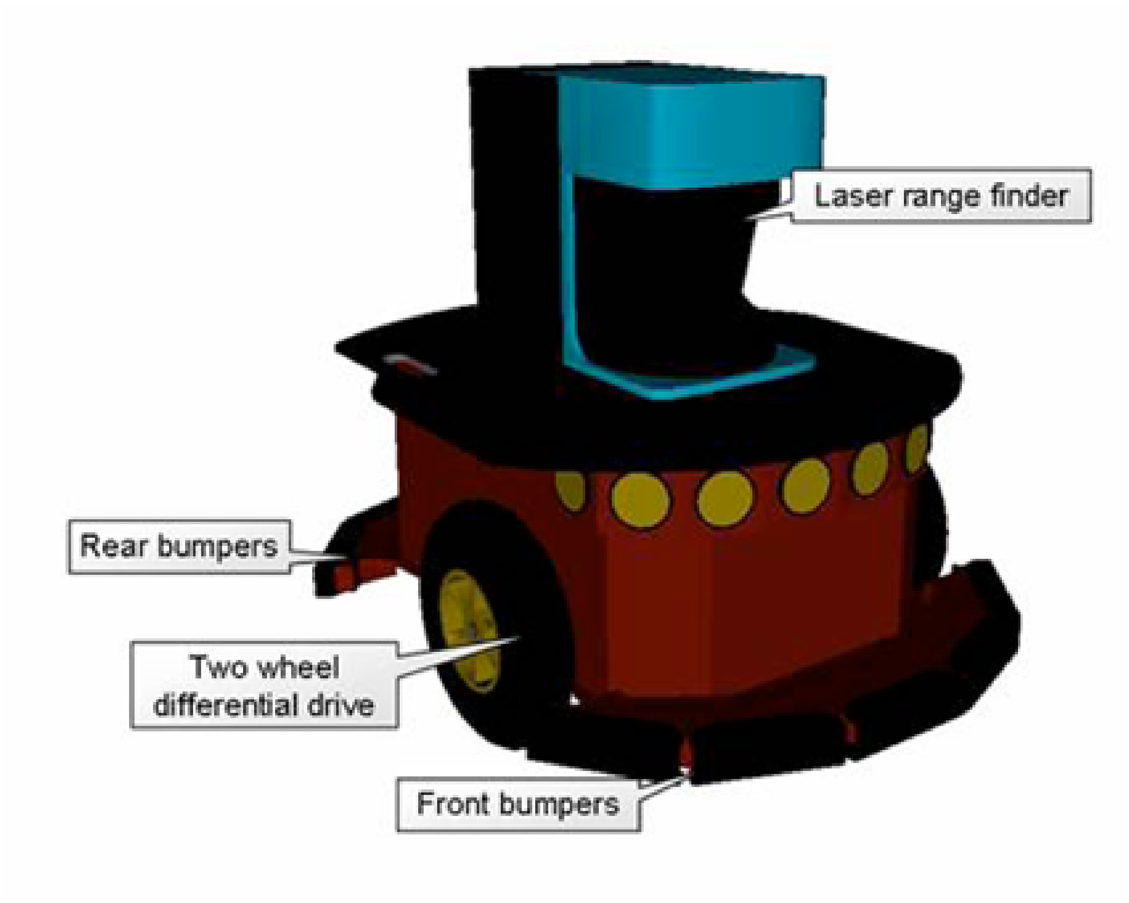


Figure 3: Simulated Pioneer 3DX robot [30].

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.

3.3 Robotic Search and Rescue Simulation Toolkit (RSARSim)

Since none of the existing simulation environments satisfied all of our experimental needs, we designed 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 editions 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.

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

The Home and Urban Intelligent Explorer (HU-IE)1.0, features a mobile base attached to an arm and gripper (Figure 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 [32], the arm from the NXT 2.0 Robotics Kit, and a Logitech Communicate STX Webcam.

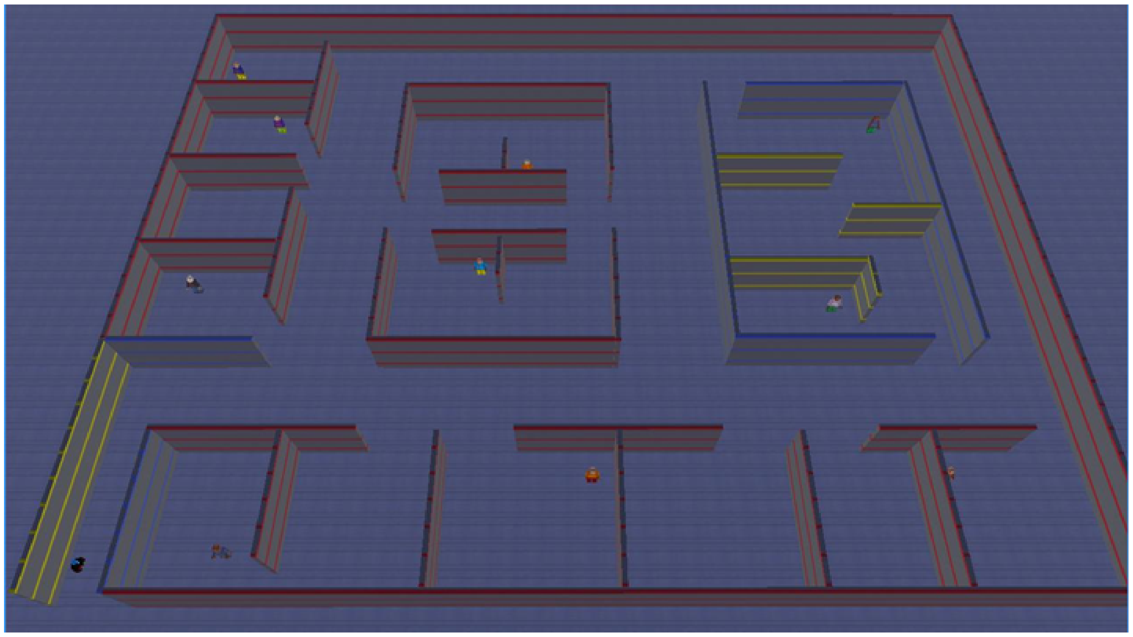


Figure 4: An search and rescue scenario simulated using RSARSim. The user is controlling a single Pioneer robot.

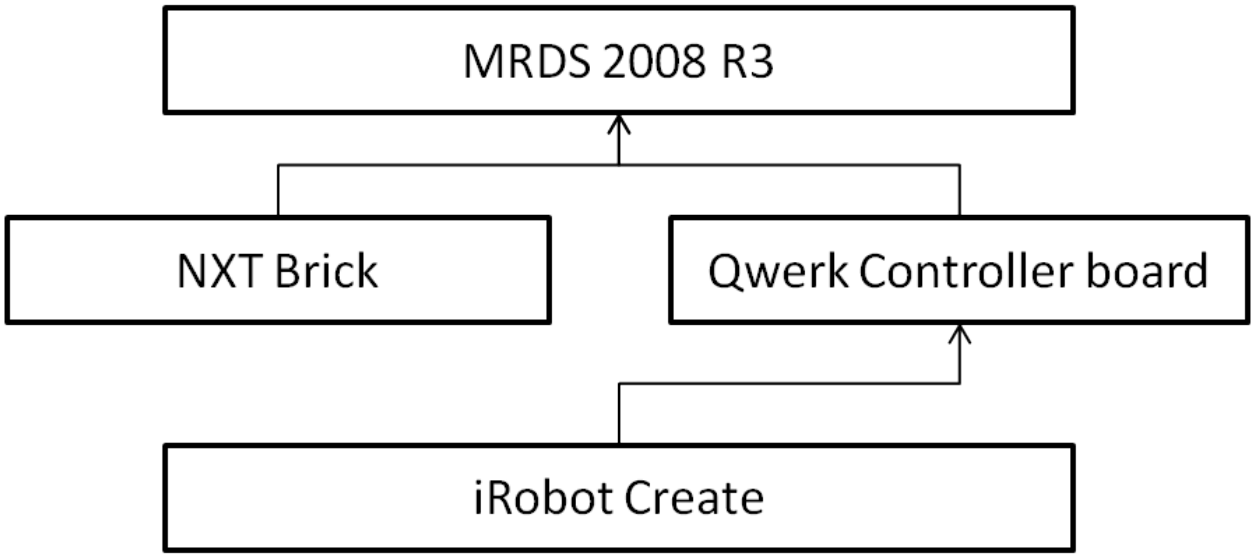


Figure 5: Overview of HU-IE Robot System hardware components.

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.

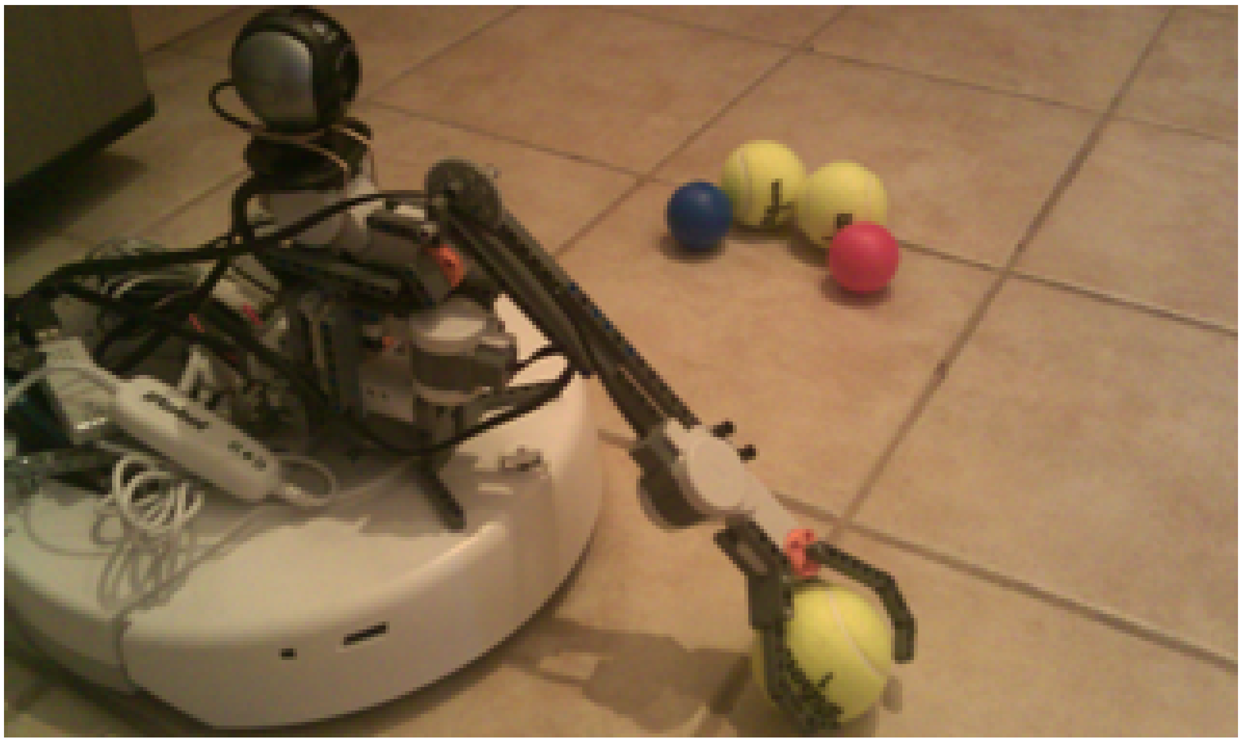


Figure 6: HU-IE combines a mobile base (3 DOF) with a robot arm (2 DOF) equipped with a gripper. This enables HU-IE to navigate indoor environments and pick up small objects. The user can wirelessly teleoperate the robot using a webcam.

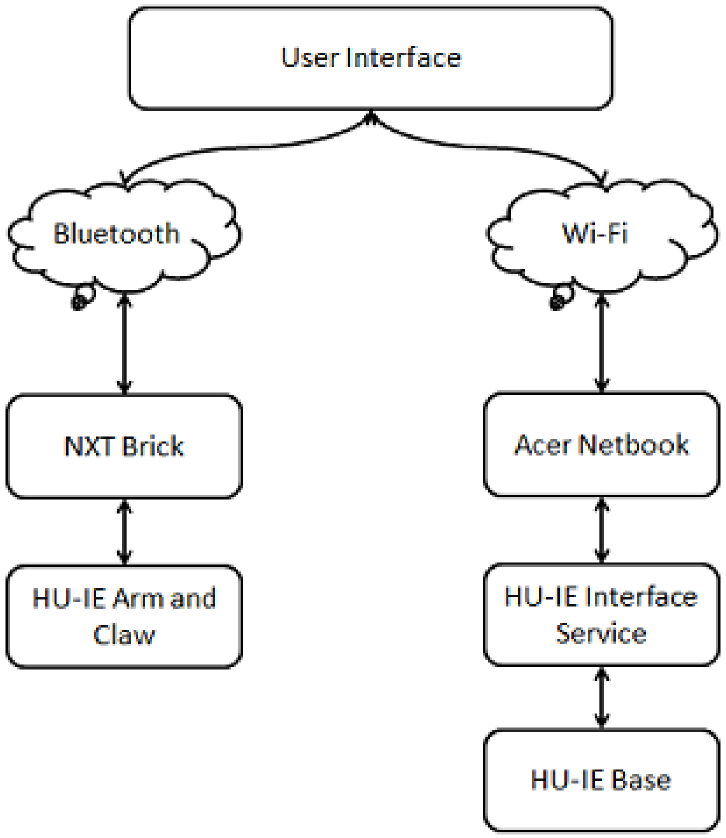


Figure 7: Connections between the HU-IE robot hardware components.

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

Due to the lack of sensors,proccessing power,and the Qwerk board becoming obsolete, we decided to upgrade the HU-IE 1.0. The Home and Urban Intelligent Explorer (HU-IE) 2.0 system is designed to be proficient at picking up light objects in a household environment with either carpets or hard floors. Having the arms on separate robots makes the pickup task more challenging but allows the user to parallelize large sections of the delivery task. Our robot 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 7 shows the robot architecture.

The iRobot Create has a differential drive that allows left and right wheel speeds to be independently specified and two bump sensors for detecting physical collisions. In addition to the internal

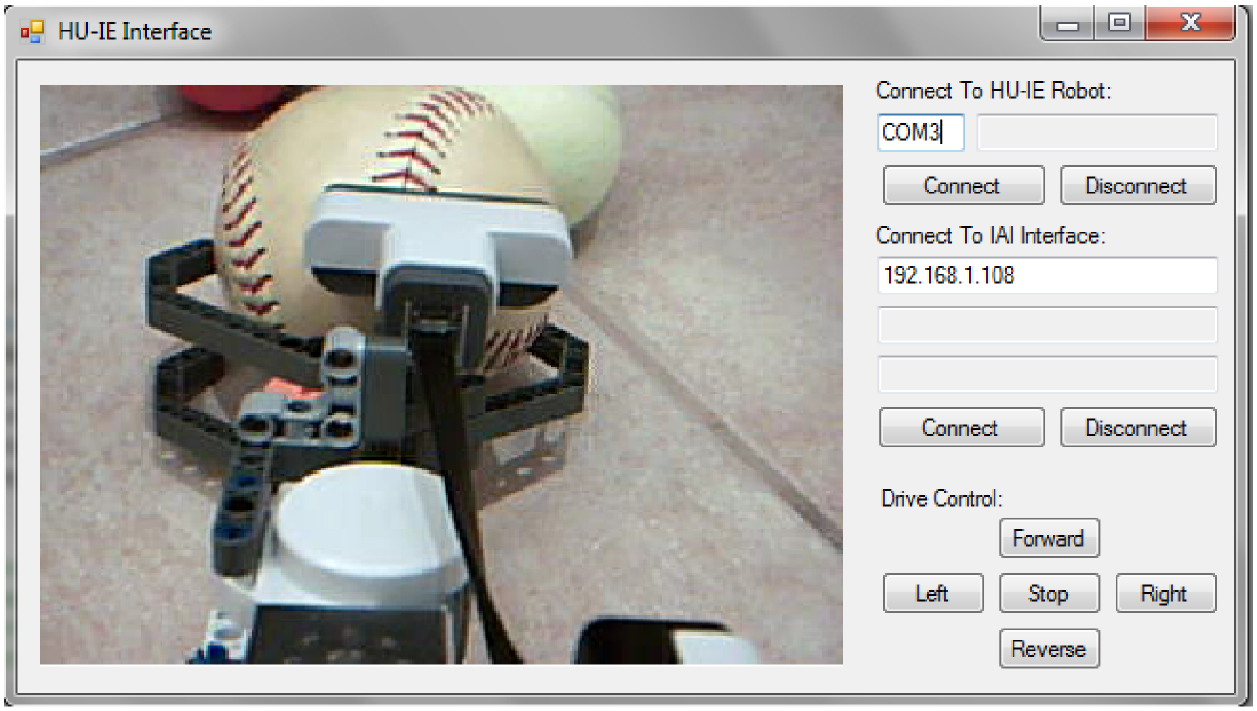


Figure 8: The user views the object being manipulated through a webcam mounted on the robot's arm.

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. A small webcam mounted on the robot arm presents a first-person perspective to the user during teleoperation. 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

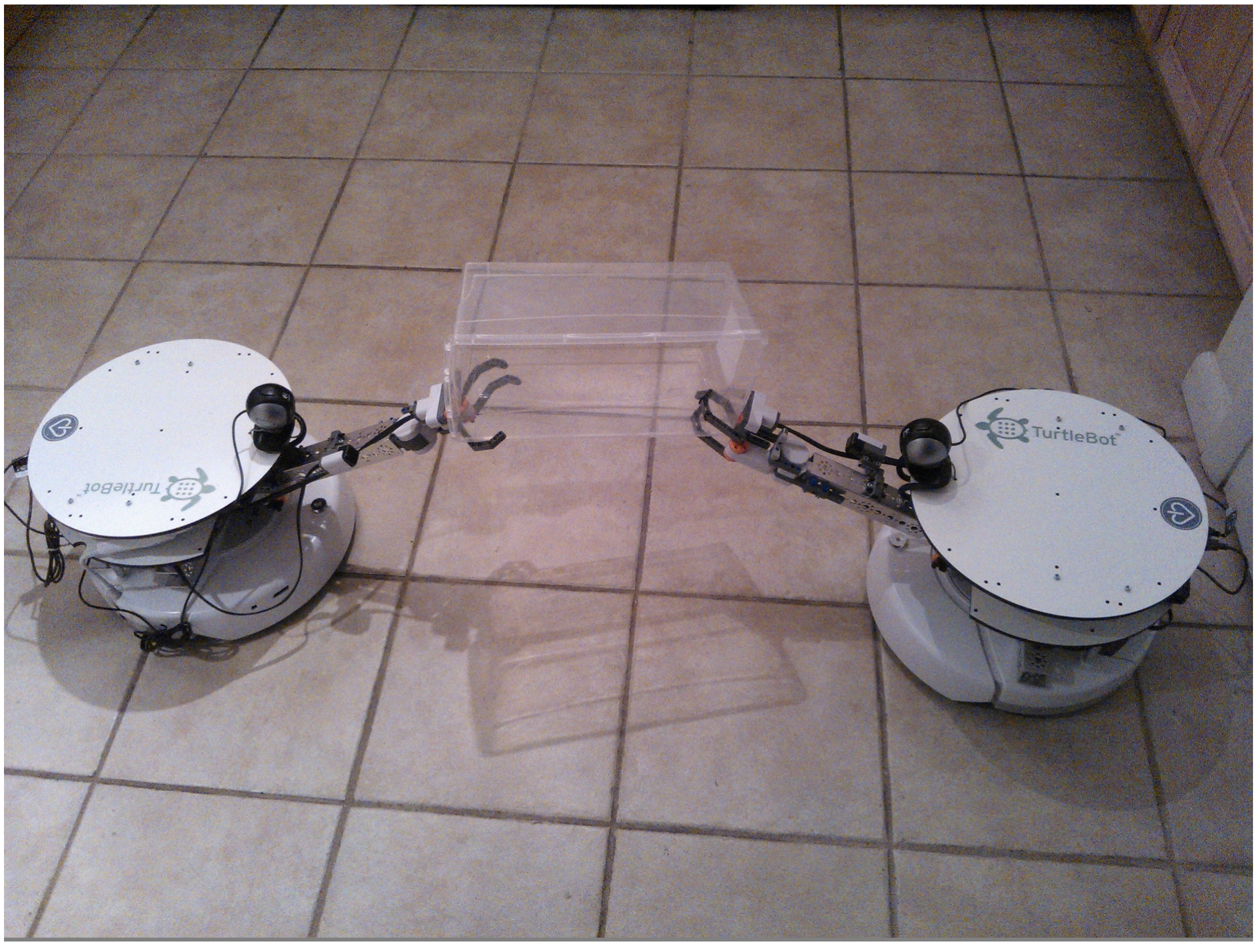


Figure 9: Two robots cooperate to lift an object under the direction of the human operator. In the multi-robot manipulation task, the robots must lift and deliver a series of objects of different sizes to the goal location.

Bluetooth, bypassing the Acer netbook.

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.

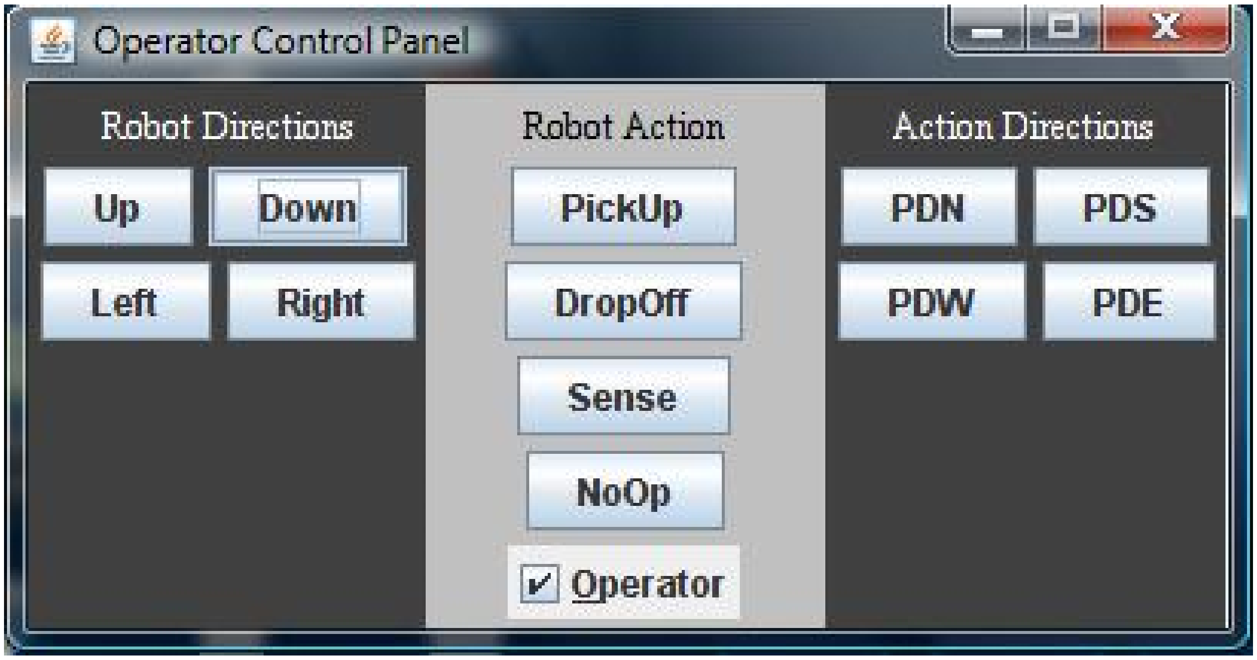


Figure 10: Operator Control Panel.

4 Completed Work

4.1 Simple USAR User Interface.

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 sitting in idle more 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 and implemented 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. Pickup and drop-off victims or objects,

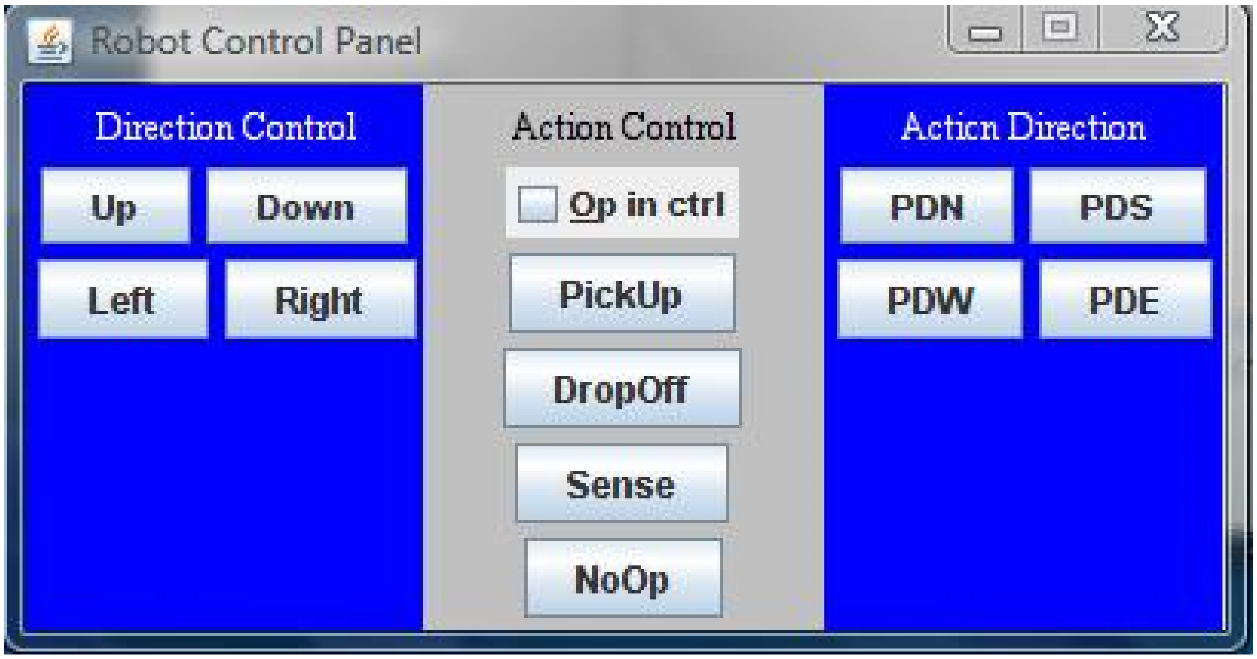


Figure 11: Robot Control Panel.

navigate and sense for victims, and switch the 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 find and rescue the victims in a specified timeframe. Each robotic agent was developed with 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 time step 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.

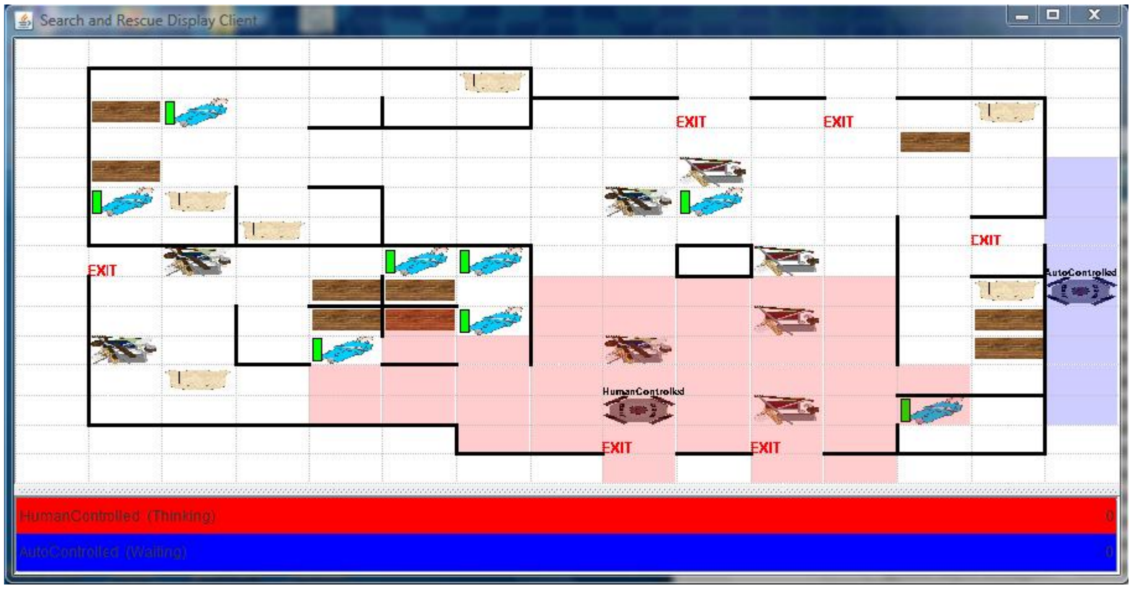


Figure 12: Gridworld Search and Rescue (GSAR) Simulator.

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 automatically that the human operator can perform 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 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.

The importance of having human operators work with robots is the main issue for search and rescue. In our study we have 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.

4.2 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.

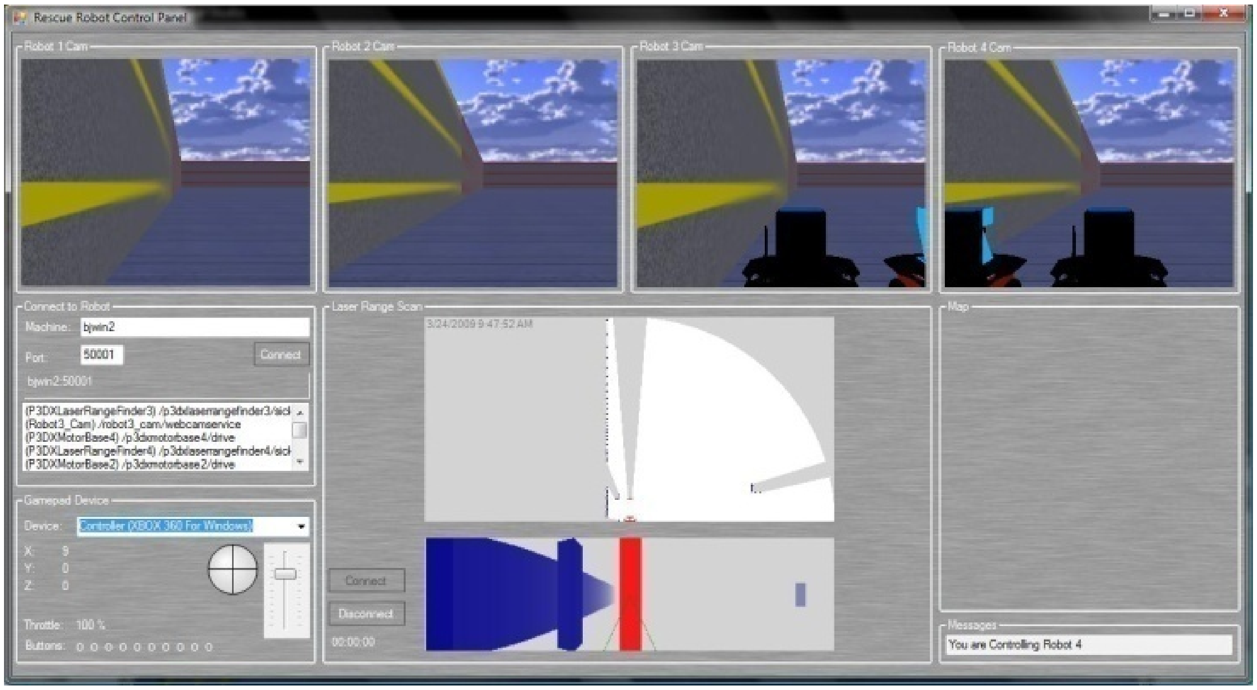
OnConnectSickLRF and OnDisConnectSickLRF: manages the connection and disconnection of the laser range finder of the four Pioneer 3DX robots.

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.

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



Machine: bjwin2
Port: 50001
bjwin2:50001

```
(P3DXLaserRangeFinder3) /p3dxlaserrangefinder3/sick
(Robot3_Cam) /robot3_cam/webcamservice
(P3DXMotorBase4) /p3dxmotorbase4/drive
(P3DXLaserRangeFinder4) /p3dxlaserrangefinder4/sick
(P3DXMotorBase2) /p3dxmotorbase2/drive
```

Device: Controller (XBOX 360 For Windows)
X: 9
Y: 0
Z: 0
Throttle: 100 %
Buttons: 0 0 0 0 0 0 0 0 0 0

3/24/2009 9:47:52 AM
00:00:00

You are Controlling Robot 4

Figure 13: Robot Rescue Control Panel.

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. Results from a small pilot study indicated 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.

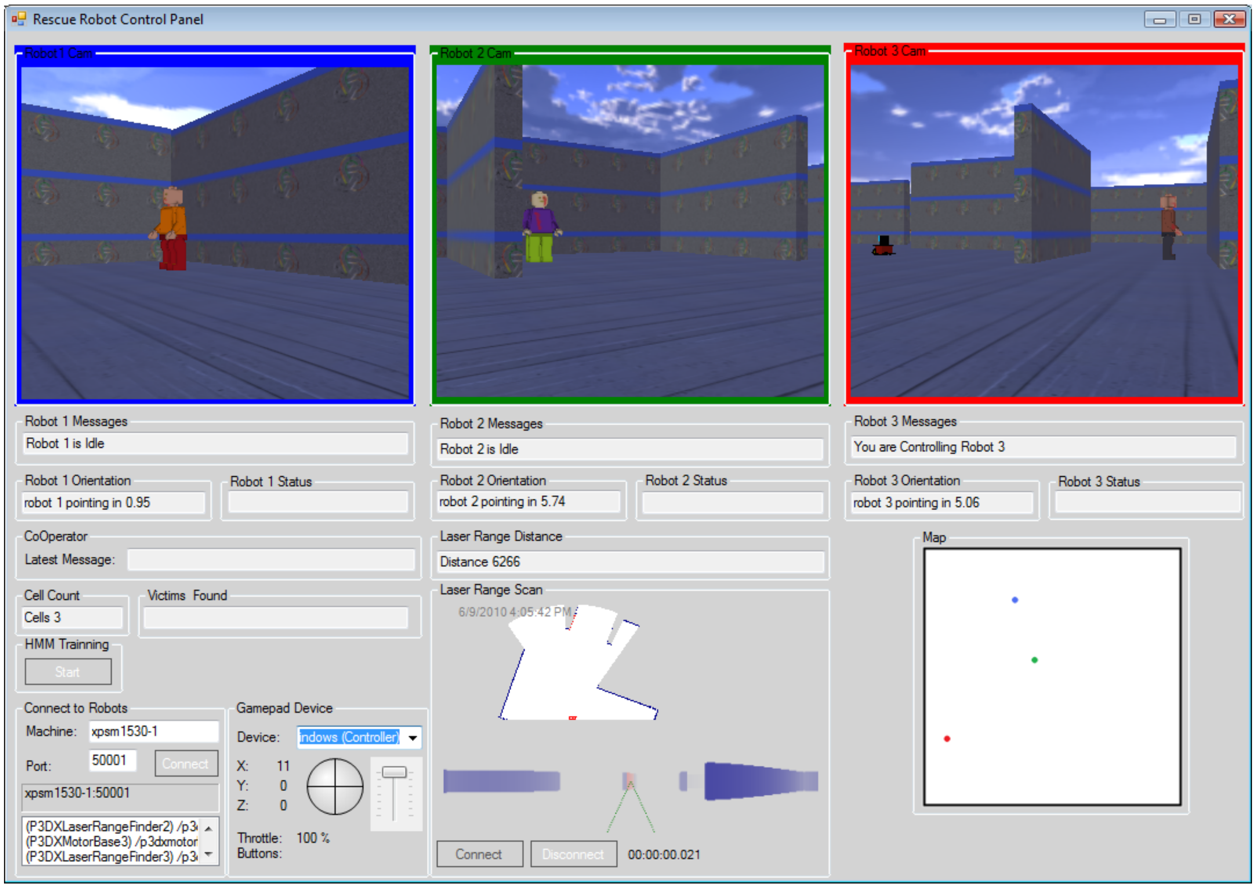
4.3 Learning Models of Operator Distraction for Human-Robot Interfaces

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 our 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. Preliminary experiments indicated that most users preferred the gamepad interface to a keyboard and mouse. 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.

CoOperator augments this basic teleoperation interface in severa 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



Robot 1 Messages
Robot 1 is Idle

Robot 2 Messages
Robot 2 is Idle

Robot 3 Messages
You are Controlling Robot 3

Robot 1 Orientation
robot 1 pointing in 0.95

Robot 1 Status

Robot 2 Orientation
robot 2 pointing in 5.74

Robot 2 Status

Robot 3 Orientation
robot 3 pointing in 5.06

Robot 3 Status

CoOperator
Latest Message:

Laser Range Distance
Distance 6266

Map

Cell Count
Cells 3

Victims Found

HMM Trainning

Laser Range Scan
6/9/2010 4:05:42 PM

Connect to Robots
Machine: xpsm1530-1
Port: 50001
xpsm1530-1:50001
(P3DXLaserRangeFinder2) /p3
(P3DXMotorBase3) /p3dxmotor
(P3DXLaserRangeFinder3) /p3

Gamepad Device
Device: indows (Controller)
X: 11
Y: 0
Z: 0
Throttle: 100 %
Buttons:

00:00:00.021

Figure 14: 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.

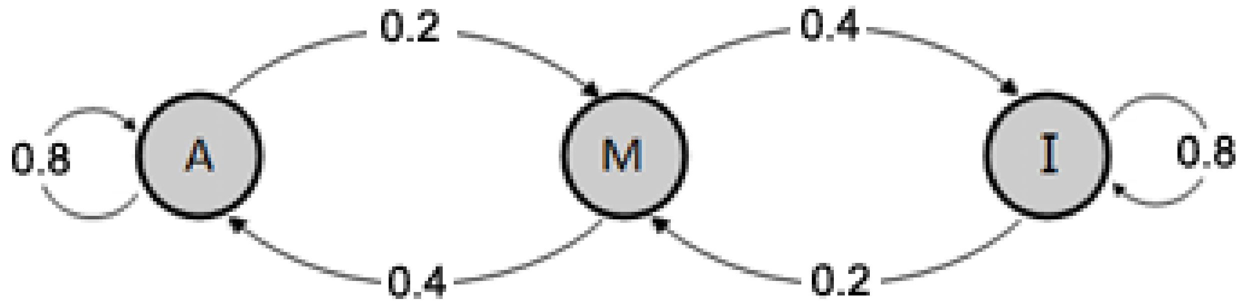


Figure 15: HMM states and transition probabilities.

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) [33] 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 [12]; 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 15.

We quantize the observable robot state into three discrete symbols that characterize the instan-

Table 2: 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

taneous activity of the robot into three levels: low, medium and high, with emission probabilities summarized in Table 2. 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 hidden states to explain the observation sequence and determines that a robot is neglected if its current hidden state is estimated to be **I**nactive.

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. As discussed in Section **??**, even 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 16) 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

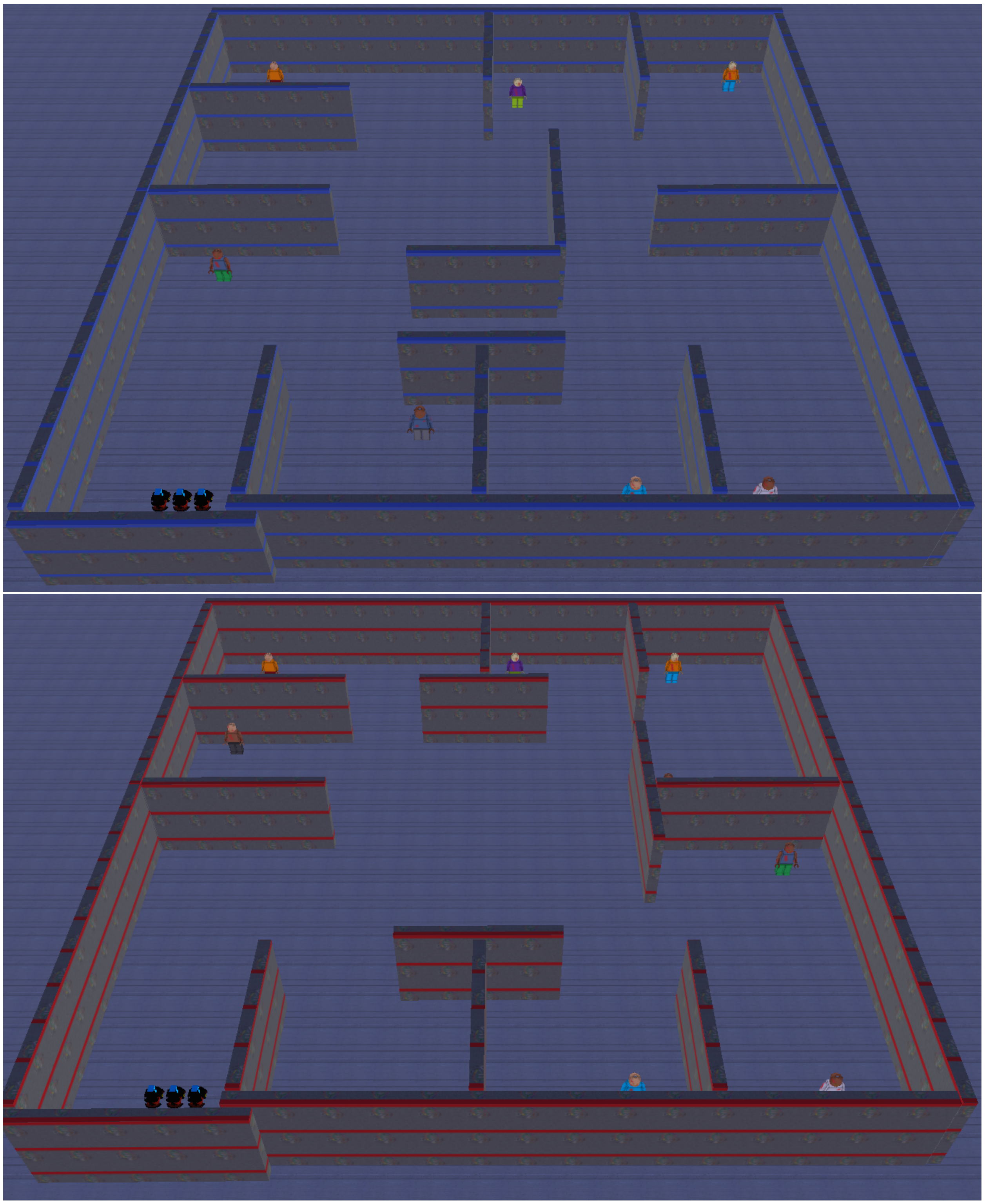


Figure 16: Overhead view of two scenarios in RSARsim. A team of three Pioneer 3DX robots is visible in the bottom left of each view and several victims that need to be rescued are scattered through the various rooms. Operators do not have access to this view and control the robots through a first-person perspective.

Table 3: 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

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

Figure 17 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 significantly 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 19 and 20 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 21). Here we see that the number of map cells explored by the robot team in each of the different scenarios

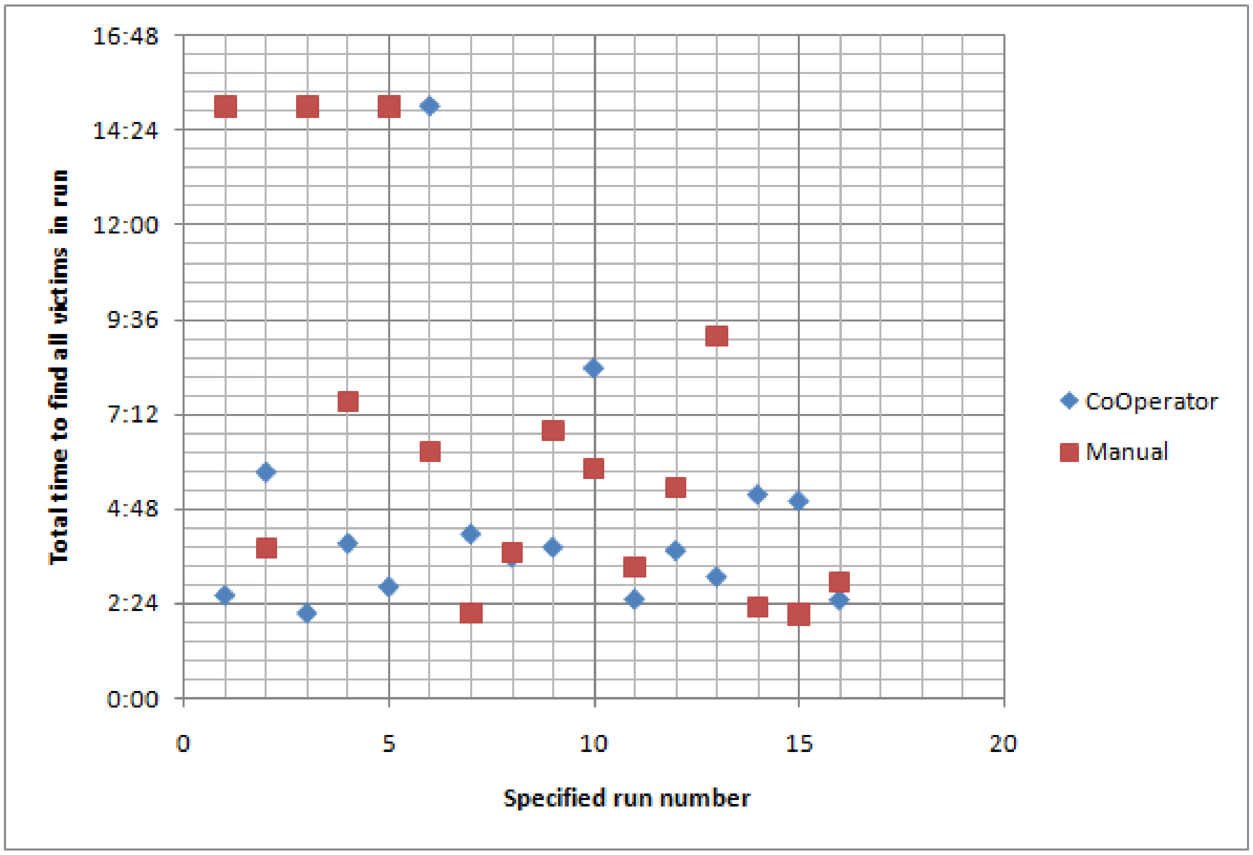


Figure 17: Scatter plot showing the time taken to find all of the victims in each scenario under the two experimental conditions: manual teleoperation (denoted as *manual*) and with the agent-assisted system (denoted as *CoOperator*). In the majority of scenarios, CoOperator significantly reduced the time needed to find the victims.

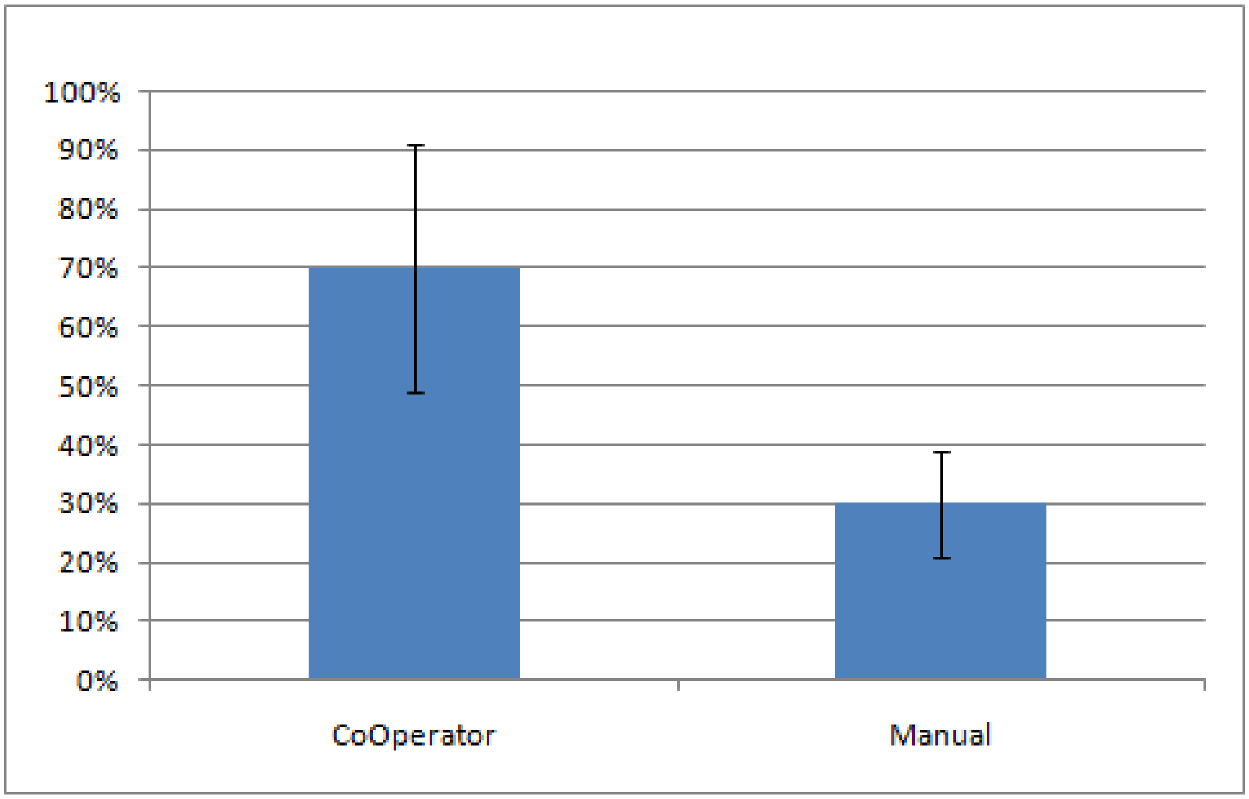


Figure 18: Fraction of users for whom each experimental condition led to a faster rescue (aggregated over all runs). Clearly, for a majority of users, the agent-assisted interface leads to faster rescues.

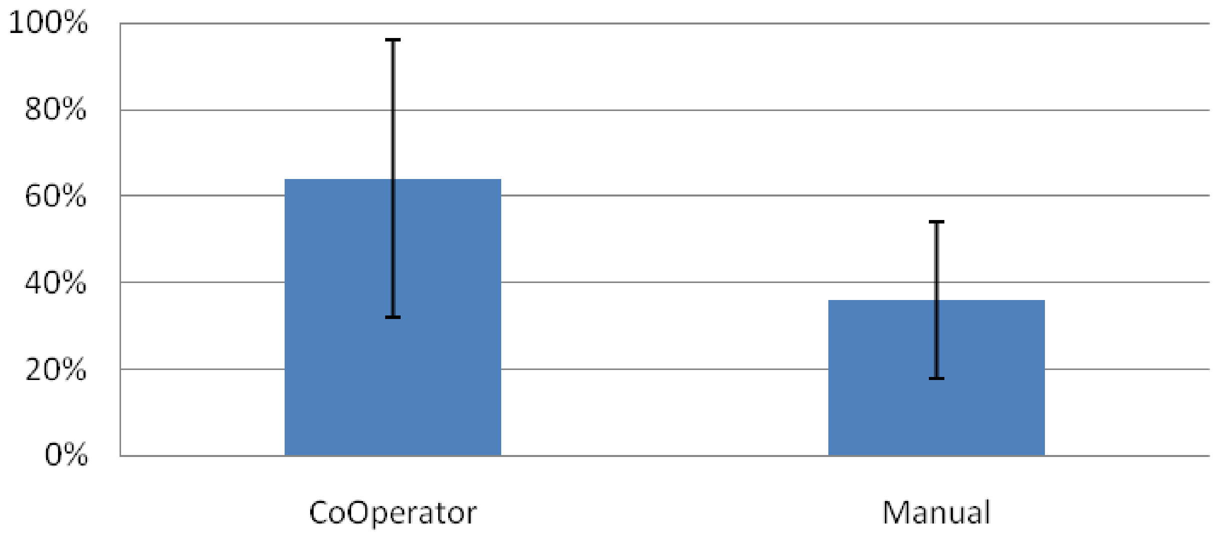


Figure 19: Fraction of users for whom each experimental condition led to a faster rescue (initial experience with system).

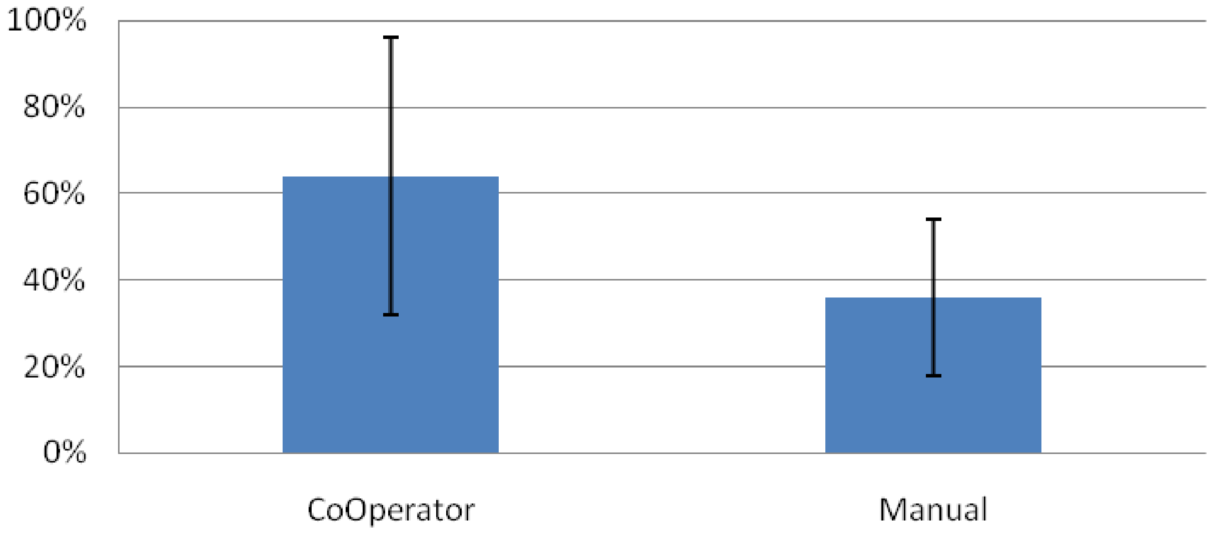


Figure 20: Fraction of users for whom each experimental condition led to a faster rescue (subsequent experience with system).

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 17.

Our final results summarize the user satisfaction reported by participants in our post-questionnaire study (see Figure 22. 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

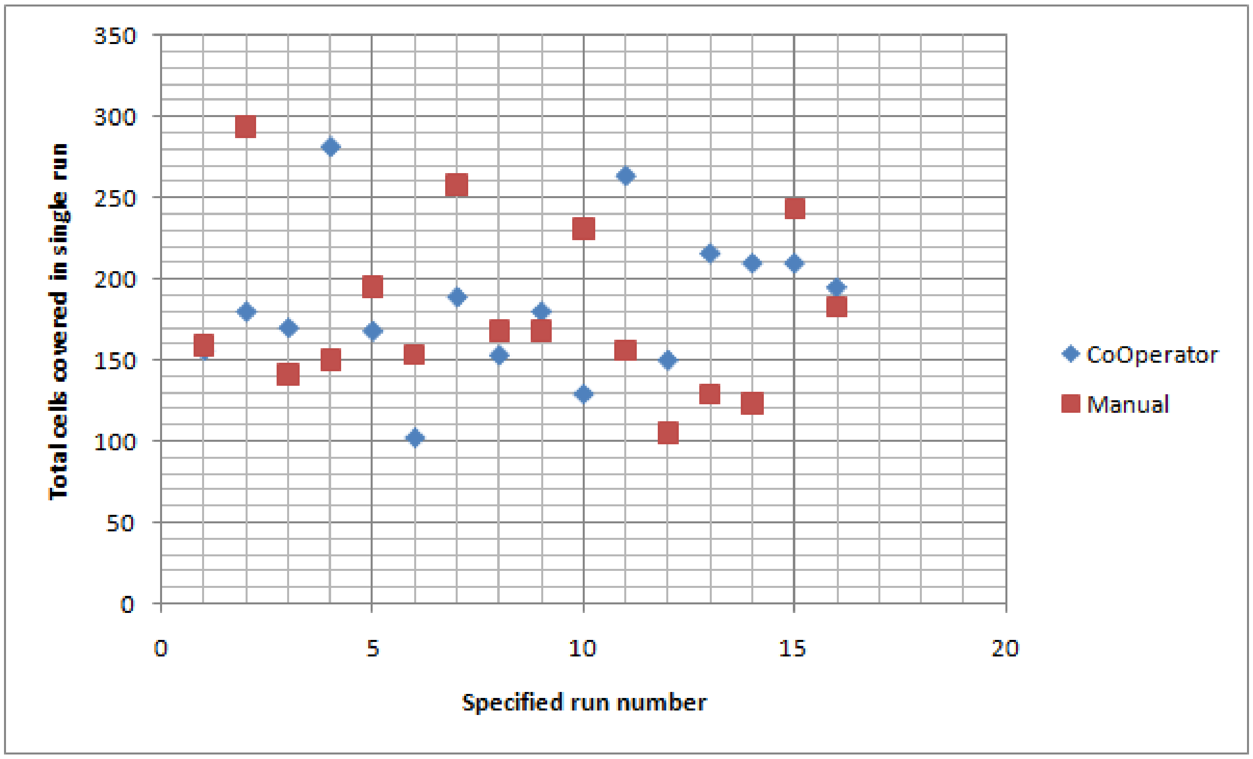


Figure 21: Scatter plot showing the area coverage by the robot team during a single run, under the two experimental conditions. We note that teleoperation with CoOperator does not necessarily cover a greater portion of the disaster area, even though it significantly reduces the time needed to find all of the victims.

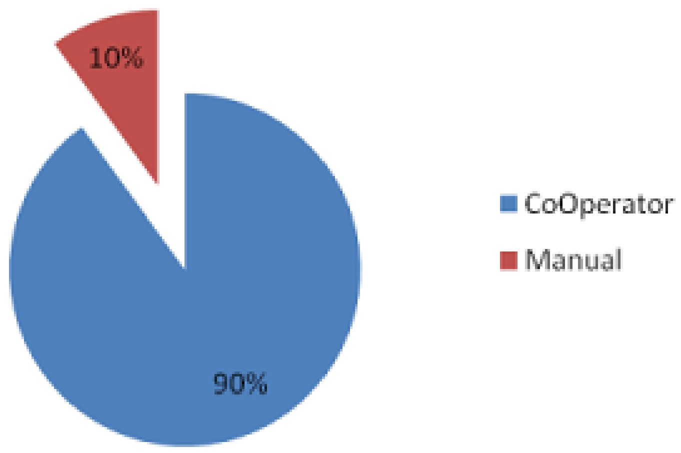


Figure 22: An overwhelming fraction of participants expressed a clear preference for using the agent-assisted (CoOperator) mode of teleoperation.

composited 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 autonomy 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.

4.4 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 23 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 24), 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 [23].

The basic user interface requires the user to fully teleoperate both robots. In this paper, 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

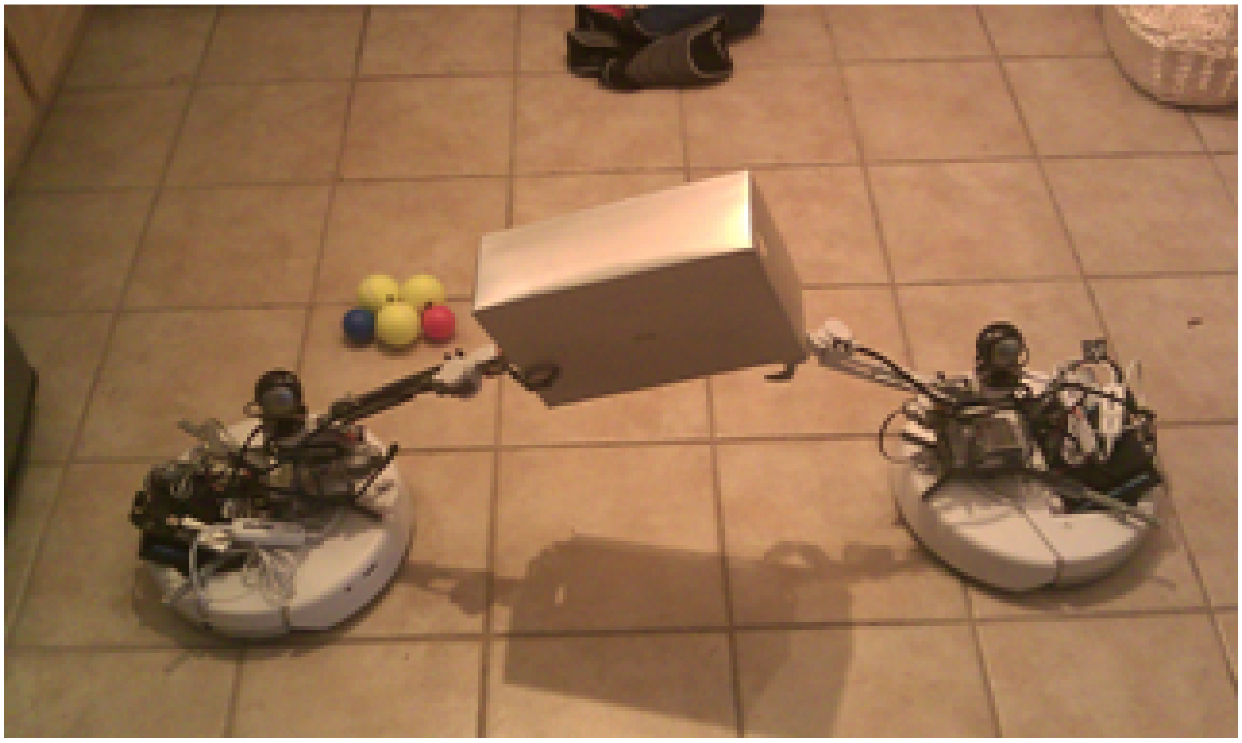


Figure 23: Two HU-IE robots cooperate to lift an object. One robot is teleoperated while the other moves autonomously to mirror the user's intentions. The user can seamlessly switch robots during such maneuvers.

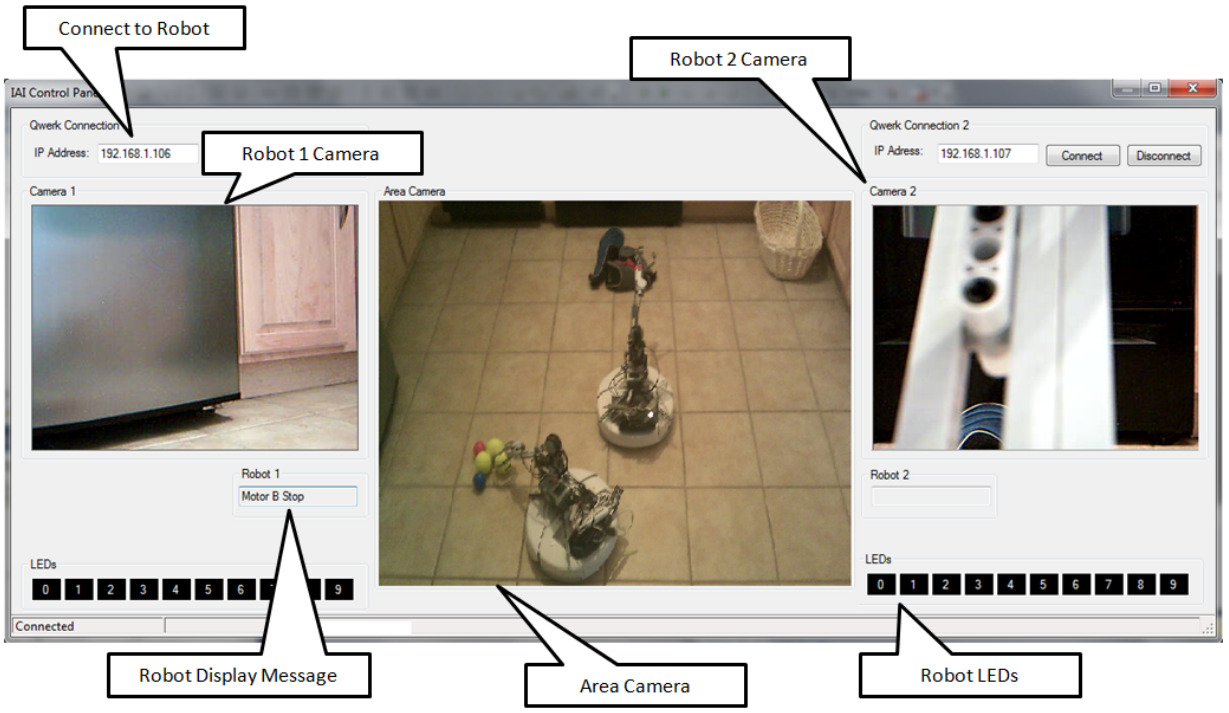


Figure 24: The intelligent agent interface is designed to enable the user to seamlessly switch tele-operation across multiple robots. The IAI supports a cooperative mode where the agent supports the user's active robot by mirroring its intentions.

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.

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; 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.

The operator controls the robots using an Xbox 360 Gamepad controller (Figure **??**) 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

[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.

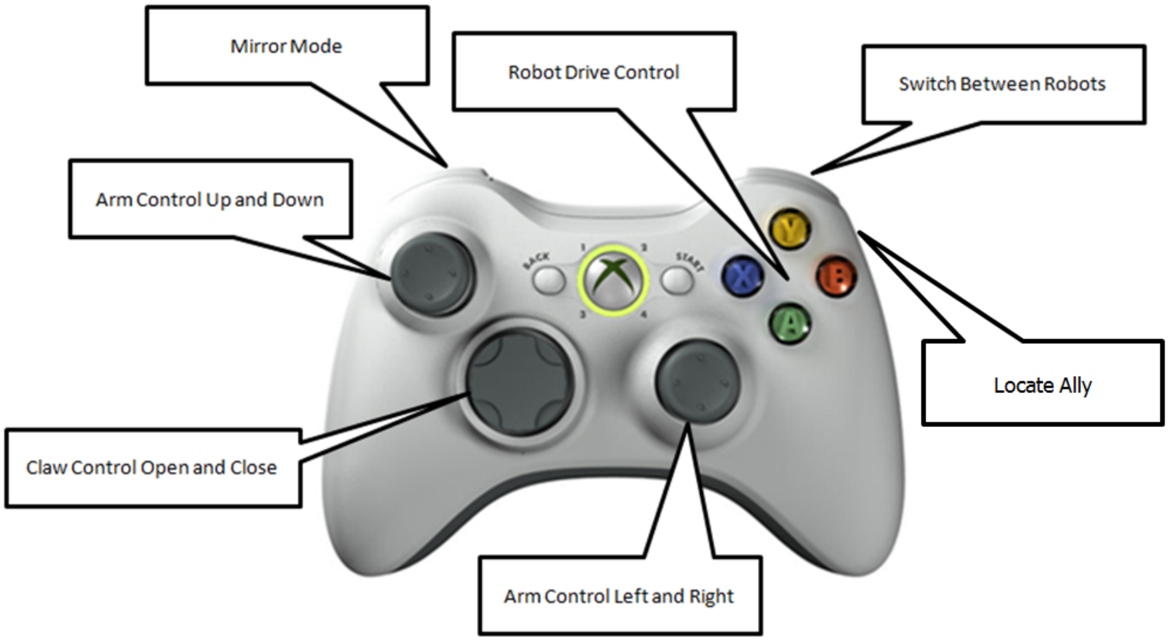


Figure 25: The physical interface to the IAI is through an Xbox 360 gamepad from which the operator can select a robot and send it teleoperation commands.

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

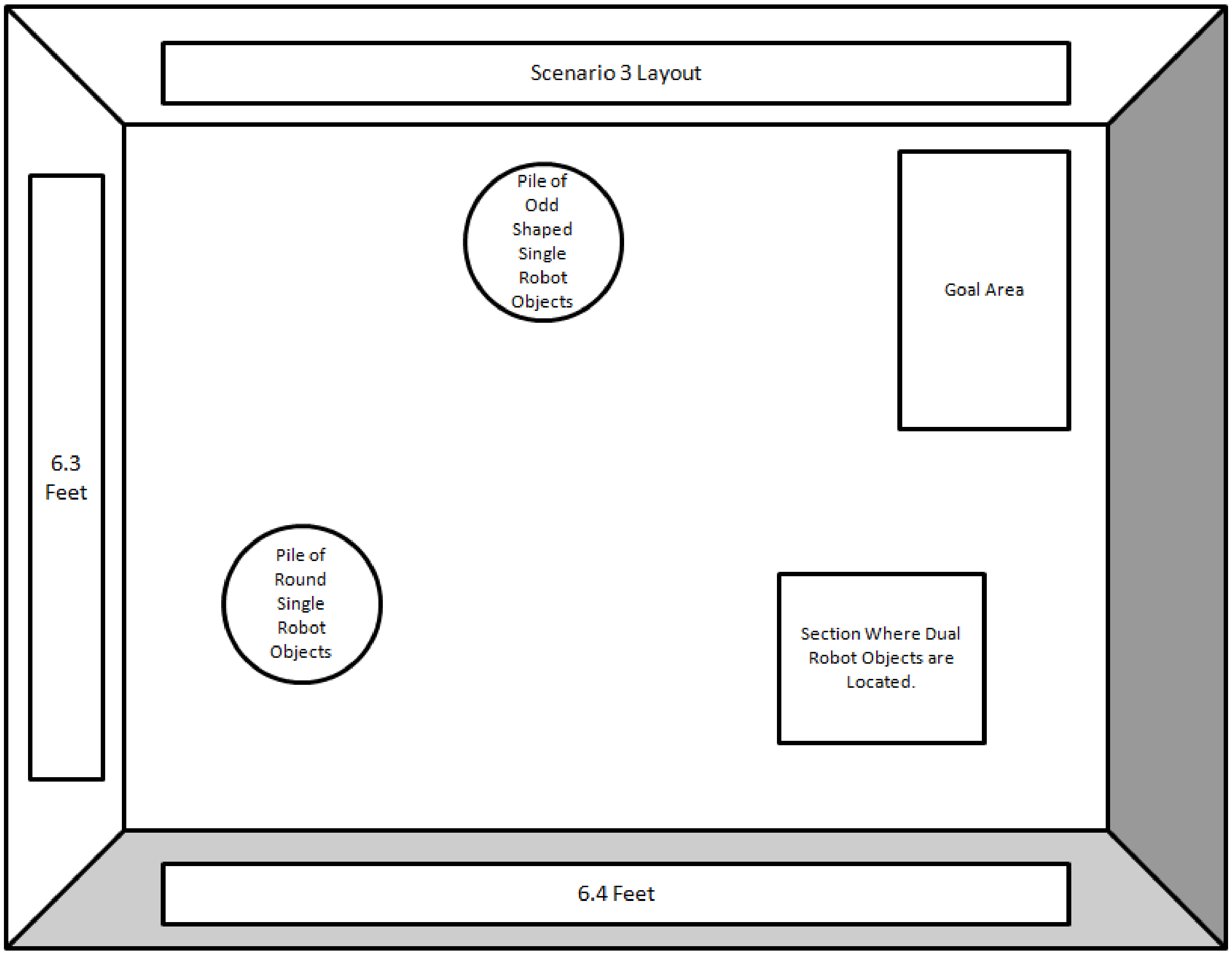

Scenario 3 Layout
Pile of Odd Shaped Single Robot Objects
Goal Area
6.3 Feet
Pile of Round Single Robot Objects
Section Where Dual Robot Objects are Located.
6.4 Feet

Figure 26: 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.

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 26: 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.

Table 4: 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

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 4 summarizes the demographics of our user study. The majority of the users had limited prior experience with game controller interfaces. Figure 27 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.

Table 5 presents an analysis of the completion time results. 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 ($p < 0.05$).

Figure 28 shows the number of failed pickup attempts by the user in each scenario, both under IAI and manual conditions. Table 6 analyzes these results to test our belief that IAI should result

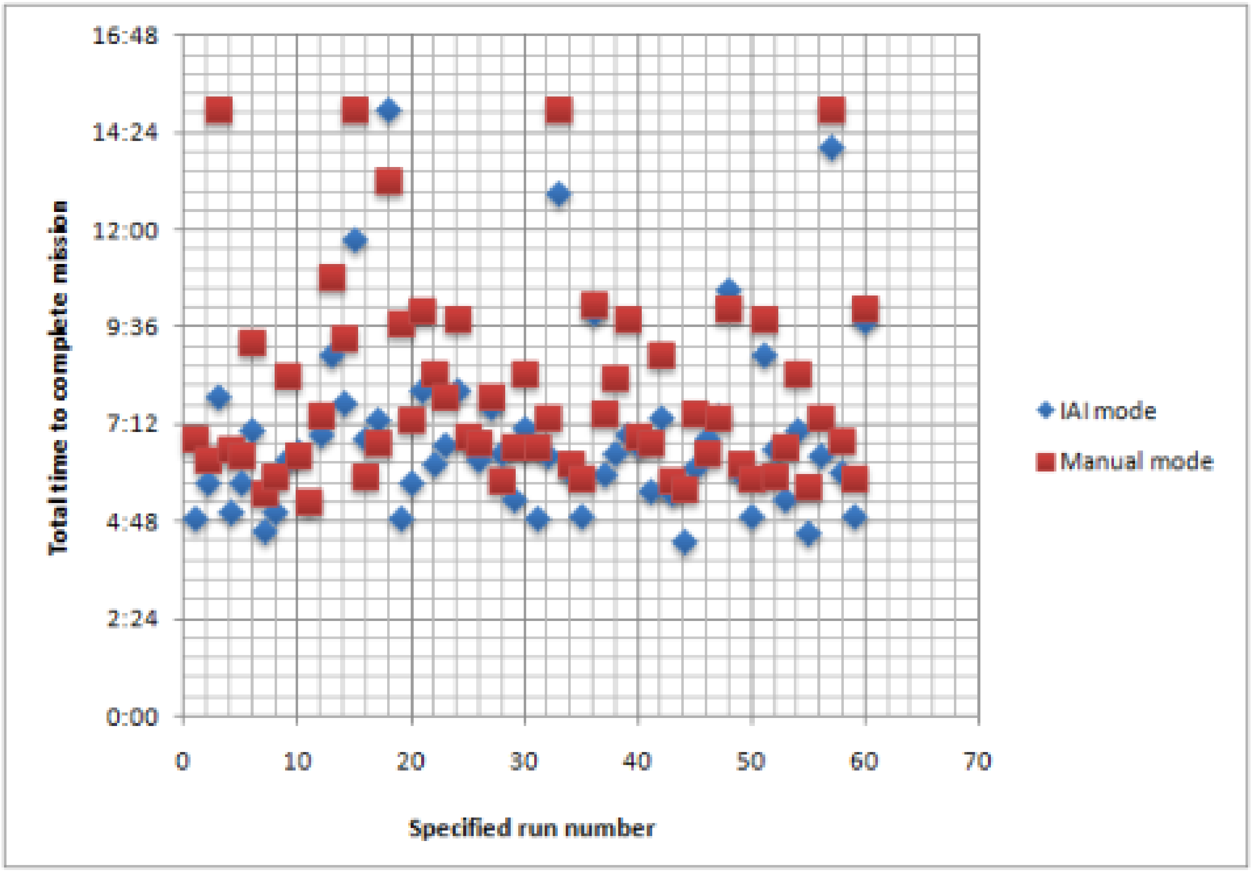


Figure 27: Scatterplot showing the time required the complete the task in each scenario under the two experimental conditions: IAI (adjustably autonomous) and Manual (pure teleoperation). We observe that in the majority of scenarios, IAI reduces the time required to complete the task.

Table 5: Analysis of average task completion time. IAI results in a significant improvement over the manual mode in all three scenarios.

Scenario	IAI	Manual	Significance
	Time $\pm\sigma$ (sec)	Time $\pm\sigma$ (sec)	$(p < 0.05)$
1	361.1 ± 63.2	411.8 ± 81.9	0.017
2	353.6 ± 56.0	407.2 ± 59.4	0.003
3	538.5 ± 157.3	627.8 ± 158.6	0.004

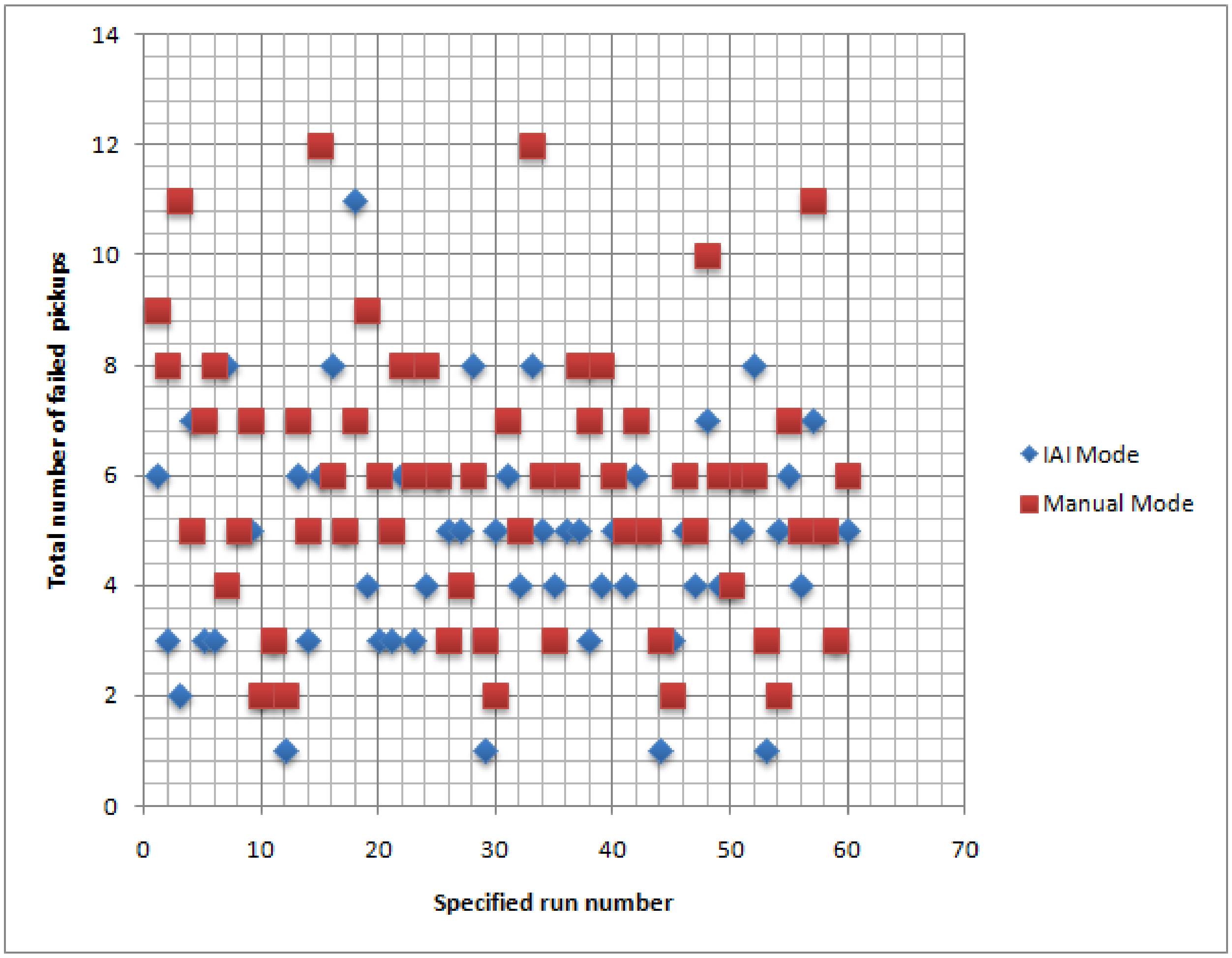


Figure 28: Scatter plot showing the number of failed pickup attempts for each user in each scenario.

in fewer dropped objects. We see that the number of average drops is lower with IAI in all three scenarios. However, the improvement in Scenario 1 is not shown to be statistically significant on the Student's t-test ($p < 0.05$). 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.

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) au-

Table 6: Average total combined failed pickups (object drops). IAI reduces the number of drops on average; this improvement is statistically significant in Scenarios 2 and 3.

Scenario	IAI	Manual	Significance
	Failed Pickups $\pm\sigma$	Failed Pickups $\pm\sigma$	($p < 0.05$)
1	5.75 ± 1.55	6.20 ± 1.64	NO
2	3.30 ± 1.21	4.70 ± 1.55	0.002
3	5.00 ± 2.22	6.80 ± 3.31	0.026

tonomously 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 29).

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 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 paper, we demonstrate and evaluate an effective human-robot interface paradigm for multi-robot manipulation tasks. Rather than increasing the workload of 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;

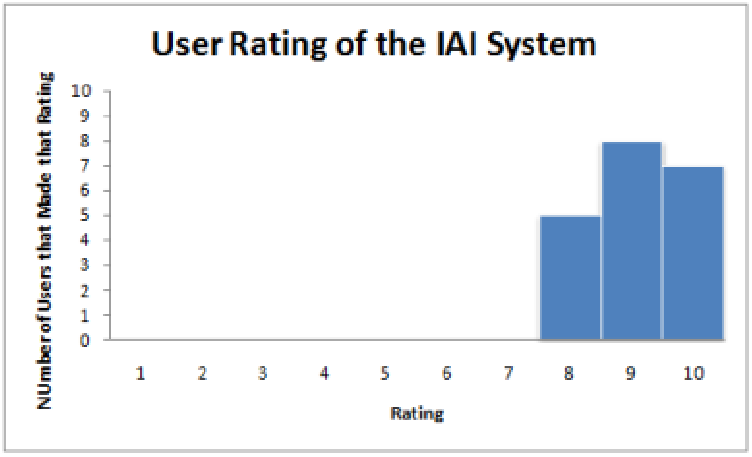


Figure 29: Histogram of user ratings of the IAI user interface on post-task questionnaires. Users were overwhelmingly positive.

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.

4.5 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 exibility 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.

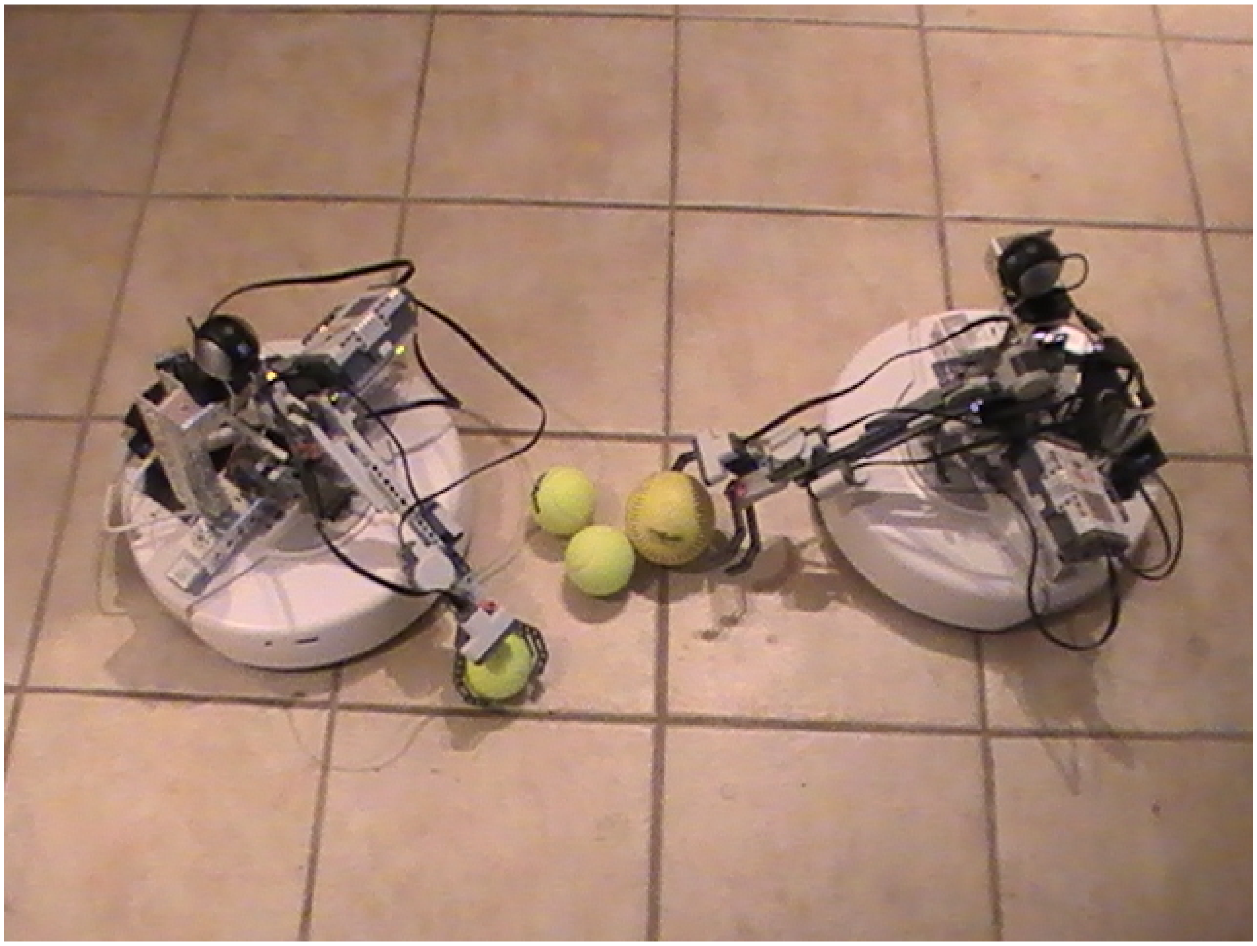


Figure 30: Two HU-IE robots cooperating together to clear the environment of objects and deposit them in the goal location.

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.

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.

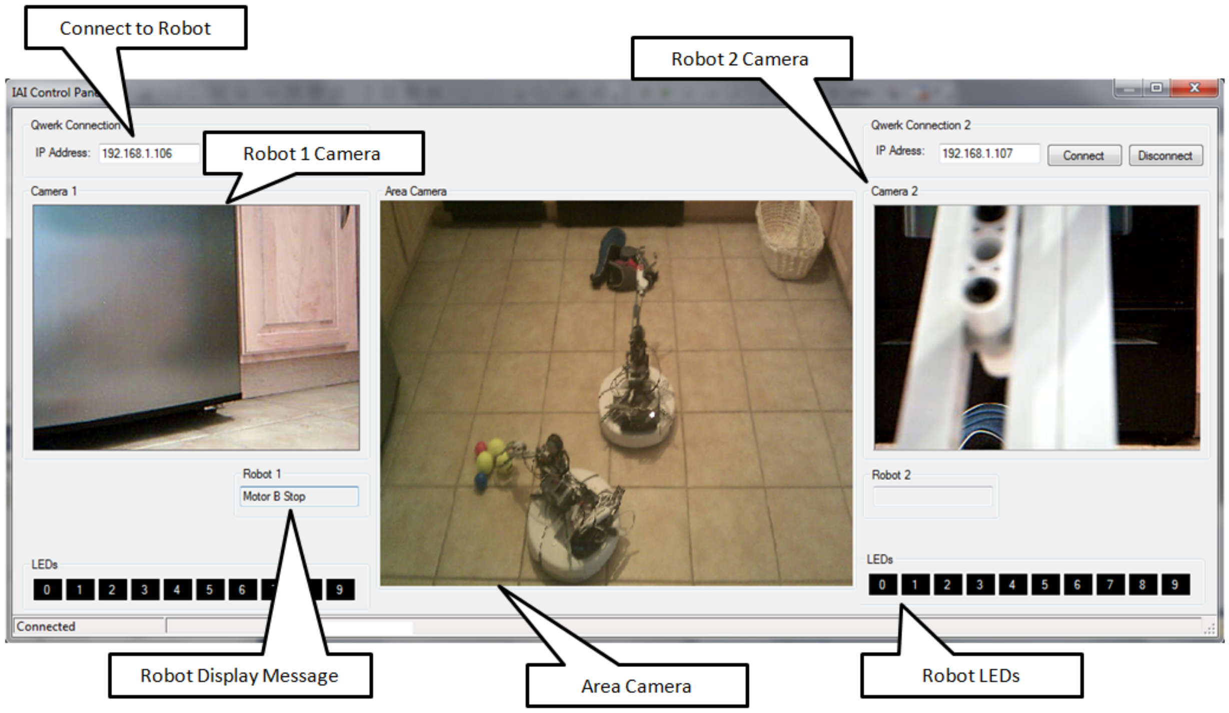


Figure 31: The user interface is designed to enable the user to seamlessly switch teleoperation between multiple robots. The IAI supports a cooperative mode where the agent supports the user's active robot by mirroring its planned actions. The user views the environment through an overhead camera and the robots' webcams.

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 32) 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

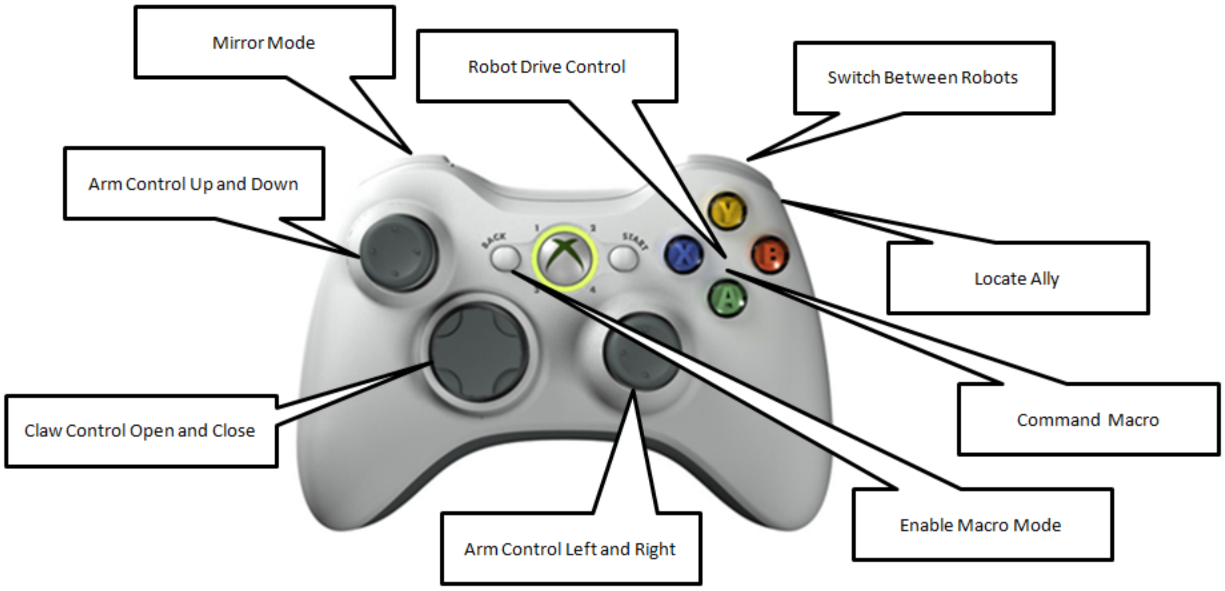


Figure 32: The physical interface to the IAI is through a Xbox 360 gamepad from which the operator can select a robot, send it explicit teleoperation commands, utilize built-in autonomous functions, and create macros.

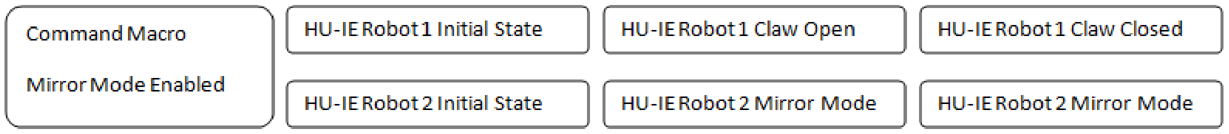

Command Macro
Mirror Mode Enabled
HU-IE Robot 1 Initial State
HU-IE Robot 1 Claw Open
HU-IE Robot 1 Claw Closed
HU-IE Robot 2 Initial State
HU-IE Robot 2 Mirror Mode
HU-IE Robot 2 Mirror Mode

Figure 33: 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.

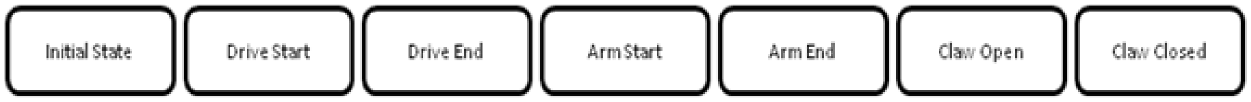


Figure 34: State representation of a recorded macro

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 33).

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 34). 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 35). The terminal position of the robot is retained in both absolute coordinates and also the relative position to the nearest object or robot.

After the macro acquisition phase, there is an acceptance phase during which the operator is

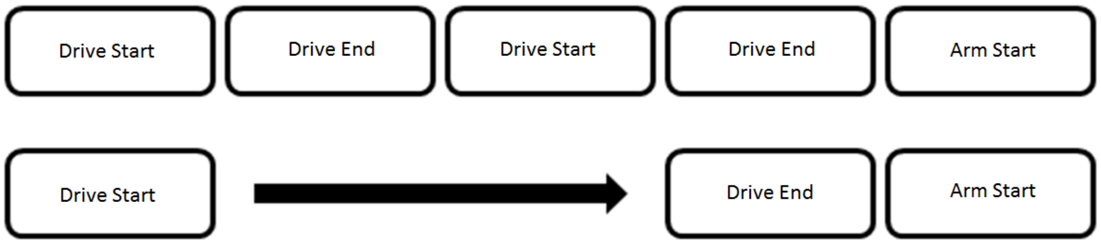


Figure 35: If the demonstration contains multiple short segments, the abstract task representation is created through merging superfluous segments.

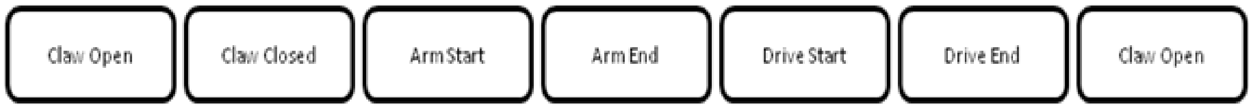


Figure 36: Example abstract task representation for driving to the goal

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.

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 36).

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 37).

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 38). 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 7 summarizes the demographics of the user group.

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

Table 7: 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

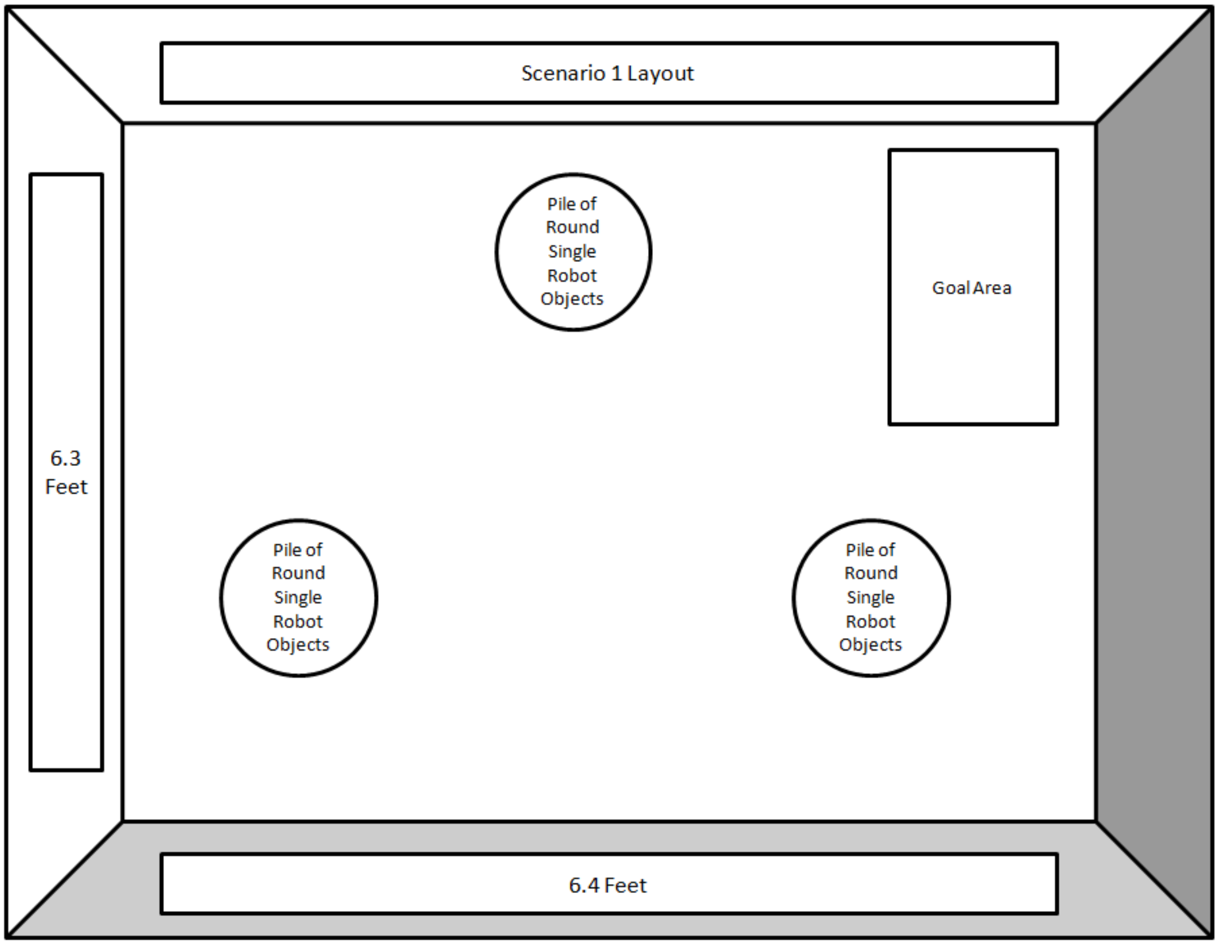

Scenario 1 Layout
Pile of Round Single Robot Objects
Goal Area
6.3 Feet
Pile of Round Single Robot Objects
Pile of Round Single Robot Objects
6.4 Feet

Figure 37: 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.

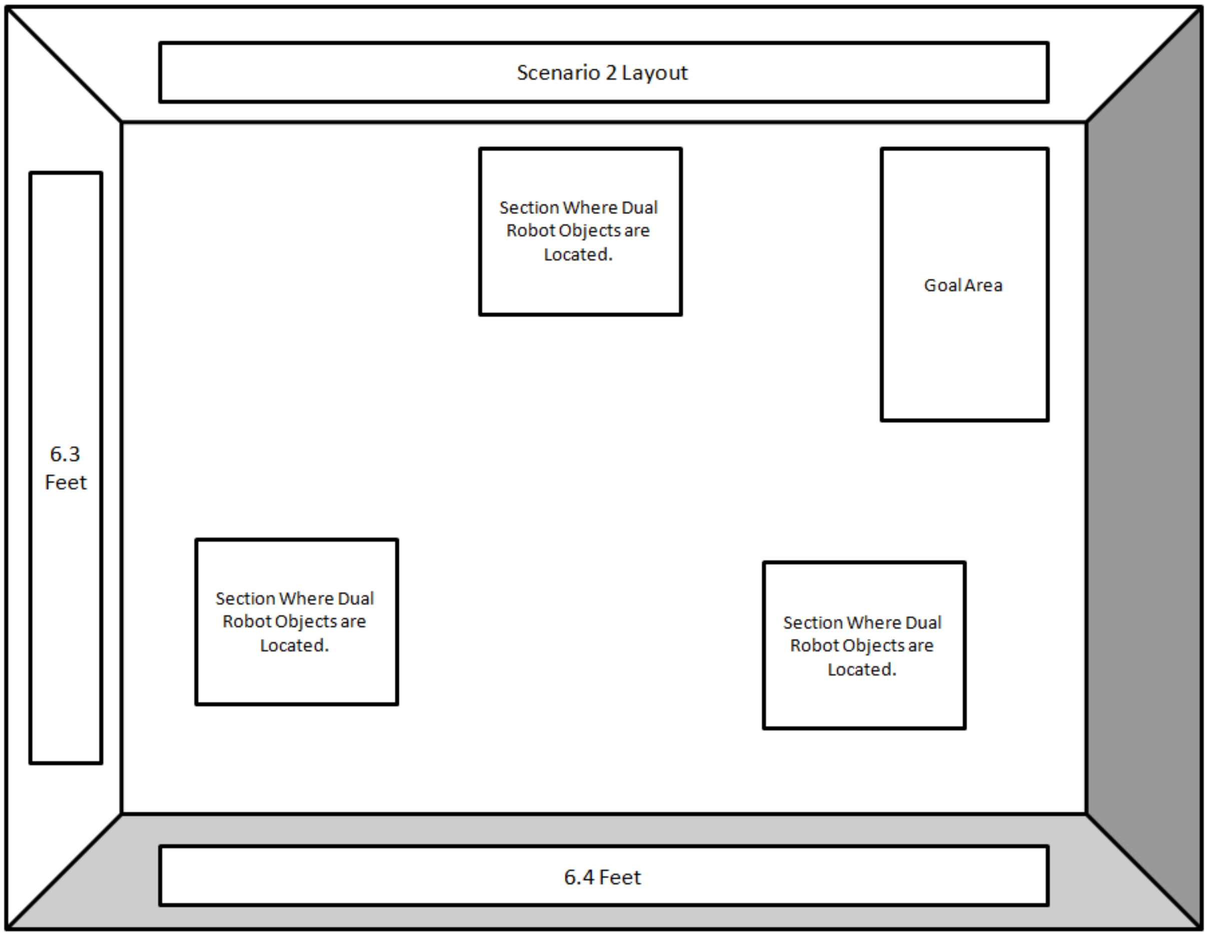

Scenario 2 Layout
Section Where Dual Robot Objects are Located.
Goal Area
6.3 Feet
Section Where Dual Robot Objects are Located.
Section Where Dual Robot Objects are Located.
6.4 Feet

Figure 38: 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.

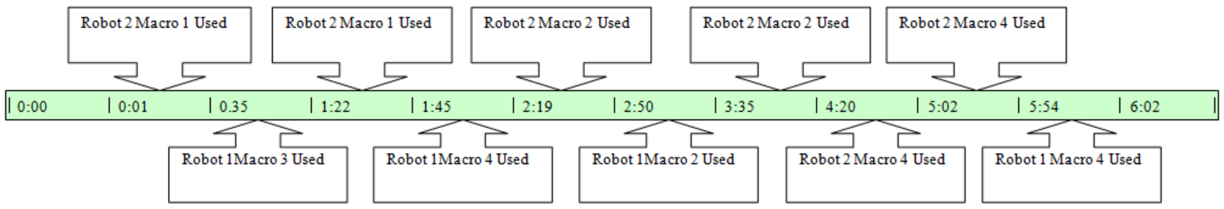


Figure 39: Timeline showing macro usage by an expert user in Scenario 1. Ten macros were used in total during the fifteen minute period. The set of macros included: 1) drive to pile, lift object, and deliver to goal 2) lift object and deliver to goal 3) lift object 4) deliver to goal.

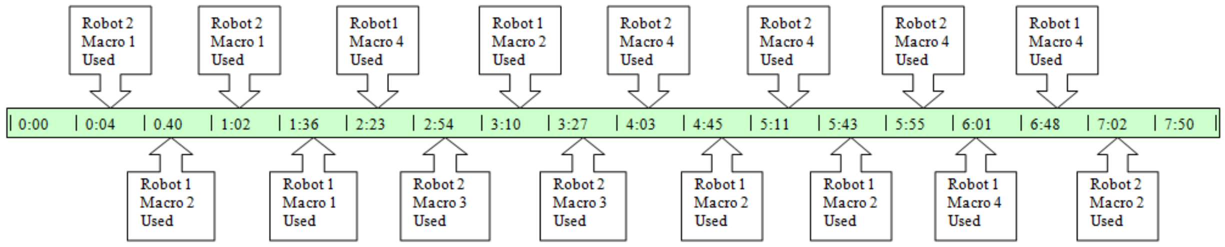


Figure 40: Timeline showing macro usage by a novice user in Scenario 1. Sixteen macros were used in total during the fifteen minute period. The set of macros included: 1) drive to pile and lift object 2) lift object and deliver to goal 3) lift object 4) deliver to goal.

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 [25].

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 39 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 40) 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.

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.

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 41).

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 42. 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 44). 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 **??**).

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.

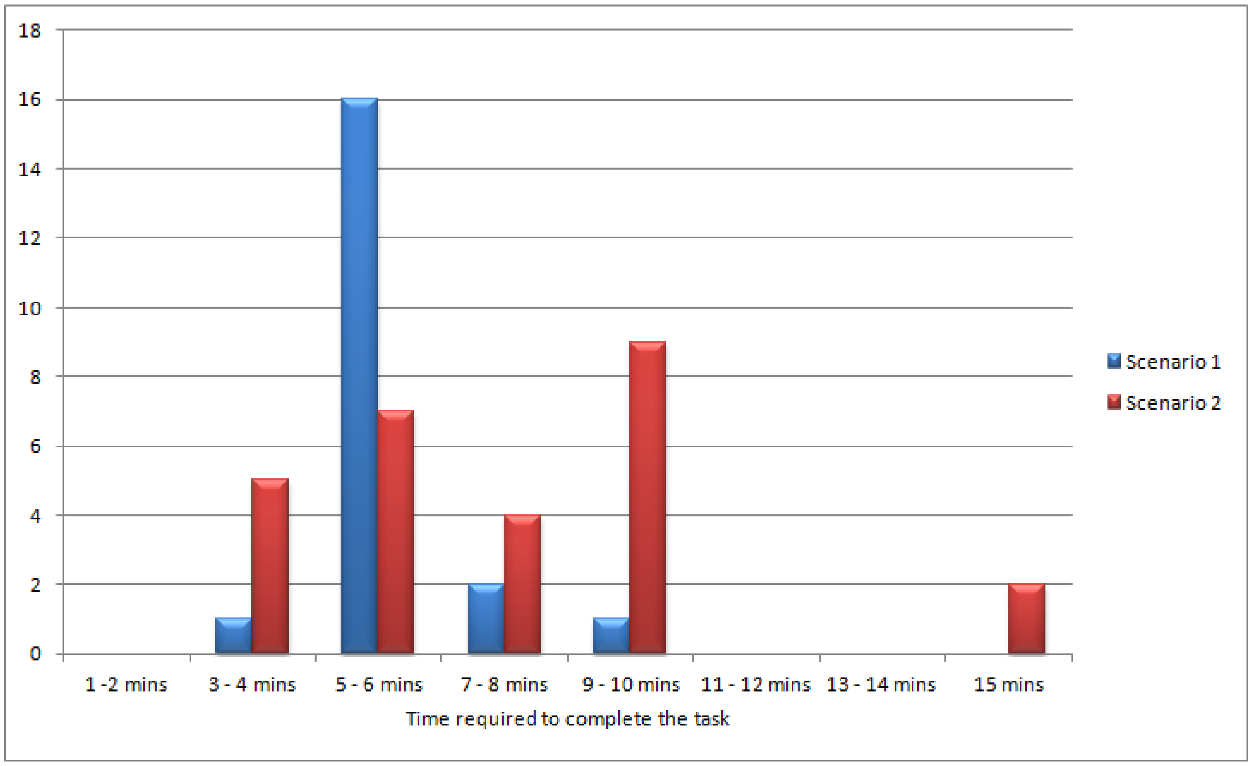


Figure 41: Histogram showing the time required the complete Scenario 1 and 2. Most users were able to complete the Scenario 1 task in one third of the allotted time. Note that there is more variance in the time required to complete the coordinated manipulation task (Scenario 2), and two users were not able to complete it in the allotted time.

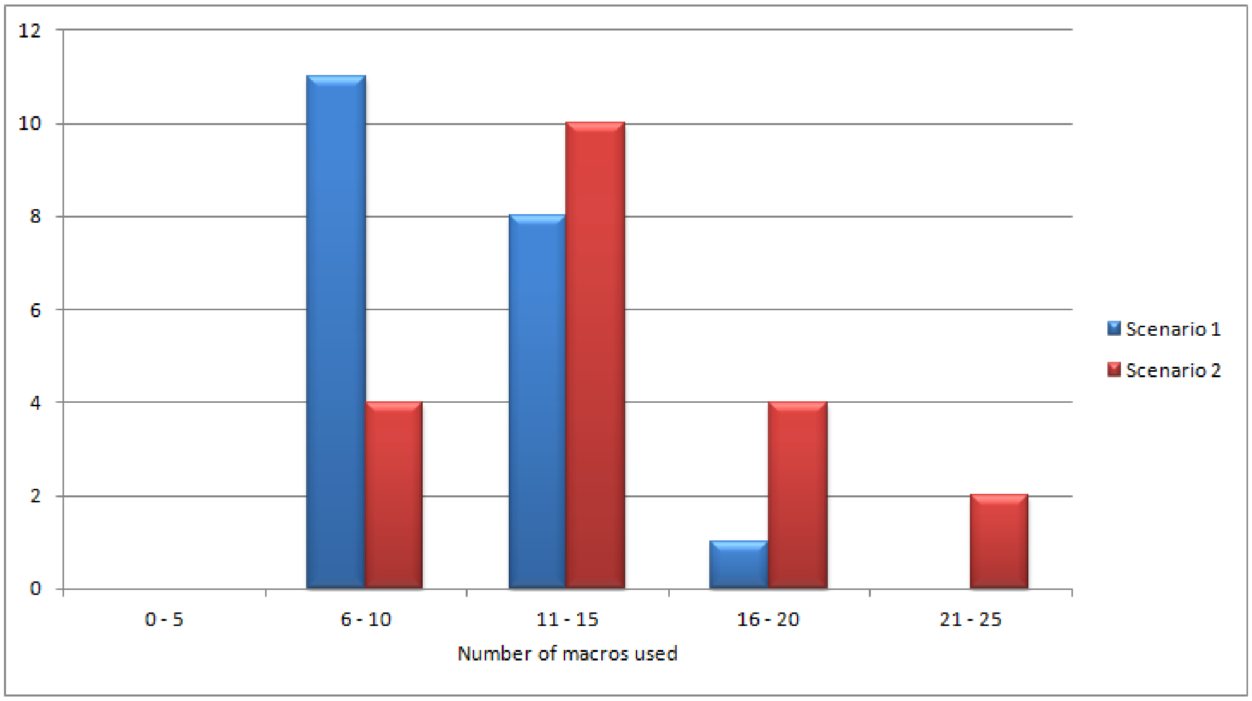


Figure 42: Histogram showing the macro usage by scenario. Bimanual manipulation (Scenario 2) was more macro intensive.

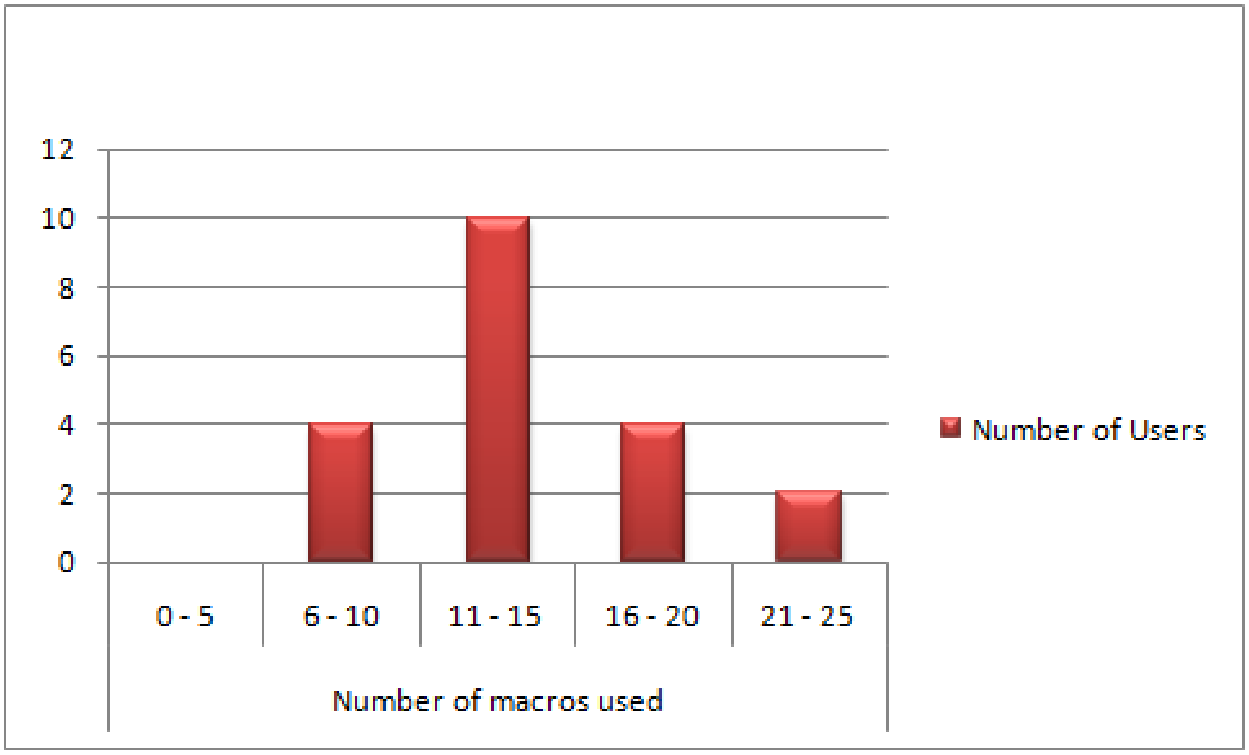


Figure 43: Histogram showing the macro usage in Scenario 2 (bimanual manipulation)

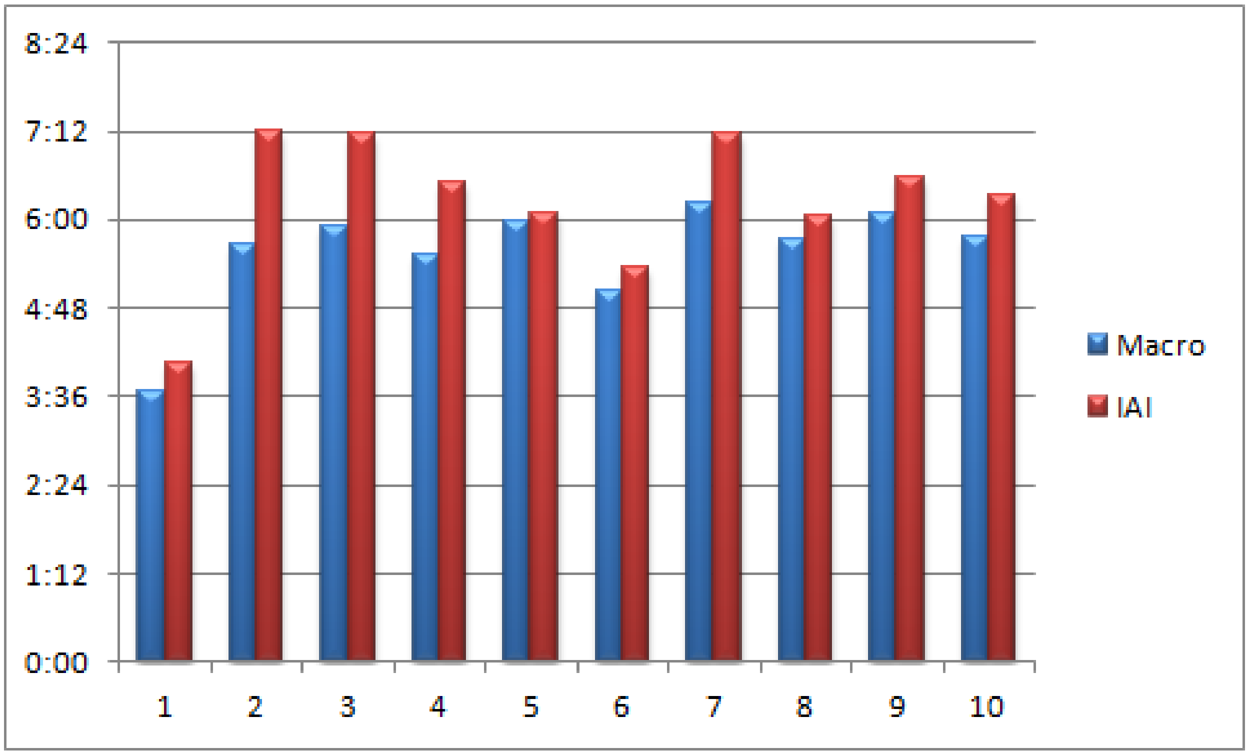


Figure 44: Post hoc analysis comparing the configurable and non-configurable user interface. The y axis shows time required to complete the scenario and the x axis the subject number. The configurable user interface appears to confer a slight time advantage.

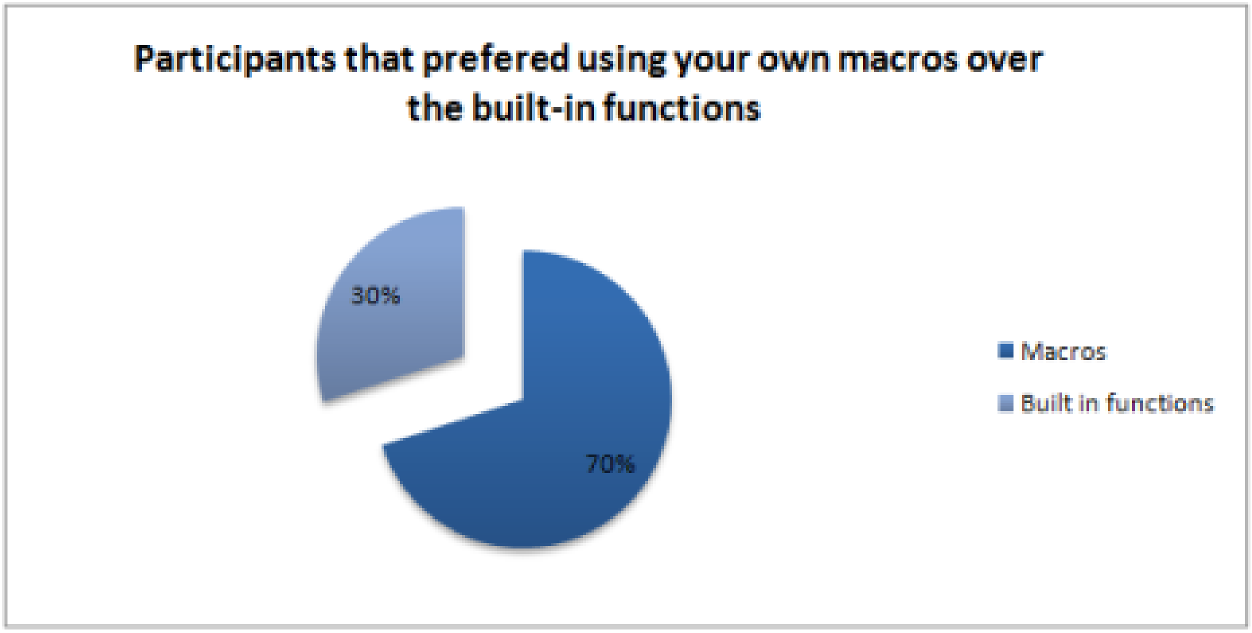


Figure 45: 70% of participants expressed a clear preference for agent-assisted mode of teleoperation; the remaining 30% expressed no preference between the two modes.

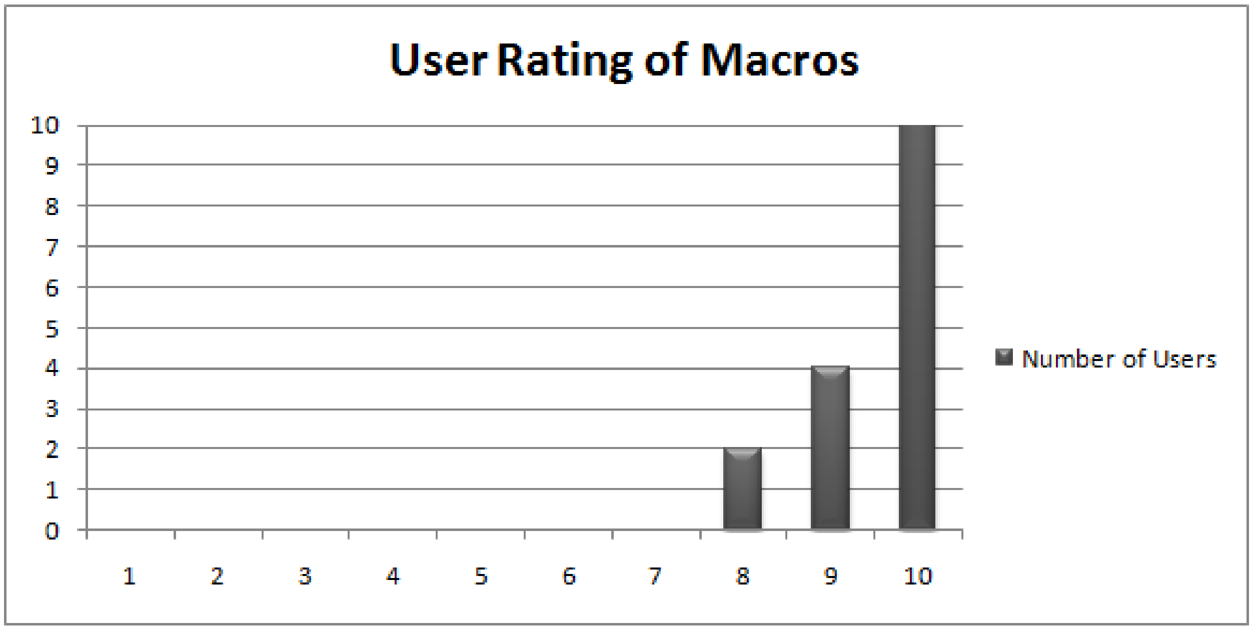


Figure 46: Histogram of user ratings of the configurable user interface on post-task questionnaires

4.6 Adapting to Expert-Novice Differences in Human-Robot Interaction

We describe an 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 a 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 47 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 [11], home service robot teams [24], urban search and rescue [7], 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 48). 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.

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

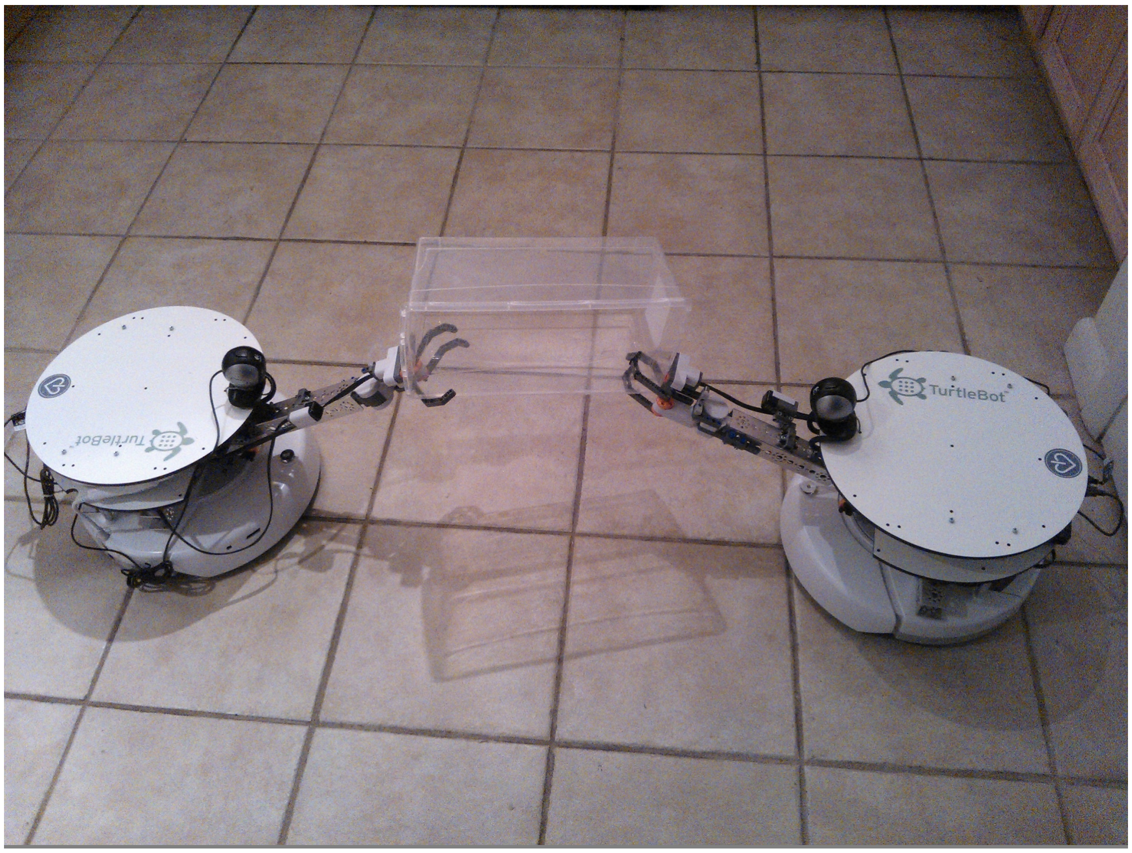


Figure 47: Two robots cooperate to lift an object under the direction of the human operator. In the multi-robot manipulation task, the robots must lift and deliver a series of objects of different sizes to the goal location.

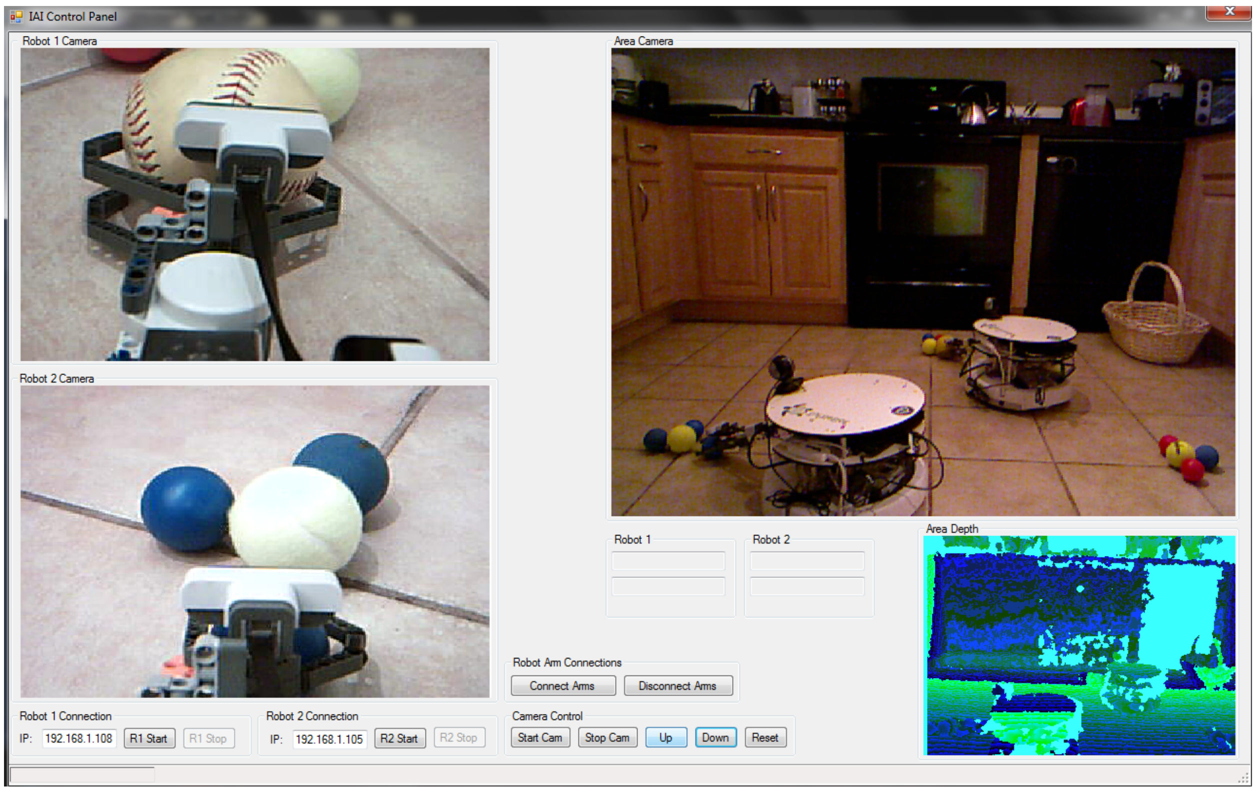


Figure 48: The user interface simultaneously provides the operator with an overhead view of the scene through a separately mounted camera (top right), a depth map of the scene from the Kinect (bottom right), and the webcam perspective from the two robotic arms (left).

the Xbox 360 controller.

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 49). 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.

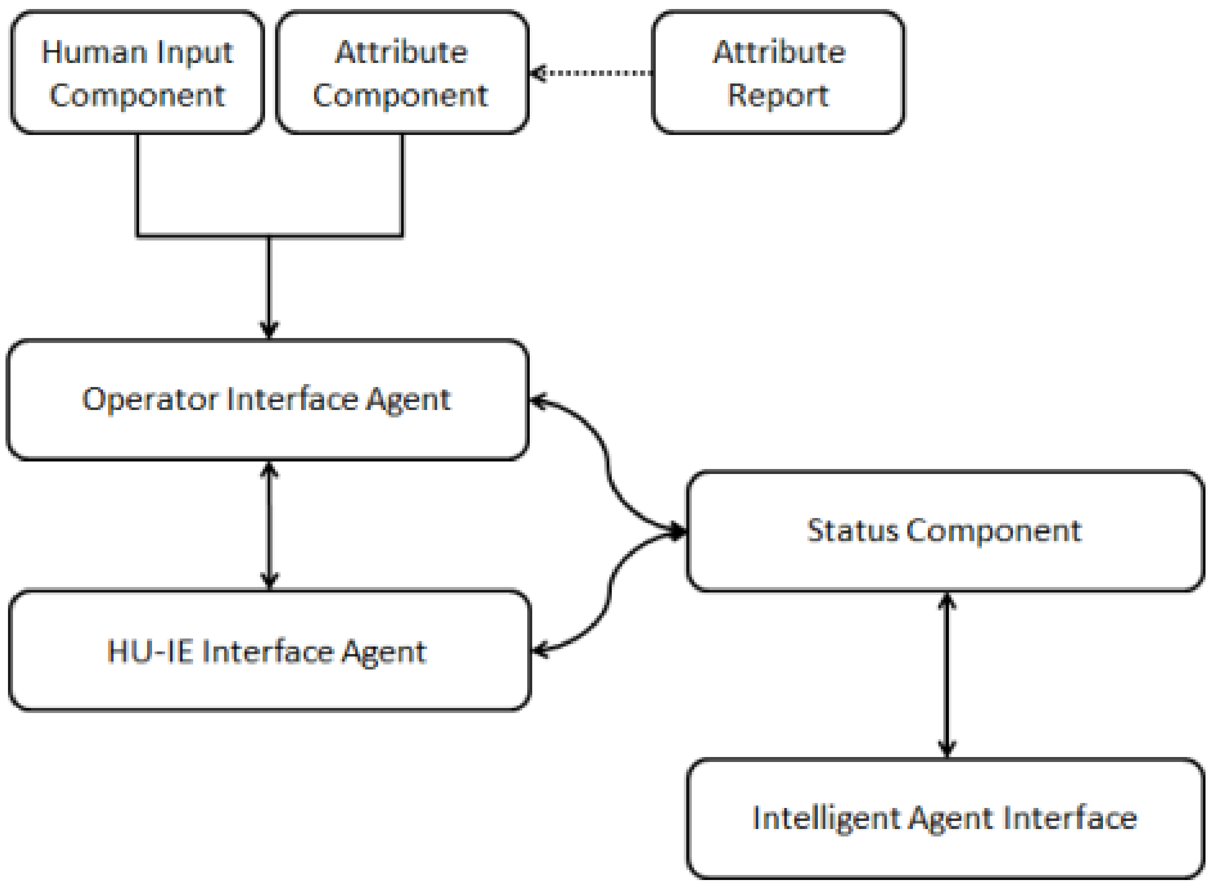


Figure 49: Overview of the Adaptive Interface Component.

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.

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 positions 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 47)); 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 50). 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 50). The participant performed this scenario twice in randomized order, once with the adaptive interface and once with the baseline version.

We compare the performance of the adaptive vs. the non-adaptive version of the user interface. Figure 51 presents a comparison of the times required for each participant to complete Scenario

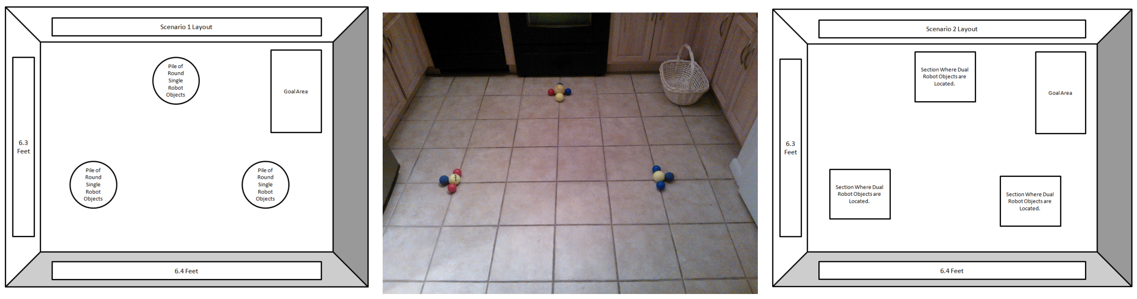

Scenario 1 Layout
Pile of Round Single Robot Objects
Goal Area
6.3 Feet
Pile of Round Single Robot Objects
Pile of Round Single Robot Objects
6.4 Feet
Scenario 2 Layout
Section Where Dual Robot Objects are Located.
Goal Area
6.3 Feet
Section Where Dual Robot Objects are Located.
Section Where Dual Robot Objects are Located.
6.4 Feet

Figure 50: 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.

Table 8: Time differences with and without the adaptive component

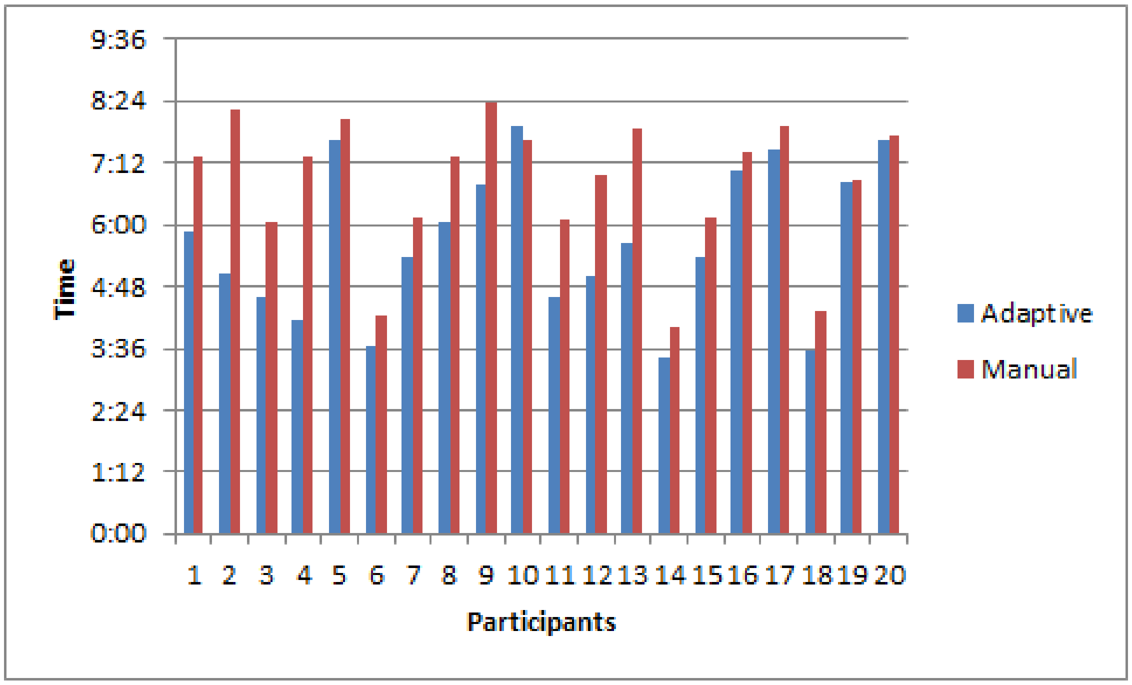
Scenario	Adaptive Time $\pm\sigma$ (sec)	Non-adaptive Time $\pm\sigma$ (sec)	Significance ($p < 0.01$)
1	340.9 $\pm$ 86.8	408.0 $\pm$ 79.8	yes
2	600.7 $\pm$ 117.7	716.4 $\pm$ 150.7	yes

1 (small objects) and Scenario 2 (bimanual manipulation) under both experimental conditions. Table 8 summarizes the completion time results. We confirm that the improvements in completion time is statistically significant under a paired two-tailed t-test at the $p < 0.01$ level for both Scenario 1 and 2.

Figure 52 presents a comparison of the object drops by each participant in Scenario 1 (small objects) and Scenario 2 (bimanual manipulation) under both experimental conditions, the adaptive and non-adaptive user interface. Table 9 summarizes the number of dropped objects in each condition. We confirm that the reductions in dropped objects is statistically significant under a paired two-tailed t-test at the $p < 0.01$ level for both Scenario 1 and 2.

Table 9: # of dropped objects with and without the adaptive component

Scenario	Adaptive Drops $\pm\sigma$	Non-adaptive Drops $\pm\sigma$	Significance ($p < 0.01$)
1	1.10 $\pm$ 1.65	3.25 $\pm$ 1.74	yes
2	3.85 $\pm$ 2.58	7.85 $\pm$ 3.15	yes



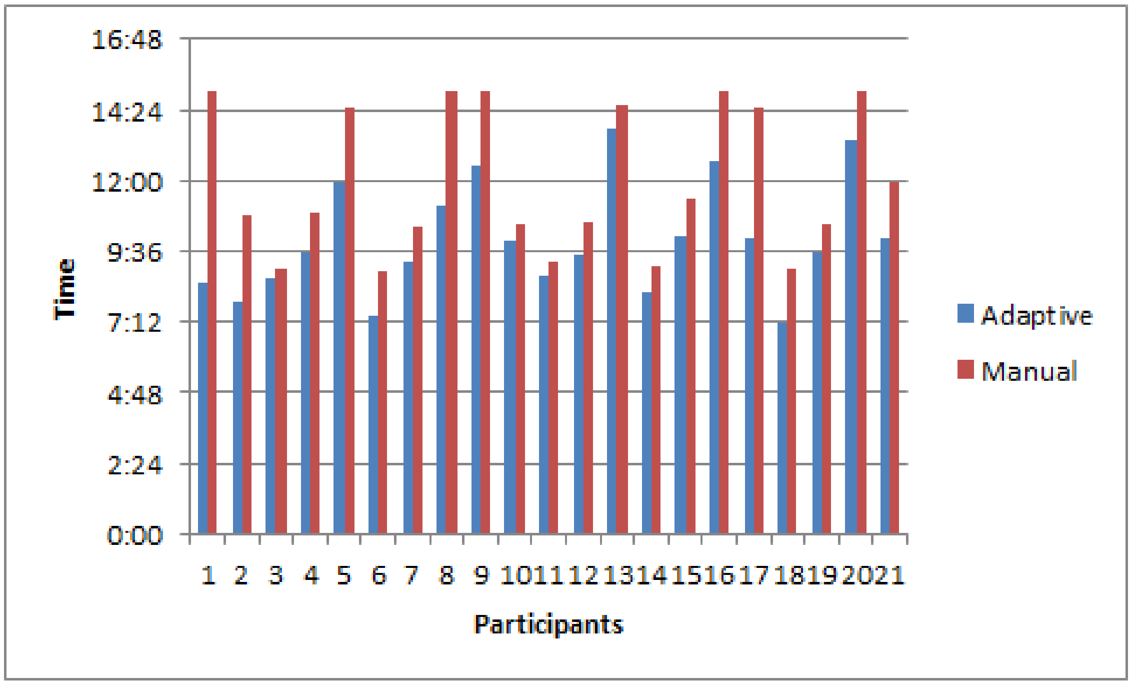
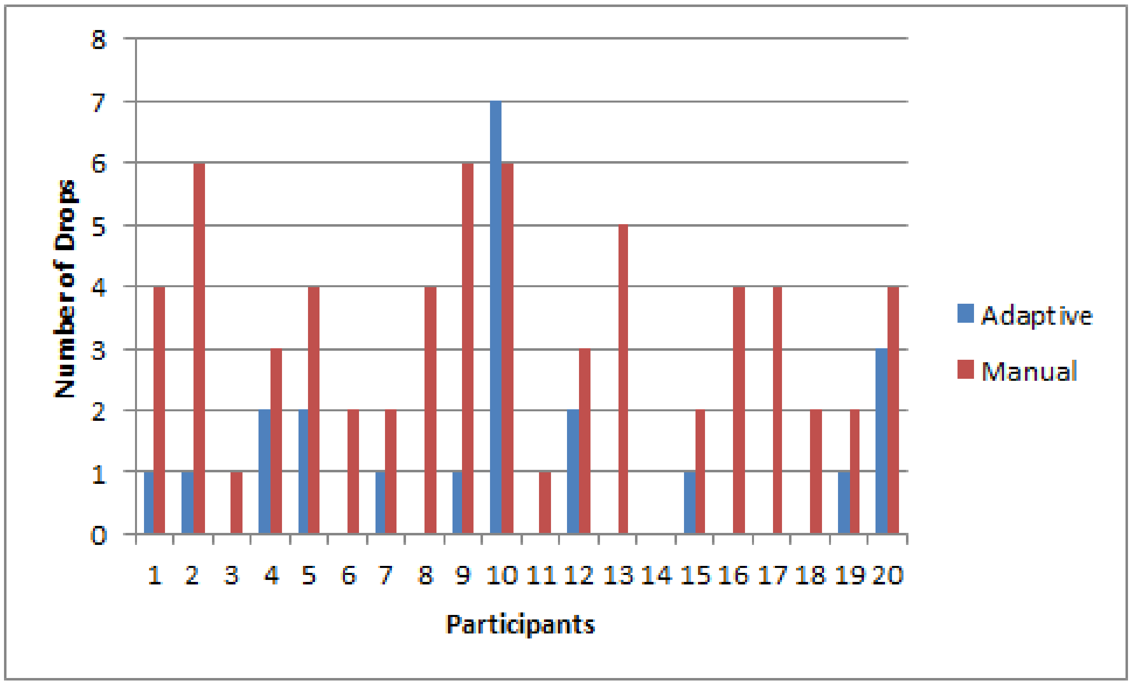


Figure 51: Time to complete Scenario 1 (left) and Scenario 2 (right) in minutes for each subject (x-axis). All of the participants (except subject #10) experience time improvements with the adaptive version of the user interface.



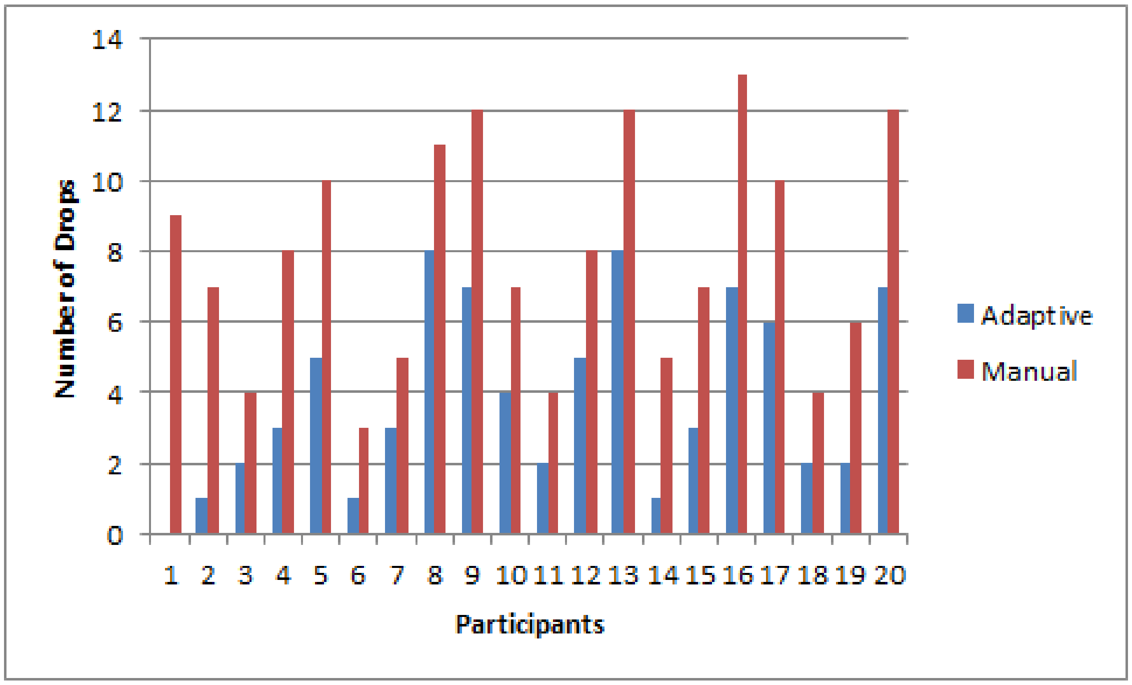
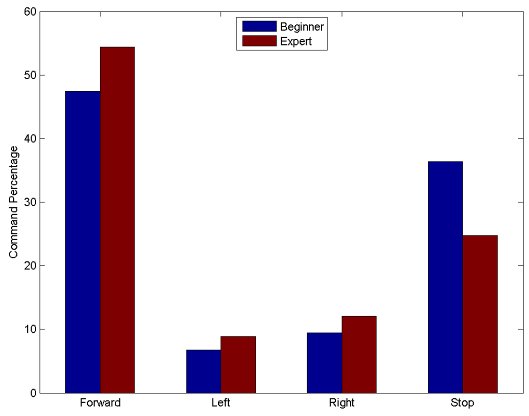
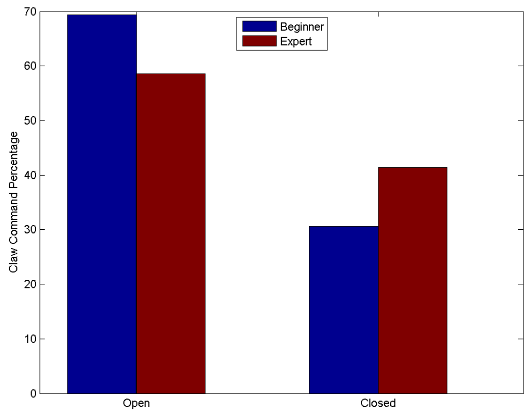
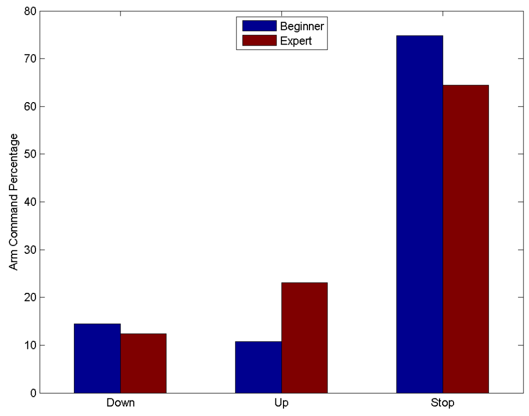


Figure 52: Number of objects dropped by each subject (x-axis) in Scenario 1 (left) and Scenario 2 (right). All of the participants (except subject #10) experience reductions in dropped objects with the adaptive version of the user interface.


Command Percentage
Beginner
Expert
Forward
Left
Right
Stop
0
10
20
30
40
50
60


Claw Command Percentage
Beginner
Expert
Open
Closed
0
10
20
30
40
50
60
70


Arm Command Percentage
Beginner
Expert
Down
Up
Stop
0
10
20
30
40
50
60
70
80

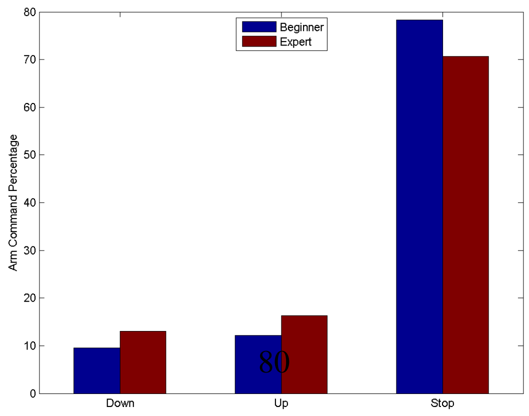

Arm Command Percentage
Beginner
Expert
Down
Up
Stop
0
10
20
30
40
50
60
70
80

Table 10: 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%

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 10 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 53 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.

5 Proposed Work

The proposed work is to further refine the IAI interface and the Adaptive Interface Component.

5.1 Multi-Modal Interface and Natural Interfacing

The first part of the proposed work is updating the current IAI interface with a more natural multi-model interaction interface between the human user and the HU-IE Robots by introducing a mixture of natural human voice communication, touch gestures, and body gestures. The current IAI interface bounds the user to a classic PC platform and limits natural communication. This proposed addition to the IAI interface will be based on lessons learned from previous user studies.

The additional updates require adding a second Kinect sensor to capture the user's voice and body gestures. A multi-touch monitor or Wii U remote will be used to capture hand and finger gestures as well as the user's navigational commands.

5.2 Learning by Demonstration for Expert and Novice Users

Following the proposed IAI upgrades the next proposed work is to evaluate whether it is more valuable for expert users or novice users to use our learning by demonstration macro technique. In our recent studies we concluded that our learning by demonstration macro technique seemed to favor the expert user. Experts were able to create more beneficial macros that helped them complete the task more efficiently then the novice users. We also concluded that novice users were able to benefit more from an autonomous system then some expert users. Some experts seemed to do poorly with a more autonomous system then novice users, and novice users seemed to struggle with a more manual system. We determined that there is a gap between learning by demonstration for expert and novice users due to the system not properly adapting to the user's specification and requirements for task completion.

We propose to evaluate a natural and non-natural learning by demonstration macro technique for multi-robot manipulation for expert and novice users. We will first establish whether a user is an expert or novice based on our classification system. Next the user will create natural macros for the system to learn based on their own perception on what they need to complete the task. Then we will evaluate the users' naturally created macros versus some predefined macros through a series of multi-robot manipulation tasks.

5.3 Timeline

The expected completion dates for the proposed work are as follows:

1.	`Fall 2013`	Initial Evaluation
2.	`Spring 2014`	Final Tests and Documentation
3.	`Spring 2014`	Complete Dissertation

5.4 Publications

Below is an outline of our expected future publications:

1.	`Fall 2013`	Attend IROS 2013
2.	`Spring 2014`	Submit to IROS 2014

Bibliography

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

[2] Abdul-Rahim Ahmad, Otman Basir, and Khaled 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] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

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

[5] 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.

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

[7] J. Casper and R.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.

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

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

[10] Eric 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.

[11] 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.

[12] 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.

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

[14] 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.

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

[16] Yasuhisa Hirata, Youhei Kume, Zhi-Dong Wang, and Kazuhiro 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.

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

[18] 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. De 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.

[19] 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.

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

[21] 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.

[22] K. Kosuge and T. Oosumi. Decentralized control of multiple robots handling an object. In *International Conference on Intelligent Robots and Systems(IROS)*, 1996.

[23] 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.

[24] Bennie Lewis and Gita 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.

[25] Bennie Lewis and Gita 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.

[26] Bennie Lewis, Bulent Tastan, and Gita 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.

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

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

[29] 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.

[30] Sara Morgan. *Programming Microsoft Robotics Studio*. Microsoft Press, 2008.

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

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

[33] Lawrence 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.

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

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

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

[37] 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.

[38] 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.

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

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

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

[42] 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.

[43] Maarten Sierhuis, Jeffrey M. Bradshaw, Alessandro Acquisti, Ron van Hoof, Renia Jeffers, and Andrzej Uszok. Human-agent teamwork and adjustable autonomy in practice. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2003.

[44] 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.

[45] Jijun Wang, Michael Lewis, and Paul Scerri. Cooperating robots for search and rescue. In *Proceedings of AAMAS Workshop on Agent Technology for Disaster Management*, 2006.

[46] Jijun Wang, H. Wang, Michael 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.

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