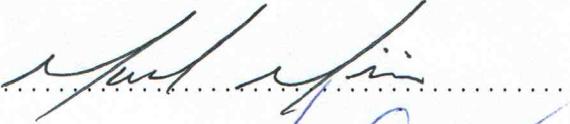


MULTI-TOUCH INTERACTION FOR
ROBOT COMMAND AND CONTROL

BY

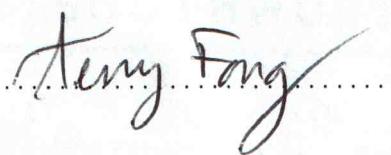
MARK JOHN MICIRE
B.S. UNIVERSITY OF SOUTH FLORIDA (1999)
M.S. UNIVERSITY OF SOUTH FLORIDA (2003)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
COMPUTER SCIENCE
UNIVERSITY OF MASSACHUSETTS LOWELL

Author:  Date: December, 2010

Dissertation Chair:  Dr. Holly A. Yanco

Committee Member:  Dr. Jill Drury

Committee Member:  Dr. Terry Fong

MULTI-TOUCH INTERACTION FOR
ROBOT COMMAND AND CONTROL

BY
MARK JOHN MICIRE

ABSTRACT OF A DISSERTATION SUBMITTED TO THE FACULTY OF
THE DEPARTMENT OF COMPUTER SCIENCE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
COMPUTER SCIENCE
UNIVERSITY OF MASSACHUSETTS LOWELL
DECEMBER, 2010

Dissertation Supervisor: Holly A. Yanco, Ph.D.
Associate Professor, Department of Computer Science

Abstract

In emergency response, gathering intelligence is still largely a manual process despite advances in mobile computing and multi-touch interaction. The labor-intensive nature of this process means that the information digested by personnel going into the field is typically an operational period old. In a day where satellite photography and mobile connectivity is becoming ubiquitous in our digital lives, it is alarming to find this is the state of the practice for most disciplines of emergency response. Recent advances in robotics, mobile communication, and multi-touch tabletop displays are bridging this technological gap and providing enhanced network centric operation and increased mission effectiveness. Our work focuses on closing the gap between the personnel in the field and the command hierarchy supporting those teams. Our research in human-computer interaction leverages these technologies for robot control through a collaborative tabletop multi-touch display. A single-robot operator control unit and a multi-robot command and control interface has been created. Users command individual or multiple robots through a gesture set designed to maximize ease of learning. Users can pan and zoom on any area, and the interface can integrate video feeds from individual robots so the users can see things from the robot's perspective. Manual robot control is achieved by using the DREAM (Dynamically Resizing Ergonomic and Multi-touch) Controller. The controller is painted on the screen beneath the user's hands, changing its size and orientation according to our newly designed algorithm for fast hand detection, finger registration, and handedness registration. In addition to robot control, the DREAM Controller and hand detection algorithms have a wide number of applications in general human-computer interaction such as keyboard emulation and multi-touch user interface design.

Acknowledgments

I would like to thank my committee, Dr. Holly Yanco, Dr. Jill Drury, and Dr. Terry Fong, for their guidance and patience through this process. In particular, I have to thank Holly for providing an environment where creativity is king and good science is the result. Holly is a catalyst for innovation and protects her students with the ferocity of a mother bear. She has provided the best qualities of a mentor, colleague, and friend. Thank you; I am better because of my time here.

Throughout this dissertation, I will use the word “we” often. I have survived because I have been supported by a wonderful group of people that work in the two robotics labs and our department staff. Thank you so much to all of you. I am a big fan of karma, so the all-night coding sessions, user tests, illustrations, and tireless editing will be remembered whenever the favor can be returned.

My family has tirelessly rallied behind me in all of my crazy adventures. Thank you to my mom and dad who had the foresight and discipline to make the start of this journey possible. Thank you to my sister who reminds me that I should keep running forward like Forest Gump despite all of my doubt and hesitation.

Thank you to Kate Tsui, who has patiently helped me through the good and bad parts of this academic struggle. Her edits, statistics, and scientific rigor are on every page of this thesis. She makes me a better scientist and a better person every day. Thank you. I cannot imagine having done this without you.

Finally, a thank you to Arnis Mangolds and John Blitch who arguably set me on this journey into the world of search and rescue robotics. Your humbleness and Zen-like disposition are constant reminders to focus the products of research in real applications. Whenever I need to gut-check a problem, I find that I can ask, “What would Mangold or Blitch do?” The answer is always nearby.

This work has been supported, in part, through grants from the National Science Foundation (NSF IIS-0415224 and IIS-0546309), Microsoft Research, National Institute for Standards and Technology (NIST 70NANB8H8168), and Army Research Office MURI (W911NF-07-1-0216).

Contents

1	Introduction	19
1.1	Disruptive and Enabling Technologies	21
1.2	Research Focus	24
1.3	Problem Statement	25
1.4	Approach	26
1.5	Thesis Statement and Research Goal	27
1.6	Contributions	28
1.7	Organization	29
2	Background	31
2.1	Multi-touch Technology	31
2.2	Multi-Touch Gestures	34
2.3	Robot Control	37
2.4	Multi-Robot Command and Control	39
2.5	3D and Immersive Telepresence	41
3	Performance Validation	45
3.1	Hypotheses	46
3.2	Experimental Design	46
3.2.1	Participants	48
3.2.2	Procedure and Data Collection	48
3.3	Data Considerations and Results	49
3.3.1	Task Completion Time	50

3.3.2	Accuracy	51
3.3.3	Performance Models	52
3.4	Impacts and Implications	53
4	Multi-Touch Interaction and Robots	55
4.1	Interface Description	56
4.1.1	Joystick Interface Design	59
4.1.2	Multi-touch Interface Design	60
4.1.3	Hypothesis	62
4.2	Experiment Design	63
4.2.1	Participants	63
4.2.2	Procedure	64
4.2.3	Data collection	64
4.3	Results and Discussion	65
4.3.1	Performance	67
4.3.2	Subjective Assessment	69
4.3.3	Learnability Assessment	70
4.4	Interaction Characterization	71
4.4.1	Participant 1	73
4.4.2	Participant 2	74
4.4.3	Participant 3	75
4.4.4	Participant 4	75
4.4.5	Participant 5	76
4.4.6	Participant 6	77
4.5	Impacts and Implications	77
5	MultiTouch Joystick Emulation	79
5.1	Design Approach	81
5.2	Related Works	83
5.3	Engineering Criteria	85
5.4	Hand And Finger Registration	88

5.5	Form and Function	92
5.6	Experiment Design	96
5.7	Results and Discussion	99
5.7.1	Hand Position	100
5.7.2	Board Position	101
5.7.3	Gender	102
5.8	Potential for Improvement	103
5.9	Impacts and Implications	105
6	Robust Multi-touch Robot Control	107
6.1	Interface Definitions	109
6.1.1	Surface:Window Interface Description	109
6.1.2	Full Screen Interface Description	113
6.2	DREAM Controller	115
6.3	Hypotheses	117
6.4	Experimental Design	118
6.4.1	Procedure	119
6.4.2	Data collection	120
6.4.3	Participants	120
6.5	Results and Discussion	121
6.5.1	Task Performance	121
6.5.2	Usability Performance	129
6.6	Impacts and Implications	135
7	User Defined Gestures	139
7.1	Experiment Design	140
7.1.1	Participants	141
7.1.2	Procedure	141
7.1.3	Tasks	146
7.2	Taxonomy of User Defined Gestures	148
7.3	Results and Discussion	151

7.3.1	Selection	153
7.3.2	Multi-hand and Multi-finger Gesturing	154
7.3.3	Handwriting	155
7.3.4	Gesture Usage as a Function of Prior Experience	156
7.4	Impacts and Implications	158
8	Multi-Touch Multi-Robot Command and Control	160
8.1	Technology Capabilities Assumptions	162
8.2	Gesture Design	165
8.2.1	Gesture Grammar	166
8.2.2	Robot Control Grammar	166
8.2.3	View Control	171
8.2.4	Menu Functions	172
8.2.5	Stop and Reset Gestures	174
8.2.6	DREAM Controller Integration	174
8.3	Gesture State Machine Implementation	176
8.4	Simulation Design	178
8.5	Experiment Design	180
8.5.1	Participants and Procedure	181
8.5.2	Data Collection	186
8.6	Results and Discussion	188
8.6.1	Gestures as a function of grammar	188
8.6.2	Gestures as a Function of Assumptions	191
8.6.3	Motivation for Further Improvement	193
8.6.4	Information Fusion and Decision Support	195
8.6.5	Questionnaire Results	196
8.6.6	User Expectations and Need for Intervention	196
8.7	Impacts and Implications	199
9	Conclusions and Future Work	200
9.1	Contributions	203

9.2 Future Work	205
9.3 Limitations of Research	206
9.4 Generalizations of Research	207
9.5 Closing Thoughts	209
A Coding Classifications for Multi-Touch Robot Control	222
A.1 Timed Event Coding Guidelines	222
A.2 Notes that may be of interest:	224
B Biographical Sketch of Author	225

List of Figures

1-1	Information gathered in the field (left) is hand annotated onto maps and then transcribed with a black marker (center) onto a global map that is entered into a GIS system and presented to the next operation period (right).	20
3-1	The Mitsubishi DiamondTouch digital tabletop (left) was evaluated in regards to task completion time and accuracy. A standard mouse, shown to the right of the participant, was used for comparison. The screen shot (right) shows a target presentation.	47
4-1	The UML USAR interface (top) is shown with a participant using the joystick configuration (lower left). This interface allows the user to operate the iRobot ATRV (lower right) though the NIST USAR course (shown in Figure 4-4).	57
4-2	Illustration of the flight-style joystick used in the interface design.	59
4-3	Screenshot of the multi-touch interface and guide to gestures that activate interface features and autonomy modes.	61
4-4	The robot operated in the NIST USAR arena (left) while experimenters uses ground truth maps (right) to record specific incidents. In the ground truth map, “Start” symbols represent the four possible starting points, “V” represents a victim location, lines represent the robot path, and circled numbers represent collisions with the arena walls.	65

4-5 Shoulder (left) and close (right) view of the drive control panel, providing control of translation (vertical) and rotation (horizontal).	71
5-1 The iRobot® Packbot® Hand Controller (left front) is an adapted Sony Playstation® style controller. The Packbot® EOD Operator Control Unit (left rear) places six degrees of freedom on each hand, requiring the operator to manage 12 degrees of freedom and 12 context sensitive push buttons for typical operation. The Foster Miller TALON robot controller (right) is operated through three two degree of freedom joysticks and a large array of switches, selectors, and dials. (<i>Photos courtesy of iRobot Corporation and the National Institute of Standards and Technology.</i>)	80
5-2 A dual-thumb joystick modeled after the Sony Playstation® controller (left) was used to inspire the design of a paper prototype (center) that was selected for the design of the multi-touch DREAM Controller (right) on the Microsoft Surface.	82
5-3 The hand characteristics and registration heuristics were, in part, inspired by Henry Dreyfuss's 1955 book "Designing for People." Reproduced with permission. Courtesy of Allworth Press.	89
5-4 Hand and finger registration is accomplished by first ensuring that the points are within the maximum size for a human hand (top left), then finding the centroid of the bounding box containing all of the points (top right), determining the two largest angles (bottom left), and determining if the angle from the thumb to the index finger is clockwise or counterclockwise (bottom right).	91
5-5 Users have a wide variety of hand characteristics. The DREAM Controller adjusts several parameters to tailor the size, orientation, and position of the controller. This design is intended to maximize the users' comfort and performance.	94

5-6	Participants in the study were asked to stand in one of eight positions around the Surface to test the sensitivity of the algorithm to hand orientation.	97
5-7	Participants were asked to place their hands in three positions. First, on the tips of their fingers (left) like a pianist or touch typist. Then in a relaxed position (center) with the pads of their fingers resting on the surface. Finally, they stretched their hands as far a possible (right) to demonstrate their maximum contact size.	99
5-8	Annotated images for failed hand and finger recognition of hand placements (left column) and successful recognition for the other hand (right column). The center point (c), thumb (t), index finger (i), and little finger (l) have been labelled. ti is the angle between the thumb and index finger and is overlaid on the angle tcl as an arc. When the arc ti extends over the line tc , then the algorithm failed. In other words, the angle between the thumb and the index finger must be smaller in order to correctly identify the fingers. . .	104
5-9	Our hand detection algorithm is played back in the Surface Simulator using the publicly available dataset (left). Additional photographic data (right) is included in this data set for researchers that wish to work with the vision system directly.	106
6-1	Surface:Window features and gestures for operating the robot with on-screen controls.	110
6-2	Surface:FullScreen features and gestures for operating the robot with on-screen controls.	113
6-3	The DREAM Controller, as configured for the ATRV-Jr robot used in this study. Illustration shows the controller functionality and many of the local controls that duplicate on-screen controls. . . .	116

6-4	The participant is observed using the multi-touch interfaces (left) to search for victim marker using an ATRV-Jr robot (middle) while test administrators (right) record video and written documentation regarding the performance of the run.	119
6-5	Participant performance was measured by counting the number of victims found (above) and the overall distance traveled in the arena (below). In the case of the Surface:Window interface, the participants were able to travel farther and find more victims. . .	123
6-6	Memorability was measured by testing the recall of participants immediately after the experiment run and then ten days later. The percentage of correct answers was recorded and compared to assess the memorability of the robot control mechanisms.	132
7-1	Participants were seated in front of a Mitsubishi DiamondTouch fastened to a round table where they provided gestures on a static front projected image.	140
7-2	Participants were presented with a physical ActiveMedia Pioneer 2Dx robot (left) that was iconically depicted in the experiment slides (right).	141
7-3	Normalized percentages of the gestures expressed by the participants in each of the classification groups.	152
8-1	An example of the Lasso gesture (left) as dragged on the screen until the finger was lifted. The result of this lasso is shown by the yellow highlighted robots (right).	167
8-2	State machine for the grammar showing the basic structure and state changes for special cases.	169

8-3	After selecting the robots, the user pressed and held his or her finger on the surface for four seconds (left). A menu appeared under the finger (center) where options could be selected such as Color, Queue, and Execute. When the user selected the desired color (right) the finger was lifted and the selected robots were changed.	172
8-4	Illustration of the DREAM Controller configured for dynamically selected robots in this interface. The left hand controller displays the data from the laser range finder in a radar-like display and allows the user to drive the robot chassis. The right hand shows the video from the robot's view and controls the pan and tilt of the camera.	176
8-5	Finite state machine for the gestures in the multi-robot command and control interface.	177
8-6	An aerial view of the urban simulation environment used to test the gesture based command and control interface. The three dimensional city was approximately 2 acres in area and used realistic hardware-accelerated physics for robot movement and interaction with the environment.	179
8-7	Circles were drawn around the robots on a transparent layer outside of the simulation. A selected robot (left) is shown with a yellow highlight and unselected robot (right). In both cases, the blue triangle represents the orientation of the robot.	180
8-8	Illustration of the fictional Loop City, Nevada, drawn in the style of a USGS topographical map showing major features, highways, and common landmark names. Participants used this map as a reference while directing search operations with the eight simulated robots.	183

8-9 Final task of the scenario required the participant to take manual control of the robot (center) and use the DREAM Controller to view the suspect construction vehicle. This screenshot shows the activated DREAM Controller and positions of the fingers are shown lightened circles.	187
8-10 Percentages of gestures used, grouped by their position in the grammar discussed in Section 8.2.2.	190
9-1 Multi-touch technology will be making its way into the field in the very near future. The UAV (above) provides real-time imagery to the command staff back at the base of operation while the robot control unit (below) allows the field technicians to comfortably control the UGV robot and neutralize the threat.	201

List of Tables

2.1	Overview of collaborative tabletop displays in active development.	34
3.1	Mean task completion time (\bar{X} in ms) and standard deviation (SD) by target size and input method. Paired t -test result of input methods shown as (t).	51
3.2	Mean error rate (\bar{X}) and standard deviation (SD) by target size and input method. Paired t -test result of input methods shown as (t).	51
4.1	Constructive performance in the USAR arena.	67
4.2	Number of destructive incidents in the USAR arena.	68
4.3	Participants' subjective assessment.	70
5.1	Hand and finger recognition rates by hand	101
5.2	Hand and finger recognition rates by board position	102
5.3	Hand and finger recognition rates by sex	103
6.1	Four versions of the operator control interface. The original joystick version (top) was compared against the DiamondTouch version from Chapter 4 (upper-middle) and two newer Surface versions (lower-middle and bottom).	108
6.2	Area explored including overlap in the USAR arena (in squared feet).	122
6.3	Number of destructive incidents per square foot in the USAR arena.	126
6.4	Victims found in the USAR arena.	128
6.5	Participants' subjective assessment.	130

Chapter 1

Introduction

Hurricane Katrina made landfall near Biloxi, Mississippi, in August 2005. The resulting disaster response revealed a large technological gap. Although satellite and aerial information existed, it was not available to or utilized by the search teams. Instead, state and federal teams were required to use topographical and urban road maps that were often outdated or incorrect. These maps were distributed to search personnel in the morning for the daily grid searches. When the teams returned in the evening, personnel on the night shift integrated updates by hand based upon search reports (Micire, 2008).

Large-scale responses require the dissemination of information through multiple law enforcement, search and rescue, and municipal facilities groups. The ability of the command hierarchy to rapidly digest, discuss, and make decisions based on new information can mean the difference between mission success and failure. Coordination and interoperability problems during Hurricane Katrina provided a catalyst for a significant overhaul in the national search and rescue (SAR) response system including mandatory adoption of the Incident Command System (ICS) and National Incident Management System (NIMS). The reorganization of the command hierarchy and multi-agency interoperability is a positive change, but unfortunately, the information gathering and planning stages have largely remained the same. This planning is still largely performed through the use of radios and paper maps (Committee on Planning for Catastrophe, 2007).



Figure 1-1: Information gathered in the field (left) is hand annotated onto maps and then transcribed with a black marker (center) onto a global map that is entered into a GIS system and presented to the next operation period (right).

Search and rescue is not the only type of response affected by this problem. In the summer of 2008 during the California wildfire responses, the aggregations and correlation of data from the field was largely performed by hand. This manual process occurred despite the use of ground GPS tracking, digital photography, and the use of the Ikhana Predator unmanned aerial system. As shown in Figure 1-1, personnel manually took data from each of these sources and manually correlated them to a centrally referenced map. Once this map was updated with black pen markers, the map was entered into a geographical information system (GIS) for paper printing and then presented to the personnel on the next operation period. Since this process took an entire operation period to complete, the map shown to the personnel reflected intelligence from nearly 24 hours ago.

The ICS and NIMS command system is flexible and can accommodate collaborative technology, even though in current practice this is far from the case. To understand problems in the current system, the state of the practice must be examined. For the most part, information is gathered by the reconnaissance or response teams in the field. At the end of the operation period, the command is

updated on findings, progress, and issues for the next operation period. All of the information is quickly digested onto paper and laptop computers for map products and briefings. Several of the problems with this strategy include:

- This information is static and therefore only provides snapshot views of the progress and goals of the operation.
- The temporal resolution of the operation is typically on the order of 8 to 12 hours.
- Each of the records on maps, papers, and laptops become small “islands” of information.
- There is no global operational picture until all of the intelligence is manually gathered and correlated by personnel.
- Planning becomes a rapid and chaotic process since there is very little time between operation periods.

The next decade will continue to exacerbate these problems since the amount of data entering the system will increase by orders of magnitude. The new sources of data such as network enabled responders and robots (discussed in the next section) will provide higher resolution, location, and temporal information, but there must be a way to digest this information in a collaborative command environment. Multi-touch tabletop collaborative displays are a well positioned solution for displaying and interacting with these new streams of data and intelligence.

1.1 Disruptive and Enabling Technologies

Much like the changes in the late 20th century, the 21st century will prove no exception to the advancement of command and control (C2). In fact, it represents the convergence of at least three technologies that will represent a significant change in the processes and systems of command and control.

First, there is the adoption of Network Centric Operation (NCO). NCO leverages global positioning, distributed network communication, and reachback capabilities to connect the command hierarchy vertically and laterally for a shared view of the operations space. The effect for group coordination and awareness can be dramatic if implemented correctly and reliably. From Alberts and Hayes (2006), “At the risk of oversimplification, [NCO] is a two-step process: first, achieving shared awareness, and second, leveraging shared awareness to achieve a greater degree of self-synchronization, leading to dramatic increases in both agility and effectiveness. The magic of [NCO] is the emergence of self-synchronizing behavior.” It has only been in the last decade that the technology to support a robust network-centric view of operations has been possible. Now that it is reaching maturity and acceptance among leaders, NCO will quickly become ubiquitous and expected in any command and control scenario, military, or otherwise.

Second, there has been the significant use of unmanned ground vehicles (UGVs) and unmanned aerial vehicles (UAVs) in military and domestic operations. Decades of funding from science and defense agencies are beginning to produce field-worthy and reliable platforms. Recent technological advances in the DARPA Grand Challenge and other military funded ground robot research has shown that robots are quickly maturing and will soon be autonomously performing dangerous tasks in real and dynamic domains (e.g. (Thrun et al., 2007; Urmson et al., 2007)). UGV manufacturers have delivered over 12,000 robots to US military and other government customers in the last five years (Singer, 2009). UAVs have not only proven their worth and become commonplace in the military, but also they have begun to replace their human-piloted counterparts. In fact, in 2009, more pilots were trained for unmanned air vehicles than manned aircraft (Singer, 2009). As these systems advance, their autonomy will increase and their coordination with ground forces (manned or unmanned) will need to become more coordinated and robust. The role of robots in SAR operations is still maturing and changing, but as robots are adopted, the increase in data and intelligence will only increase the workload for planning teams.

Finally, the use of large multi-touch interactive tabletops or displays has brought value to the task of command and control through increased interaction and group collaboration. These tabletop displays have existed for several decades (Buxton, 2007; Krueger, 1983), but it is only recently that they have come into popularity for viewing geospatial and temporal data for group decision making (TouchTable, 2007; Rowe et al., 2009; Taylor, 2006). A horizontal tabletop view of the task space, similar to a paper map, helps create a shared workspace that catalyzes group discussion and decision making. Additionally, multi-touch displays and tabletops are now being packaged for use in mobile command centers in the field (Tse et al., 2006). These multi-touch tabletop interfaces significantly benefit operations because, unlike paper maps, they can be automatically updated and benefit from the digital information gathering from all of the elements described above in the NCO and robot assets.

In contrast to the earlier analysis of the manual planning strategy, these three technologies enable the following improvements:

- Information is dynamic and provides live or slightly delayed views of the progress and goals of the operation.
- The temporal resolution of the operation is typically on the order of seconds, minutes, or tens of minutes.
- Each of the records are correlated to a common computing and control platform that can be collaboratively viewed and discussed.
- There is a global operational picture as data is automatically gathered, and intelligence can also be manually correlated by personnel.
- Planning becomes informed and iterative since the command hierarchy can observe progress as teams progress through their operation periods.

1.2 Research Focus

Command and control has been developed largely in government and military domains. SAR borrows from the military’s experience and expertise in this area. While the end goal is different, the foundation and intermediate steps in these two domains are very similar. In both, there is a requirement to identify people (such as enemies or disaster victims), assemble people or equipment to address the problem (such as weapons or rescue equipment), and provide a solution to the problem (such as military force or rendering assistance). Both of these domains require the coordination of large numbers of players and logistical support. The similarity is not surprising when one considers the large number of former military personnel that later become trained and utilized in SAR operations.

The research in this dissertation focuses on the SAR domain. SAR can include many sub-disciplines including urban search and rescue (USAR), wilderness rescue, wildfire response, and avalanche rescue. Ultimately, if the rescue sub-domain can benefit from NCO, robots, and multi-touch enabled C2, then this research can be applied to rescue operations.

This research and dissertation focuses on two fundamental use cases. First, there is the case of a high level commander requiring a “theater wide” view of the response space. This view will include the affected area of interest, all of the tracked assets deployed into the scenario, and all of the agents that can be tasked, including robots and humans. Additional aids such as threat modeling and probabilistic search strategies can optionally be viewed at this level of interaction.

The second use case centers on the person responsible for monitoring and collaboratively interacting with the robots should they require interaction or corrective measures. In the human-robot interaction literature, this person is typically termed the “robot operator.” For UAVs and UGVs, it is expected that a certain level of autonomy will allow the robot to carry out navigation and basic obstacle avoidance. Regardless of the implementation and reliability of the autonomy, it should be expected that a human operator will at some point

need to control and observe the robot directly. This intervention might include manual teleoperation of the platform or manipulation of sensors such as cameras. The interface described in this research provides teleoperation-level “fine grained” interaction to individual robots.

The overall goal of this research is to bridge these two use cases in a natural and intuitive manner. The “theater wide view” will show tasking and planning on a large scale, but the users should then be able to select and zoom in to a robot of interest and begin manual intervention of the robot’s sensors or manipulators. With this functionality, the users should feel that they have a unified display and control interface.

1.3 Problem Statement

As the command staff and hierarchy of emergency response organizations has strengthened over the years, the effectiveness and execution of each person’s responsibilities has improved. Roles and responsibilities have been iteratively tuned to maximize impact and minimize idle personnel. As workload is maximized in finite time, there may not be the opportunity for in depth training on new technologies. Lack of advanced training, therefore, leaves gaps in the staff’s ability to fully utilize these advanced tools and limits their ability to further optimize their workload. This cycle presents an artificial upper bound that is created by the efficiency of the human-computer or human-robot interface. It is this demographic of emergency response managers that represents our target community. More specifically, the group that will benefit the most from this research are emergency response managers who have seen their workflow increase due to technological advancements but still lack an intuitive and collaborative method for interpreting, digesting, and making informed decisions related to the command and control of people, equipment, and robot assets.

If multi-touch tools are to be used effectively in SAR, learnability is the most important key to widespread acceptance and use. In the midst of a disaster

response (real or training exercise), support personnel cannot stop the command staff and say, “We will now have an hour long demonstration of the gesture set needed to operate the multi-touch C2 board in front of you.” The command staff must be able to walk up to the apparatus and begin using it with little to no instruction. The user interface must be intuitive enough and designed to “do the right thing” with little training.

As the command staff becomes familiar with the interface and the ease of learning is achieved, efficiency becomes the next most important feature. While seeking efficiency, we may include smaller or less obvious gestures that would be analogous to “hotkey” shortcuts in the mouse and keyboard interface domains.

This research explores this basic requirement of learnability and efficiency in the gesture set for multi-touch command and control interfaces for robots. To this end, the strategies for multi-touch interaction have been iteratively validated and tested.

1.4 Approach

This research represents an iterative, bottom-up, user-centered approach to interface design. Our methodology began with validation performance testing using the multi-touch tabletop compared to a well studied user interface using more traditional robot control methods such as a joystick to ensure that there was not a significant degradation of performance measures. The participants’ gestures and responses to these experiments also provided insightful hints regarding their expectations of the user interface elements. It is from this data that we derived the models and guidelines for the next iteration of the interface. We also began to evaluate learnability by evaluating the number of questions asked by the participants.

It was from the above experimentation and analysis that we realized that the gestures used to manipulate the user interface are the key to maximizing learnability. As humans, we naturally use our hands and arms to provide rich

expression. Leveraging these natural tendencies in a software user interface will provide a wealth of information when compared to traditional input methods. The key is to ensure that our gestural tendencies (natural or learned) are correctly captured in the user interface design. In many cases, these gestures may be subtle and difficult to detect. They may also be borrowed from disparate learned phenomena such as automobiles or consumer electronics. In some cases, they may even be mutually exclusive and impossible to implement in a holistic fashion. Regardless of the correctness of the expressed gestures, they are “right” in the context of learnability if that gesture set is what the users naturally want to use on the interface.

Identifying a gesture set that all users could begin using immediately and without instruction would be the “holy grail” of interface learnability. While this may not be feasible for many of the reasons stated above, a entirely user generated gesture set is a good place to begin. Towards this end, we presented participants with specific robot command tasks to complete in a static digital prototype interface analogous to the paper prototypes used in other human-computer interface studies. It is the results from this simple but rich data set that we are using to bootstrap the user-centered iterative design process.

1.5 Thesis Statement and Research Goal

To support the mission effectiveness of emergency response operations, my claim, which I will prove in this thesis, is that:

Multi-touch interfaces improve human-robot interaction for single robot teleoperation and multi-robot command and control. This is particularly valuable for supporting novice users and reducing training time in domains such as search and rescue.

While the focus of this research is specific to SAR and robot control, the implications of novice user generated gesture sets as a practice has far reaching

applications outside of this particular domain. As multi-touch devices become more available to the general populous, it would be ideal if software designers would begin with the users' expectations of interaction rather than the technology itself. This user-centered design ensures that the wealth of new interaction options does not overshadow the need for a succinct and understandable design. Interface design is more often helped by constraints than freedoms. So it follows that over the long-term, the design constraints of the users' natural interaction are more constructive than new technological freedoms when designing new and useful interfaces.

Based upon the application domain and the interaction technology explored, the following research goal was established: *Multi-touch is a rich and interactive method of computer interaction that can provide enhanced learnability in time-critical and safety-critical domains when implemented correctly. By carefully studying the biomechanics of the human hand and leveraging the natural responses of users to multi-touch displays, human-robot interaction can be enhanced for single and multiple robot control. The goal will be to maximize learnability and therefore lessen the amount of training time required for proficient control of the robot or robot teams.*

1.6 Contributions

This research has resulted in a complete user interface for single and multi-robot control. Research contributions include the following:

- Performance model of table-top touch interfaces based on Fitts's Law.
- Validation of multi-touch human-robot interaction compared to a traditional joystick-based single robot interface.
- An Algorithm for five-point hand identification and finger registration.
- Interface using the dynamically resizing, ergonomic, and multi-touch (DREAM) controller for joystick emulation.

- A user generated multi-touch gesture set tailored for ease of learning.
- An integrated interface designed specifically for multi-touch interaction that combines high level command and control of robot teams and individual control of single robots.

1.7 Organization

The dissertation is organized as follows. In Chapter 2, I explore the previous work and foundational research upon which this dissertation is based. Where applicable, I compare, contrast, and examine gaps in this literature relative to the investigation at hand.

Chapter 3 describes the performance model used to ensure that the multi-touch interface does not impair user performance for basic tasks. Specifically, Fitts's Law is used to test the performance of the multi-touch display compared to classic mouse-based interaction. Surprisingly, this fundamental performance validation of multi-touch tabletops was not found in the human-computer interaction literature. Since mouse-based interfaces are well established for most computer interfaces, this step becomes necessary to ensure that there is not a baseline decrease in user performance before metrics for gestures, learnability, and usability are examined.

After establishing comparable performance, Chapter 4 compares a well studied joystick-based and keyboard-based robot control with a prototype multi-touch robot control user interface for robot teleoperation. Again, the research goal here is to ensure that the change in input and output methods does not dramatically decrease the user performance in an SAR related task. Although quantitative data is presented in detail, incorrect interface design assumptions provided a wealth of qualitative information for the next iteration of the robot control interface.

A ground-up redesign of the single robot control interface is described in Chapter 5 that incorporates a design based on the biomechanics of the hand and borrows from muscle memory gained through video game usage. In Chapter 6, we describe how this new joystick design and a new interface approach was tested in

an identical experimental setting as Chapter 4 to allow for a four-way comparison.

Chapter 7 explores the usability of a command and control interface from the perspective of a novice user. Specifically, robot related tasks are presented to the user to determine the gestures that people would naturally use, rather than the gestures they would be instructed to use in a pre-designed system. This chapter presents the details of these findings, a taxonomy of the gesture set, and guidelines that provide the basis for gesture design for following chapters.

The findings gathered in Chapter 7 are then used to create an interface for controlling multiple robots in a simulated environment in Chapter 8. A usability test involving six representatives from a FEMA search and rescue team is discussed along with findings and recommendations for future command and control interfaces.

Finally, Chapter 9 discusses the broader impact of this research and the barriers that will need to be overcome for widespread acceptance. This discussion includes a narrative example of multi-touch used in the field and in the command center for enhanced operational effectiveness.

Chapter 2

Background

This work presents the convergence of three technologies. As discussed in Chapter 1, the combination of multi-touch tables, robots, and command and control is novel and timely. Interestingly, little research on the confluence of all three of these topics exists in the literature. As such, this background discussion focuses around each of these three research areas individually and in cross sections when applicable.

2.1 Multi-touch Technology

Recent commercial successes such as the iPhone have brought attention to multi-touch devices. Interestingly, very few people know that this technology actually goes back to the early 1980s when personal computers were in their infancy. Bill Buxton (2007) provides one of the most insightful and thorough surveys of touch technology. He is careful to note that the interest in multi-touch technology and its seemingly late acceptance is not unlike the computer mouse. Originally developed in 1965, the mouse was not truly ubiquitous until 1995 with the release of Microsoft Windows 95. If multi-touch devices are gaining acceptance at the same rate at the original mouse, then the decade following 2010 may prove to be an excellent time for multi-touch software development. Buxton's 2007 survey article and historical timeline is far too in-depth for this background discussion, but it is important to

recognize that the ideas and innovation around multi-touch are not particularly new. Like personal computers in the early 1980s, multi-touch devices (as a market) appear to have been looking for the right combination of price, portability, and a “killer application” to guarantee success.

Some of the first work in rich gesture recognition for screens and tabletops occurred in 1983, done by Myron Krueger (Krueger, 1983). At this time, the technology did not yet exist to detect multiple touch points on CRT screens, so Krueger used computer vision to track hand motions. His system was one of the first to interact with multiple hands and fingers. Since touch could not be detected, dwell time was used as a trigger for events. Early videos of this system and the many papers that followed (including (Krueger et al., 1985; Krueger, 1991)) provide the foundation for many gestures that are now common including pinch and reverse pinch.

From 1984 to 2000, multi-touch devices were largely a novelty and targeted specific research applications. Almost two decades later, in 2001, researchers at Mitsubishi Electric Research Laboratories (MERL) began the development of an experimental multi-user interface device (Dietz and Leigh, 2001). Several years of development resulted in the commercially viable DiamondTouch board. Although low quantity runs of these boards have been created, they were never fully marketed through Mitsubishi. As of 2008, the DiamondTouch is now developed and marketed exclusively by Circle Twelve, Inc. (Circle Twelve, 2009). The DiamondTouch screen uses an array of antennas below the laminated touch surface to transmit unique signals corresponding to the antennas’ respective *x* and *y* position. From these antennas, a small radio signal is coupled through the user’s finger to a receiver connected to the user’s body. This receiver can take the form of a wristband, seat cover, or floor mat. The use of multiple receivers allows for unique identification of individuals. From these signals, the computer software is able to determine where the person is touching the interface, who is touching the interface, and in how many locations each respective person is touching the screen.

Two groups have recently contributed to the multi-touch technology, although

their publications are largely design documents and not usability studies. The first was created at New York University (NYU) by a rear-projected surface that uses frustrated total internal reflection (FTIR) to create a multi-touch interface (Han, 2005). In this case, a covered Plexiglas sheet is illuminated at its edges by infrared (IR) Light Emitting Diodes (LEDs). When touched, an IR filtered camera in parallel with the projector captures the reflected light. The detection of multiple touches then becomes an image segmentation process, in which the centroid of each “blob” created by the finger touches is identified. This process has reportedly allowed for high frame-rate interaction with the interface and rich gesture recognition. While elegant in the simplicity and affordability of the mechanical design, this implementation has not seen any formal evaluation or open software development. After significant media exposure and commercial interest(Han, 2006), Han founded the company Perceptive Pixel to commercially develop a product for television and defense customers. Unfortunately, this new device is very cost prohibitive for most customers in the emergency response market and is not packaged for field use.

During the same time as Han’s work, Northrup Grumman and Applied Minds developed an IR based tabletop touch screen (TouchTable, 2007). This system has two incarnations: a large front projected touch table and a touch screen covered high definition television. As Northrup Grumman is a defense contractor to the United States government, there has been very little published about this interface. We do know, however, that this system only supports single points of contact despite its large size and collaborative capabilities. It is unclear if rich user interactions such as gestures are supported in this implementation. The touch-screen high definition TV is of particular interest to this research because it is packaged for use in the field by military personnel. Many of these screens have already been deployed by the US military and are used to coordinate mission objectives and command resources. While time will tell if this practice becomes ubiquitous, what this does signify is the need for advanced command and control collaborative technologies is increasing and tabletop touch screens have been identified as a worthwhile research investment.

Table 2.1: Overview of collaborative tabletop displays in active development.

Name	Research Group	Projection Technique	Multi-Touch	Simultaneous Multi-Person	Identifiable Multi-Person
DiamondTouch	Mitsubishi (MERL)	Front	Yes	Yes	Yes
FTIR	New York University	Rear	Yes	Yes	No
Touch Table	Northrop Grumman	Front	No	No	No
Surface	Microsoft	Rear	Yes	Yes	No

Finally, Microsoft has recently begun manufacturing a diffuse illumination interface similar in many ways to the NYU interface described above (Microsoft, 2007). Notably, Microsoft only offers this product in a tabletop configuration and appears to be targeting it for the entertainment and hospitality industries. The system uses multiple cameras below the projection surface to track multiple points of contact from multiple people simultaneously. Unlike the DiamondTouch from Mitsubishi, it cannot uniquely identify the individual interacting with the interface. Of all the platforms, the Surface is one of the most mature from a software design architecture standpoint. It has bindings for Windows Presentation Foundation and XNA Graphics Frameworks, and the software development kit includes a robust emulation environment.

2.2 Multi-Touch Gestures

Most prior work in multi-touch gesture design has taken the approach of letting human-computer interaction (HCI) experts design gesture sets, and then conducting user studies to verify whether these sets are, in fact, natural or easy to use. Rekimoto (2002) and Tse et al. (2006) are the two most recognized and cited HCI-expert generated gesture sets. Although this method has been shown to produce satisfactory results, this method may not produce the most natural gesture set for novice users. The gesture designers may not necessarily place learnability over other engineering requirements.

One of the most influential analysis of gesture sets on tabletop devices was by Wu et al. (2006). This research proposed a systematic procedure for designing

multi-touch interaction. They identified three major issues that were not addressed with previous design attempts: incorporating multi-finger and multi-hand gestures into an environment which has been traditionally pointer-based, occlusion issues, and access of areas on the surface which are physically uncomfortable to reach. They defined gestures as having three phases: registration, relaxation, and reuse. Gestures would be registered either statically or dynamically, and, after being registered, the user would not be constrained to maintaining the same hand position (relaxation). In different contexts or when performing different tasks, users could reuse a previous gesture to operate differently, perhaps by using some sort of gestural cue to change tools.

Epps et al. (2006) conducted a similar study, using a combination of multi-touch and computer vision inputs to allow users to interact with the system while not in contact with the board. Participants were asked to perform common desktop computing tasks on the tabletop, but most employed largely off-the-surface, 3D gestures to accomplish these tasks. The researchers concluded that participants prefer using their index finger only, and that for more complicated tasks, there is a strong need for the ability to gesture above the table surface. Epps et al.'s work is an important data point for this multi-touch research since their participants had a significant bias toward single finger interaction. This result would tend to indicate that the participants carry a bias from single pointer user interface (UI) paradigms or that the index finger is the most natural method for expression.

Although Epps et al. (2006) indicates that 3D off-the-surface gestures are the most natural for users, this gesturing technique falls outside of the scope of this study and is prohibitively difficult in the selected application domain. The current state of the art for off-the-table gestures require range sensing or camera technologies that are sensitive to ambient light, changes in light, and visual noise. The complexity of setting up a secondary gesture detection system is also prohibitive for the field requirements of the command and control domain.

Wang and Ren (2009) conducted a study to determine how hand geometry and finger properties affect participants' abilities to convey information to the

touch surface. Participants performed a set of pointing and rocking gestures while various finger properties, such as fingertip position and orientation, were recorded using an FTIR interface. From this data, they developed a set of design guidelines for multi-touch widgets with the natural performance of the human hand as the guiding factor. Their results agree with our findings in Chapter 3 that on-screen widgets should maintain a certain minimum size and be oriented in a useful way.

Koskinen et al. (2008) examined the possibility of improving power station control rooms using multi-touch tables, instead of the single-touch displays that are currently used. Their study investigated the natural gesture space for this environment by asking participants to demonstrate how they would perform common mouse-driven tasks on a multi-touch screen. They concluded that, in general, participants prefer single-hand and single-finger interaction over more complicated gestures, and that participants preferred gestures that required less contact with the screen. However, we believe that the study most likely introduced a bias: when asked to perform a well-known task which involves a mouse, participants are likely to be biased towards using a single finger.

Wobbrock et al. (2009) conducted a study to create a natural gesture set by designing around unbiased user input. They presented participants with tasks to perform with both one and two hands, and no prompting as to what was an acceptable gesture. They found that participants used an arbitrary number of fingers to perform many tasks, so differentiating gestures based on the number of fingers on the tabletop may be a poor choice. Additionally, they found that participants preferred using one hand rather than two. Their work was focused on tasks such as word processing and managing documents. As their study confirmed, this domain is heavily influenced by desktop computing and WIMP (window, icon, menu, and pointing device) paradigms.

The arguments from Koskinen et al. (2008) and Wobbrock et al. (2009) are relevant and well formed, but not overly surprising. In both cases, the participants were asked to create gestures to perform well known tasks that typically involve using a mouse pointer and keyboard. To find that the participant was significantly

biased toward single hand or single finger input is an important user bias to note, but something that an experienced UI designer would expect.

The task of command and control of robots is fundamentally different in many ways from desktop tasks. The temporality of the movements of the robots is a good example. The robots' movements are not immediate and there is an expectation that the robots will move at the best pace possible to achieve their individual goals. There are very few examples of desktop tasks that exhibit this behavior (outside of video games that are slowed down to emulate the time of the "real world"). A second difference is the notion that the user will be tasking various robots to perform different tasks. One robot may scout for information while a second robot is commanded to stay in place and recharge its batteries. There are very few examples in the desktop metaphor that require a user to independently task icons in the user interface.

Given the above differences, the research related to user defined gesture sets is certainly taken into consideration, but not taken in its entirety. Chapter 7 experimentally explores this area and determines the extent to which users are biased to single pointer desktop paradigms when using robots in a command and control environment.

2.3 Robot Control

Interaction with remote robot systems, such as those used in search and rescue scenarios, requires a user to be able to obtain situation awareness using the system's graphical display. Situation awareness (SA) is defined by Endsley (1988) as "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future." Yanco and Drury (2004) have modified this definition for human-robot interaction (HRI) to be the perception of the robot's location, surroundings, and status; the comprehension of their meaning; and the projection of how the robot will behave in the near future.

A robot operator control unit display usually includes a video window and status information about the robot (Yanco and Drury, 2006). Input devices for interaction with a remote robot system most often are joysticks, keyboards, or mice. Other input devices have also been used, including stylus-based interaction in (Skubic et al., 2002, 2003), and (Beard et al., 2005). While the use of speech and gestures have been studied for applications where the robot is collocated with its user (Perzanowski et al., 2001), they do not transfer well to a remote robot system, as the gesture information is lost without visual contact.

The problem with joysticks, mice, and keyboards is that they add a layer of indirection to HRI. Due to the distance between the user and the robot, the interaction already includes a layer of abstraction; the user must be able to interpret the on-screen information to understand the robot's current situation. With robot system manipulation mapped to a joystick or keyboard, the user's focus moves away from the need to interpret the video on the screen to the need to understand how to manipulate the images on the screen with the provided input device. By removing the joystick, mouse, or keyboard from the interaction, we remove a layer of interface abstraction and thereby increase interaction by increasing the degree of direct manipulation (Shneiderman, 1983). In the case of HRI, the lessening of indirection should allow users to more directly interact with the robot and affect its behavior.

Despite the recent interest in touch technology, single point touch based displays are not new. Their original incarnation was in compact personal data assistants (PDA) and tablet-based personal computers during the late 1980s and early 1990s. These commercially available devices largely emulated mouse pointer interaction and provided little in the way of further interaction. It is not surprising that there have been few successes in HRI using these small, and often computationally limited, devices.

In (Perzanowski et al., 2001), a PDA is part of the multi-modal robot control; the user can issue commands from the PDA or select a destination on a map. Keskinpala et al. (2003) use a PDA with simple compass-style buttons to drive the

robot forwards, backwards, left, and right. Fong et al. (2003) use a PDA to drive a remote robot using waypoint navigation on a video panel or a two-axis velocity control using a widget. Beard et al. (2005) use a PDA and voice commands to provide high level flight control of an unmanned air vehicle. Skubic et al. (2002) also uses a PDA to drive a robot; the user sketches a top-down view map of the environment and draws a path for the robot to traverse. A tablet has been used to perform laser laparoscopy using a “what you draw is what you cut” sketch control scheme of a 4-degree of freedom prototype robot (Tang et al., 2003, 2005).

In all of these cases, the use of the stylus or finger touch is limited to mouse-like emulation where the user interaction is limited to pressing buttons, moving sliders, interacting with generated maps, or drawing paths. In most cases the widgets are standard UI elements where the size and finger occlusions are not optimal. Higher level control is typically expressed in a “go here” command when coupled with a map of the area to be explored. Our tests detailed in Chapter 4 and Chapter 6 indicate that we can achieve a much richer experience through the use of gestures and UI element affordances specifically tailored for touch interaction. Additionally, we believe that these lessons and design recommendations can directly improve interfaces regardless of the touch surface size or input method.

2.4 Multi-Robot Command and Control

More recent studies have begin to explore multi-robot control through multi-touch interaction in various contexts. Kato et al. (2009) investigate using a top-down view of multiple iRobot Roomba robots in an indoor environment. This implementation shows the location of each robot, and the user drags their fingers over the multi-touch surface to manipulate a two dimensional vector field representing the robots’ path. While an interesting implementation of potential fields, this interface only addressed navigation and does not allow for individual robot tasking or higher level commands. The gesture space is also limited to drag for modifying the potential field, and single touch to clear the field.

Hayes et al. (2010) uses a tablet laptop with multi-touch capabilities for tasking simulated robots. These tasks include specifying paths, waypoints, and regions for the robot to follow and explore. Two versions of the interface were studied: one with stylus based mouse emulation and the one with full multi-touch input capabilities. Hayes et al. (2010) conducted a within subjects experiment and the findings agreed with many of the results detailed in Chapter 3 and Chapter 8. Specifically, scenario completion time, task specification time, and region specification time were all faster with the multi-touch interaction. Additionally, Hayes et al. (2010) found that the NASA TLX overall workload and frustration levels were significantly lower with the multi-touch interface.

As mentioned in Chapter 1, the investment in unmanned systems for the military has been significant. Generally speaking, UAVs are intended to increase safety and lower the man-power required to operate the aircraft. In practice, this is far from the case as detailed in (Taylor, 2006). In this study, an Air Chief Marshal describes how a “Predator A can orbit for 20 hours and requires 2 crew who operate for 8 hours each, totaling 6 crew for a single Predator.” To lessen this personnel burden, the Air Force Research Lab has been developing the Vigilant Spirit Control Station that is intended to allow a single operator to supervise and control multiple vehicles (Rowe et al., 2009). Few quantitative testing results have been released to the public, but an external review in (Taylor, 2006) indicates that the system currently has the capability to provide “support for air-to-air refueling, . . . augment operator task engagement, enhance operator situation awareness and trust in automation . . . , and provide intuitive and integrated pictorial information to support operator supervision and intervention decision making (Taylor, 2006).”

The Navy has also actively developing systems for single-person heterogeneous multi-robot command and control since 2001. The Space and Naval Warfare (SPAWAR) Systems Center Pacific has been developing a Multi-robot Operator Control Unit (MOCU) with capabilities in the land, air, sea, and undersea robot domains (Bruch, 2006). This modular architecture allows the MOCU to be modified depending on the task requirements and robot capabilities. The physical joysticks,

buttons, and screen can be field changed and the underlying software will adapt to the new input methods. Much like the Vigilant Spirit Control Station, little quantitative testing has been released to the public, but Nguyen et al. (2009) report that the system in development employs “task management including task existence and completion status, attention management using visual and auditory prompts, control-display compatibility employing similar layouts for each vehicle, and proximity implemented as adjacent or overlaid information including route, waypoints, vehicle status, and video.”

Although the nature of military development has not allowed quantitative analysis of the Vigilant Spirit Control Station and MOCU to be publicly released, screen shots and design requirements provide a wealth of information for the UI designer. These systems have been in development for many years, so their evolution and maturity allows one to infer “lessons learned” from design changes and qualitative reviews as detailed above. While they are not specifically designed for multi-touch control, their capabilities were considered when forming features for the multi-touch command and control interface in this dissertation.

2.5 3D and Immersive Telepresence

The argument for augmenting emergency response and robot control with interactive information sources is not new. Before the popularity of multi-touch tabletop devices, there was a significant interest in the use of three dimensional (3D) or immersive displays (Barfield and Furness III, 1995; Kalawsky, 2004). Minimally, these interfaces modeled the world in a 3D simulation that permitted the user to interact with objects, data sources, and other participating users. Some of the implementations placed sensors on the user and modeled their body movements with their virtual representation, or avatar, in an effort to increase the level of direct manipulation with the environment. Head movement and viewpoint management could be adjusted through head mounted displays (HMDs) that would coordinate the user’s head and neck position with the camera view in simulation. Although

the use of fully immersive and 3D world metaphors has not been entirely successful over the last few decades, it is instructive to explore this line of human-computer and human-robot interaction to ensure that positive and negative lessons can be captured.

In Gyorfi et al. (2008), the Motorola Corporation developed an application that modeled incident response in a simulated 3D world called the virtual incident command center (VICC). The argument was made that the incident command system (ICS) required all management personnel be in the same location for effective execution which made ICS vulnerable to failure. If the management personnel had the opportunity to be remote, then the response was geographically redundant and less of a threat for attack. To achieve interpersonal communications with collaborators, the VICC used a 3D command-center room metaphor. Personnel in the simulation had the ability to share documents, images, and 3D location data. The participants in the simulation were represented as 2D avatars that floated in the space of the command center simulation.

Clients connected to this simulation through standard TCP/IP connections over the Internet or private network connections. Application interfaces included standard desktop workstations, small handheld devices, and a variety of HMDs. The researchers noted that “HMDs [created] a more immersive experience, but [made] interaction more difficult.” They did not explore this limitation in any detail and there was no accompanying user study to support performance claims of the prototype. It is interesting to note that the researchers conceded that with standard video-conferencing and telepresence methods, “The ability to read body language and gestures is lost. Information sharing is also hampered by this approach. In the classic ICS methodology, the participants can open up a map, lay it on the table, draw on it, point to features, and plan with it.” The interpersonal nature and interactivity of the tabletop paper map is supported throughout this work, although it is unclear if this implementation provided the fidelity of interaction supported by the research in this thesis.

Attempts have been made to control robots through 3D interactive display

technology (Fong and Thorpe, 2001). Research by Crescenzi et al. (2009) demonstrated the control and supervision of Unmanned Aerial Vehicles (UAVs) using a touch screen and 3D stereoscopic virtual display. Although this research focuses mostly on the aspects of situation awareness and shared control of air vehicles, the use of the touch screen in a tabletop configuration in addition to the 3D display makes it relevant to this research. A desktop monitor-sized touch display was placed in a near-horizontal position facing the user. A map was displayed in the center of the screen surrounded by buttons representing various functionalities specific to the control of the UAV. The touch display, in this case, represented the teleoperation and flight path control of the UAV, while the larger projected 3D stereoscopic display provided the view from the UAV. The prototype interface was tested with 12 student pilots, and the researchers found that the “touch screen was considered a good tool to command the vehicle by means of high level commands.”

One of the most ambitious attempts in the fields of robotics and immersive telepresence has been the development of NASA’s Robonaut platform. This humanoid-like robot is extremely complex in an effort to emulate the form and function of a human torso. The argument was that a sufficiently high fidelity apparatus that can simulate human characteristics would not only be compatible with crew spaces on existing spacecraft such as the International Space Station (ISS) and Space Shuttle Orbiter (STS), but also allow the robot to use tools and apparatus designed for human astronauts. This robot system is scheduled for launch via the STS for delivery to the ISS in November of 2010. It has additionally been considered for teleoperation tasks on the Moon and Mars (Landis, 2008).

As described in (Diftler et al., 2003), the complexity of the arm and hand and need for fine motion and force-torque control place Robonaut far outside of traditional joystick-based controller design. The hand, forearm, and upper arm account for 19 degrees of freedom and 42 sensors for feedback control. To achieve comparable performance to the robot’s human counterparts and achieve a high ease of learning, a full telepresence interface was designed that used virtual-reality-based telepresence gloves and helmet to sense the position of the hand, arms, and neck

of the operator. These sensor readings are processed and sent to the Robonaut so that it may emulate the operator's pose. It is argued in (Ambrose et al., 2000) that Robonaut's pose emulation occurs fast enough that the operator can achieve high situation awareness and high proficiency in little time.

In general, 3D immersive and telepresence systems have not found the ubiquitous “killer application” that was once hoped. Multi-touch technology seems to currently exist in the same awkward infancy and time will tell if it can overcome barriers in acceptance and economies of scale. Regardless, applications like the command and control of robots will help explore and support the enhanced interaction capabilities of multi-touch devices.

Chapter 3

Performance Validation

One of the first tasks with any new input device is to establish its relative performance to other input devices that perform the same function. When surveying the literature, we were surprised to find that there appeared to be no formal performance model for large tabletop multi-touch devices. As such, we determined that this would be a good starting point from a human-computer interaction standpoint. It is important to ensure that the device and horizontal configuration will not inhibit the users' performance. This also establishes an important baseline of performance for future user interface designs.¹

Performance models allow researchers and interface designers to understand and predict human aiming performance. An example of an aiming task is the activation of a control in a graphical user interface. The most frequently applied performance model is Fitts's Law (Fitts and Deninger, 1954; MacKenzie, 1995; Zhai, 2004). Fitts's Law models movement time (MT) as the tradeoff between speed and accuracy characterized by the ratio of the movement amplitude (A) and target width (W):

$$MT = a + b(ID) \tag{3.1}$$

where ID is the Index of Difficulty of the movement, defined as

¹Portions of this chapter appear in (Micire, Schedlbauer, and Yanco, 2007)

$$ID = \log_2 \left(\frac{A}{W} + 1 \right) \quad (3.2)$$

The constants a and b are experimentally derived regression coefficients. Equation 3.2 presents the formulation of ID proposed by MacKenzie, which is the generally accepted form (ISO 2000, 1998) (Zhai, 2004) (Soukoreff and MacKenzie, 2004).

The main goals of this study were to assess the performance characteristics of digital tabletop devices and to determine if Fitts's Law holds for them. This chapter specifically explores the hypothesis that target selection is faster and more accurate on horizontal digital tabletop input devices compared to a mouse.

It should be noted that this study did not explore multi-target touch or multi-user collaboration. In the interest of establishing a lower bound of performance, we limited the interaction with the tabletop to the same single point and click paradigm used by most tabletop mouse hardware.

3.1 Hypotheses

The study sought to determine the validity of the following hypotheses:

- H1:** Target selection is faster using touch than using a mouse on a tabletop.
- H2:** Target selection is more accurate using touch than using a mouse on a tabletop.
- H3:** Fitts's law holds for touch selection on a tabletop.

3.2 Experimental Design

The experiments were conducted while standing at a laboratory workbench with a front projected Mitsubishi DiamondTouch (Dietz and Leigh, 2001) screen surface 92 cm (36.2 in) above the floor. This height was static throughout all participants and provided a relatively neutral height just above the waist of the participant. As

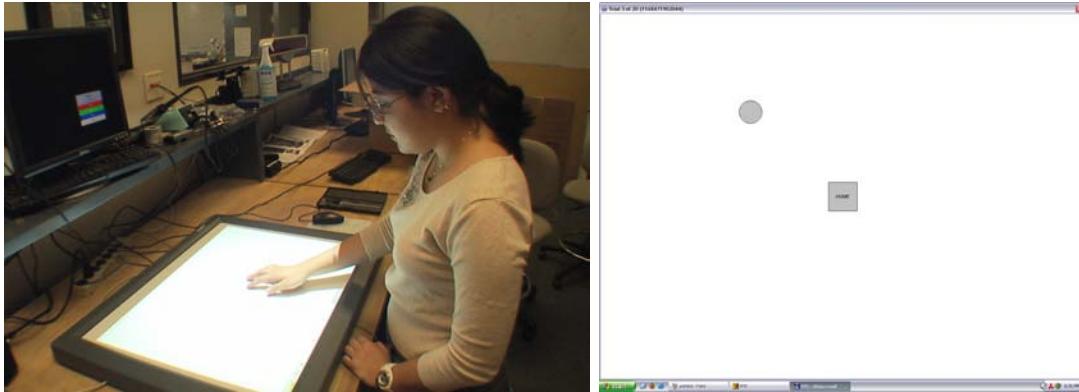


Figure 3-1: The Mitsubishi DiamondTouch digital tabletop (left) was evaluated in regards to task completion time and accuracy. A standard mouse, shown to the right of the participant, was used for comparison. The screen shot (right) shows a target presentation.

seen in Figure 3-1 this placement allowed the participant to comfortable reach to the extents of the screen. Depending on the height of the individual, this position and table height provided a 45 degree or greater viewing angle of the screen and presented targets.

The DiamondTouch tabletop was connected to a Dell Precision 360 (3.0Ghz CPU, 1GB RAM) and a ceiling mounted Optima EP737 Projector (1024×768 pixel resolution) located directly above the horizontal table top. The projected screen was at an angle of ninety degrees to the user in a flat tabletop configuration. The effective screen size on the projected surface was 610 mm (240 in) by 460 mm (181 in). The mouse used for comparison was a Dell optical mouse connected to the same computer and horizontal tabletop screen. Figure 3-1 illustrates the testing apparatus.

The experiment was conducted using the Movement Time Evaluator (MTE)² software, an open and configurable platform for Fitts's experiments written in Java (Schedlbauer, 2007). The Mitsubishi DiamondTouch SDK software (Esenther et al., 2002) was used for mouse emulation. The software allowed the participants to simply tap their fingers to emulate a left click of the mouse. Since the Dia-

²Open source software available under GNU Public License from <http://research.cathris.com/mte.html>

mondTouch cannot move the mouse cursor through movements above the tabletop, position and velocity tracking was not compared between the mouse and the DiamondTouch. This software configuration provide data and analysis for error rates and task completion time for the two devices.

3.2.1 Participants

Nineteen participants (5 female, 14 male) were recruited from the university. They did not receive any compensation for their participation. The average age for the participants was 25 years ($SD = 5.4$). All participants were experienced computer users, but had only minor experience in using the digital tabletop interface. They had normal or corrected-to-normal vision with no other physical impairments being reported. All participants were right-hand dominant in their daily computer activities.

3.2.2 Procedure and Data Collection

Before testing, participants were instructed to hit the targets as quickly as possible while minimizing errors. Any click or tap outside the target area was recorded as an error. A 1-2 minute rest period was provided between input device changes. A target acquisition trial consisted of clicking a home region at the center of the screen which started the timing and caused the home region to be hidden. The participant then selected the target. Auditory feedback confirmed successful acquisition of the target or warned of a touch outside the target area. The experiments for mouse and touch were conducted with the participants standing as shown in Figure 3-1. As the targets were positioned at various angles, a circular target shape was used in all experiments, which presented the same target width regardless of the approach angle. The participants were instructed to use only their dominant hand to eliminate two-handed interactions.

Each participant was presented with four blocks of twenty trials each. Each block varied the target size, and, within each block distance and angle to the target

were randomly assigned. Each participant saw the same sequence of targets in the same positions for each of the two input devices: mouse and digital tabletop. Therefore, the independent variables were target size, distance to the target, and input method. The dependent variables were movement time and error rate.

In keeping with the recommendations by Soukoreff and MacKenzie (2004), the experiment tested a broad range of *ID* values ($\min = 0.5$, $\max = 5$, $\text{mean} = 3$, $SD = 1$). The experiment used a land-on selection strategy, which means that the tap was recorded as soon as the finger touched the screen. Time measurements were taken at a resolution of 10 ms, the smallest granularity supported by the Sun JVM on Microsoft Windows XP (Green, 2008). Amplitudes were calculated using the Euclidean distance between the starting point and the end point of the movement. The recorded movement time was not adjusted to remove the initial reaction time. Therefore, the measured time reflects the total task time (Zhai, 2004).

To ameliorate any latent learning effects, each subject was given a set of warm-up trials before each experiment. The time of the warm-up trials was not recorded. After each block, the participants were allowed to rest. The presentation of the conditions was randomly varied.

3.3 Data Considerations and Results

The collected data contained a few outliers which were not removed from the data set as it was not clear whether they were due to the complexity of the task or the characteristics of the input device. The coefficient of determination for the correlation (R^2) was calculated using averaged *MT* values over 20 *ID* ranges. There is considerable debate over whether to use the raw data values in the correlation calculations or averaged *MT* values over fixed *ID* ranges (Thompson et al., 2004). While the use of the raw data makes the correlation results more meaningful, a few far outliers can markedly affect the correlation. Averaging the values attenuates the effect of outliers by bringing them closer to the mean, but it

may hide some effects. For instance, for finger touch, the smallest target size had a much higher selection time. When using averaged MT values, this effect may be hidden. Therefore, certain factors that significantly affect performance may not be taken into account. On the other hand, most published studies on Fitts's law report correlations based mean MT over a fixed range of ID values, so the use of the correlations obtained from the averaged data are more appropriate.

Soukoreff and MacKenzie (2004) state that obvious outliers should be removed from the calculation of ID , which they define as being farther than three standard deviations from the mean. They attribute the presence of outliers to misfires where a subject accidentally double-clicks on a target or pauses during the movement. The outliers observed in this experiment do not fall into any of these categories. Rather, they appear to be caused by the imprecision of touch input for small targets. The driver for the DiamondTouch device reports a single coordinate position to the testing software even though the probe covers much more than a single pixel on the screen. The reported position is an average of the covered pixels. Therefore, targets that are smaller than the probe often require repeated attempts before a successful selection occurs. Consequently, the trial completion time measured by MT captures the actual difficulty of the task and outliers generally represent selections of small targets. Because an overall performance model was sought, all data points were included in the analysis.

3.3.1 Task Completion Time

The average task completion time was 861 ms ($SD = 284$) for the mouse and 772 ms ($SD = 593$) for the digital tabletop. A paired t -test showed the difference of 89 ms to be significant ($t_{1519} = 6.495, p < 0.001$). As shown by a one-way ANOVA, target size is a factor in the task completion time for both devices ($F_{3,1516} = 270.31, p < 0.001$ for the mouse and $F_{3,1516} = 184.78, p < 0.001$ for the tabletop). Interestingly, for the smallest target size of 10 mm, selection on the tabletop was 182 ms slower compared to the mouse, although for all other target sizes, tabletop selection was faster. As summarized in Table 5.2, the differences in

Table 3.1: Mean task completion time (\bar{X} in ms) and standard deviation (SD) by target size and input method. Paired t -test result of input methods shown as (t).

Target Size (mm)	Mouse		Tabletop		t
	\bar{X}	SD	\bar{X}	SD	
10	1094	296	1276	896	-3.97*
20	941	219	757	371	9.66*
30	751	214	568	239	14.09*
40	661	170	488	188	15.99*
all	861	284	772	593	6.49*

* $p < 0.001$

performance across the four different target sizes were all significant.

3.3.2 Accuracy

The mean error rate across all target sizes for the mouse was 0.041 ($SD = 0.214$) compared to an error rate of 0.192 ($SD = 0.699$) for the digital tabletop, a significant increase of 370% ($t_{1519} = -8.11, p < 0.001$). As illustrated in Table 3.2, accuracy of the mouse was better for the two smaller target sizes, but essentially the same for the two larger sizes. The accuracy results for the mouse are consistent with other published studies such as MacKenzie (1995) and Thompson et al. (2004).

Table 3.2: Mean error rate (\bar{X}) and standard deviation (SD) by target size and input method. Paired t -test result of input methods shown as (t).

Target Size (mm)	Mouse		Tabletop		t
	\bar{X}	SD	\bar{X}	SD	
10	0.050	0.230	0.595	0.120	-8.51*
20	0.034	0.196	0.103	0.360	-3.35*
30	0.037	0.227	0.039	0.243	-0.15†
40	0.042	0.201	0.032	0.190	0.73†
all	0.041	0.214	0.192	0.699	-8.11*

* $p < 0.001$, † $p > 0.05$

The error rate for the digital tabletop was almost 60% for the smallest target size of 10 mm, but reached a more reasonable rate of 3.9% for the 30 mm target and 3.2% for the 40 mm target. As illustrated in Table 3.2, the differences between mouse and digital tabletop accuracy are not statistically significant for the two larger target sizes ($p > 0.05$).

Spatial variability (i.e., dispersion) of the selection endpoints about their mean is another measure of accuracy. It is calculated as the mean least square distance of the selection end points to the mean selection end point. For the mouse the mean deviation was 8.81, whereas for the digital tabletop it was 9.86, a statistically significant increase in the dispersion ($t_{79} = 3.96, p < 0.001$). This analysis suggests that touch selection is overall less precise and that it exhibits more variability leading to an increase in targeting errors.

3.3.3 Performance Models

The linear correlation between MT and ID has an R^2 of 0.98 for the mouse and 0.89 for the tabletop ($p < 0.001$). Linear regression of MT against ID results in the following Fitts's models for task completion time:

$$MT_{Mouse} = 193 + 219 \times \log_2 \left(\frac{A}{W} + 1 \right) \quad (3.3)$$

$$MT_{TableTop} = -187 + 329 \times \log_2 \left(\frac{A}{W} + 1 \right) \quad (3.4)$$

The regression intercepts are within the range suggested by Soukoreff and MacKenzie (2004), who have argued that intercepts outside the range of $[-200, 400]$ should be interpreted with caution as they might point to problems with the experimental methodology. The increased slope for the tabletop device suggest that movement time increases more rapidly as the difficulty of the task increases compared to the mouse, which is evidenced by the high movement times for smaller targets.

The accepted measure of the efficiency of input devices is throughput (*TP*), which is defined as the reciprocal of the regression slope and is calculated in bits per second (bps). ISO 9241-9 proposes that between-study comparisons of input device evaluation results should be based on throughput rather than task completion time (ISO 2000, 1998) (Soukoreff and MacKenzie, 2004) (Douglas et al., 1999). In this study, throughput was found to be 4.57bps for the mouse and 3.04bps for the digital tabletop, making the mouse more efficient by 1.53bps.

3.4 Impacts and Implications

Fitts's law was found to be a good predictor of target selection time on a horizontal digital tabletop operated in a standing posture. Consequently, hypothesis H3 (Fitts's law holds for touch selection on a tabletop) cannot be rejected. Specifically, task completion time for the tabletop was faster than the mouse in all targets except those that had a diameter of 10 mm, leading to a conditional acceptance of hypothesis H1 (Target selection is faster using touch than using a mouse on a tabletop). The mean error rate was comparable to the mouse interface for targets of 30 mm and 40 mm, but substantially higher for target diameters of 20 mm and smaller, which leads to a rejection of hypothesis H2 (Target selection is more accurate using touch than using a mouse on a tabletop).

Efficient task completion can be expected only when target elements in the user interface are larger than approximately 30 mm in size. Furthermore, due to the increased spatial variability of target selections on the digital tabletop, user interface controls should be spaced further apart to avoid false selections. Compensatory techniques can also be investigated in future interfaces, such as magnifying the anticipated target before selection (e.g. iPhone keypad).

The initial results from in this study indicate that the mouse is overall slightly more accurate, particularly for smaller targets, and more efficient as measured by throughput than the digital tabletop. However, this initial study did not address multi-finger or multi-handed input, an input method not available for the mouse.

As such, multi-touch table interaction is a viable alternative to mouse input as long as appropriate provisions for target size adjustment are made. The following design considerations will be made for future interfaces on horizontal touch surfaces:

- Target elements in the user interface must be larger than 30 mm in size.
- Task completion time will meet or exceed mouse based performance for the same tasks. As such, mouse based selection and completion models such as goals, operators, methods, and selection rules (GOMS) (Card et al., 1983) can be used to provide an upper bound on task completion.

These findings seem basic, but they are fundamental for moving forward in this research. Establishing lower and upper constraints on the interface design will ensure that we do not naively hinder performance at a low level.

Chapter 4

Multi-Touch Interaction and Robots

Regardless of the design of the multi-touch interface, it is difficult to argue against the enhanced interactivity of such a display. By removing the joystick, mouse, or keyboard from the interaction, we increase the degree of direct manipulation, thereby increasing interaction by removing a layer of interface abstraction (Shneiderman, 1983). To our knowledge, our study represents the first use of a multi-touch table with a physical agent (Micire et al., 2008). Many unexpected events occur when a system contains a moving, semi-autonomous physical object that is affecting the world. As such, we must determine if multi-touch interaction decreases the performance of systems in the real, dynamic, and noisy world.¹

A mature and well-studied joystick-based interface formed a baseline for comparison (Keyes, 2007)(Yanco et al., 2007). The University of Massachusetts Lowell (UML) Urban Search and Rescue (USAR) Interface system encompasses a wide range of robot functionality and autonomous capabilities. While leaving the visual presentation the same, this system was ported from a joystick and keyboard interface to a Mitsubishi DiamondTouch (Dietz and Leigh, 2001) with minimal modification. A description of the original joystick design and multi-touch features

¹Portions of this chapter appear in (Micire, Drury, Keyes, Yanco, and Courtemanche, 2008) and (Micire, Drury, Keyes, and Yanco, 2009b)

is provided in Section 4.1. The similarity in design enabled us to test whether we are impairing performance with the new interaction method.

This study assists in the design process by providing a detailed analysis of users' varied interaction styles. Our analysis, described in Section 4.4, sheds light on how users perceive the interface's affordances (Norman, 1988) and highlights mismatches between users' perceptions and the designers' intentions. These mismatches point towards design changes to better align users' expectations and interface realities.

4.1 Interface Description

The UML USAR interface, which has evolved as a result of several usability studies, was originally designed to test the recommended guidelines produced by Scholtz et al. (2004) and Yanco and Drury (2004) to improve situation awareness. These guidelines proposed that all USAR interfaces should include a map of where the robot has been, more spatial information about the robot in the environment, indications of the current camera position, and fuse data to lower the cognitive load on the user (Keyes, 2007).

Video Panel: The interface consists of six panels that make up the interface. The most frequently used is the main video panel. It is in the center of the interface and acts as the center of focus for the user. We observed in many studies that all users rely heavily on the main video screen and very rarely notice other important information presented on the interface (Yanco and Drury, 2004). For this reason, all of the important information, such as ranging information, is presented on or around the main video panel. The main video panel has a cross-hair overlay to indicate the current pan and tilt orientation of the video camera.

ADR Mode Panel The rear view panel, which displays the video feed from the robot's rear camera, is located to the upper right of the main video panel. This panel is placed in this location to mimic the location of a car's rear view mirror. Similarly, the rear camera's video stream is mirrored to imitate the view seen in a rear view mirror. The rear view panel is smaller than the main video screen, so the

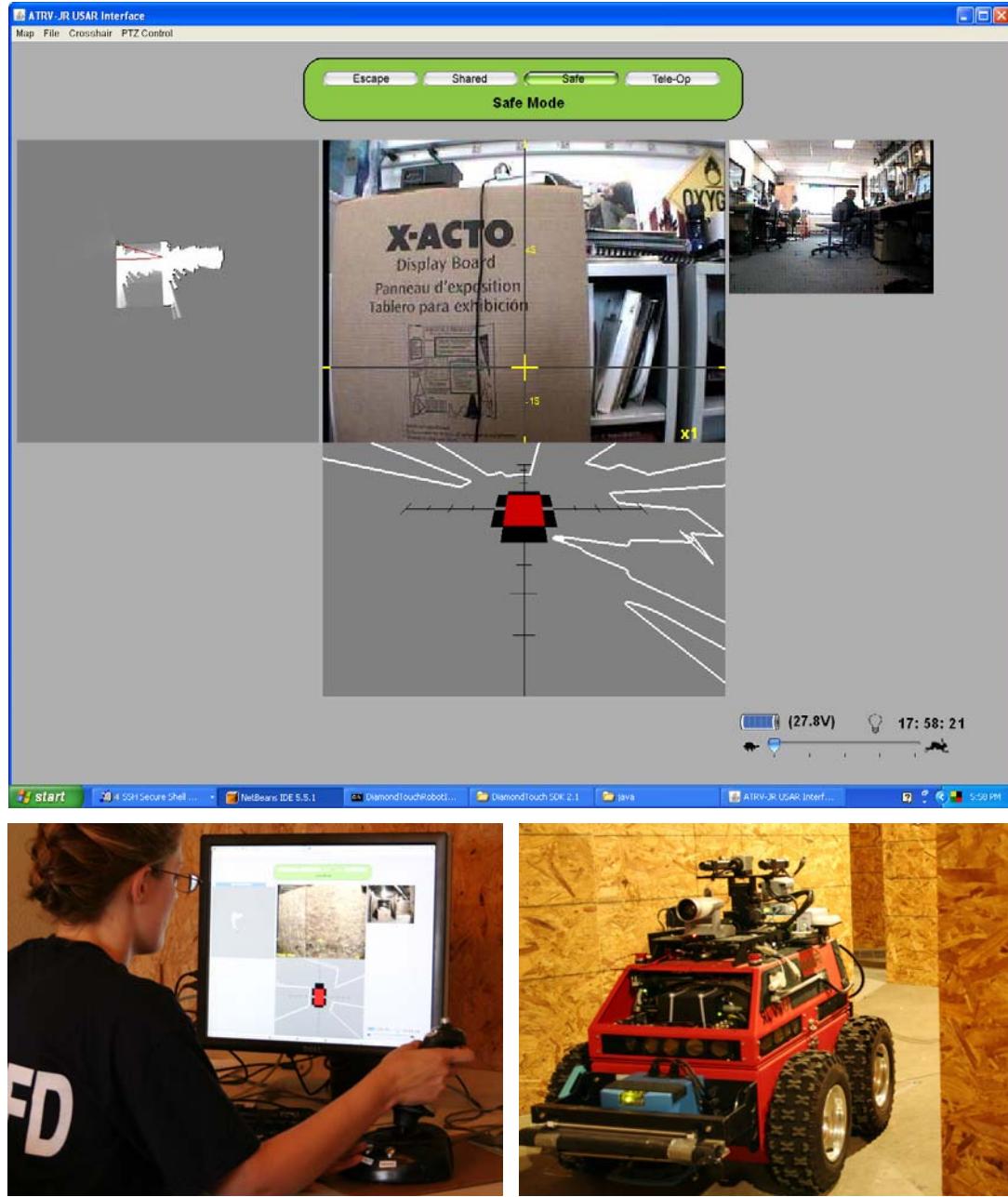


Figure 4-1: The UML USAR interface (top) is shown with a participant using the joystick configuration (lower left). This interface allows the user to operate the iRobot ATRV (lower right) through the NIST USAR course (shown in Figure 4-4).

operator does not get confused as to which one is the main camera. However, if the user wants to see a larger view of the rear camera's video, he/she can switch to Automatic Direction Reversal (ADR) mode. This mode causes the rear camera's video to be displayed in the larger main video panel, while relegating the front camera's video to the smaller rear view mirror panel. This act also reverses the driving commands as well as the ranging information displayed on the distance panel. This reversal makes driving while looking out the rear camera appear the same as driving while looking out the front video camera, except the robot is moving in reverse.

Distance Panel: The distance panel is located directly under the main video panel. This panel consists of an image of the robot's chassis, with white lines displayed around it representing distance readings from the sonar sensors and the laser range finder. There are also black tick marks on each side of the robot, each representing 0.25 meters. These markings help to give the user a frame of reference regarding how close objects may be to the robot. When a user pans the video camera, this panel rotates in the opposite direction of the pan to line up what the operator is seeing in the video with the ranging information being displayed. This panel is rendered in a perspective view by default; the operator can toggle it to a top down view.

Map Panel: To the left side of the video panel is a map. This map, which uses a simultaneous localization and mapping (SLAM)-based algorithm (Howard, 2006), is dynamically generated as the robot is maneuvered around an area. The map shows open space as white space, obstacles are depicted as black lines, and grey represents unexplored space. The map shows the robot's location as a green triangle and its trail is a red line.

Autonomy Mode Panel: The mode panel is displayed on top of the main video panel. This panel consists of four buttons that each represent one of the four autonomy modes of the system. When a mode is selected, the corresponding button is highlighted, and the background color is changed.

Status Panel: The status panel is located on the bottom right of the interface.

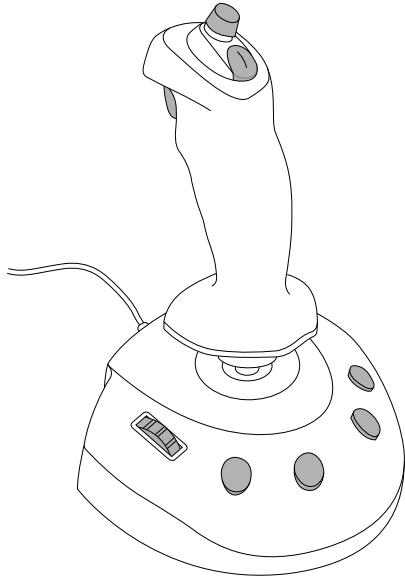


Figure 4-2: Illustration of the flight-style joystick used in the interface design.

It is the only panel that does not border the video panel; it contains information that is not as critical as the video displays and distance information. This panel contains the current battery level, whether or not the lights are on, and the robot's maximum speed indicator.

4.1.1 Joystick Interface Design

The interface uses a ten button joystick, shown in Figure 4-2 that consists of a trigger, five standard buttons, and a hat sensor (similar to a small joystick) located on the top of the joystick. To move the robot, the user must press and hold the trigger, so that if the joystick is accidentally pushed, the robot will not move. The robot is controlled by pressing the trigger and pushing the directional gimbal on the joystick in the direction of the desired motion. Full mixing of translation and rotation is provided. If ADR mode is active, forward and backward is reversed, as explained in the previous section.

Camera controls occupy all but two of the buttons on the joystick. We decided that all camera controls should be on the joystick because maneuvering the camera is the task that takes the most time after navigation. The pan and tilt actions are

controlled by the hat sensor on the top of the joystick. Directly beneath the hat sensor on the joystick is a button that will “home” the camera by bringing it back to its centered position.

On the base of the joystick are four buttons. The left-most button toggles the ADR mode. The bottom left and bottom right buttons control the zoom feature of the camera. The button to the right of the handle toggles the robot’s brake. There is also a scroll wheel on the left side of the handle that adjusts the robot’s maximum speed.

As the joystick does not have enough buttons to fulfill all the functionality of the interface, six actions have been relegated to the keyboard. Changing autonomy modes is set to buttons F1-F4. The lighting system on the robot is toggled on and off by pressing the caps lock key. Changing the distance panel from its perspective view to the top down view is accomplished by pressing the F8 key.

4.1.2 Multi-touch Interface Design

We spent a considerable amount of time ensuring that the multi-touch interface was as visually identical to the above mentioned joystick interface design as possible. The goal was to duplicate all of the functionality without creating any confounding issues in presentation or arrangement of display elements. Each of the discrete interaction elements is shown in Figure 4-3 and described below.

Autonomy Mode Panel: At the top of the interface is a rectangular panel populated with the four autonomy modes as described above. The user simply needs to tap the corresponding button to engage the requested autonomy mode. Visual feedback is provided by changing the background color of the panel depending on the mode.

ADR Mode Panel: As mentioned in the joystick interface description, the upper right of the interface shows the view from the rear camera mounted on the robot. For the multi-touch interface, the mirrored image is also the panel that selects the ADR mode. The user taps the small video panel, and it switches with the main video display. While in ADR mode, the drive control panel functions are

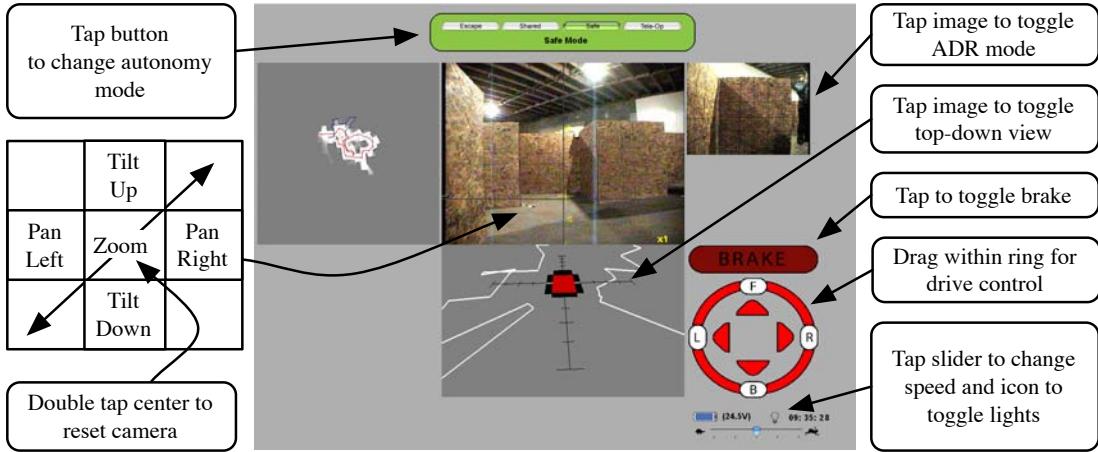


Figure 4-3: Screenshot of the multi-touch interface and guide to gestures that activate interface features and autonomy modes.

inverted so that moving forward in the camera view moves the robot backwards and vice versa.

Drive Control Panel: The drive control panel is the only visual element that was not included in the original joystick design. This panel addresses the need to duplicate the proportional velocity control of the translation and rotation of the robot. The interface panel is a visual analog to the joystick, from a top-down view. The user places their finger inside of the ring, and the relative position of their fingertip within the panel, horizontal and vertical, is translated into rotation and translation vectors respectively. The panel changes color from red to green when the user engages the control. This panel is positioned in the lower right hand corner of the interface to position it near the user and maintain the right-handed configuration of the original joystick interface. Directly above this panel is a simple button that engages and disengages the emergency stop mechanism on the robot. Visual feedback for the brake is provided through the button brightness, “lighting” and “dimming” as the brake is engaged or disengaged respectively.

Status Panel: Directly below the drive control panel is a group of icons representing the battery state of the robot, the external light state, and the time since the start of the run. The user taps the icon for the light to engage or disengage the lights on the robot. The icon turns yellow or gray relative to the

state of the lights. A slider below these icons provided a speed limiting scalar. The user taps or “slides” the control to the desired top speed for the robot.

Distance Panel: Directly below the main image, the distance panel gives a combined display of the sonar and laser range readings relative to the robot. The user can choose between a perspective display (shown in Figure 4-3) or an overhead display by tapping in the panel.

Camera Control Panel: The camera control panel allows the participant to affect all of the functions of the pan and tilt zoom cameras on the robot. As shown in Figure 4-3, the user presses and holds his or her finger on the region of the screen corresponding to the direction of movement. The left center quadrant is pressed for pan left, upper center quadrant for tilt up, etc. To zoom in, the user begins with his or her fingers in the center of the image and then rapidly expands them diagonally to the corners of the screen. This movement increases the zoom factor by two times for each motion. The user then can tap twice in the center of the image to recenter the pan and tilt and reset to a one times zoom factor.

It should be noted that, outside of the drive control panel, we made no visible changes to the interface. Despite this, the multi-touch interface was immediately able to provide more functionality than the joystick could alone. For example, the autonomy mode selection was offloaded to the keyboard in the joystick interface due to a limited number of buttons. In the case of the multi-touch interface, the buttons that were already displayed were used for this purpose. This “free functionality” was also true for the distance panel and light control.

4.1.3 Hypothesis

The study sought to determine the validity of the following hypotheses:

H1: Multi-touch interface does not degrade performance.

H2: Users experienced higher ease of learning with the multi-touch interface compared to the joystick interface.

4.2 Experiment Design

The goal of the experiment was to compare participants' performance and interaction with two different versions of the same interface: one based on a "traditional" PC and joystick, and the other based on a multi-touch table. Accordingly, we designed a within-subjects experiment so that each participant would use both interfaces.

We conducted the experiment in the Reference Test Arenas for Autonomous Mobile Robots at the National Institute of Standards and Technology (NIST) (Schipani and Messina, 2007). We used a portion of the arena that was configured as a maze of wooden panels. The panels formed corridors that required tight turns and close attention to the relationship of the robots to the wooden walls. NIST personnel placed mannequins and baby dolls within the maze to represent victims of a catastrophe.

4.2.1 Participants

Participants consisted of six people (4 men, 2 women), ranging in age from their 20's to 60+, who are members of the search and rescue community. All have used PCs for at least five years. Four considered their computer expertise to be moderate and two assessed their expertise to be at the expert level. Five participants have never previously used robots, and the remaining participant had taken part in an experiment of ours several years ago during which time he used a much different control interface. Three participants have previously used remote control cars occasionally or many years ago. Four participants never play video games and two participants play video games 4 and 8 hours a week, respectively. Five participants have previously used a joystick, with one of them self-assessing his joystick expertise as good, three as average, and one as poor.

4.2.2 Procedure

After signing a consent form, participants filled out a pre-experiment questionnaire requesting demographic information and probing their relevant experience with computers, robots, remote control vehicles, video games, and joysticks. We showed the participants what the robot looks like (Figure 4-1) and then trained them on how to control the robot using one of the interfaces. We allowed participants time to practice using the robot in a location outside the test arena and not within their line of sight so they could become comfortable with remotely moving the robot and the cameras. We then moved the robot to the arena and asked them to maneuver through the area to find as many victims as possible during a 25-minute period. We asked participants to “think aloud” (Ericsson and Simon, 1980) during the task so we could determine when participants were having trouble with parts of the interface and/or had a different mental model of how the interface works than was intended by the designers. After task completion, an experimenter asked six semantic differential scale questions. After a break, we then repeated these steps using the other robot interface.

We counterbalanced the experiment in two ways to avoid confounding factors. Three of the participants started with the joystick interface and the other three started with the multi-touch interface. Additionally, we used two different starting positions (A and B in Figure 4-4) in the arena so that knowledge of the arena gained from using the first interface would not transfer to the use of the second interface. The two counterbalancing techniques led to four different combinations of initial arena entrance and initial interface.

4.2.3 Data collection

We collected four types of data: video, logs, observer notes, and annotated maps. Besides a video of the robot’s progress through the arena, we videotaped over the shoulder of each participant to capture his/her interactions, and we mounted a video recorder pointing down at the multi-touch table. We videotaped a direct



Figure 4-4: The robot operated in the NIST USAR arena (left) while experimenters used ground truth maps (right) to record specific incidents. In the ground truth map, “Start” symbols represent the four possible starting points, “V” represents a victim location, lines represent the robot path, and circled numbers represent collisions with the arena walls.

output of the PC’s screen to record the state of that interface at all times. Custom logging software captured each time the participants changed modes, moved the camera, or activated other controls. An experimenter sat with the participant and hand-wrote observations. Finally, an experimenter following the robot manually marked its progress through the maze on a run sheet that also provided space to note when and where each bump, scrape, or “e-stop” (emergency halting of the robot) occurred. Figure 4-4 contains a reproduction of the map portion from a run sheet that shows numbers coded to specific incidents that are enumerated on the run sheet’s second page.

4.3 Results and Discussion

The two interfaces differ in at least two major ways: in ergonomics and in the degree of direct manipulation that each attains.

The differences in ergonomics can best be explained by briefly describing the necessary physical interactions. During one run, participants sat at a table containing a standard 17 inch LCD monitor, a multi-button joystick, and a

keyboard. Using the joystick necessitates pulling a trigger and moving the whole hand in the desired direction, sometimes while simultaneously activating other buttons such as the one on the top of the joystick. Participants moved a hand to the keyboard to activate autonomy mode changes. For the other run, participants sat in front of the 36 inch multi-touch surface, which was canted up at a slight angle, and extended their arms over the interface surface to activate the desired control. Thus, the nature of the movements necessary for the two interfaces differ substantially. Also, the visual perception of the interfaces differ because the same number of pixels viewed on a 36 inch table look much less “crisp” than when viewed on the 17 inch display.

Differences in the degree of direct manipulation (Shneiderman, 1983) – characterized by highly-visible, rapid, incremental, and reversible actions – have implications for the cognitive load required by each interface. A joystick is a pointing device and thus inserts a layer of indirection between the user and the interface. The participant must mentally translate from the desired robot movement to the hand movements necessary to give the commands for those movements. In contrast, the multi-touch table affords more direct manipulation: to move the camera, for example, the participant puts a finger on the video display and moves the finger in the direction of the desired motion.

Since the two interfaces make use of the same graphical elements and provide the same functionality, we hypothesized that performance using the two interfaces would be comparable. But because of the differences just described, we could not be sure. Thus, we tested this hypothesis by measuring the number of victims found, the amount of new area covered, and the number of destructive incidents incurred by participants when using each interface. Also, because of the differences described above, we hypothesized that participants might form different subjective judgments regarding each interface’s helpfulness, comfort, efficiency, ease of use, ease of learning, and pleasure/irritation level.

Table 4.1: Constructive performance in the USAR arena.

Participant	Joystick Interface		Multi-touch Interface	
	New Area Discovered (sq. ft.)	Victims Found	New Area Discovered (sq. ft.)	Victims Found
1	272	3	304	6
2	288	3	288	2
3	352	3	240	3
4	480	8	480	7
5	384	7	464	6
6	480	6	464	10
\bar{X}	376	5	373.3	5.7
SD	90.4	2.3	107.4	2.9

4.3.1 Performance

We assessed the positive, or constructive, aspects of performance based on measuring the number of victims found and the amount of new or unique territory the robot covered while traversing the arena. These measurements are related because it is difficult to find additional victims if the operator is not successful in maneuvering the robot into previously unexplored areas. Table 4.1 shows the participants explored an average of 376 square feet ($SD = 90.4$) and found an average of 5 victims ($SD = 7.3$) when using the joystick-based interface. The multi-touch interface shows remarkably similar results: the participants directed robots to 373.3 square feet ($SD = 107.4$) of territory and found 5.7 victims ($SD = 2.9$). Thus, there is no significant difference in the constructive performance of the two interfaces.

We also assessed the negative, or destructive, aspects of performance. Damage to the robot may delay or curtail real-life rescue operations, and damage to the robot's surroundings may result in causing unstable structural members to fall and injure trapped victims. We categorized the destructive incidents as pushes (the robot moves an obstacle away from its normal position), scrapes (some part of the robot brushes up against an obstacle), bumps (the robot impacts an obstacle),

Table 4.2: Number of destructive incidents in the USAR arena.

Part.	Joystick Interface				Multi-touch Interface			
	Pushes	Scrapes	Bumps	E-stops	Pushes	Scrapes	Bumps	E-stops
1	1	1	1	0	0	0	0	0
2	5	5	20	6	2	0	4	3
3	0	0	1	0	11	0	1	6
4	0	0	0	0	1	1	0	0
5	0	0	0	0	6	2	0	6
6	1	0	1	0	1	0	1	1
Average	1.2	1.0	3.8	1.0	3.5	0.5	1.0	2.7
Std Dev	1.9	2.0	7.9	2.4	4.2	0.8	1.5	2.8

and e-stops (experimenters fear severe damage will occur if operations continue and so halt the robot).

Table 4.2 contains the numbers of destructive incidents for the joystick and multi-touch interfaces. Note that the numbers vary widely: the standard deviations in each case are larger than the average values. While there are more scrapes and bumps using the joystick interface and more pushes and e-stops with the multi-touch interface, none of the differences are significant. (Paired, two-tailed *t*-tests with five degrees of freedom result in *p*-values of 0.32, 0.64, 0.33, and 0.31 for pushes, scrapes, bumps, and e-stops, respectively.) Thus, we confirmed that there was no difference in constructive or destructive performance when using the two interfaces as they are currently designed.

Note that the interface design was originally developed with the joystick in mind and has previously gone through multiple iterations as a result of earlier user testing (see (Keyes, 2007)). Now that we know that performance is not degraded by the act of porting the interface to the multi-touch table, we can optimize the design for use with multi-touch interaction based on incorporating what we learn from participants' subjective feedback and a detailed understanding of how they interacted with the interface. Consequently H1 (Multi-touch interface does not degrade performance) cannot be rejected.

4.3.2 Subjective Assessment

To get a first look at participants' preferences, we asked them six semantic differential scale questions. Using a scale of one to five, we asked how they would rate each interface along six dimensions: hindered in performing the task/helped in performing the task, difficult to learn/easy to learn, difficult to use/easy to use, irritating to use/pleasant to use, uncomfortable to use/comfortable to use, and inefficient to use/efficient to use.

Prior to the experiment, we conjectured in H2 that participants would find the multi-touch interface easier to learn and use and to be more efficient. The rationale for the ease of learning is that the controls are more dispersed over the table and incorporated into the areas that they relate to, as opposed to being clustered on the joystick where users must remember what motions and buttons are used for what functions. The predictions for ease of use and efficiency spring from the postulation that an interface with a higher degree of direct manipulation will be easier and faster to use.

Table 4.3 shows that the multi-touch interface scored the same or higher on average in all categories, although four of these categories evidenced no statistically significant difference. We found weak significance using a paired, 1-tailed *t*-test for ease of learning ($p = 0.088$, dof=5) and efficiency ($p = 0.055$, dof=5), and assert that it is likely we would have attained true significance if we had had access to several more participants.

We believe that the scores given the multi-touch interface for ease of use and irritating/pleasant to use suffered because of several implementation problems. Sometimes the robot did not receive the "recenter camera" command despite the fact that the participants were using the correct gesture to send that command, requiring the participants to frequently repeat the re-centering gesture. At other times, the participants attempted to send that command by tapping on the very edge of the region in which that command could be activated, so sometimes the gesture was effective and at other times it failed; it was difficult and frustrating

Table 4.3: Participants’ subjective assessment.

Semantic differential scale Scale range 1 / 5	Joystick		Multi-touch	
	\bar{X}	SD	\bar{X}	SD
Hinder / Help	3.7	1.2	4.2	0.4
Difficult / Easy to learn	4.7	0.5	5.0	0.0
Difficult / Easy to use	3.5	1.4	4.2	1.2
Irritating / Pleasant	2.8	1.3	3.7	1.2
Uncomfortable / Comfortable	3.8	1.0	3.83	1.0
Inefficient / Efficient	3.3	1.2	4.33	0.8

for the participants to understand why the failures occurred. Additional visual feedback may have helped the participants to know if their command gesture registered or not.

It was not always clear to participants how to form the optimal gestures to direct the robot’s movement. We discuss what we mean by this and also characterize a number of gesture styles used by the participants in Section 4.4.

4.3.3 Learnability Assessment

Because differences in semantic differential scale scores for ease of learning were on the edge of significance, we looked for other supporting or disconfirming evidence. We noted that participants asked questions about how to activate functions during the runs, which we interpreted as indication that the participants were still learning the interface controls despite having been given standardized training. Accordingly, we investigated the number of questions they asked about each system during the runs as well as the number of times they showed uncertainty in finding a particular function such as a different autonomy mode. We found that five of the six participants asked a total of eight questions about the joystick interface and one participant asked two questions about the multi-touch interface ($p = 0.072$, dof = 5 for paired, 1-tailed t -test). This result, while again being on the edge of significance due to the small sample size, tends to support the contention that the multi-touch interface is easier to learn than the joystick interface. Our data appears

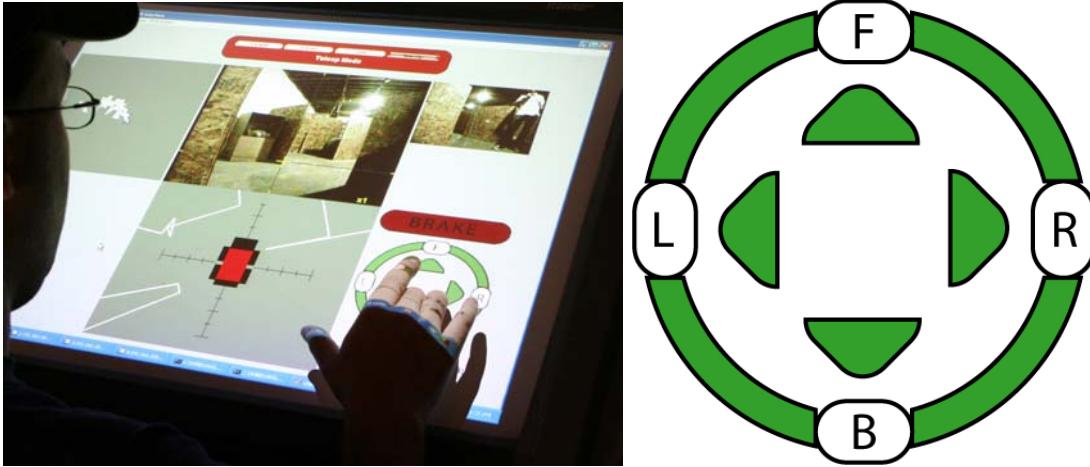


Figure 4-5: Shoulder (left) and close (right) view of the drive control panel, providing control of translation (vertical) and rotation (horizontal).

to support H2, but would need more participants for statistical significance.

4.4 Interaction Characterization

We concentrated on the camera and driving controls when characterizing the approaches participants used with the multi-touch interface. The other controls, such as to turn the lights on and off, required only simple tapping motions that were easily mastered by participants. In contrast, the two movement controls involved more degrees of freedom and, in the case of the moving the robot (versus the camera), with variable speeds. As described earlier, the camera movement is controlled by directly touching the main video panel, and the robot movement is controlled by touching a movement control panel that looks like a cross between a top-down view of a joystick and a steering wheel. Figure 4-5 depicts a close-up of this control mechanism.

We then looked for large-scale patterns of movement. We noted that participants tended to work using one of two mental models. One model holds that movement and speed are controlled together: the location of the finger with respect to the origin or center of the movement control panel determines both future direction and speed, with speed increasing with the distance from the center of the panel.

We call this the “proportional velocity” model. The other model, which we call the “discrete velocity” model, states that the user expects to control direction of movement independent of speed. There are two major refinements to these models: when participants confine their gestures to a cross-like area consisting of up-down and side-to-side motions (which we term “on-axis” movement) and when they make gestures outside of these x - and y -axes (which we term “off-axis” movement). Finally, there were two other classes of movement that occurred sufficiently frequently to warrant their own categories: “trackpad”-type movement and “ring”-type movement. Trackpad movement is reminiscent of how users work with trackpads: with short, repetitive motions. Ring movement occurred along the steering-wheel-like ring that formed the outer circle for the movement control panel.

Once we identified these patterns, we reviewed the video to broadly characterize each participants’ interaction with the multi-touch interface (described below). We noted that three participants had largely orthogonal approaches that, when taken together, provided a good cross-section of mental models and responses to the interface. Accordingly, we isolated the data from these participants for further, more detailed analysis. We prepared for the analysis by further refining the models of interaction described in the previous paragraph into categories of actions that were described precisely enough to be able to match instances of participants’ gestures against them to code each of their actions. The detailed rules for coding each gesture can be found in the Appendix A. To ensure standardization and reproducibility of the data analysis, we computed Cohen’s Kappa statistic for two coders and found very good agreement: $\kappa = 0.84$ after chance was excluded ($\kappa = 0.88$ if chance was not factored out).

The patterns and coding showed that every participant exposed some base assumption for which we had not accounted in the interface design. Through the detailed post-hoc data analysis, we noticed each user seemed to develop his or her own individual style when interacting with the multi-touch interface. In some cases, these variations helped the robot move through the course and identify

victims. In other cases, the style variants did not hinder their performance enough for users to notice a degradation of control or interactivity. Regardless, we noted these “nuggets” of human and robot interaction and analyzed them qualitatively and quantitatively. To illustrate these user-developed interaction styles, we provide a narrative of each participants’ interactions.

4.4.1 Participant 1

This participant had a continuous, flowing movement on the drive control panel. Exclusively using his middle finger on his right hand, he only made use of proportional control. The finger movements can be best described as a continuous line that began at the origin or middle of the panel and then curved smoothly throughout the control panel. The participant appeared to grasp the concepts of proportional and mixed-axis control due to his ability to not only adjust the speed of the robot’s movements, but also to mix translation and rotation constructively. Mixed “analog” movements likely indicate an understanding that the drive control panel was a direct analogy to the joystick. Besides using proportional movement 100% of the time, 55% of his movements were in off-axis areas of the drive control panel.

Interestingly, the participant insisted on using this continuous motion on the camera control panel even though an experimenter explained to him at the beginning of the run that the camera was controlled through discrete grid “buttons” on the panel, as shown in Figure 4-3. Although the buttons caused discrete camera movement, the participant held his fingers to the surface and moved them continuously in the direction of desired motion as he would in a proportional mode control panel. The participant continued this action throughout the run even though the camera provided absolutely no proportional control actions as feedback to the user. Fortunately, this continuous movement did not negatively affect the camera control buttons, so the participant did not appear to notice any unanticipated reactions from the robot. This example reinforces the often-referenced design principle of consistency (see (Nielsen, 1993)): in this case, that

the control panels should all adopt the same movement analogy.

4.4.2 Participant 2

The second participant chose several interaction methods throughout her run and provided the widest variety of unexpected interaction methods. She began her movements in the center of the control panel and quickly moved to the top or bottom of the control surface indicating an initial understanding of proportional acceleration. Interestingly, she never made movements off of the vertical axis until she was at the top or bottom of the control surface. She would then trace the outer ring of the control surface with her finger and repeat this action approximately every two seconds. It was only when she rotated her wrist slightly that we realized the incorrect assumption she was making. The participant was attempting to “turn” the outside ring of the control surface like a steering wheel in a automobile.

After approximately five minutes, an experimenter explained that the outer ring was not a steering wheel and restated that the robot could be rotated by moving to the left and right components of the control panel. The participant acknowledged that she understood and placed her finger correctly on the edge of the control panel to rotate the robot. Rather than holding her finger constantly on the control surface and moving in a constant motion as before, she began tapping the control panel rapidly in the desired direction. The tapping became her preferred mode of interaction for the rest of the run, accounting for 89% of her drive control velocity movements. These “button” movements were sometimes very rapid, exceeding four taps per second in some cases. Strangely, this tapping caused the robot to visibly “bounce” the video image since it was receiving commands to translate, rotate, and brake in rapid succession. The participant did not appear to notice this effect.

4.4.3 Participant 3

The third participant began his run with concise and deliberate motions on the drive control panel. He would begin his motions in the center of the movement control panel and then move in a straight line to the desired translation and rotation position. In this respect, he seemed to grasp the proportional control aspects of the interface. Unlike the previous two participants, he would lift his finger immediately and restart the motion for every subsequent robot movement. This created an unexpected mix between the proportional velocity control seen in Participant 1 and the discrete control bias seen in Participant 2. After approximately five minutes, the hybrid proportional and discrete finger movement began to resemble a “trackpad” movement that one might make with modern laptop mouse control surfaces. This finger action with a mouse would also be equivalent to directly translating a map in an application such as Google MapsTM or the page surface in Adobe® Acrobat®. In this way, the participant seemed to want to “push” or “drag” the robot in the desired direction, but in small steps instead of continuous button presses or proportional control. Similar to Participant 2, this approach created a noticeable bounce in the video display for the end of every finger movement, but the participant did not seem to have a negative reaction to this phenomenon. Even more interestingly, this persistent “trackpad” movement did not manifest itself in any interaction with the camera pan and tilt control. The participant pressed very deliberately at the sides of the video control display and interacted with the video control panel in a way intended by the system designers.

4.4.4 Participant 4

This participant appeared to have a very natural and smooth approach to the drive control panel. The participant would begin in the center of the panel and then proportionally accelerate to the desired speed. It was only through the post-hoc analysis that we noticed a very subtle technique being used. The participant used multiple fingers, much like a piano player, to shorten the amount of area that his

fingertips were required to transverse. For example, if the middle fingertip was at the top of the control, indicating 100% forward translation and 0% rotation, and he wanted to switch to a 100% left rotation, he would just lower his index finger. Upon making contact with his index finger on the left side of the drive control panel, he would slowly raise his middle finger and allow drive control to transfer to the index finger. This finger movement had the unintended effect of providing very smooth transitions in what would otherwise have been a “button” style non-proportional acceleration. This technique was mixed with standard proportional mode control, although his fingers were switched at seemingly random times. Like Participant 1, this participant insisted on using proportional control of the camera control panel even though discrete control was the only method described by experimenters and demonstrated by the robot’s pan-tilt unit.

4.4.5 Participant 5

The participant began the run with proportional acceleration but after two minutes of the run he began pressing the inner directional triangles exclusively. His interaction with these buttons was a mix of proportional and discrete velocity control, but one interesting effect emerged. Regardless of the control method, he never moved outside of the circular boundary created by the outside of the triangular button images. This artificial boundary meant that the robot never accelerated to full translation or rotation at any time during his run. Like Participant 4, he used multiple fingers to activate the drive control panel but maintained very discrete finger contact with the triangular buttons. He did not perform any of the subtle “mixing” of multiple fingers used by Participant 4. Although Participant 5 did not take advantage of the proportional control of the drive control, his discrete button presses allowed him to interact with the camera control panel without issue.

4.4.6 Participant 6

Immediately upon starting her run, the sixth participant established a clear style that used only the vertical and horizontal axis. She would begin in the center of the control panel and then quickly and deliberately move to the outer ring, establishing 100% translation or rotation, but only one at a time. The post-hoc analysis confirmed this, as she showed 100% of her movements on axis, 76% ended at the outer ring, and 76% of these were proportional velocity commands. She would regularly switch fingers, although no pattern could be detected. Her hand posture was muscularly tight and she held her non-used fingers high like a pianist or touch typist. Another interesting aspect was her interaction with the camera control. She would only touch the edge of the image even though she had been shown that the pan and tilt control buttons were much larger. In fact, there were many accidental triggers of surrounding panels like ADR mode and the distance panel view. This finding indirectly reinforces the design criterion that the borders between panels should minimally be the width of the participant's fingertips to avoid accidental interference.

4.5 Impacts and Implications

A joystick interface limits the user to a relatively small set of interaction possibilities. Digital buttons, analog gimbals, and analog sliders are the three common modes of input. The multi-touch surface is quite different, allowing for almost limitless interaction methods on a 2D plane. Where the joystick limits the user through mechanical and physical constraints, the multi-touch surface serves as the “blank canvas” on which control surfaces are dynamically created. However, the flexibility and freedom of the interface also presents a problem for the designer. Namely, the designer must carefully choose control methods that give extremely clear affordances and appropriate feedback to the user. Users are accustomed to haptic feedback, such as spring loaded buttons and gimbals, and auditory feedback, such as clicks, even from a non-force-feedback joystick controller.

In robotics, the term “emergent behavior” is used to describe unintentional or surprising combinations of behaviors or interactions with the environment. These emergent behaviors are unintentional artifacts that may or may not contribute to the desired outcome of the robot’s task. During user testing, we found that the novelty of the multi-touch surface created a catalyst for many “emergent interactions” that were not planned or anticipated by the system designers. Although each participant was trained on the interface in the same way, they adopted their own interaction styles borrowed from various devices in the world. While the system designers intended the interface to evoke a joystick and button affordance, the participants also demonstrated motions similar to those they would use with mouse track-pads, piano keys, touch-typing, and sliders.

The gestures used by our participants tells us that we need to revise the design to better align perceived affordances and actual functionality. Since it was clear that participants would bring biases from from disparate learned phenomena such as automobiles or consumer electronics, the next natural step was to develop a test to expose these biases so that we can engineer the next interface to maximize the ease of learning. This user-centered design is described further in Chapter 7.

Chapter 5

MultiTouch Joystick Emulation

For robots used for explosive ordinance disposal (EOD) and search and rescue (SAR), there are several ways to control the movement of the platform and all of the various sensors, effectors, and lighting. The most common method involves a mix of a large number of joysticks, switches, and dials that each manage some degree of freedom or functionality. Operators will often employ a technique called “chording” to cope with this situation. In the same way that a piano player will use multiple finger positions to create a harmonic chord consisting of multiple complimentary notes, the robot operator will, at times, use all fingers available to manage complex and coordinated movement of the robot.

For example, while using the successful Foster Miller TALON EOD robot (Foster Miller, 2010), an operator may use his or her right thumb on a two degree of freedom joystick to drive the robot left, right, forward or backwards. The left index finger is used to rotate a dial that scales and governs the speed of the robot. The robot will need to move rapidly when traveling down range, but once it is close to the bomb or victim, it needs to move very slowly and deliberately. As seen in Figure 5-1 around the drive controls, several switches are toggled with the middle, ring, or little finger to control light intensity, laser range sighting, and other on-board systems. As complex as this arrangement may seem, a cluster of almost identical controls are situated for the right hand where camera control, manipulator control, disruptor engagement, and sensors can all be manipulated.



Figure 5-1: The iRobot® Packbot® Hand Controller (left front) is an adapted Sony Playstation® style controller. The Packbot® EOD Operator Control Unit (left rear) places six degrees of freedom on each hand, requiring the operator to manage 12 degrees of freedom and 12 context sensitive push buttons for typical operation. The Foster Miller TALON robot controller (right) is operated through three two degree of freedom joysticks and a large array of switches, selectors, and dials. (*Photos courtesy of iRobot Corporation and the National Institute of Standards and Technology.*)

Although there is a large variety of robot configurations and capabilities in the field, it is not uncommon to see a robot operator managing as many as ten or more degrees of freedom for a single robot platform.

What is different between this domain and many other multi-touch applications is that this robot control interface *mandates* the use of multiple hands and fingers simultaneously. The application must work reliably and repeatedly because both EOD and SAR place the interface in a mission critical context where human lives may hang in the balance. It has only been recently that technological advancements and economies of scale have allowed multi-touch capable devices to be considered in these difficult and mission-critical field domains.

In the course of re-evaluating the joystick controller described in Chapter 4 and designing the implementation in Chapter 6, we recognized the need for a fast and highly robust algorithm for hand detection and finger registration. Once this algorithm worked sufficiently well for our testing purposes, we then

continued building the robot controller described in Chapter 6 that provided performance and ergonomic characteristics that met or exceeded the performance of standard physical joystick based controller systems. This chapter explores this design process and the verification of the performance of the hand detection and finger registration. We call our system the **dynamically resizing, ergonomic, and multi-touch controller**: the “DREAM Controller.”

5.1 Design Approach

One approach to user design is to borrow from outside experiences and interfaces that the user has already encountered. This design method not only helps with ease of learning, but may also exploit muscle memory that the user has developed over time while using the other interfaces. A popular controller paradigm established in the late 1990’s by the Sony Playstation® and the Microsoft® XBox® for video games used a dual-thumb joystick design that allowed both of the thumbs to manipulate four degrees of freedom (two on each thumb) and for various digital buttons and analog pressure sensitive buttons to be incorporated. A survey of popular first-person games showed the most common mapping involved placing camera movement (look) on the right thumb and placing character movement (run and strafe) on the left thumb. Fortunately, the capabilities of real robots closely mirror those of virtual game characters: cameras can look left, right, up, and down, while the robot can drive forward, backward, turn left, and turn right.

Coupling the ergonomics of the hand with the familiarity of the dual-thumb joystick paradigm, we developed several paper prototypes (Snyder, 2003) to determine the feasibility of function and comfort for the user. After several revisions, the multi-touch joystick design shown in Figure 5-2 was chosen as the best candidate for further software development. Rather than forcing the close left and right hand positions, as in the case of a physical game controller, we decoupled the left and right hands so that the user could maintain all of the functionality of the original dual-thumb joystick design while allowing independent hand movement to any

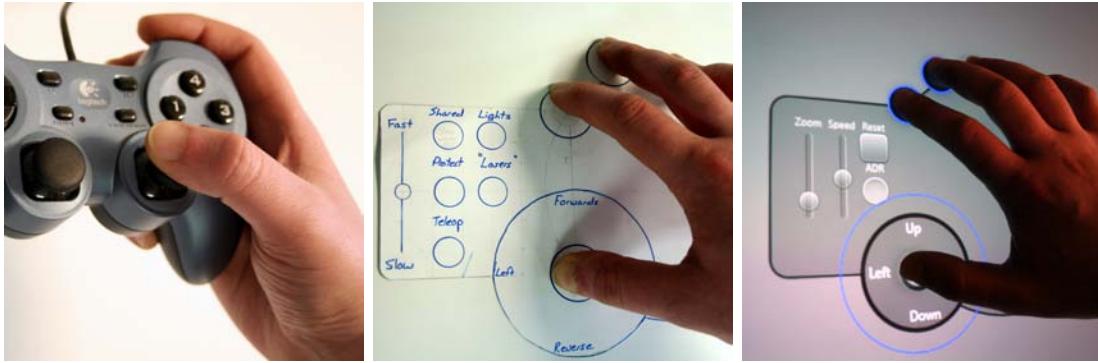


Figure 5-2: A dual-thumb joystick modeled after the Sony Playstation® controller (left) was used to inspire the design of a paper prototype (center) that was selected for the design of the multi-touch DREAM Controller (right) on the Microsoft Surface.

position on the screen.

A fundamental design goal of this interface was to allow untrained users to quickly experiment with the touch surface and realize that the interface would conform to them reliably. Users only needed to place their five fingers on the tabletop, then the algorithm would then do the difficult task of re-arranging, re-sizing, and rotating the controller for that individual hand placement.

In his book “Everyware: The Dawning Age of Ubiquitous Computing,” Adam Greenfield discusses this type of natural interaction and ad-hoc user experimentation very succinctly.

“People figured out how to do that by themselves, without some designer having to instruct them in the nuances... The more we can accommodate and not impose, the more successful our designs will be” (Greenfield, 2006).

This philosophy was adopted very early in the design of the multi-touch controller. We should allow the user to make mistakes throughout the learning stages of interaction. Over time, the design should intuitively lead the user to a working and individually tailored solution for their specific ergonomics and style of interaction.

5.2 Related Works

The algorithms used in most multi-touch interfaces do not consider which finger is touching the contact surface. For most applications, this design is preferred since users have been shown to regularly use any finger or combination of fingers for interface operations like dragging, selection, and zooming (Wobbrock et al., 2009; Micire et al., 2009a). However, in our design of a multi-touch controller, registration of the individual fingers and, consequently, the identification of the hand as a right or left hand was needed. The thumb, index, middle, and little finger were vital to the position, orientation and sizing of the joystick before any further interaction occurred.

Agarwal et al. (2007) developed a method of accurately finding fingertips and detecting touch on regular tablet displays using an overhead stereo camera. In their paper, they describe a machine learning approach in which a classifier was trained to identify points at the end of fingers (“tip point”) based on shape and appearance given overhead images with the backgrounds removed (Agarwal et al., 2007). Individual finger tips are identified when several tip points are detected in a cluster. The researchers have also investigated machine learning for hand poses which does not take into consideration their fingertip detection (Izadi et al., 2007). While these approaches are robust and able to find fingers using overhead cameras, it is not clear that if fingers themselves were identified as thumb, index, etc. The Surface is able to easily find finger tips and return the centroid points, so the finger tip detection problem was solved for us by the hardware and software provided with the device. Instead, our algorithm needed to specifically label the fingers relative to the anatomy of the hand.

Wang et al. (2009) developed an algorithm for detecting orientation of finger pads for camera-based multi-touch devices. First, an ellipse is fit to the contact region. The touch is classified as an “oblique touch” (as opposed to a vertical touch from a finger tip) if the area of the contact region is at least 120mm^2 and if the ratio of the major and minor axes of the ellipse is greater than 120% (Wang et al.,

2009). To determine the directed orientation of the contact region, the contact center is tracked as the user puts his or her finger on the surface. The orientation is derived from the angle of the vector from the contact center in the previous frame through the current frame which points away from the user’s palm. This algorithm has been shown to work well for individual fingers. However, for our DREAM Controller, we needed to be able to uniquely identify finger types. Finger orientation for multiple fingers can be used to help determine if the fingers are on the same hand, but this does not necessarily indicate handedness. Additionally, people have varying flexibility to spread their fingers (e.g., a pianist or a person with arthritis), and it is difficult to robustly identify fingers based directly on contact orientation when this surface area may change dramatically while on the tips of the fingers.

Dang et al. (2009) developed an algorithm to map fingers to unique hands also based on ellipses created by fingertip contact regions. As described in the previous approach, the researchers first detect the individual finger orientations using the major and minor axes of the ellipse and the angle of orientation of the major axis vector. Their algorithm then projects the major axis vectors of each of the fingers backwards towards the palm such that the vectors intersect. Given these finger orientations and vector intersection points, the algorithm ensures that first the maximum distance between all points is less than 10.55 inches and adjacent points must be less than 3.5 inches apart. The algorithm then checks for the intersection point of two finger’s orientation forward vectors (i.e., parallel in the same direction, parallel in opposite directions, pointed towards each other, and pointed away from the other). Next, the algorithm examines the interior angles of fingers’ orientation backward vectors to ensure that they are less than 45 degrees for adjacent fingers. To accommodate finger touches on opposite sides of the surface, the line from the backwards intersection point of two fingers must be less than 14.06 inches. Finally, the algorithm maintains temporality of finger IDs to allow accurate identification for when hands move close together. This combination of heuristics allows for detection of fingers mapped to a unique hand even when not all of the fingers from

a given hand are placed on the surface. While this approach is robust in certain hand positions, the reliance on contact region ellipses makes it a difficult choice when we expect a portion of our users to use the tips of their fingers while learning the system’s behaviors.

Matejka et al. (2009) successfully emulated a mouse on a multi-touch platform using simplistic and elegant heuristics for finger tracking. In their paper, they describe a method that uses four primary methods for interaction: chording, side, distance, and gesture. At the foundation of each of these interaction methods, basic finger tracking was needed to determine if the user was pressing any combination of three virtual mouse buttons. Rather than performing explicit finger identification, they constructed a state machine that used timing for registration of the initial tracking finger or fingers. Once the tracking finger was identified, subsequent finger contact was measured in pixels to the right or left. For example, the “Distance Technique” defines a short (< 150 px), medium (150-250 px), and far (> 250 px) distance to the right of the index finger for activating the left, middle, and right button (Matejka et al., 2009). While this method is not computationally expensive, it makes the base assumption that the user knows to begin with the index finger before any subsequent finger contact. It also assumes that the size of the user’s hands conform to these static offsets. Since explicit finger identification, handedness, and automatic sizing was needed for our application, the technique in this research was not appropriate despite other benefits in its design.

5.3 Engineering Criteria

Engineering criteria for the DREAM Controller were as follows: learnability, efficiency, memorability, error prevention and recovery, and satisfaction. These were derived from Nielsen’s five usability attributes (Nielsen, 1993). Specifically, our design attempted to deliberately incorporate these characteristics in the following ways:

Learnability: “The system should be easy to learn so that the user can rapidly start getting some work done with the system” (Nielsen, 1993). As previously mentioned, the design can exploit learning effects by borrowing from past user experiences or experiences that the user may have observed in passing. The ubiquity of the dual-thumb joystick design implies that a large number of users will already know how to use the DREAM Controller. The controller would also be labeled explicitly regarding the functions that each of the components provide. Buttons, sliders, and control surfaces will have each of their function directly displayed on or near the control itself. This labeling is a positive aspect of the software-based controller when compared to physical devices that can only generically label buttons as “A”, “1”, “Start”, or use special iconic characters such as “ Δ .”

Efficiency: “The system should be efficient to use, so that once the user has learned the system, a high level of productivity is possible” (Nielsen, 1993). The user is given four degrees of freedom and an array of buttons and sliders to choose from. Any of these functions can be used sequentially or simultaneously, so efficiency can increase over time. Much like gamers who can achieve very high levels of efficiency on gaming consoles that use the dual-thumb joystick paradigm, we believe that this design should afford the same efficiency characteristics.

Memorability: “The system should be easy to remember, so that once the casual user is able to return after some period of not having used it, without having to learn everything all over again” (Nielsen, 1993). Since using the left joystick for robot movement and right joystick for camera movement is ubiquitous in game design, the users will effectively be training for our system while using most first person games. This constant reinforcement and coordinated movement should provide significant reinforcement for recall on the DREAM Controller’s usage. As mentioned in the discussion of learnability, all of the functions of the controller will also be clearly labeled on the controller graphic for quick recall if needed.

Error prevention and recovery: “The system should have a low error rate, so that users make few errors during the use of the system, and so that if they do make errors they can easily recover from them. Further, catastrophic errors must not occur” (Nielsen, 1993). The user only needs to lift his or her hand from the touch surface to recover from an error. In the case of the robot’s movement, the robot will stop immediately and set the brakes for safety when the hand is lifted. Errors related to hand detection and finger registration may also be a problem. We specifically designed the algorithm for hand detection and finger registration to be fast and responsive to the user. This algorithm allows the user to rapidly experiment with different hand and finger configurations to determine what works best for himself or herself and minimizes errors.

Satisfaction: “The system should be pleasant to use, so that users are subjectively satisfied when using it; they like it” (Nielsen, 1993). A system designer would be hard pressed to find a more satisfying source of inspiration than one that is used to entertain people for hours at a time. Our attempts to model the robot control after first person game control not only affords all of the characteristics above, but most users are quick to recognize the similarity and enjoy moving the robot through the world with as much control and fluidity as they achieve in virtual game play. This is not to say that the real-world movement of the robot is without the “problems” of inertia, friction, and other physics, but from the user’s perspective, he or she is working with a familiar interface that has brought him or her entertainment and satisfaction in prior experiences.

Ergonomic engineering criteria were also taken into consideration during the design process. We engineered the controller with a significant respect for the resting poses for the human arm, wrist, and hand. The paper prototypes in the early design process helped minimize flexion (decrease of angle), extension (increase of angle), and pronation (downward rotation) of the muscles in the wrists and fingers based on recommendations found in Saffer (2008). We also considered

that the movements of the thumb, index finger, and little finger have been shown to have much more individualized movement characteristics than the middle or ring fingers (Hager-Ross and Schieber, 2000). In particular, the movement of the thumb for managing the two degrees of freedom for robot movement and camera positioning must be appropriate for accurate and long-term use.

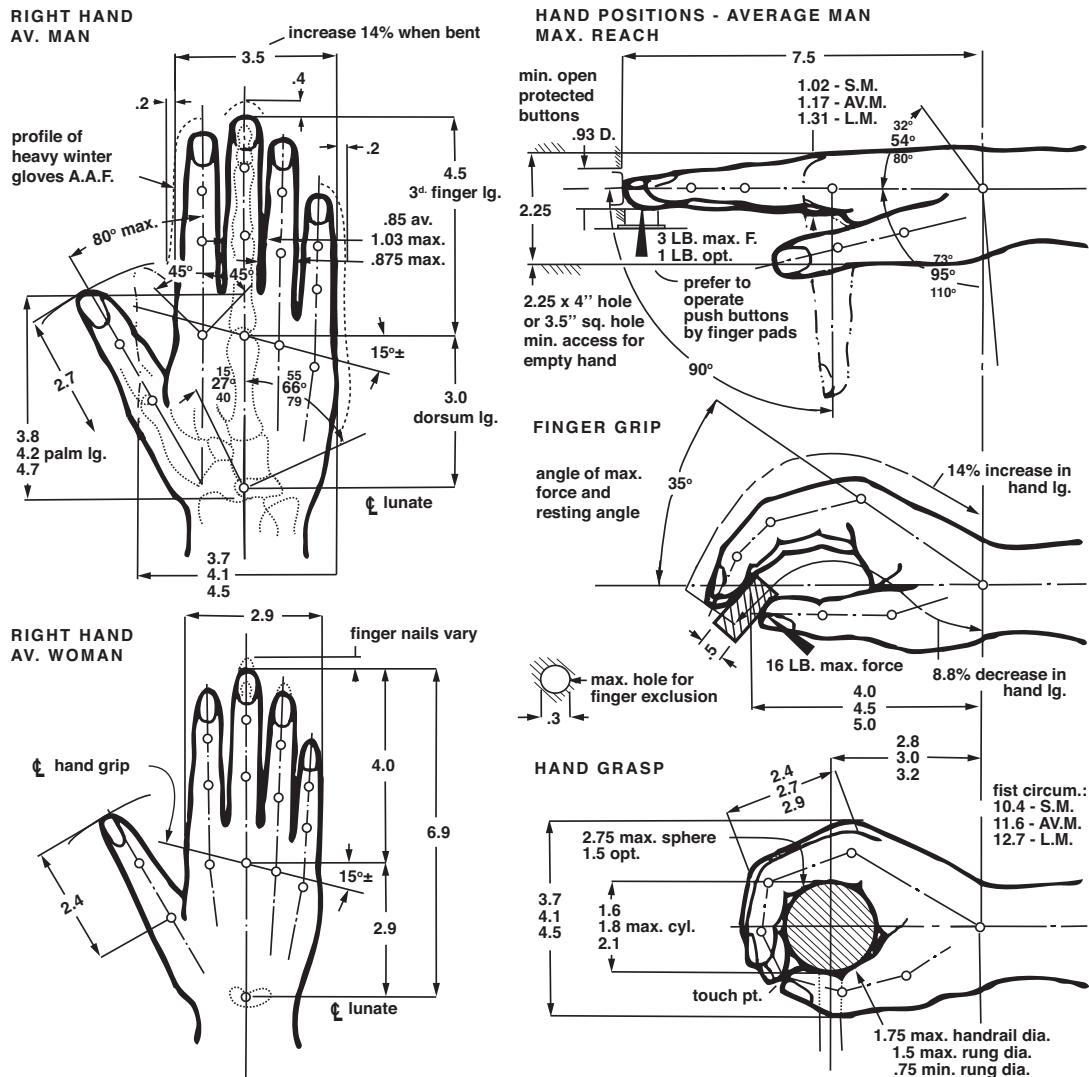
Two sources of information were important in establishing the ergonomic requirements of the DREAM Controller. A wealth of ergonomic information related to gestural interfaces can be found in (Nielsen et al., 2003). In this paper, the authors suggest six key principles of ergonomics: avoid outer positions, avoid repetition, relax muscles, relaxed neutral position is in the middle between outer positions, avoid staying in static position, and avoid internal and external force on joints and stopping body fluids. Each of these principles dramatically influenced our design during the prototyping phase.

Another source of anatomical information (and inspiration) was found in the 1955 book by Henry Dreyfuss titled “Designing for People” (Dreyfuss, 1955). The book contains composite figures of human anatomy gathered from years of research and data collection. In particular, there is significant commentary on the ergonomics of the human hand as seen in Figure 5-3. It was through these illustrations and text that we began decomposing the anatomy of the hand and recognized that the fingers were limited in their lateral deflection angles. Even in extreme stretching, there were key characteristics that we could exploit to identify the components of the hand.

5.4 Hand And Finger Registration

Not only does the system need to have a low error rate, but it also needs to allow for quick recovery from an error. If an error is made, the user should be able to quickly realize that something has gone wrong and recover from the error. As such, hand detection and finger registration became one of the most important aspects of the interface design. The algorithm not only needed to be fast, but it needed to

HAND MEASUREMENTS OF MEN, WOMEN AND CHILDREN



HAND DATA	MEN			WOMEN			CHILDREN			
	2.5%tile	50.%tile	97.5%tile	2.5%tile	50.%tile	97.5%tile	6 yr.	8 yr.	11 yr.	14 yr.
hand length	6.8	7.5	8.2	6.2	6.9	7.5	5.1	5.6	6.3	7.0
hand breadth	3.2	3.5	3.8	2.6	2.9	3.1	2.3	2.5	2.8	—
3 ^d . finger lg.	4.0	4.5	5.0	3.6	4.0	4.4	2.9	3.2	3.5	4.0
dorsum lg.	2.8	3.0	3.2	2.6	2.9	3.1	2.2	2.4	2.8	3.0
thumb length	2.4	2.7	3.0	2.2	2.4	2.6	1.8	2.0	2.2	2.4

Figure 5-3: The hand characteristics and registration heuristics were, in part, inspired by Henry Dreyfuss's 1955 book "Designing for People." Reproduced with permission. Courtesy of Allworth Press.

be reliably insensitive to extreme finger configuration and hand orientation cases that the user might present while experimenting with the interface.

Finger detection is the first step in our algorithm. When a finger touches the device, the centroid of the contact is added to a list of possible point candidates. If there are five or more candidates in the list, the candidate list is passed through a heuristic to determine if those points could contain a subset of five fingers from a single hand. If five of the points pass the heuristic, the points are removed from the candidates list and added to a separate hand list.

Currently, the heuristic for the hand is a pair-wise evaluation of the candidate points to determine if a subset of those points are within the maximum possible distance for a human hand. Based on Dreyfuss's measurements (Dreyfuss, 1955) for the largest male hand and some empirical testing, we determined that the distances between the points need to be clustered within eight inches. This heuristic was chosen because of its simplicity, but improvements on this heuristic are discussed in the section on future work.

Once the heuristic has determined that a hand has been detected, the finger registration algorithm then attempts to figure out which of the five points correspond to specific fingers. To compare the angles between the points, a relatively accurate centroid of the hand needs to be located. A bounding box is created around the five points. The centroid of the box roughly represents a point above the center of the palm, but below the knuckle on the middle finger.

A sorted list of angles between adjacent points and the centroid of the bounding box is then calculated. The largest angle in this list represents the angle between the thumb and the little finger. The second largest angle represents the angle between the thumb and the index finger. By taking the intersection of these two sets, the algorithm is able to determine the point representing the thumb. The complimentary point on the largest angle is then the little finger and the complimentary point on the second largest is the index finger. Likewise, the complimentary point to the index finger that is not the thumb is the middle finger. The remaining point is the ring finger.

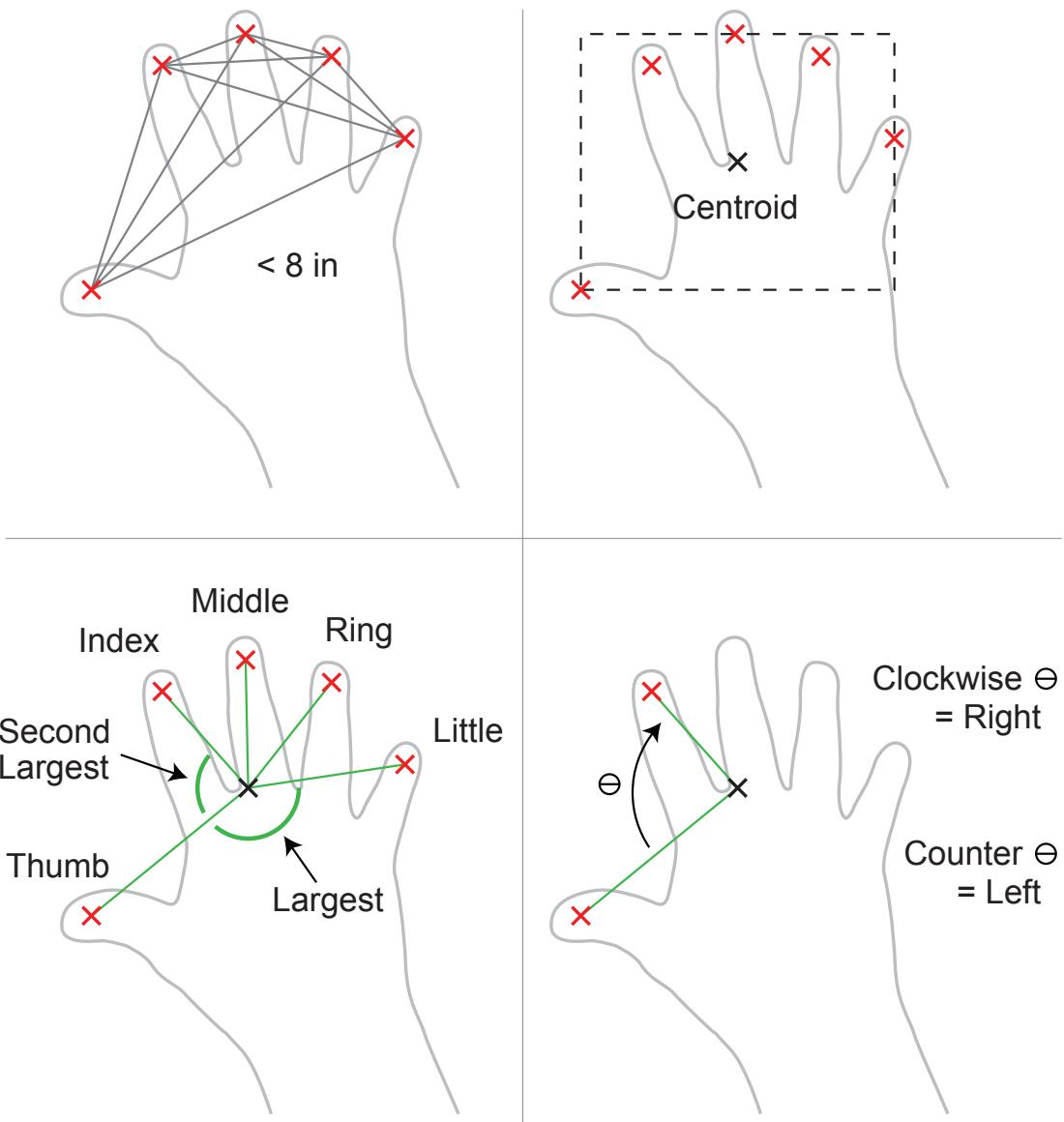


Figure 5-4: Hand and finger registration is accomplished by first ensuring that the points are within the maximum size for a human hand (top left), then finding the centroid of the bounding box containing all of the points (top right), determining the two largest angles (bottom left), and determining if the angle from the thumb to the index finger is clockwise or counterclockwise (bottom right).

Now that the fingers have been identified, the algorithm can determine if the fingers correspond to a right or left hand. If the angle from the thumb, centroid, and index finger is clockwise, then it is the right hand. If the angle is counterclockwise, then it is the left hand. To determine the direction of rotation, we take the sign of the cross product of two vectors. For the purpose of derivation, we assign the vector from the centroid to the thumb as \vec{a} and the vector from the centroid to the index finger as \vec{b} . Assuming that \vec{a} and \vec{b} are in the XZ plane, then the cross product vector will be positive (point upwards) if the rotation from \vec{a} to \vec{b} is counterclockwise, and be negative (point downwards) if the rotation is clockwise. Since these are two dimensional vectors, the cross product derivation expands to Equation 5.1 where (x_t, y_t) correspond to the thumb coordinates, (x_i, y_i) correspond to the index finger coordinates, and (x_c, y_c) correspond to the centroid of the bounding box.

$$(x_c - x_t) \cdot (y_i - y_c) - (y_c - y_t) \cdot (x_i - x_c) \quad (5.1)$$

At this point, the algorithm has all of the information that it needs to begin building the DREAM Controller underneath the user's hand. With the exception of the initial bounding box to determine the centroid of the hand, it should be noted that the algorithm does not rely on a Cartesian coordinate system and is insensitive to user orientation. The algorithm only uses the highly optimized atan2 standard math library function and arithmetic functions, making it inherently fast and applicable for execution on limited processors. Finally, since the algorithm does not use any of the contact surface area information, it can be used on other multi-touch technologies that only return single pixel touch points.

5.5 Form and Function

The algorithm uses the hand geometry and handedness to adjust the size, orientation, and arrangement of the DREAM Controller elements. As shown in Figure 5-5, there are some key measurements that determine these features. The following

description of the widget algorithm will focus on the right hand controller, but the left hand controller is the same algorithm, just mirrored.

First, the angle from the thumb to the index finger determines the orientation of the controller and the button grid. The top right corner of the button grid is placed at the index finger and the lower right corner is placed at the thumb. The width of the grid is determined by the size of the buttons and sliders with the addition of padding for layout and visual balance. The height is the distance between the thumb and the index finger.

A circle containing the points corresponding to the thumb, index finger, and little finger is calculated. This bounding circle provides a purely aesthetic visual, showing the users that the controller is tailored to their specific hand size. Programmatically, this circle is also used to protect user elements in lower panels from detecting erroneous events from the finger movement on the controller layer above. Since the controller can be created and moved to any part of the screen, this protection for lower event triggers becomes important.

The circle is calculated based on the circumcircle that surrounds the triangle created by the points representing the thumb, index, and little finger. First, the coordinates of the circumcenter is calculated using the equation shown in Equation 5.2 through 5.4. The distance from this circumcenter to any of the finger points is the radius of the desired circle.

$$x_c = ((y_t^2 + x_t^2) * (y_i - y_l) + (y_i^2 + x_i^2) * (y_l - y_t) + (y_l^2 + x_l^2) * (y_t - y_i)) / D \quad (5.2)$$

$$y_c = ((y_t^2 + x_t^2) * (x_l - x_i) + (y_i^2 + x_i^2) * (x_t - x_l) + (y_l^2 + x_l^2) * (x_i - x_t)) / D \quad (5.3)$$

$$D = 2 * (x_t * (y_i - y_l) + x_i * (y_l - y_t) + x_l * (y_t - y_i)) \quad (5.4)$$

Two circles are placed under the center of the index and middle fingers. These are analogous to the “shoulder” buttons used on dual-thumb joystick controllers. The circles were not needed to provide button functionality in this particular implementation, but since they can receive their own events (above the controller object), they can easily be used as momentary or toggle buttons if needed. Additionally, these have the potential to be used as pressure sensitive buttons just

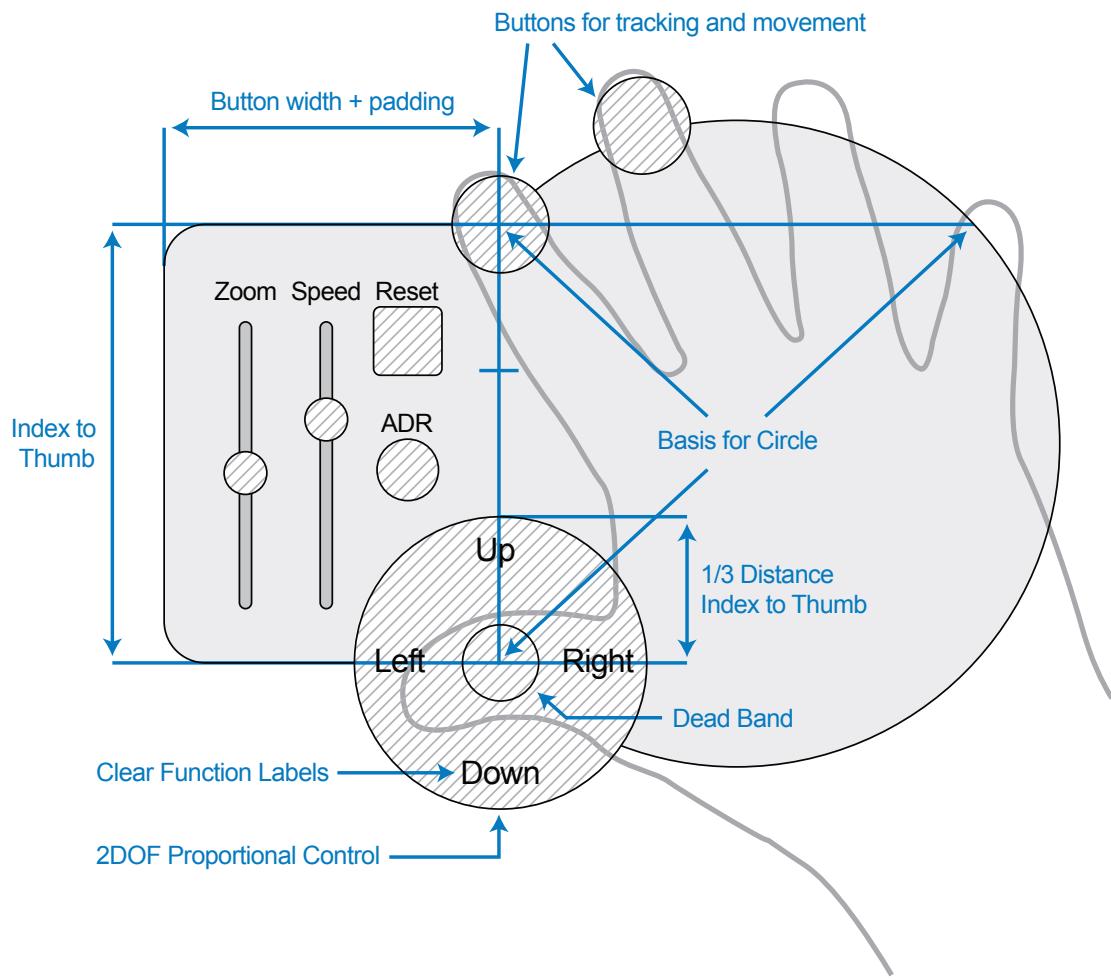


Figure 5-5: Users have a wide variety of hand characteristics. The DREAM Controller adjusts several parameters to tailor the size, orientation, and position of the controller. This design is intended to maximize the users' comfort and performance.

like the dual-thumb joystick controllers since the relative size of the finger blob detected can be determined as the user changes the amount of pressure placed on the surface.

A circle pad is placed under the thumb representing analog control for two fully proportional degrees of freedom (DOF). Much like the Playstation controller, the thumb is then moved up, down, left, and right corresponding to the desired movement. The thumb pads have their function clearly labeled within the boundaries of the circle. The left hand is labeled for robot movement where the top of the circle corresponds to moving forward, bottom to moving backwards, left for counterclockwise rotation, and right for clockwise rotation. For the right hand, movement to the top of the circle tilts the camera up, bottom of the circle tilts the camera down, and left and right movement rotate the camera left and right respectively. In both hand cases, the user has full proportional control and can fully mix the analog ranges for both degrees of freedom. If the thumb is not in contact with the analog control pad, then motion of the camera pan-tilt mechanism or drive system is immediately halted.

The text labeling in the camera thumb control visual is important because users may come to the interface with two different competing paradigms for camera manipulation. In the case of people familiar with first person shooters, moving the thumb joystick toward the top causes the character view to look in an upwards direction. In the case of airplane or submarine simulators, moving the joystick down causes the vehicle to fly in a more upward angle. The downward motion of the joystick is commonly referred to as “pulling up” on the joystick even though in actuality the joystick is being pushed to the bottom of its freedom of movement. Interfaces should carefully implement labeling and optionally give the user an option to select either paradigm.

The index and middle finger can be moved to adjust the angle of the controller dynamically if the thumb is not in contact with the joystick. The controller will simply follow the user’s fingers and maintain the orientation of the controller relative to the respective contact points. This freedom of movement is stopped

once the thumb makes contact with the 2DOF analog control pad. The shunting of movement is important for safety and continuity of control. Many secondary movements are transferred to the index and middle finger when the thumb muscles are actuated. The thumb position relative to the analog control pad changes if this movement of the index finger and middle finger then rotate or translate the position of the controller. This, in turn, causes the user to move their thumb, which causes the index and middle finger to again move. This feedback loop would quickly become very frustrating to the user and would result in the user pushing down on the screen surface to limit secondary movement. Muscle fatigue would quickly occur and the user experience would be very diminished. By simply stopping movement on thumb contact, the controller maintains position and allows the user to relax his or her hand muscles without consequence.

Removal of the DREAM Controller requires the user to simultaneously lift their thumb and their index or middle finger. As mentioned above, the user can lift their thumb to stop the motion of the robot and allow repositioning of the controller.

At any time while using the controller, the user can lift or lower their ring and little fingers with no effect on operation. We made this design decision for the comfort of the user after noticing this relaxation behavior with the early paper prototypes.

5.6 Experiment Design

We needed to test the reliability of the above algorithms before investing effort integrating the DREAM Controller into the interfaces described in Chapter 6. To test the accuracy of the hand detection and the finger registration algorithm, we designed an experiment to collect data from a wide sampling of people to get a range of hand sizes and natural angles between fingers. For this experiment, we used a Microsoft Surface which uses diffuse illumination of infrared (IR) light.

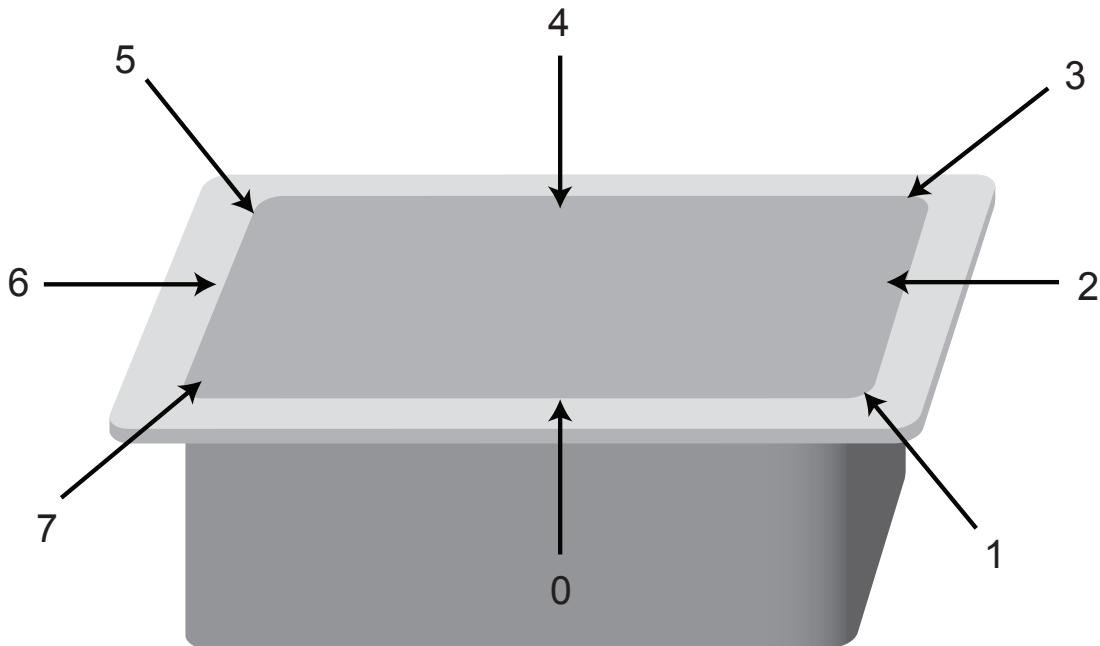


Figure 5-6: Participants in the study were asked to stand in one of eight positions around the Surface to test the sensitivity of the algorithm to hand orientation.

At the start of the experiment, the participant would stand facing the Surface.¹ Participants were asked to stand in one of eight positions: four corresponding to the sides of the Surface and four to the corners of the Surface, as shown in Figure 5-6. A participant would stay in his or her location for the entire data collection session.

Once positioned, the participant was asked to place his or her fingers down on the table in three configurations, first with the right hand, then with the left. Each hand posture was demonstrated by the researcher for the participant and is shown in Figure 5-7.

First, participants were asked to curl their fingers so that the tips touched the board simultaneously, similar to a pose used by piano players or touch typists. After the Surface registered the touches, our algorithm would run, displaying how it identified the fingers and hand. Participants were then asked to remove their hand from the Surface. If the Microsoft-supplied vision processing system failed

¹Our Surface is mounted on a wheeled dolly, making operation from a standing position comfortable.

to identify all five touches when a participant put his or her hand down,² the participant would be asked to try the hand position again.

Second, participants were asked to relax their hand so that their muscles were in a neutral position and pose. In the ergonomic literature, this hand position would be referred to as a position of function (e.g., see (Bunnell, 1942)). This position is the same pose that is used after a hand injury when a hand is placed in a cast. It is the most natural and ergonomic pose for the human hand. Participants were again asked to place the pads of their fingers on the Surface simultaneously. After the Surface registered the five points and our algorithm ran, participants were told they could remove their hand from the table. If the Surface did not register all five points, the participant was asked to put his or her hand down in that position again.

Third, participants were asked to stretch their fingertips as far as they could while maintaining reasonable comfort. This position represents the maximum hand size that the participant can create. Again, participants were asked to place the pads of their fingers on the Surface simultaneously.

The entire process took under two minutes per participant. Each participant was given a gift card for their participation in the study.

Data was collected in three ways. First, the test administrator wrote down on paper whether the algorithm correctly identified the hand and fingers. Correctness was determined by visually inspecting the labels over the fingers. Because the algorithm could not measure its own performance, this manual step was necessary for evaluation.

Data was also collected using the Surface Simulator Recording Utility. This utility creates a script file for all of the touch events on the Surface. A script was created for each user individually. A concatenation of these scripts provides a comprehensive way to test future versions of the algorithm and compare them directly to this study.

Finally, each time the hand detection and finger registration was executed, a

²Sources of the interference could be sunlight, fluorescent lighting, or other sources of IR light.

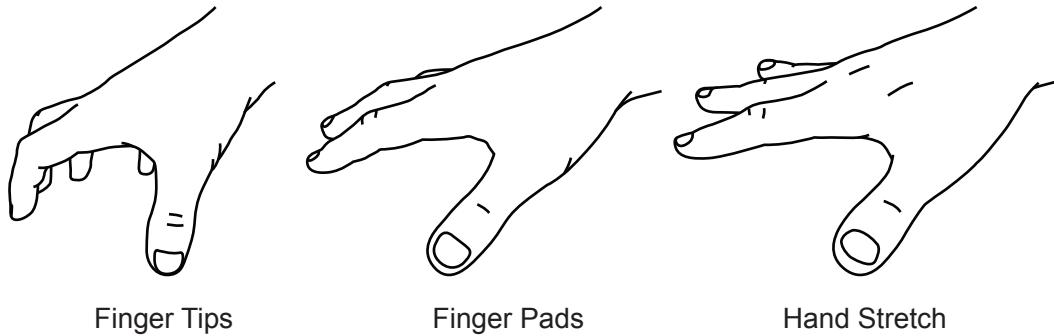


Figure 5-7: Participants were asked to place their hands in three positions. First, on the tips of their fingers (left) like a pianist or touch typist. Then in a relaxed position (center) with the pads of their fingers resting on the surface. Finally, they stretched their hands as far as possible (right) to demonstrate their maximum contact size.

photograph of the raw image of the diffused illumination surface was taken. This image capture used the same cameras in the Surface that provide all of the image processing of touch points. This final data collection step was done in the event that a future hand detection or registration algorithm is developed that wishes to bypass the standard Surface API libraries and process the images directly. Since the Surface cameras can see fingers and hands slightly above the surface of the screen, information outside of touch contacts may be an interesting source for future information gain about the orientation and registration of fingers. In the interest of direct comparison, this photographic data set could be used to evaluate a new algorithm relative to algorithms that strictly use the script method above.

5.7 Results and Discussion

Sixty-five people participated in this experiment (21 female, 44 male), which resulted in a data set of 390 hand placements on the Surface. The algorithm described in this paper correctly recognized 92.31% (360 of 390) of the hand placements. It correctly recognized both right hand placements (91.79%; 179 of 195) and left hand placements (92.82%; 181 of 195).

5.7.1 Hand Position

The user may or may not place his or her hand in a comfortable or correct posture for efficient use of the DREAM Controller. While it is hoped that this will eventually be the case, during the experimentation and learning phase, the user may present many different hand configurations. The algorithm must determine the correct finger registrations and arrange the controller correctly in even severely sub-optimal hand positions. If not, the user will doubt that the interface will perform reliably.

The three hand postures shown in Figure 5-7 represent two extreme cases and one desired case. When the participant touched the Surface with the tips of their fingers, this represents a very compact and minimal size for the overall surface area of the fingers and will result in the smallest controller size. This position is also an expected experimentation approach for any users who are familiar with touch typing. The anatomy of the hand in this configuration (as detected by the Surface) is not as well defined since the fingers are arched and do not have their normal representative angles of deflection. From the view of the Surface, the points are shortened fingers relative to the centroid of the bounding box and would therefore have potentially different angle characteristics.

In the second case, the relaxed pose puts the hand in the most ergonomic position and maximizes the surface area of the finger pads. The anatomy of the hand in this configuration is closest to the position of function and should represent a detectable configuration for the algorithm. This position is the one that we hope most users would use after some experience with the system. This posture should minimize fatigue and give the highest degree of manipulation for the thumb joystick.

The final posture represents the maximum size that the participant can generate with their hand. Since the thumb is fully extended in this case, it would not be a usable configuration for control of the thumb pad. This said, one would expect a inexperienced user to test this hand configuration during their learning time to

Table 5.1: Hand and finger recognition rates by hand

	Tips	Pads	Stretch
Right Hand	56 of 65 (86.15%)	61 of 65 (93.85%)	62 of 65 (95.38%)
Left Hand	59 of 65 (90.77%)	60 of 65 (92.31%)	62 of 65 (95.38%)
Both	115 of 130 (88.46%)	121 of 130 (93.08%)	124 of 130 (95.38%)

see how the system responds. This position also presented an opportunity to see if the hand detection heuristic for a cluster of points within 8 inches was appropriate for the population tested.

We conducted 1-tailed paired t -tests with $\alpha=0.05$ examining the three different hand postures. Table 5.1 shows the recognition rate for the right, left, and combined hand placements. We found that the algorithm recognized the participants' hands significantly better when people used the pads of their finger versus the tips of their fingers overall ($p<0.03$, $t(265)=2.20$).

The algorithm also recognized the participants' hands significantly better when people had their hands fully stretched versus the tips of their fingers overall ($p<0.01$, $t(265)=2.99$). Further, in comparing the tips and stretched hand positions, we found that the algorithm performed significantly better in both in right hand placements ($p<0.02$, $t(129)=2.43$) and in left hand placements ($p<0.05$, $t(129)=2.06$). The third hand posture in which the participant was asked to stretch his/her hand to maximally comfortable position is another instance of the pads hand posture. As such, we found no statistical significance between the recognition rates between the pads and stretch hand postures.

5.7.2 Board Position

One of the inherent values of multi-touch devices is the ability to bring people around a common table. Unlike traditional WIMP (windows, icons, menus, and

Table 5.2: Hand and finger recognition rates by board position

	Position 0	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7
All	54 of 54 (100%)	44 of 54 (81.48%)	48 of 48 (100%)	52 of 54 (96.30%)	40 of 41 (97.62%)	38 of 42 (90.48%)	42 of 42 (100%)	38 of 53 (70.37%)
Right overall	27 of 27 (100%)	24 of 27 (88.89%)	24 of 24 (100%)	24 of 27 (88.89%)	20 of 21 (95.24%)	17 of 21 (80.95%)	21 of 21 (100%)	21 of 27 (77.78%)
Right tips	9 of 9 (100%)	8 of 9 (88.89%)	8 of 8 (100%)	8 of 9 (88.89%)	6 of 7 (85.71%)	5 of 7 (71.43%)	7 of 7 (100%)	5 of 9 (55.56%)
Right pads	9 of 9 (100%)	8 of 9 (88.89%)	8 of 8 (100%)	8 of 9 (88.89%)	7 of 7 (100%)	6 of 7 (85.71%)	7 of 7 (100%)	8 of 9 (88.89%)
Right stretch	9 of 9 (100%)	8 of 9 (88.89%)	8 of 8 (100%)	8 of 9 (88.89%)	7 of 7 (100%)	6 of 7 (85.71%)	7 of 7 (100%)	8 of 9 (88.89%)
Left overall	27 of 27 (100%)	20 of 27 (74.07%)	24 of 24 (100%)	27 of 27 (100%)	21 of 21 (100%)	21 of 21 (100%)	21 of 21 (100%)	20 of 27 (74.07%)
Left tips	9 of 9 (100%)	7 of 9 (77.78%)	8 of 8 (100%)	9 of 9 (100%)	7 of 7 (100%)	7 of 7 (100%)	7 of 7 (100%)	5 of 9 (55.56%)
Left pads	9 of 9 (100%)	6 of 9 (66.67%)	8 of 8 (100%)	9 of 9 (100%)	7 of 7 (100%)	7 of 7 (100%)	7 of 7 (100%)	7 of 9 (77.78%)
Left stretch	9 of 9 (100%)	7 of 9 (77.78%)	8 of 8 (100%)	9 of 9 (100%)	7 of 7 (100%)	7 of 7 (100%)	7 of 7 (100%)	8 of 9 (88.89%)

pointer) interfaces which have a clear sense of orientation, people may use the multi-touch device from any side. In this experiment, participants stood at one of eight positions. We wanted to test if our hand and finger recognition algorithm was insensitive to orientation. Table 5.2 shows the hand and finger algorithm recognition rates for each of the eight positions.

We examined the recognition rate for the hand placements for which the participant was perpendicular to a major axis (i.e., positions 0, 2, 4, and 6) versus those who were at a 45 degree angle between the major axes (i.e., positions 1, 3, 5, and 7). For participants facing a major axis, the algorithm recognized 99.46% (185 of 186) of the hand placements and 85.78% (175 of 204) of the hand placements for participants who were not. We conducted 1-tailed unpaired *t*-test with $\alpha=0.05$ and found that our algorithm performed significantly better with on-axis hand placements ($p<0.01$, $t(388)=5.51$).

5.7.3 Gender

Table 5.3 shows the hand and finger algorithm recognition rates for males, females, and all participants combined. We conducted 1-tailed unpaired *t*-test with $\alpha=0.05$ and found that our algorithm performed significantly better overall

Table 5.3: Hand and finger recognition rates by sex

	Overall	Tips	Pads	Stretch
Male	251 of 264 (95.08%)	81 of 88 (92.05%)	84 of 88 (95.45%)	86 of 88 (97.73%)
Female	109 of 126 (86.51%)	34 of 42 (80.95%)	37 of 42 (88.10%)	38 of 42 (90.47%)

on hand placements from male participants versus female participants ($p<0.01$, $t(388)=2.79$).

Interestingly, we find that in 97% (28 of 29) of the failed recognitions of hand placements for both male and female have to do with placement of the thumb relative to the fingers. Figure 5-8 shows a sampling of the failure cases in which the angle between the thumb and index finger is greater than the angle between the thumb and little finger. Visually, the thumb might seem to be nearly equal in distance to both the index and little fingers. Physically, the thumb is positioned under the palm. At this point, it is unclear if the effect is also a function of the difference in hand size between men and women, of an effect that would reduce with a larger population. Regardless, we believe that the user's hand would quickly tire in this position while manipulating the proportional joystick because of the restricted thumb movement. We hope that when actually using the DREAM Controller, the user would adjust his or her hand to a more comfortable and less tiring configuration.

5.8 Potential for Improvement

Although the hand detection and finger registration algorithm performed well for this user study, there is still room for improvement. For instance, the use of the bounding box for determining the relative center of the hand loses accuracy as the user approaches the 45-degree positions relative to the Cartesian coordinate system. A more accurate and orientation insensitive approach might use an equation that

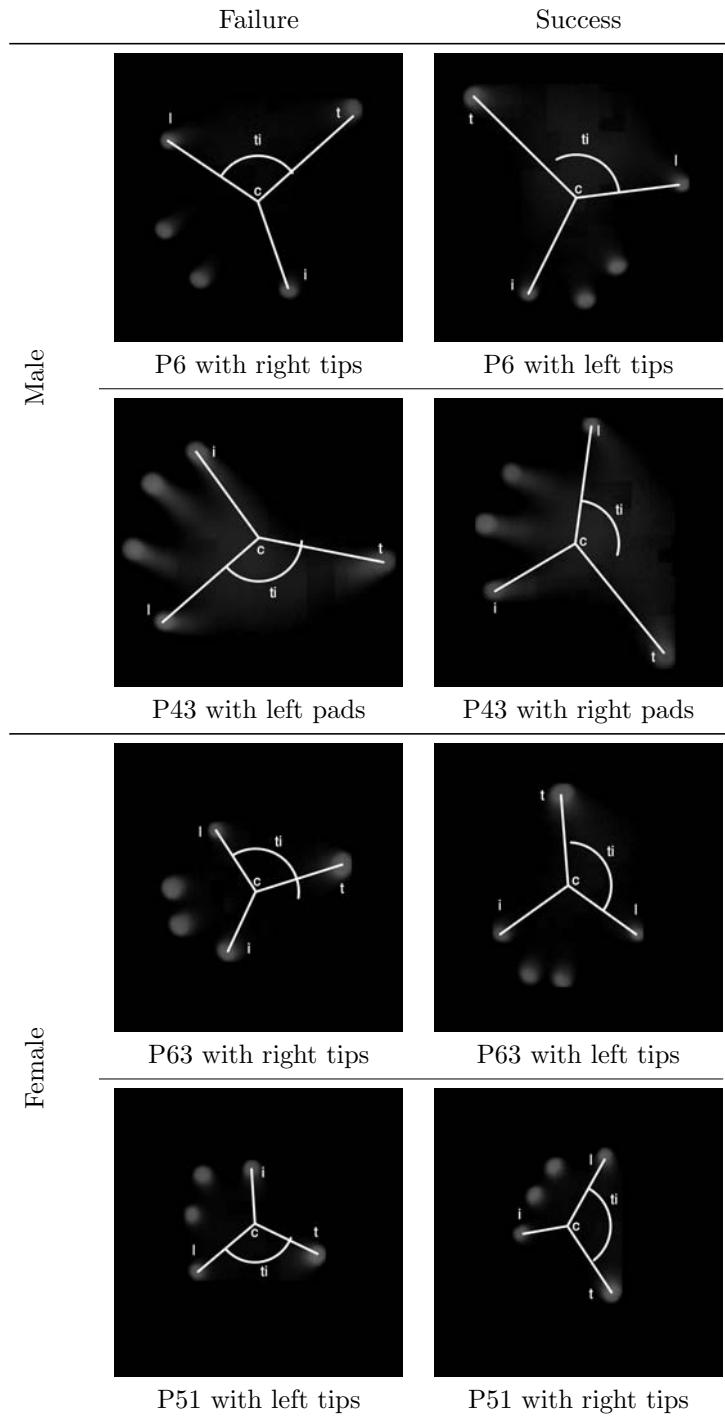


Figure 5-8: Annotated images for failed hand and finger recognition of hand placements (left column) and successful recognition for the other hand (right column). The center point (c), thumb (t), index finger (i), and little finger (l) have been labelled. ti is the angle between the thumb and index finger and is overlaid on the angle tcl as an arc. When the arc ti extends over the line tc , then the algorithm failed. In other words, the angle between the thumb and the index finger must be smaller in order to correctly identify the fingers.

uses all or a subset of the points to create a circumscribed polygon. The center of this circle, or some combination of circles, may provide a closer approximation and would certainly be less orientation sensitive.

There is an interesting point of information gain that was not exploited by this algorithm that may also help in determining the centroid of the palm and help start the finger registration process. The thumb is typically the point furthest from all of the other finger points. By performing the reverse of a nearest neighbor search, the algorithm could find the point that is furthest from all of the other points in the set. Since this point is arguably the thumb, a weighted average for the points in the Cartesian plane or some other heuristic might be able to better determine the center of the palm and bootstrap the registration process.

By all intuitive measures, the algorithm described and tested in this document should work well with multiple users at different orientations. Since the DREAM Controllers are always located relative to the hands of the operators, concepts of territoriality in collaborative workspaces are preserved (Scott et al., 2004). However, the simple hand detection heuristic of five points clustered within eight inches may not be sufficient for users that are in close proximity to each other and place their hands down on the surface simultaneously. The heuristic would need to make a much closer anatomical evaluation of the candidate points to disambiguate the points in nearby hands that have not been fully detected and registered.

5.9 Impacts and Implications

Given the high success rate of the current algorithm, it is the hope of the author that this algorithm, and algorithms like it, become a standard component of multi-touch software development kits. As future interfaces begin taking advantage of the full anatomy and ergonomics of the human hand, the identification and registration of fingers becomes vital. It is to this end that the collected Microsoft Surface Simulator scripts and Surface photographic data is being released into the public domain and available upon request. An example of the Surface Simulator data

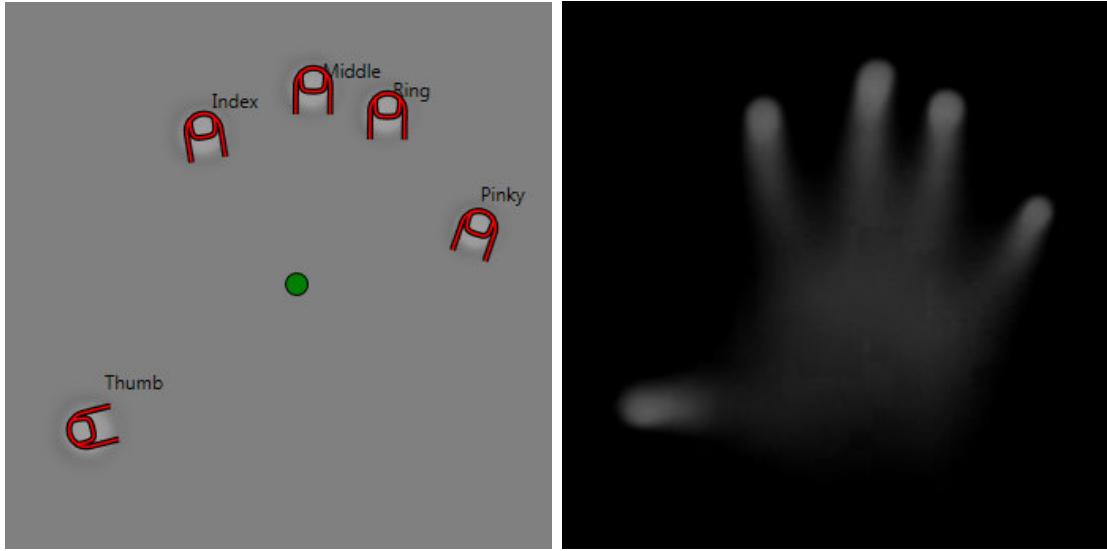


Figure 5-9: Our hand detection algorithm is played back in the Surface Simulator using the publicly available dataset (left). Additional photographic data (right) is included in this data set for researchers that wish to work with the vision system directly.

and photographic data is shown in Figure 5-9. Others are encouraged to compare their algorithms to our results and improve upon the detection and registration performance. When coupled with the Microsoft Surface Simulator API, this data has the potential to be a powerful tool for testing new algorithms.

Although this research is set in the context of physically situated robots, we foresee this type of controller being used effectively for game play and other domains when multiple degrees of freedom must be managed simultaneously. First person games are an obvious application, but others such as crane or forklift operation, manufacturing equipment, or aircraft control may benefit from experimentation with controllers like the one described.

Chapter 6

Robust Multi-touch Robot Control

Having established a potentially useful model for joystick emulation on multi-touch devices in Chapter 5, the next natural step in this iterative design process was to see how the DREAM Controller compared to the performance of the DiamondTouch interface validated in Chapter 4. The interface and experiment design for this part of the study were two-fold. First, we wanted to apply the newly designed DREAM Controller to the well tested and studied UML USAR interface. The original joystick-based design described in Chapter 4 and also in Keyes (2007) would provide a baseline for comparison. We kept the visual presentation constant, which would allow a three-way comparison between the joystick, DiamondTouch, and new Surface DREAM Controller. It was our hope that the new design would increase performance above both of the prior implementations.

The second part of this chapter's study drastically changed the visual presentation while keeping the DREAM Controller implementation constant. In Yanco and Drury (2006), one of the suggested guidelines for robot control interfaces was to use larger video on the control screen since that was the primary sensor used by the operators. The changes in this interface included a full-screen display with transparent sensor readouts. Taking inspiration from the displays on military helicopters and aircraft, the video was stretched to the extents of the screen and a

green overlay would augment the view of the surrounding environment with sensor readings and robot state.

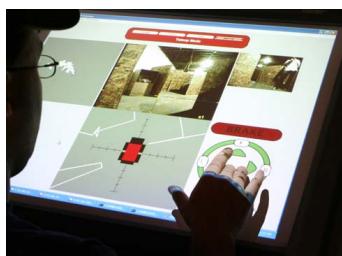
By carefully controlling the variables in the experiment, it was our intent to provide a two-step improvement in the same user study. If either of the interface changes degraded performance, we would be able to identify the component of performance degradation and adjust future versions of the interface accordingly.

Table 6.1: Four versions of the operator control interface. The original joystick version (top) was compared against the DiamondTouch version from Chapter 4 (upper-middle) and two newer Surface versions (lower-middle and bottom).



Joystick:Window

Interface that uses joystick and keyboard for input with the visual display detailed in Chapter 4 and also in Keyes (2007).



DiamondTouch:Window

Interface that uses multi-touch control with the static round widget. This implementation was detailed in Chapter 4 and otherwise uses the identical graphical presentation as Joystick:Window.



Surface:Window

Interface that uses the DREAM Controller described in Chapter 5. Graphical presentation is based on Joystick:Window and DiamondTouch:Window with only minor layout adjustments.



Surface:FullScreen

Interface that uses the same DREAM Controller design with a full-screen video display. The interface borrows from the lessons of the previous interfaces, but is a significant departure from the previous graphical layouts.

The UML USAR interface has seen an almost constant improvement through multiple iterations over eight years of development. A significant improvement in input method and display organization would hopefully provide the next step in this interface evolution.

6.1 Interface Definitions

For this chapter, there are four interfaces that will be discussed. They are shown and defined in Table 6.1. For the experiments based on these interface definitions, there are two independent variables. The first is the control input method, where the new DREAM Controller in Surface:Window will be compared to the performance of the multi-touch joystick widget on DiamondTouch:Window and the physical joystick with Joystick:Window. The other independent variable is the presentation of the interface. The window-based interface in Surface:Window will be compared to the full screen display in Surface:FullScreen.

We will also use this notation to maximize the clarity and brevity of comparisons in the following descriptions and analysis. References to the “Window” interface include Joystick:Window, DiamondTouch:Window, and Surface:Window, as opposed to the “FullScreen” interface which includes Surface:FullScreen. A reference to the “Joystick” or “DiamondTouch” interface refer to the Joystick:Window and DiamondTouch:Window respectively. Finally, “Surface” refers to both the Surface:Window and Surface:FullScreen interfaces.

6.1.1 Surface:Window Interface Description

The Joystick:Window and DiamondTouch:Window interfaces provided the look and panel arrangement for the Surface:Window interface. The earlier interfaces are discussed in depth in Chapter 4. The focus of this section is the Surface:Window interface and the interaction techniques implemented.

The Window interfaces use four panels and a user adjustable joystick for control of the robot functions. Each discrete interaction element is shown in Figure 6-1.

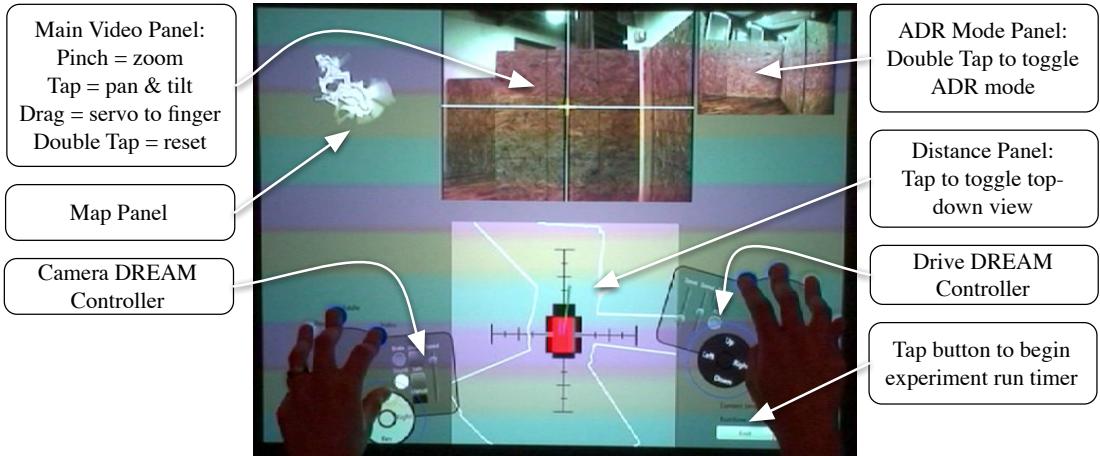


Figure 6-1: Surface:Window features and gestures for operating the robot with on-screen controls.

Main Video Panel: This panel provides the view from the forward facing camera and occupies the center of the display. The video image is augmented by a crosshair that represents the position of the camera pan and tilt unit (PTU) relative to the body of the robot. The crosshair is calibrated so that when the camera is facing straight forward, the crosshairs are centered on the vertical and horizontal view of the camera image. The crosshair coordinate system corresponds to the limits of movement for the PTU, where panning rotation from 0 to 90 degrees leftward will move the crosshair proportionally from the center to the left of the image. This similarly happens for movement to the right, top, and bottom. The crosshairs move proportionally across both degrees of freedom, which provides a quick visual reminder of the rotation of the camera PTU.

If the user taps the main video panel with his or her finger, the PTU will servo to the corresponding (x, y) location with the crosshair providing real-time position feedback. If the user drags the finger, the crosshair and PTU will “chase” the finger and attempt to maintain the crosshair under the finger tip. The PTU is mechanically limited and the motors have to overcome inertia, friction, and other physical properties, so the tracking movement is best-effort and optimized to minimize lag as much as possible. A double-tap on the main video panel will reset the PTU to the origin $(0, 0)$ where the camera is looking straight forward

and level to the robot chassis.

ADR Mode Panel: The robot has a rear-looking camera with a wide-angle lens that allows the user to see what is behind the robot at any time. The video from this camera is placed in the upper right hand corner of the user interface and is analogous to a rearview mirror in an left-hand drive automobile. This image is mirrored on the vertical axis to make it appear natural as a rear-view mirror. In Keyes et al. (2006), it was shown that simply providing this view of the robot helps significantly increase situation awareness and many measurements of user performance.

A double tap on the rear view will cause the rear image to switch with the larger main image in the center of the screen. This feature is called Automatic Direction Reversal (ADR) mode and it allows the user to drive the robot backwards while looking through the larger central image. To further facilitate the ease of operation, the driving commands are reversed so that a normally forward movement of the joystick now moves the robot backwards. Steering is similarly reversed. This effectively allows the user to drive the robot physically in reverse while it is perceived to be driving forward. Since the robot has skid-steering and is longer than it is wide, this feature is helpful when the robot is driven into a corner and cannot easily turn around in place.

Distance Panel: Directly below the main video panel, a distance panel depicts the robot's chassis with white lines displayed around it representing the distance readings from the sonar sensors and the laser range finder. These distance lines help give the user frame of reference regarding how close the robot is to obstacles that may, or may not, be in the view of the camera. The axis immediately under the robot are shown with 0.25 meter tick marks for reference. Lines surrounding the robot in this panel are a fusion of the laser range sensor in the front of the robot and the sonar data that covers 360 degrees around the robot. Since the laser range finder is typically more accurate and has higher resolution, the sensor display for the front of the robot is more reliable than the sides and rear. With a tap, this panel can be toggled between a perspective view (similar to popular GPS

navigation systems) and a top down view (similar to a paper map).

Map Panel: The upper left corner of the display contains a map that is dynamically generated as the robot travels through the environment. This map, which uses a simultaneous localization and mapping (SLAM)-based algorithm (Howard, 2006), is dynamically generated as the robot is maneuvered around an area. The map shows open space as white space, obstacles are depicted as black lines, and grey represents unexplored space. The robot location is shown with a green triangle and the robot path is a red line.

Although it was not the intent of the original design, the arrangement of panels left two open areas on each side of the distance and main video panels sufficiently large for two hands. This area was used as a convenient location to place five fingers and spawn the DREAM Controller. The user was not restricted to this area, but the openness of this area lent itself to joystick control very easily. When the user places five fingers down, the DREAM Controller is activated and when the fingers are lifted, the robot is stopped and the controller is removed.

It should be noted that this Surface:Window interface omits two panels that were used in the previous revision of the DiamondTouch:Window interface. First, the autonomy mode panel that was previously above the main video was removed and this functionality was moved to three radio buttons on the DREAM Controller (to be described in Section 6.2). The proportional drive control widget in the lower right was removed along with the speed sliders and brake control. Again, this functionality is moved to the control panel on the DREAM Controller.

One of the motivations for this change was a minimalist approach to the main window display. Specifically, when the user was not touching the interface, we wanted to maximize the resolution and layout of each of the components of video, range sensing, and mapping. Having the interface cluttered with static buttons and other widgets seemed wasteful when we had determined that the dynamic DREAM Controller could provide this functionality and optimize button placement relative to the users' hands.



Figure 6-2: Surface:FullScreen features and gestures for operating the robot with on-screen controls.

6.1.2 Full Screen Interface Description

The full screen version of the interface has all of the same interface elements of the Window display, but presented in a slightly different arrangement. The most significant change is the maximization of the video to the full extents of the screen. Since the video now takes up the entire screen, all of the interface elements need to be transparent and allow the user to see the video underneath the graphical rendering of the information presented. Taking inspiration from fighter-jet and helicopter heads-up display implementations, we outlined all of the interface elements in a high-contrast green with their backgrounds at full transparency or slightly tinted black.

Camera Control: To control the camera, the user needed to only tap on the part of the image that he or she wanted to have at the center of the screen after rotation. Functionally, this meant that the user could just “tap on what you want to see” and the PTU would servo to center this object in the middle of the screen. If the user dragged their finger on the screen, the camera would “chase” the finger much like the Window interface.

Unlike the Window interface, we abandoned the crosshair design and displayed the pan and tilt degrees in a scrolling counter on the top and side of the display region respectively. The intent in this change was to give the user a direct reading

of the pan and tilt values rather than them having to visually interpret each side of the screen representing -90 to 90 degrees as in the Window interface. The user need only to look at the sliding scale and it would display the numerical number of degrees left, right, up, or down that the PTU is rotated. As before, a double-tap on the image will reset the PTU to the origin (0,0) where the camera is looking straight forward and level to the robot's chassis.

ADR Panel: The ADR and map functionality stayed relatively the same as the Window design. A double tap in the rear view would cause the interface to switch to ADR mode as previously described. Rather than locking the ADR and map panels in a single location on the screen, the individual panels could be disconnected from the corners and dragged to any location of the display for closer inspection. The panels could be moved, resized, and rotated. When the user was done manipulating the panel, a quick flick of the finger toward the desired corner of the screen would cause the panel to fly to that corner and snap into place. This feature was useful for close inspection, and it allowed the user to resize the map and rear view panels before snapping them back into the corners.

Distance Panel: The distance panel was identical to the Window interfaces in position and functionality. The panel was fully transparent with the exception of the outlines of the range readings and the distance scale. This transparency provided the user with all of the same distance information without interfering with the use of the camera video below.

Like the Surface:Window interface, the user places five fingers on the surface of the screen to activate the DREAM Controller functionality. Since the video takes up the entire screen, the DREAM Controller was made fully transparent with a bright green outline and a slight dark tint to match the rest of the FullScreen color scheme. The transparency allowed the user to gain all of the functionality of the DREAM Controller with minimal occlusion of the video underneath. When the user's hands were lifted, the controller disappeared and the robot immediately halted movement.

6.2 DREAM Controller

As reported in Chapter 4, we found that users bring many different experiences, learned behaviors, and biases to large touch screen interfaces. We found that the novelty of the multi-touch surface created opportunities for “emergent interactions” that were not anticipated by the system designers. Unfortunately, in most cases these alternate interaction styles degraded user efficiency, comfort, and ergonomics.

To counter these effects as seen in the DiamondTouch:Window interface, we decided to abandon the concept of static widgets that provided the affordances of a joystick, dial, knob, or switch. Rather than having the user conform to metaphors and artificially emulated affordances, we would have the user place their five fingers on the multi-touch surface and then shape the controller around their individual touch profile. This control scheme is novel, and to the our knowledge, this method of adaptive multi-touch control has never been applied to robots or game control.

To control the robot, we implemented a version of the DREAM Controller described in Chapter 5 for the iRobot ATRV-Jr robot platform. The design of the DREAM Controller is very flexible, so the control panel and thumb controls were tailored to the specifics of this robot. Specifically, the robot had functionality for brakes, speed limiting, and autonomy mode changes. All of the associated buttons and sliders were implemented on the panel extending from the index finger and thumb as described below.

As in many first person shooter games, the right hand was used for camera control. The control encircling the thumb would rotate the camera up, down, left, and right when the thumb was placed at the top, bottom, left and right of the circle respectively. The center of the thumb control represented zero rotational velocity, and this velocity increased as the thumb moved outward from the center. Full mixing of the two degrees of freedom was permitted which allowed for fully proportional control of the pan and tilt. A double tap in the thumb area would re-center the camera so the camera is looking straight forward and level to the

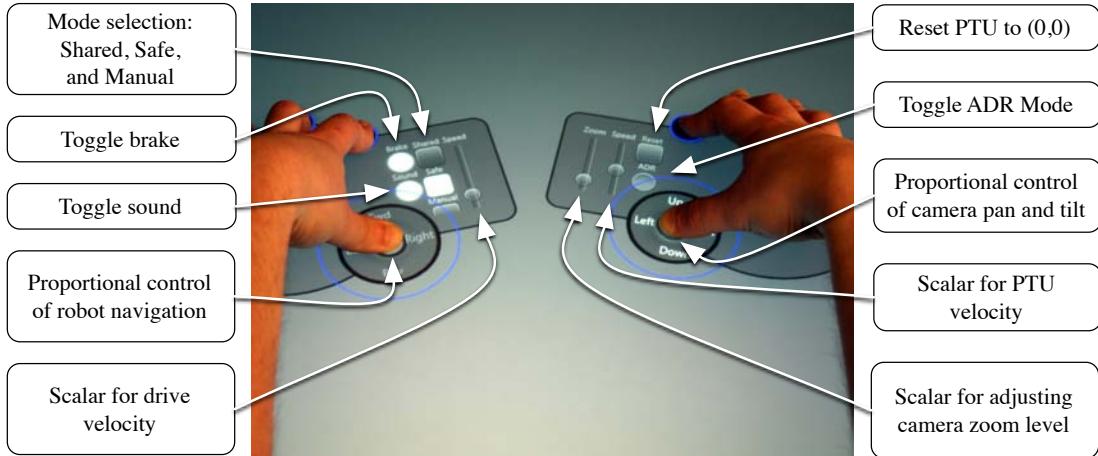


Figure 6-3: The DREAM Controller, as configured for the ATRV-Jr robot used in this study. Illustration shows the controller functionality and many of the local controls that duplicate on-screen controls.

robot's chassis.

The panel that extended from the right hand's thumb and index finger included buttons to re-center the camera and switch into ADR mode. These two controls duplicated the functionality of the on-screen gestures, but provided quick local access without the user having to lift a hand from the joystick. Two sliders provided the ability to zoom the camera and adjust the gain on the rotational velocity of the PTU control provided by the thumb.

Also as in first person shooter games, the controller on the left hand was used for the movement of the robot's base. In the thumb control area, movement to the top and bottom moved the robot forward and backwards respectively. The left and right of the circle rotate the chassis via skid steer to the left and right respectively. As with the camera movement, this control surface was fully proportional and permitted simultaneous operation of both degrees of freedom.

The thumb and index panel for the left hand also contained buttons and sliders for additional robot control. There were two buttons to toggle ADR mode and sound.¹ The ADR mode button provided identical functionality as the on-screen control for Window interfaces.

¹We hoped to have sound functionality working by the time of the experiment. Unfortunately the sound feature was not fully implemented so the button was non-functional.

Three buttons on the left hand panel also controlled the navigation mode of the robot. These were identical behaviors as described in Chapter 4 where the robot could be placed in shared, safe, and manual mode. In shared mode, the user input would be mixed with an algorithm that would seek out the most open space and would help guide the robot through tight passageways. Safe mode would give the user control of the robot and only slow or stop the robot if an impending collision with objects in the environment was detected. Finally, manual mode gave the user full control of the robot and provided no interaction or protection.

The user would create these joysticks by simply placing their hands on the surface of the screen. The size and controls would then be configured to the individual user's finger contact points. If the hand was removed from the screen, the joysticks were destroyed and the robot immediately stopped. The user could adjust the position of the DREAM Controller by lifting their thumb and dragging their index and middle finger to a different location of the screen at any time. The user could relax their ring and little finger once the joystick was created if they desired by lifting the fingers off of the surface.

6.3 Hypotheses

Before our experiment, we hypothesized:

H1: The new multi-touch proportional robot control widget in Surface:Window will increase task performance when compared to the Joystick:Window or DiamondTouch:Window interface. That is, we hypothesize that participants will be able to cover more area, find more victims, and create less damage to the environment.

H2: The new presentation in Surface:FullScreen will increase task performance when compared to the Joystick:Window, DiamondTouch:Window or Surface:Window interface. That is, we hypothesize that participants will be able to cover more area, find more victims, and create less damage to the

environment.

H3: Participants will prefer the Surface:FullScreen interface over the Surface:Window interface.

6.4 Experimental Design

In August 2007, we conducted a within subjects study of the Joystick:Window and DiamondTouch:Window interfaces with six participants from the emergency response field described in Chapter 4 (hereafter referred to as the “2007 study”). We concluded that the DiamondTouch:Window interface performed no worse than the Joystick:Window interface, thus establishing the baseline. In this experiment, we repeated the experimental protocol with six new emergency responders. The participants used the Surface:Window and Surface:FullScreen interfaces to drive the robot through a maze to locate victims.

To the extent that it was feasible, we created an exact duplicate of the Reference Test Arenas for Autonomous Mobile Robots at the National Institute of Standards and Technology (NIST) (Schipani and Messina, 2007) used in the 2007 study. A 2000 square foot maze was built using the same materials and construction in an controlled indoor environment. In place of mannequins and dolls, victim markers were 8.5 x 11 inch white pieces of paper with the letter “V” printed on them, as shown in the middle of Figure 6-4. The ten victim markers were placed in the same locations as the 2007 study, and two starting conditions were established.

We compared the task performance data of the 2007 study with this study. Since we had two independent participant pools (between subjects), unpaired *t*-tests were used when comparing the Joystick:Window and DiamondTouch:Window interfaces to the Surface interfaces. It would have been ideal to directly compare the Joystick:Window condition with the same participant pool as the Surface conditions. A within subjects experiment would help to directly correlate the performance of the two Surface interfaces to the Joystick:Window interface. Unfortunately, the



Figure 6-4: The participant is observed using the multi-touch interfaces (left) to search for victim marker using an ATRV-Jr robot (middle) while test administrators (right) record video and written documentation regarding the performance of the run.

nature of the USAR personnel demographic makes it extremely difficult to capture large numbers of personnel for multiple hours of their limited time. As such, our experiment design attempted to mirror the two testing environments as closely as possible to limit confounders and complications.

6.4.1 Procedure

For consistency between this experiment and the 2007 study, the procedure stayed the same. After signing a consent form, participants filled out a pre-experiment questionnaire requesting demographic information and probing their relevant experience with computers, robots, remote control vehicles, video games, and joysticks. We showed the participants what the robot looks like and then trained them on how to control the robot using one of the interfaces. We allowed the participants time to practice using the robot in a location outside the test arena and not within their line of sight so they could become comfortable with remotely moving the robot and the cameras. We then moved the robot to the start location in the arena and asked them to maneuver through the area to find as many victims as possible during a 25-minute period. We asked participants to “think aloud” (Ericsson and Simon, 1980) during the task so we could determine when participants were having trouble with parts of the interface and/or had a different

mental model of how the interface works than was intended by the designers. After task completion, an experimenter asked several post-run questions related to their experience with the interface. After a break, we then repeated these steps using the other robot interface.

We counterbalanced the experiment in two ways to avoid confounding factors. Three of the participants started with the Surface:Window interface and the other three started with the Surface:FullScreen interface. Additionally, we used two different starting positions (A and B in Figure 4-4) in the arena so that knowledge of the arena gained from using the first interface would not transfer to the use of the second interface. The two counterbalancing techniques led to four different combinations of initial arena entrance and initial interface.

6.4.2 Data collection

We collected four types of data: video, logs, observer notes, and annotated maps. In addition to a video of the robot's progress through the arena, we videotaped over the shoulder of each participant to capture his or her interactions, and we mounted a video recorder pointing down at the multi-touch table. Custom logging software captured each time the participants changed modes, moved the camera, or activated other controls. Two experimenters sat with the participant and hand-wrote observations. Finally, an experimenter following the robot manually marked its progress through the maze on a run sheet that also provided space to note when and where each bump, scrape, or "e-stop" (emergency halting of the robot) occurred. Figure 4-4 contains a reproduction of the map portion from a run sheet that shows numbers coded to specific incidents that are enumerated on the run sheet's second page.

6.4.3 Participants

Four men and two women with an average age of 43 years ($SD = 9.5$) participated in this study. All were active members of the USAR community with an average

of 8 years ($SD = 4.5$) experience in the field. Disciplines included technical information specialist, communications specialist, canine search, and search. One member was additionally a search team leader for a wilderness search and rescue group. All had used computers for five years or more, and four assessed their expertise to be at expert or better. All but one had some experience with touch or pen based computing technologies. Half of the participants played video games for an average of 4.5 hours per week ($SD = 4.4$) and half reported that they were pilots. Four of the six had never previously used robots. All participants were right hand dominant in their daily computing activities.

6.5 Results and Discussion

For the analysis of the experimental data, we performed a 4-way comparison on all of the task performance measures between Joystick:Window, DiamondTouch:Window, Surface:Window, and Surface:FullScreen. These are general task performance measures related to search coverage, victim identification, and damage to the environment. Through experience gained through the 2007 and other later studies, we incorporated additional usability metrics used in a 2-way comparison between Surface:Window and Surface:FullScreen interfaces.

6.5.1 Task Performance

The positive, or constructive, task performance of the interfaces was measured in two ways. First, we measured the number of victims found by the operator accounting for overlap in identification. If the participant identified a victim twice or more, the victim only counted as one “find.” Second, we measured the distance that the robot traveled in the area. This metric was computed slightly differently from the analysis reported in Chapter 4 due to the fact that both Surface interfaces allowed several participants to travel so far into the maze that they effectively “lapped” the maze in the allocated time. As such, the distance traveled for both studies is reported including areas that had already been searched. These two

Table 6.2: Area explored including overlap in the USAR arena (in squared feet).

Participant	2007 Study		Participant	2010 Study	
	Joystick: Window	DiamondTouch: Window		Surface: Window	Surface: FullScreen
1	576	352	7	736	960
2	512	320	8	1040	352
3	560	304	9	720	592
4	544	624	10	1056	912
5	464	544	11	192	352
6	896	752	12	736	928
Total	3552	2896	Total	4480	4096
Average	592.00	482.67	Average	746.67	682.67
SD	154.13	185.33	SD	313.15	288.65

metrics are closely related, since the farther that the robot can travel, the higher the search coverage and probability of detection. The results are shown in the top chart in 6-5.

Area Explored: We compared the area explored for the four interfaces. First, people drove farther using the Joystick:Window interface than the DiamondTouch:Window interface with weak significance ($p = 0.07, t(11) = 2.02$) using a one-tailed paired t -test with $\alpha = 0.05$. One should note that this result differs from the result presented in Chapter 4, when we found these two interfaces to have no significant significance. This change is due to the need to change the metric for area explored from non-overlapping to overlapping. Eliminating overlapping search areas is generally a good practice when using this performance metric, as it is important to cover new ground in a search task. However, as mentioned above, the distance traveled using the two Surface interfaces was artificially limited by the construction of the 2000 square foot course. As such, the revised metric was analyzed with the earlier data.

We found that people drove farther using the Surface:Window interface ($\bar{x} = 746.67, SD = 313.15$), than the DiamondTouch:Window interface ($\bar{x} = 682.67, SD = 288.65$) with weak significance ($p = 0.056, t(10) = 2.16$), using a one-tailed un-

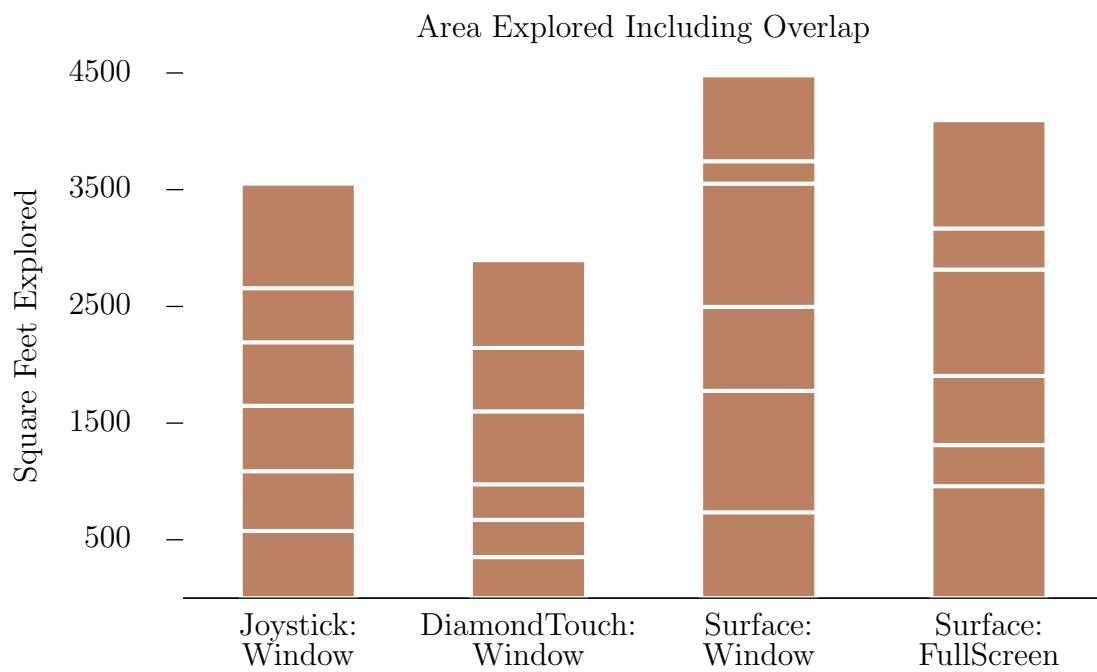
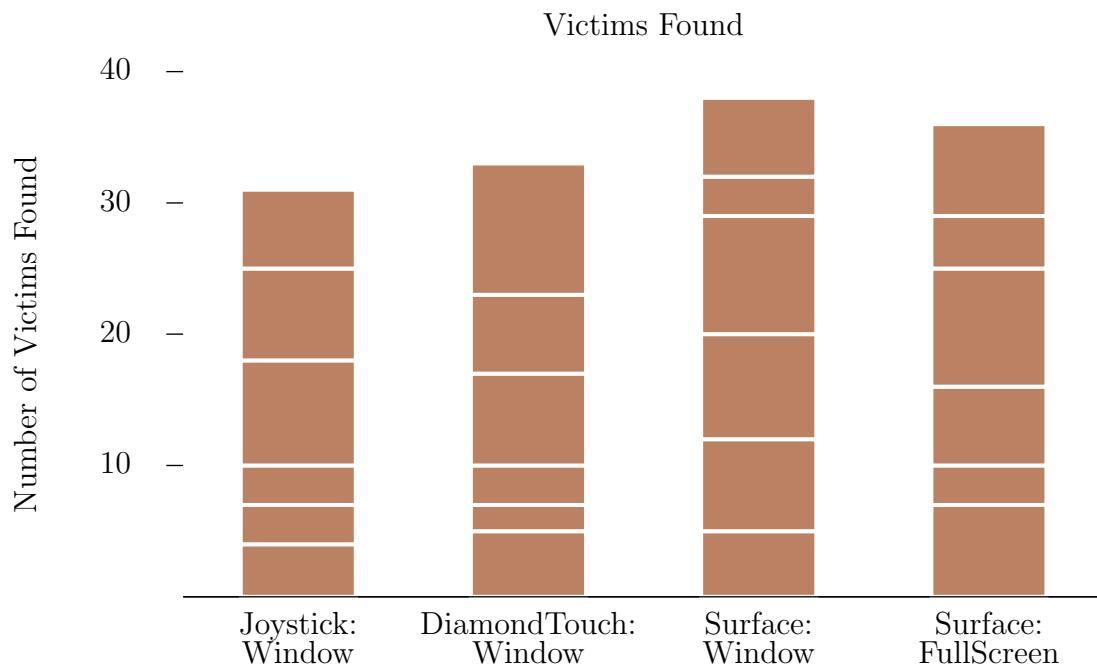


Figure 6-5: Participant performance was measured by counting the number of victims found (above) and the overall distance traveled in the arena (below). In the case of the Surface:Window interface, the participants were able to travel farther and find more victims.

paired *t*-test with unequal variance with $\alpha = 0.05$. We also found that people drove farther using the Surface:FullScreen interface ($\bar{x} = 682.67, SD = 288.65$) than the DiamondTouch:Window with weak significance ($p = 0.09, t(10) = 1.85$) using a one-tailed unpaired *t*-test with unequal variance with $\alpha = 0.05$. There was no significant difference in the area explored including overlap between the Surface:Window and the Surface:FullScreen interfaces ($p = 0.335, t(11) = 1.01$).

Based on the experimental design, the use of the DREAM Controller is most likely the contributor to this finding since the control method was the independent variable in the first three cases, and the presentation remained constant between the Window interfaces. In short, this result is an indicator that the human-centered design of the DREAM Controller is a significant step in the right direction when compared to traditional static widgets for robot control.

Qualitatively, this result is supported by several observations by the test administrator. One of the ways that participants move quickly through the maze is by focusing on the search task itself and not the robot controls on the screen. In virtually all cases, by the end of the first run, the participants appeared to be confident that the controller would reliably appear under their fingers in the correct configuration. This interaction was noted since it eliminated the need to look down to their hands and confirm that the controller was configured correctly and ready for input. By gaining the participant's confidence through the reliability of the hand detection algorithm, the new controller was able to take seconds off every hand placement. Over the course of the 25 minute run, this time adds up and cumulatively allow the participant to achieve higher levels of performance over a static widget. Much like the touch typist or pianist does not need to look at their hands to achieve significant performance, our participants did not need to look at the DREAM Controller to maneuver the robot through the maze.

A second observed contributor to the area explored was lack of use of on-screen controls outside of the DREAM Controller. Our interface design team spent a significant amount of time ensuring that the mixed interface metaphors of the DiamondTouch:Window interface were not repeated in the Surface interfaces.

Responsive and accurate on-screen proportional camera control is an example of interaction methods that were designed specifically to ensure that the participant did not have problems like we observed in Chapter 4 where proportional robot and discrete camera control confused participants. While these improvements alone should have helped align the interface with participant expectations, the participants more often used the controls on the DREAM Controller for camera movement, zoom, and resetting while driving. In effect, the Surface interfaces allowed the participants to not only interact more naturally with the on screen controls when not driving the robot, but also gave them a more efficient way of adjusting those controls while they were engaged in the manipulation of the virtual joystick controller.

The use of a single hand to trigger buttons and sliders on the DREAM Controllers was one final observation related to efficiency while driving. Before this user testing, we had assumed that the user would lift the opposite hand to trigger buttons on the DREAM Controller. For example, imagine that the user has the left hand engaged in the DREAM Controller and is using it to drive the robot. We assumed that one would use the right hand fingers to move the speed limited sliders or toggle the autonomy mode buttons while keeping the left hand in a static position. In many cases, the participants would angle their hand with the wrist at a slightly upward angle and toggle the nearby button with their thumb. Since the DREAM Controller had the ability to rotate and track the person based on their index and middle finger positions, participants found they could just quickly flip the position of these two fingers several degrees and place the buttons under their thumb rather than lifting their opposite hand. While not the most ergonomic technique, this quick and elegant trick permitted quick mode changes and slider adjustment outside of our original design intent.

Destructive Performance: We also measured the negative, or destructive, performance by recording the number of critical events in which the robot damaged the environment. This metric was categorized as pushes (the robot moves an obstacle away from its resting position), scrapes (some part of the robot brushes

Table 6.3: Number of destructive incidents per square foot in the USAR arena.

Participant	2007 Study		Participant	2010 Study	
	Joystick: Window	DiamondTouch: Window		Surface: Window	Surface: FullScreen
1	0.005	0.000	7	0.031	0.008
2	0.070	0.034	8	0.013	0.051
3	0.002	0.069	9	0.026	0.036
4	0.000	0.002	10	0.008	0.022
5	0.000	0.026	11	0.037	0.009
6	0.002	0.004	12	0.015	0.025
Average	0.013	0.023	Average	0.022	0.025
SD	0.028	0.027	SD	0.012	0.016

up against an obstacle), bumps (the robot impacts an obstacle), and e-stops (experimenters fear severe damage will occur to the robot if operations continue and press a button on the robot to stop it). Due to the fact that the distance traveled in this study was higher than the 2007 study, this performance metric was adjusted to accommodate the greater distance traveled. Rather than a straight count of events, we computed this metric using the number of critical events per square foot of distance traveled. This data is shown in 6.3. Note that the numbers vary widely in both studies and, in the case of the 2007 study, the standard deviations were larger than the average values.

We found that the number of destructive incidents per square foot showed no statistically significant difference across all four interfaces. The consistency in the number of destructive incidents is a positive finding since it indicates that neither of the Surface interfaces dramatically decreased performance or safety of the environment while operating the robot. We believe that all of the performance benefits and observations in the above discussion on Area Explored also apply here.

Victims Located: The number of victims over all participants for the Surface:Window interface for victim detection showed a 23% and 16% increase com-

pared to Joystick:Window and DiamondTouch:Window respectively (Figure 6-5). The Surface:FullScreen interface showed similar increased performance with a 16% and 9% increase in victims when compared to Joystick:Window and DiamondTouch:Window respectively. In Yanco and Drury (2004), it was found that robot operators on average used 47% of their time operating the camera to look around the environment. If the participant was more comfortable moving the camera while moving through the environment when compared to the Joystick:Window and DiamondTouch:Window interfaces, then it follows that the participants would be able to keep the destructive performance on par while increasing the distance traveled and number of victims located.

However, several factors in this experiment did not allow the data to achieve statistical significance to support the claim that victim detection performance increased. As mentioned above, the overall distance traveled for the Surface interfaces allowed many of the participants to visit areas that they had already searched. Since victims can only be counted once, the participant could no longer receive credit for duplicate victims found. It is not known how many victims that the participants neglected to identify when they realized that they had looped back to their starting location (and several of them did realize they had looped), so this data cannot be easily generated from the raw data set post-hoc. To mitigate this problem in future user testing, the maze needs to be enlarged to exceed the maximum performance expected from the participants.

As a result of the area explored, incidents, and victim performance metrics above, we are unable to support H0 (Surface:Window will increase task performance when compared to the Joystick:Window or DiamondTouch:Window) and H1 (Surface:FullScreen will increase task performance when compared to the Joystick:Window, DiamondTouch:Window or Surface:Window interface) given the task performance data. However, the summative data seems to indicate that our approach is in the right direction particularly since we observed that participants were able to drive farther using the Surface interfaces than the DiamondTouch:Window interface. Also, the number of destructive incidents per

Table 6.4: Victims found in the USAR arena.

Participant	2007 Study		Participant	2010 Study	
	Joystick: Window	DiamondTouch: Window		Surface: Window	Surface: FullScreen
1	4	5	7	5	7
2	3	2	8	7	3
3	3	3	9	8	6
4	8	7	10	9	3
5	7	6	11	3	4
6	6	10	12	6	7
Total	31	33	Total	38	36
Average	5.17	5.50	Average	6.33	6.00
SD	2.14	2.88	SD	2.16	2.19

square foot was not statistically different across all four interfaces, supporting that we are not impairing performance.

Quantitatively, the testing administrators did notice one major contributor to support that the participants were working more efficiently and able to more easily identify victims. In the earlier Joystick:Window and DiamondTouchWindow interfaces, the participants were largely driving or searching throughout their run, but rarely both at the same time. The top-hat camera control in the Joystick interface allowed the actuation of drive and camera control at the same time, but this hand movement was difficult and ergonomically unnatural for most people. The DiamondTouch interface also allowed simultaneous drive and camera control, but both required the participant to look at their hands to ensure they were touching the right position on the screen in the absence of tactile feedback. As a result, in both cases, we rarely saw the participants confidently using both position controls simultaneously.

In the Surface interfaces, the DREAM Controller allowed the participants to rest both of their hands on the surface and engage all four degrees of freedom without ergonomic awkwardness or the need for visual positioning of the finger. This hand position allowed them to operate the robot control and the camera

at the same time. In all six users, we observed continuous periods during their runs where both hands were fully engaged with the surface and their thumbs were moving, or ready to move, simultaneously. This observation beckons to the need for “chording” motions as described in the beginning of Chapter 5. In our own bodies, we are able to simultaneously walk and look around our surroundings in a natural and intuitive fashion. This ability increases our situation awareness of the world around us and is a characteristic of most successful land creatures. As such, we should not be surprised to see them moving, looking, and increasing their situation awareness when the design of the controller makes this possible.

6.5.2 Usability Performance

According to Nielsen, the usability of a system can be measured by its learnability, efficiency, memorability, satisfaction, and resistance to errors. Nielsen (1993) defines the composition as follows:

“Learnability: The system should be easy to learn so that the user can rapidly start getting some work done with the system.

Efficiency: The system should be efficient to use, so that once the user has learned the system, a high level of productivity is possible.

Memorability: The system should be easy to remember, so that the casual user is able to return after some period of not having used it, without having to learn everything all over again.

Errors: The system should have a low error rate, so that users make few errors during the use of the system, and so that if they do make errors they can easily recover from them. Additionally, catastrophic errors must not occur.

Table 6.5: Participants' subjective assessment.

Semantic differential scale Scale range 1 / 5	Surface:Window		Surface:FullScreen	
	\bar{X}	SD	\bar{X}	SD
Hinder / Help	4.7	0.5	3.7	0.8
Difficult / Easy to learn	4.5	0.6	3.8	1.0
Difficult / Easy to use	4.3	0.5	3.8	0.4
Irritating / Pleasant	4.2	0.4	3.5	1.0
Uncomfortable / Comfortable	3.7	0.8	3.8	1.0
Inefficient / Efficient	4.5	0.5	4.0	0.6

Satisfaction: The system should be pleasant to use, so that users are subjectively satisfied when using it; they like it.” (Nielsen, 1993, pg. 26)

When placed in the context of usability performance, Nielsen’s measures can provide a useful framework for interface comparison. When we designed the experiment for this study, we wanted to make sure that these data points were captured through observations and qualitative questions for the participants. As such, the following analysis is a usability comparison of the Surface:Windows and Surface:FullScreen interfaces.

Learnability: We measured learnability through the number of clarifications the participant requested during the run regarding the use of the interface. Examples of this might include questions about the interface and corrections about interface assumptions. Participants received fewer clarifications about the interface using the Surface:Window interface ($\bar{x}_{S:W} = 4.67$ clarifications, $SD = 1.90$) versus the Surface:FullScreen interface ($\bar{x}_{S:FS} = 6.67$, $SD=2.25$). Using a one-tailed paired t -test, the results show weak significance ($p = 0.08$, $t(11) = 1.90$).

Participants asked a total of 34 questions for which they received clarification. Only two of which were related to the interaction with the interface; both questions were asked when the participants were using the Surface:Window interface. Participant 3 asked if it mattered that his hand touched the distance panel when

summoning the DREAM Controller. Participant 4 asked if his thumb needed to be outside the blue joystick controller area when selecting buttons on the DREAM Controller panel.

The remaining questions could be categorized into categories relating to the interface and the robot system; note the one question could have more than one categorization. Participants asked questions relating to details of pmap (9 questions), the distance panel (9 questions), the video panel (4 questions), and ADR mode (4 questions). There were eight questions relating to the hardware capabilities of the robot system, and two about the status of the robot.

Efficiency: In the case of this experiment, the participants were only able to use the interface for 25 minutes. In that amount of time, a good measure of efficiency is not realistic since the participants are still in a learning phase. Since it cannot be quantitatively measured through a direct performance metric, we measured the perceived efficiency through two semantic differential questions asked upon the completion of the task (i.e., on a scale from 1 to 5, how did the interface hinder/help in performing the task, and how inefficient/efficient was the interface to use). The participants reported that the Surface:Window interface helped them in performing the task significantly better than the Surface:FullScreen interface ($\bar{x}_{S:W} = 4.7$, $SD = 0.5$; $\bar{x}_{S:FS} = 3.7$, $SD = 0.8$; $p < 0.01$, $t(11) = 3.41$), using a one-tailed paired t -test with $\alpha = 0.05$. The participants reported that both Surface interfaces were efficient to use ($\bar{x}_{S:W} = 4.5$, $SD = 0.5$; $\bar{x}_{S:FS} = 4.0$, $SD = 0.6$), with no statistically significant difference.

Memorability: Immediately following each run, the participant was asked to recall all of the gestures and describe the resulting actions. Participants were not instructed to remember these answers for later use. Each of the participants was contacted ten days after the experiment and asked to recall the gestures used to control the interface.

Understanding is a side-effect of the memorability measure, since the number of correct gestures immediately after the experiment can indicate missed metaphors

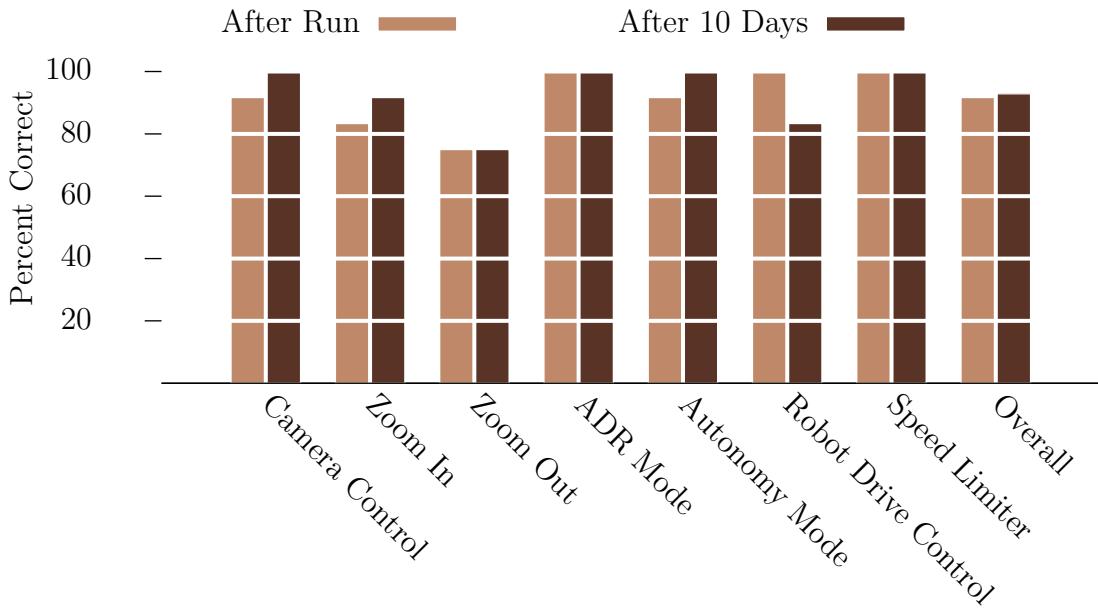


Figure 6-6: Memorability was measured by testing the recall of participants immediately after the experiment run and then ten days later. The percentage of correct answers was recorded and compared to assess the memorability of the robot control mechanisms.

or affordances. As such, understanding was measured as the number of correct gesture descriptions immediately after the experiment over the total number of gestures.

The percentage of questions answered correctly in each of the interface categories is shown in Figure 6-6. Overall, the percentage of correct recall after the run and ten days later was 91.7 and 92.7 respectively. This indicates that the interface design is highly memorable, as the users were able to recall more than 90% of the interface functionality ten days after using the system. Learnability of an interface is highly desirable to the USAR demographic, since responders may have limited training time and need to recall functionality quickly during emergency use.

We found a statistically significant learning effect for the zoom out command ($p < 0.04, t(11) = 2.36$) for the first interface the participants used versus the second. This effect was compared using a one tailed paired t -test with $\alpha = 0.05$. This finding simply indicates that by the second 25 minute run, the users were better able to recall how to use the zoom functionality of the interfaces. This

effect was enhanced due to the fact that the zoom was not widely used across the participants' runs. The close proximity of the walls of the maze did not require the use of the zoom to identify victim markings. As such, this feature was not used much by the participants and therefore took longer for retention to occur. There were no other statistically significant learning effects.

Errors: Errors are considered with respect to the usage of the interface, and not errors relative to the robot and the environment. An example of an error may include unintended events such as triggering ADR mode or not being able to reset or zoom the camera. An interface error is an error by the system; the participant executed the correct gesture, but the system did not register that gesture. A gesture error was counted when the participant did not execute the gesture correctly.

There were a total of 89 interface errors and 28 gesture errors when the participants used the Surface:Window interface ($\bar{x}_{interfaceerrors} = 14.83$ errors, SD=13.47; $\bar{x}_{gestureerrors} = 4.67$ errors, SD=5.39 respectively). For the Surface:FullScreen interface, there were a total of 70 interface errors and 32 gesture errors when the participants used the Surface:Window interface ($\bar{x}_{interfaceerrors} = 11.67$ errors, SD=8.85; $\bar{x}_{gestureerrors} = 5.33$ errors, SD=4.13 respectively). There was not a statistical difference between the total number of errors, interface errors, or gesture errors.

The most common gesture error seen was single tapping the rear view mirror for ADR mode instead of double tapping; single or triple tapping the thumb on the right DREAM Controller to recenter the video instead of double tapping; and tapping the index finger on the DREAM Controller as if there were a button below the finger. The most common interface errors were not registering double tap for re-centering the camera on both the right joystick and on the video panel, and interpreting the heel of a hand when a joystick was activated as the point to which the camera should servo.

Fortunately, these errors can be easily fixed and should dramatically increase the consistency of the user experience. In all of the cases where single, double, or triple taps were misinterpreted, these tap sequences can easily be adjusted to receive any number of taps to toggle the state of the display. We learned from this experiment and applied this lesson to the selection of robots in the interface that is described in Chapter 8. The errors related to the heel of the hand can be solved through a more accurate heuristic for the detection of a finger, as opposed to any part of the body contacting the surface.

Finally, the error related to the index finger tapping on the DREAM Controller is an interesting one. In this case, the circular pad underneath the finger did nothing to affect the robot or the interface so there were no direct negative impacts. What this does indicate is a willingness to use the index finger as a button should the complexity of control require functionality out of these fingers. Since many of the game controllers use “shoulder” buttons for the same purpose, we will be able to leverage all of the outside learning and muscle memory benefits that we see with the thumb controls.

Satisfaction: We measured satisfaction through two subjective semantic differential scale questions after the study (i.e., on a scale from 1 to 5, how difficult/easy was the interface to use, and how irritating/pleasant was the interface to use). The participants reported that the Surface:Window interface was more pleasant ($\bar{x}_{S:W} = 4.2$, $SD = 0.4$) and easier to use ($\bar{x}_{S:W} = 4.3$, $SD = 0.5$) than the Surface:FullScreen interface ($\bar{x}_{S:W} = 3.5$, $SD = 1.0$; and $\bar{x}_{S:FS} = 3.8$, $SD = 0.4$ respectively). We believe that with a larger participant pool that these findings would have been significant ($p_{pleasant\ to\ use} = 0.12$, $t(11) = 1.70$; and $p_{ease\ of\ use} = 0.10$, $t(11) = 1.79$ respectively).

Finally, the participants were also asked directly which interface they preferred. This preference provides an overall assessment of satisfaction and also indicates the interface that was the most “right” from the participant’s perspective. Five out of the six participants reported preferring the Surface:Window interface over

the Surface:FullScreen interface, therefore we cannot support H2 (Participants will prefer the Surface:FullScreen interface over the Surface:Window interface).

We believe that this preference was due to the occlusion of the video by the hands and on-screen indicators. In the case of the Surface:FullScreen, the participants interacted on top of the video with their hands and all of the visual elements were rendered on top of the video. The occlusion by the hands is unlike the heads-up displays used to inspire this type of interface. Also, since the participants were using the video for searching, the visual elements on top of the video may have cluttered their view of the surrounding area. The Surface:Window interface did not suffer from these problems since the video and sensors were separated and there was unobstructed areas where the participant could place their hands and use the DREAM controllers.

6.6 Impacts and Implications

At the end of Chapter 4 for the 2007 study, analysis showed that people interpreted the joystick widget in a number of different ways, leading to sub-optimal usage. The DREAM controller did not exhibit these interpretation problems since explicit visual affordances were not used. While we were not able to support our original hypothesis, there is a strong indication that this anatomy-based design is a good approach and additional improvements should be investigated. The small population size, unpaired participants across all four interfaces, and wide variability in task performance make statistical significance particularly difficult, but we strongly believe that using representatives from the target population is extremely vital in these early stages of development even if the consequence is a smaller population sample.

Two main points did arise from this preliminary exploration. First, the design of the interface achieved the qualitative benchmark for ease of learning (4.5 out of 5) and an extremely high score for memorability (over 90%). These two features were key criteria in the design of the controller and interface and those benchmarks

were achieved. Second, victim detection and area explored did increase over the performance found in Chapter 4. This finding is a good indicator that multi-touch technology can be used in the place of traditional joystick control and that there may be performance and ergonomic benefits for the user.

A common and reasonable negative reaction to multi-touch interfaces is based around the objection that high precision control requires direct mechanical feedback to the operators' hands. In the case of classic joystick control, this long standing belief is embodied in the volumes of ergonomics literature and decades of successful product design. The tension of the spring-loaded gimbals, the debounce of the buttons or triggers, and the shape of the control stick are just a few examples of ways engineers have tuned the "feel" of a joystick to maximize the sensory return.

After careful investigation using the DREAM Controller as an existence proof, we believe that the need for direct mechanical feedback is not the only way to achieve high precision control. In fact, we believe that a departure from traditional mechanical input device design in favor of multi-touch interaction will not only maintain or increase performance, we believe that a significant increase in ergonomics and posture will also result from this change.

One of the reasons that the mechanical feedback becomes so important in traditional joystick design is because the user needs a clear way to conform to the shape of the control surface and understand the position of the control mechanism is in its various degrees of freedom. When this conformity is sufficiently succinct, the psychology literature calls it an affordance (Norman, 1988). Affordances are one or multiple properties that give some visual indication of how to interact with an object. Properly designed door knobs *afford* the property of being turned and pulled. Buttons on a control panel *afford* the property of being pushed. Switches *afford* the property of being flipped. These elements of design walk the line between engineering and aesthetics that, when executed properly, can become sublime and appear to be the "correct" answer regardless of prior experience or bias.

Once the users' hands have conformed to the device, the spring tension and other mechanical feedback properties allow the user to look away from the joystick

and concentrate on the task. The user can trust that the nerves in their hands and arms will feel the position of the joystick and they will not have to repeatedly look at their hands to verify that their input to the system is correct. This learned behavior is directly analogous to a touch typist or pianist, who after enough practice, can interact with the mechanical device without looking at their hands and achieve considerable levels of performance over time.

Just as our users bring biases and multiple metaphors to new interfaces, we as engineers come to the proverbial design table with preconceived ideas of how we are going to overcome the lack of dimensionality and physical interaction in multi-touch interface design. Unfortunately, when working on a 2D glass surface, designing (or visually emulating) physical affordances may not be the best approach. Flattening the 3D world to a 2D screen and expecting the same affordances while (by virtue of the device) eliminating the mechanical feedback is a strategy doomed for failure. In the earlier example of the touch typist or pianist, it is not unexpected that a literal 2D projection of a keyboard or piano on a multi-touch surface would not provide the same performance as the real physical device.

As demonstrated by the DREAM Controller, a design approach that centers closely around the bio-mechanical design of the human hand may be an appropriate solution. This focus is not in the traditional mechanical design for physical input devices where the designer attempts to find the most correct design for the largest number of people. Instead, the interface should *conform* to the *user* individually every time that their hands touch the control surface. Hand sizes, finger lengths, and degrees of dexterity are all variables between individuals. Additionally, all of these properties change as the user fatigues while using the interface for extended periods of time. So, even within a single user experience, there may be multiple optimal interaction configurations to be employed as the interaction progresses.

In a presentation in 2006, Jeff Han provided a very succinct argument for user centered design on multi-touch devices.

“Now, when you have initiatives like the 100 dollar laptop, I kind of cringe at the idea that we are going to introduce a whole new generation

of people to computing with this standard mouse-and-pointer interface.

... there is no reason in this day and age that we should be conforming to a physical device. That leads to bad things like RSI. We have so much technology nowadays that interfaces should start conforming to us (Han, 2006)."

It is the ability to dynamically adapt to the users' configurations that gives multi-touch interaction a significant advantage over traditional mechanical device design. Mechanical devices cannot dynamically conform to the user on every instance of interaction. They cannot instantly reconfigure themselves for different user profiles and abilities. Multi-touch interaction can adjust dynamically and change on every contact with the surface. It just takes a departure from traditional design methods based around physical affordances to make this happen.

The DREAM Controller provides this interaction by "wrapping" the joystick around the fingers of the individual user. The size of the thumb control is automatically sized based on the size of the users hands. Buttons, dials, and control surfaces are all tailored specifically for the user's comfort and performance. Even in the case where the user moves his or her hand to a different location of the screen, the controller will dynamically track to the new location and position itself underneath the user's fingertips.

While testing the DREAM Controller, we observed users who had never interacted with a multi-touch device controlling four degrees of freedom without looking at their hands during their 25 minute runs. The advantage to our user-centered approach was illustrated through the realization of one of our participants when he explained, "It is taking me a little while to understand that the joystick is going to conform to me and not the other way around. It is a little strange, but I like it." Just as touch typists and pianists take time to trust that their hands and muscle memory will act correctly, the users willingness to trust that the interface will act correctly in the absence of mechanical feedback will increase over time. When this does occur, new levels of performance and ergonomic comfort will be the result.

Chapter 7

User Defined Gestures

Historically, multi-touch interfaces have used carefully designed gesture sets and UI elements. The gesture sets are often tailored around detectability and repeatability. These requirements vary depending on the enabling multi-touch technology and the capabilities of the touch sensor mechanisms. Despite the best intentions of the system designers, often the detectability of a gesture is at odds with its ease of learning. In an ideal setting, a novice user should be able to begin interacting with the multi-touch interface quickly, naturally, and without explicit instructions. In the case of command and control for military operations or disaster response, ease of learning is especially crucial since the commanders typically do not have an abundance of time to learn new user interfaces and must be able to quickly achieve operational proficiency.¹

To maximize the ease of learning for a command and control interface for teams of autonomous robots, this chapter aims to find the most natural gestures for controlling robot teams, regardless of detectability or input technology. We designed an experiment in which the participant was presented with a number of tasks that had varying numbers of robots and the need for different types of control. With no other visual user interface elements such as menus and windows, the participant was asked how he or she would express the task to the robot(s). The result is a unique look at how users wish to control multi-agent teams.

¹Portions of this chapter appear in (Micire, Desai, Courtemanche, Tsui, and Yanco, 2009a)



Figure 7-1: Participants were seated in front of a Mitsubishi DiamondTouch fastened to a round table where they provided gestures on a static front projected image.

7.1 Experiment Design

Our goal was to determine the gestures participants would use naturally. The experiment is purely exploratory. It was an effort to find “first impressions” from users that had not interacted with this type of input device. The tasks were designed to elicit responses from the participants that were free-form and not constrained by pre-determined graphics and visual feedback. Care was taken to avoid standard UI window conventions such as menus, title bars, and buttons. In this regard, the experiment can be considered analogous to the paper prototype method often used in early user interface designs (Snyder, 2003).

For this experiment, we used a 64×48 centimeter Mitsubishi DiamondTouch by Circle Twelve and a Dell 5100MP projector. The DiamondTouch was securely fastened to a round wood tabletop with a large steel base, ensuring that participants could interact naturally and rest their arms without accidental movement. The projector was fastened to the ceiling with a custom mount and front reflecting



Figure 7-2: Participants were presented with a physical ActiveMedia Pioneer 2Dx robot (left) that was iconically depicted in the experiment slides (right).

mirror for image adjustment. An overhead camera was secured on a custom camera mount that aligned the camera field of view with the projected image and tabletop.

7.1.1 Participants

Thirty-one people participated in the study; each received a movie ticket. The average age was 27.5 years ($SD=10.1$), and nine of the participants were female. All of the participants had some experience with computers. Seventeen participants reported playing video games for an average of 7.8 hours per week ($SD=7.4$). Nine of the video game players reported that they played real time strategy (RTS) games such as StarCraft, Civilization, and Sins of a Solar Empire.

All but two participants reported prior experience with touch screen or stylus-based devices. Eighteen had experience with some type of touch screen phone; sixteen of these were the Apple iPhone. Sixteen participants had used a Palm OS stylus device and thirteen had experience with tablet-based PCs.

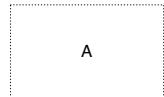
7.1.2 Procedure

Each participant was first briefed on the experiment and introduced to the physical robot, an ActiveMedia Pioneer 2Dx, that would be iconically depicted in the experiment as shown in Figure 7-2. After answering any questions, the participant

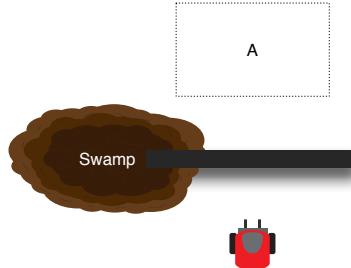
completed an informed consent form and a demographic survey. The participant was then presented with the tasks and asked to “think aloud” (Ericsson and Simon, 1980) while completing them.

For each task, the participant was presented with a slide with a written description of the task at the top of it; the experimenter also verbally stated the task. Participants were asked to use their finger, fingers, hand, or hands on the tabletop to express how they would command the robot(s) to complete the tasks; there was no time limit on responses. We videotaped the participants’ interactions and commentary using the overhead camera and logged the movements using custom software. The experimenter also took notes. In addition to the think aloud protocol, the participant was asked to talk about any aspects of the interface that he or she would expect if the interface were active. These could include, but were not limited to, menus, dialog boxes, or interface features.

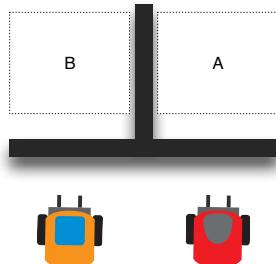
Table 7.1: Illustration and descriptions of some of the 26 tasks performed by participants. Similar tasks have been omitted from this table due to size constraints, but full descriptions are given in the text.



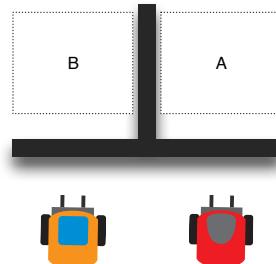
(1-2) Command the robot to area A.



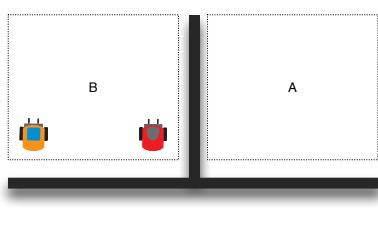
(3) Command the robot to area A.



(4-5) Command the red robot to area A and orange robot to area B.



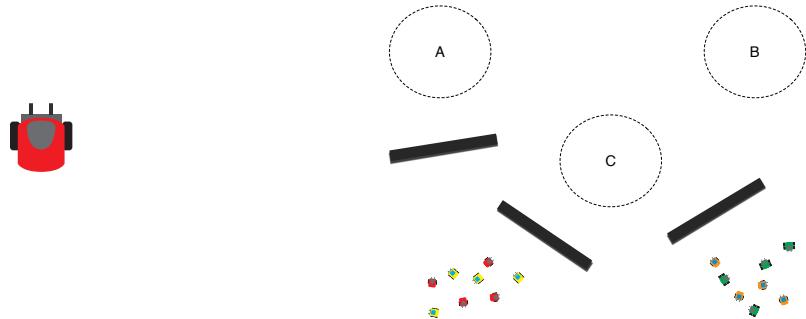
(6-7) Command both robots to area A.



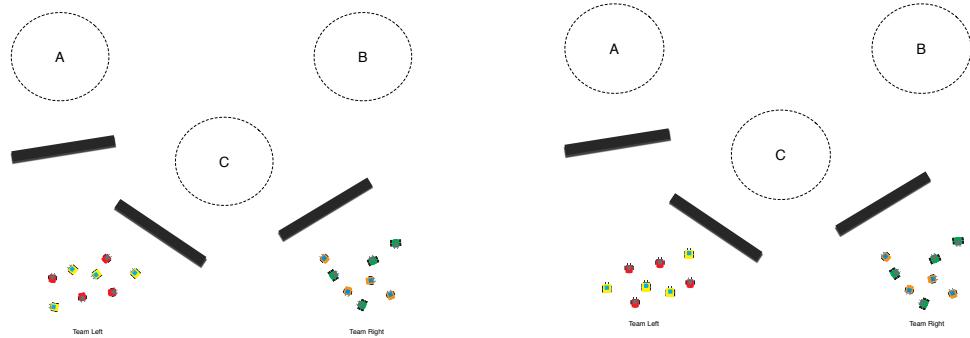
(8-9) Command the red robot to go below the wall and the orange robot to area A.



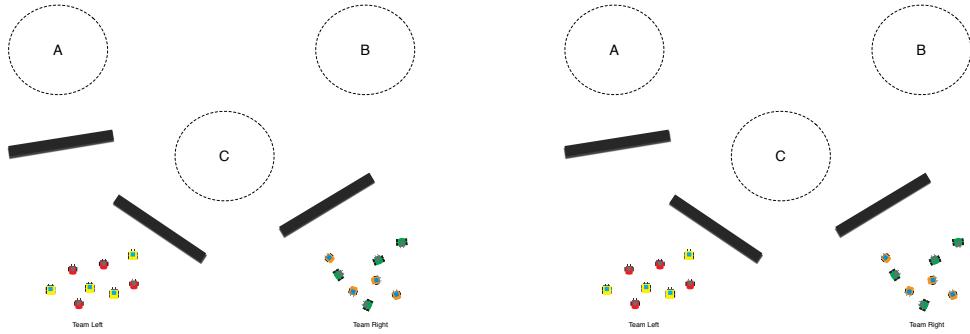
(10) Command the robot to go straight and keep going straight.



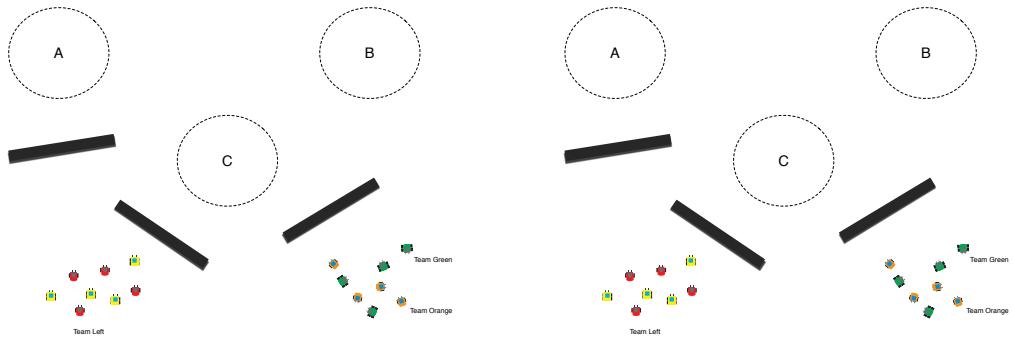
- (11-12) Command the robot to face the opposite direction.
 (13) Create a team consisting of all robots on the left side and label them as "Team Left" and on the right as "Team Right."



- (14-15) Have all the robots in Team Left face area A.
 (16) Command all the robots in Team Right to area B.

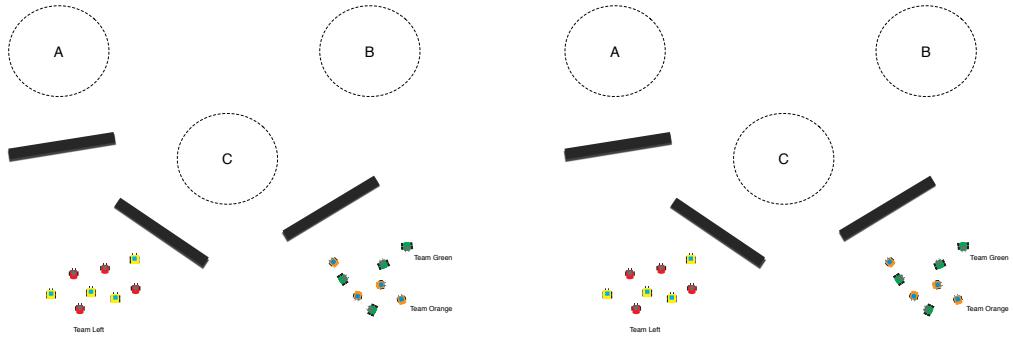


- (17) Command all the robots in Team Right to area B and back.
 (18) Split Team Right into orange and green robots and label them as Team Orange and Team Green.



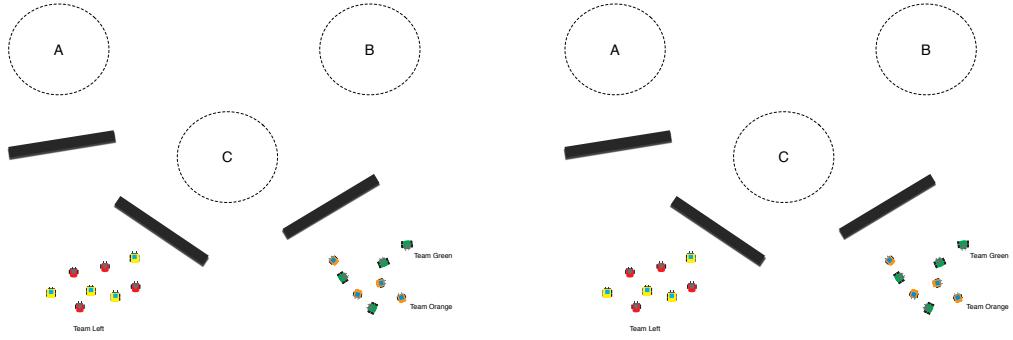
(19) Command Team Left to area A, Team Orange to area C, and Team Green to area B.

(20) Command Team Green to loop around area C one time and then command them to area B.



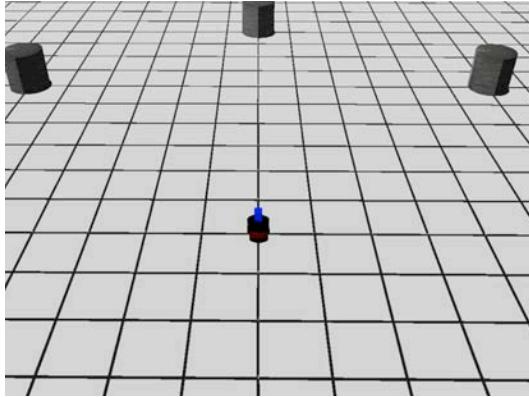
(21) Reorient the map so that area A is at the center of the screen.

(22) Turn the map 90 degrees counter-clockwise.

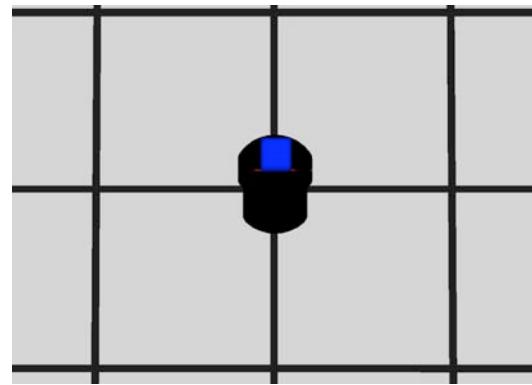


(23) Zoom into the map.

(24) Have all of the robots in Team Left form a horizontal line facing area A.



(25) You are viewing the robot from behind. Change to a view where you are looking from the front.



(26) You are viewing the robot from the top down. Change to a view where you are looking from behind.

7.1.3 Tasks

Twenty-six slides were presented sequentially which showed either one robot, two robots, or two teams of 8 robots each (shown in Table 7.1). The tasks on the slides introduced a range of desired interactions for robot command and control. Some tasks required very little interaction, while others forced the participant into situations where multiple hands or complex UI strategies were required. Twenty-four showed top-down views of the robot(s), while two showed 3D views. The slides are shown in Table 7.1. The omitted slides were visually identical and had small changes in the required task (detailed below).

The first three tasks involved only one robot and provided a simple starting point for the participants' understanding of the experiment and talk aloud method. Task 1 displayed only one robot and one labeled area; the participant was instructed to move the robot to the labeled area. Building on the previous task, Task 2 added a wall between the robot and the destination, requiring the participant to either express that the robot needed to go around the wall or make the assumption that the robot was capable of finding a path around the wall on its own. Task 3 extended this one step further by displaying a impassable “swamp” area on the left side. Again, the participant needed to express that the robot should travel around the wall, but not through the swamp.

Two robots were controlled in Task 4 through Task 9. Tasks 4 and 5 asked the participant to command each robot to a separate area. Task 5 asked the participant to command the robots at the same time, encouraging some type of group gesture, multi-touch interaction, or command queuing. Tasks 6 and 7 extended this idea by having the participant command both robots to the same area. This variant was explored since it could allow the participant to use group selection and a single destination gesture. Like Task 5, Task 7 asked the participant to perform the task for both robots at the same time. Tasks 8 and 9 displayed the robots in an area to the left and required the participant to move them to different locations on the screen. If the participant was using multi-handed gestures, this sequence created an arm crossing situation. Again, Task 9 required concurrent actions.

Task 10 asked the participant to simply command the robot to move forward and continue moving. Since there is no destination, this task asks the participant to form a gesture that had no predicate. Tasks 11 and 12 asked the participant to label the robot and rotate the robot respectively.

Groups of robots were displayed in Tasks 13 through 24. In Task 13, the participant was asked to give each of the teams of robots a name. This task required the participant to think about grouping and group selection. Tasks 14 and 15 then asked the participant to have a robot team face a specific area on the map, which extended the group selection gesture to now include a group action. Task 15 was identical, but asked for the team to face the direction at the same time. In Task 16, the team on the right side of the screen was then required to move to Area B. This iteration required group selection, position change, and destination designation. Task 17 then took the previous task one step further and asked the participant to command the team on the right to Area B, and back. This combined action required some sort of command queuing since there were two destinations.

Task 18 was unique since the participant needed to sort the robots and label them by color. Then for Tasks 19 and 20, the participant needed to maneuver the robot groups to various areas and paths on the map. Tasks 21 through 23 explored

the gestures to describe map translation, rotation, and zoom. Task 24 asked the participant to command the robots into a line formation, which required a hybrid between group selection and the need for independent movement of robots.

The final two slides were rendered in 3D to explore how participants would control the viewpoint of the robot using only a 2D tabletop surface. In Task 25, the view was from the rear and slightly above the robot; the participant was asked to rotate to a forward view of the robot from the same angle. Task 26 displayed the robot from above; the participant was asked to adjust the view so the robot was seen from behind.

7.2 Taxonomy of User Defined Gestures

We began data analysis by looking for large-scale patterns of movement in the overhead video. We refined our description of the patterns of interaction by isolating components of actions using open and axial coding from grounded theory (Glaser and Strauss, 1967). Open coding involves analysts noting data that can be grouped or categorized while axial coding is a process of refining the groupings.

After several iterations of group discussions and individual video analysis, clear patterns were seen across the majority of the participants. We coded instances of participants' gestures to the consensus of the gesture classifications. Inter-rater reliability was established using Cohen's Kappa statistic ($\kappa=0.74$ excluding chance, $\kappa=0.76$ if chance was not factored out). The data set provided a total of 3197 gestures over 31 participants and 26 tasks.

We identified five groups of interaction classifications for these 26 tasks: selection, position, rotation, viewpoint, and user interface elements.

Selection gestures were used to select a robot, multiple robots, other objects of interest in the environment. For example, a common occurrence of selection was when the participant tapped on a robot (selecting the robot to move) and then dragged their finger on the path that they would like the robot to follow. The initial finger tap on the robot would be classified as a selection.

Table 7.2: Taxonomy of user generated gestures based on 3197 gestures over 31 users and 26 tasks.

	Name	Description
Selection	Tap	Single finger taps object to be selected (See Sequence select for multiple taps)
	Double tap	Single finger double taps object to be selected (See Sequence select for multiple taps)
	Lasso	Single finger draws line encompassing objects to be selected
	Meta	Object selected with some external modifier (e.g. Ctrl, Alt)
	Sequence select	Objects selected in a serial fashion (Supersedes Tap and Double Tap)
	Press & hold	Object touched for a duration of longer than 1 second
	Bounding box	Opposite corners of bounding box are shown with fingers
	Palm	Palm of hand placed on object or objects
	2-finger select	Two fingers on the same hand simultaneously used for selection (Supersedes Tap)
	<i>n</i> -finger	More than two fingers on same hand used simultaneously for selection (Supersedes Tap)
Position	Drag	Single finger slides across surface to robot destination with immediate lift at end
	Drag & hold	Single finger slides across surface to robot destination with finger hold greater than one second at end
	Waypoint	Tap sequence providing waypoints for robot to follow ending at destination
	Pinch & move	Two finger pinch and then position change to robots' destination
	Flick	One or more fingers placed on robot and finger tip(s) accelerated rapidly in direction of movement
	Path to edge	Finger placed on object and dragged to the edge of screen in direction of movement
	Arrow	Arrow-like gesture drawn with arrowhead at end of vector
	Direction segment	Like drag, but smaller segment (vector) not terminating at goal
	Palm drag	Palm placed on object and dragged
	2-finger drag	Two fingers on the same hand are simultaneously used for drag
Rotation	<i>n</i> -finger drag	More than two fingers on the same hand used simultaneously to perform drag
	Finger rotate	Finger placed on object and finger tip rotated
	Pinch & rotate	Two finger pinch and then rotation change
	Off center rotation	Finger placed on object outside of center of mass and rotated
	C-style rotation	Finger begins in the center of the object, extends outward, and begins rotation
	Palm rotation	Palm placed on object and rotated
	2-finger rotation	Two fingers from the same hand placed on the object and fingers rotated
Viewpoint	<i>n</i> -finger rotation	More than two fingers on the same hand used simultaneously to perform rotation
	Pinch	Thumb and finger(s) converging using one hand
	Spread	Thumb and finger(s) diverging using one hand
	Finger pinch	Two or more fingers converging using two hands - one or more finger per hand
	Finger spread	Two or more fingers diverging using two hands - one or more finger per hand
Elements	Vanishing point	Hands placed on side parallel to each other and then angle outward
	Menu selection	Menu appears with more than one object property or action
	Button selection	A button selected by pressing on it, allowing for object modification or action
	Keyboard	A keyboard appears for annotation
	Handwriting	Handwriting recognition modifies object
	Voice recognition	Voice recognition modifies object
	Widget	A widget verbally described and interacted via specialized functionality

Position gestures indicated a desired change in location of the robot or some object. In the previous example, the drag movement of the finger providing the path would be classified as a drag representing a position change.

Rotation gestures expressed rotation of robots, objects, or the map. Many of the tasks required the reorientation of the robots to face areas of the map. In the simplest of cases, participants used a two finger rotation gesture over the robot image. Other participants creatively used single finger gestures where the rotation was outside of the center of mass of the robot.

Viewpoint gestures were used to change the view by either moving the world or changing the position of the camera. Most commonly, either case was achieved using a single finger on each hand to reorient the screen. Several participants came up with unique and unexpected gestures to accomplish the tasks.

User Interface Elements included commonly used interface elements like buttons, menus, virtual keyboards, handwriting recognition, and voice recognition. This classification was used when the participant described some additional interface element outside of the touch gesture space. Most often this was a verbal statement like, “I would expect to have a menu here,” while pointing at the area that would contain the menu.

The grouping of these user defined gestures does not imply that gestures across groups are not interrelated or would not be mixed in sequences. In fact, sequencing of multiple groups of gestures was important for many of the tasks. For example, the simple task of moving a robot from its resting position to another area of the screen might require selection (e.g., tap), position (e.g., drag), and then another selection (e.g., double tap). Another method for directing the robot may be simply to select the robot (e.g., tap) and then select the destination (e.g., tap), expecting the robot to determine the best path.

Although the grouping of the gestures is important, a developer should be careful to not use these high level groupings to drive state-based inspection of the

gesture intent. In the first example, the selection (tap) can be thought of as the subject of the action, the position (drag) as the verb, and the selection (double tap) as the predicate. This grammar can drive the state of the gesture recognition in this simplistic case. Unfortunately, the grammar quickly falls apart in the second example where only the noun (tap) and predicate (tap) are provided; the verb is omitted from the sentence analogy and must be inferred.

7.3 Results and Discussion

The data set produced 3197 coded gestures using the gesture taxonomy described in the prior section. For each gesture, we recorded the selected object (if applicable), the gesture type, and the destination of the gesture. Since participants had varied levels of verbosity and gesturing, we normalized the data for each task by dividing the examined feature by the total number of gestures used by the participant in the task. This scalar could then be equally compared to other participants that might have given duplicate explanations or extended talk-aloud narratives. Unpaired two-tailed t -tests assuming equal variance with $\alpha = 0.05$ were used to determine significance. The results of the coding are shown in Figure 7-3. Of particular interest is the low percentage of participants expressing a desire for voice recognition (1.3%) and keyboards (1.5%).

The display provided no visual or audio feedback to the participants; while this eliminated any potential biasing, it also removed any indications that the participant might be providing inconsistent or nonsensical input. We observed that this lack of feedback resulted in tendency for participants to leave their fingers or hands engaged with the tabletop while thinking. These pauses resulted in the coding of more “press and hold” gestures than would have been seen if the interface reacted to the gestures.

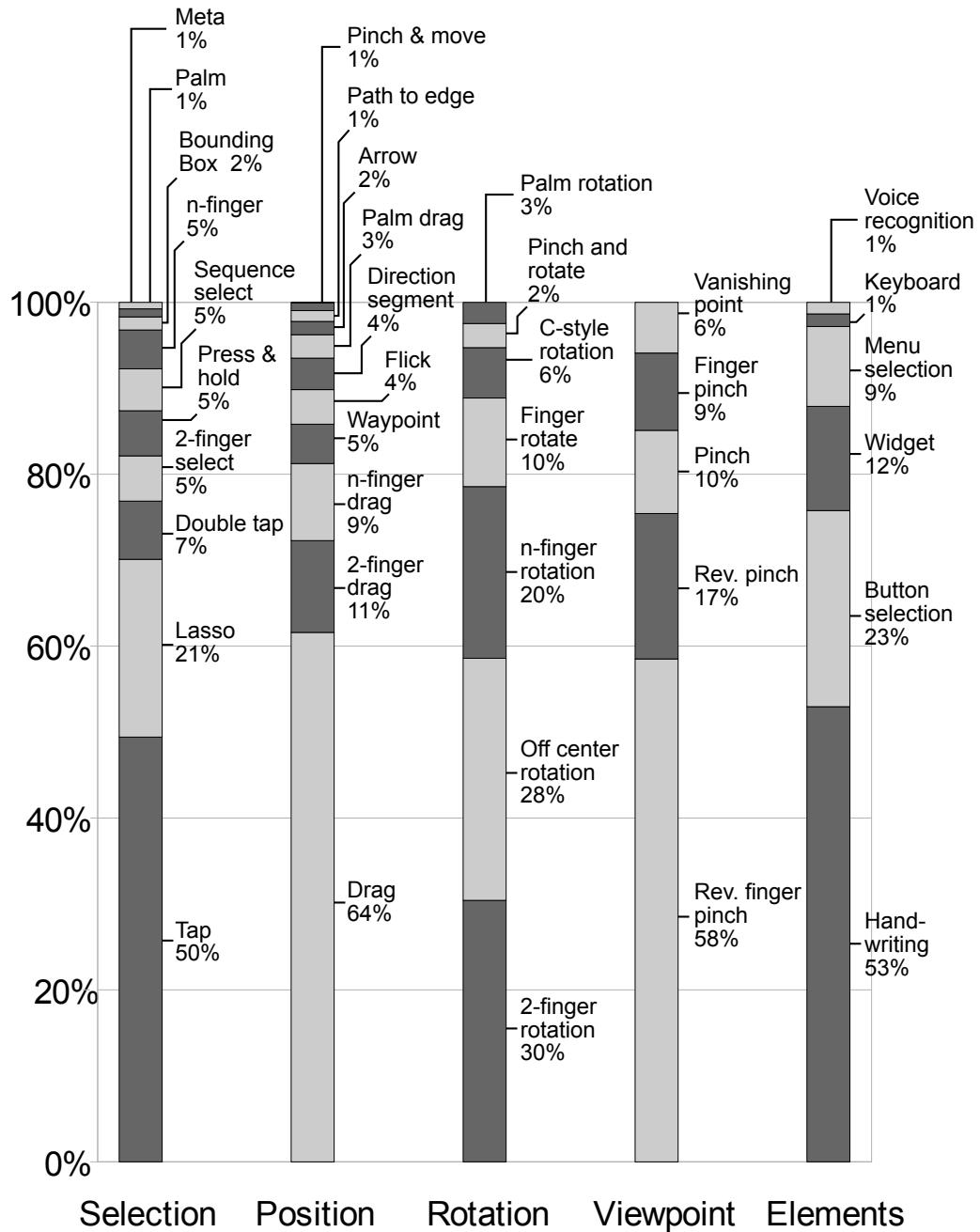


Figure 7-3: Normalized percentages of the gestures expressed by the participants in each of the classification groups.

7.3.1 Selection

From an overall gesture space perspective, the selection classifications followed several expected trends (shown in Figure 7-3). Tap accounted for 49% of the selection gestures. This large percentage was expected since this gesture is roughly analogous to the “click” of the mouse in the window user interfaces. Lasso was the second most occurring selection gesture (20%) primarily due to the natural action of drawing an ellipse around the objects to be selected. There was a cluster of selection techniques totaling 27% that included double taps, 2-finger select, press and hold, sequence select, and n -finger select. These are also analogous to mouse click methods. The low percentage of bounding box gestures at 1.5% and palm select at 1% is notable since these gestures have been used in several tabletop applications in the past. Overall, the selection results indicate that there is a bias to classic mouse “click” paradigms, but a gesture to circle or lasso the object would appear to be a very natural alternative.

We found an effect with selection gestures that has implications for gesture registration: in situations with one or two robots, most participants had no explicit selection step. The participant would simply gesture a drag or waypoint for the robot to follow; the selection was implied through the source of the gesture. The movement started at the robot and ended at the destination with no explicit selection of either. In contrast, with a group of three or more robots, the participant would select the robot group explicitly and then gesture the drag or waypoints that the group was to follow. The implications of this finding are important for gesture registration because, although the task is the same, the start of the gesture is different depending on the number of robots being considered for the task.

For tasks that had two robots, there was statistical significance between individual selections (e.g., tap or double tap) and group selection gestures (e.g., sequence select, lasso, and n -finger select). Participants used significantly fewer group select gestures ($\bar{X}=0.13$) than individual select gestures ($\bar{X}=0.95$), ($t_{60} = 3.0, p < 0.004$). Participants found it easier to individually select each of the two robots than to

use a gesture such as a lasso for a group selection. We had expected to see lasso used more often for selection of two robots, since this is the way that it would be accomplished in the mouse-driven case; however, participants preferred using individual selections in the case of two robots.

In tasks with three or more robots, participants used significantly more group select gestures ($\bar{X}=2.75$) than individual select gestures ($\bar{X}=0.43$), ($t_{60} = 6.5$, $p < 0.001$), and the use of group selects was significant against all of the other selection gestures.

7.3.2 Multi-hand and Multi-finger Gesturing

All participants in our study used a multi-handed gesture without explicit prompting by the experimenters. Slightly fewer (90%) used gestures that involved multiple fingers on the same hand; the other 10% used two hands for at least one of the tasks, but never more than one finger on each hand. This result contradicts other studies that found that most users preferred using a single finger on a single hand for their interactions (Epps et al., 2006; Koskinen et al., 2008). Instead, we found an almost unanimous willingness to use multi-hand and multi-finger gestures. In the other studies, the tasks performed were largely desktop-computing tasks. Since our task domain is outside of the traditional desktop paradigm, we had predicted that we would see different results. As the data above confirms, we did indeed see this difference. This result is important because it shows that when given unfamiliar tasks, users are more likely break away from mouse-driven paradigms to take advantage of multi-touch capabilities.

We also observed that the number of contact points on the multi-touch surface was influenced by the number of objects requiring manipulation in the task. For tasks that required the selection of two robots, 42% of the participants (13 of 31) used two fingers to select the robots, one finger per robot, rather than use a “group select” gesture. The two fingers used for the robot selection were most often on the same hand if the robots were close to one another and on different hands if the robots were farther apart. Participants used two hands to gesture significantly

less often when there were three or more robots ($\bar{X}=0.58$) as compared to tasks where there were one or two robots ($\bar{X}=4.76$), ($t_{60} = 5.6$, $p < 0.001$). For these tasks with three or more robots, participants tended to perform a group select, using a single hand.

We also observed that the type of task influenced the number of contact points, particularly in cases where prior computer usage introduced a bias. For example, the analogy of dragging an item from one location to another is ubiquitously used in WIMP interfaces. We found that when asked to move a robot or a group of robots, participants continued to use this dragging paradigm, with single finger drag (64%), 2-finger drag (11%), and n -finger drag (9%) as the most used position gestures. While all three gestures convey the same intent, they use differing numbers of fingers. Given that 20% of the movement gestures used two or more fingers to accomplish the same task as a single finger drag, a gesture set for robot control must be designed to allow these gestures to be used interchangeably. This finding that the number of fingers is not significant to the use of the gesture is consistent with Wobbrock et al. (2009). We believe that it is the shared context of dragging an item on a screen that leads to this agreement between the studies.

In contrast to position gestures, rotation gestures showed a tendency toward multi-finger gestures. The single finger rotation classification only accounted for 10% of the total number of rotation gestures. Since rotation is not a very common movement in WIMP interfaces, participants had fewer biases towards a mouse-like single point gesture. These multi-finger rotations follow the same motion that one would perform with physical objects on the table. Since the robot icons were analogues to physical objects in the world, the off-center rotation is a natural response and how one might rotate a robot in the physical world.

7.3.3 Handwriting

We observed that the participants tended towards handwriting for annotation tasks when given the freedom to gesture without constraint. 87% of the participants (27 of 31) expected to be able to use handwriting for labeling objects. Only

13% of the participants (4 of 31) described using onscreen keyboards, and 6.5% of the participants (2 of 31) used both keyboards and handwriting. We found this surprising since most ATMs, iPhones, and tablet-based PCs use on screen keyboards as the primary text input method. We had hypothesized that ubiquitous keyboard usage would create a bias toward the description of virtual keyboards or keypads. However, since participants could be free-form in all of their other gestures, they expected to be able to use handwriting and have it be recognized by the system or stored as an image label in their handwriting. Even further emphasizing this free form expectation, many participants did not write on the robot itself: there is an expectation of locality. 92.5% of the participants (25 of 27) that used handwriting gestures performed the gesture on the ground and only 7.4% (2 of 27) of the participants performed the handwriting gesture directly on the target robot(s).

We also observed that 10% of the participants (3 of 31) used both of their hands for handwriting, meaning that these participants were using non-dominant hands for handwriting. The participants abbreviated the labels for the robots in these cases, substituting “L” for “Team Left” and “R” for “Team Right.” Watching the video for this effect, we believe that the ambidextrous handwriting was due to the simplistic abbreviation and the placement of the annotation on the tabletop. Objects on the left side of the screen were annotated with the left hand and objects on the right by the right hand. The handedness based on position is an important property since, if implemented, handwriting recognition may need to work correctly with possibly degraded registration from multi-handed handwriting. Alternatively, handwriting may need to be stored as a label, without recognition, particularly in the case of writing with the non-dominant hand.

7.3.4 Gesture Usage as a Function of Prior Experience

Unsurprisingly, we found that the gestures that people wanted to use often correlated to their prior experience with computers, computer gaming, and multi-touch devices. Since the mainstream adoption of personal computers in the 1990’s, we

expected that desktop computing and WIMP paradigms would affect how the participants commanded the robot(s). The use of computer gaming has also become pervasive. In the past few years, personal electronics have begun to incorporate multi-touch technology, such as the iPhone.

Every participant had used computers, meaning that they had been exposed to common mouse paradigms. Despite the fact that 90% (28 of 31) of the participants used more than two fingers for gestures in their tasks at some point, multi-finger gestures only constituted 8.9% of all the gestures, indicating that the use of WIMP interfaces may be heavily influencing the expectations of the participants. Drag, 2-finger drag, and n -finger drag were the most used position gestures, totaling 84%. While these are natural responses, their use is also encouraged by the ubiquitous use of dragging in mouse paradigms.

We found that for tasks that were uncommon in mouse paradigms, the gesture space did not show such influence. For example, rotation is not extremely common in window interfaces outside of graphic design programs. The single finger rotation and the off-center rotation classifications accounted for 10% and 28% of the gestures respectively, whereas the 2-finger rotation and n -finger rotation accounted for 40%.

Since most computer applications have some type of menu system, we expected to see participants describing the use of drop down or on-screen menu systems. However, this was not the case as only 29% of the participants (9 of 31) expressed the need for menus during some point in the experiment. Similar effects were noted for the use of buttons (32% of participants; 10 of 31) and specialized widgets (41% of participants; 13 of 41). Participants that played games expressed the desire for significantly more “widget” gestures ($\bar{X}=0.41$) than those participants that did not play games ($\bar{X}=0.13$), ($t_{29} = 2.2$, $p=0.0304$), which may be attributed to the fact that games tend to have many custom widgets to perform certain activities.

We believe that iPhones have biased the “pinch” since 53.8% of the zoom gestures were some form of pinch gesture. This effect is not surprising, but indicates that established gestures cannot be ignored regardless of their “correctness” (or

lack thereof) from a HCI perspective. Combined pinch gestures include “pinch,” “spread,” “finger pinch,” and “finger spread.” Participants that had prior experience using the iPhone used significantly more combined pinch gestures ($\bar{X}=0.68$) than those participants that had no experience using the iPhone ($\bar{X}=0.39$), ($t_{29} = 3.9$, $p<0.001$).

Participants that played RTS games had fewer combined pinch gestures ($\bar{X}=0.68$) than those participants that did not play RTS games ($\bar{X}=0.48$), ($t_{29} = 2.3$, $p<0.028$). The prior experience of these participants might have influenced them against using pinch gestures, since most RTS games are played using a mouse and a keyboard and do not have any form of pinch gesture.

7.4 Impacts and Implications

As robots become more commonplace in large teams, the need to manage their actions as individuals and groups becomes important. Our experiment was designed to determine the gestures that people would find the most natural for a variety of tasks in this domain. The assumption was made that the robots would have the ability to be very autonomous in the execution of the specified tasks; without a great deal of autonomy in each individual robot, it would not be feasible to control a group of more than a dozen robots.

We found that the prior experience of the participants introduced some bias into the gestures that they wanted to use. In particular, selection and movement gestures were heavily influenced by standard mouse paradigms. Additionally, we saw that participants who had used iPhones used significantly more pinch gestures for zooming, while people who have spent many hours playing real time strategy (RTS) games expect to have similar controls in the multi-touch domain. However, we also found that when presented with unfamiliar tasks, users are willing to break away from standard mouse-driven paradigms and use multi-touch capabilities freely.

This research has identified several guidelines for designing gesture sets for the

robot control domain:

- If we want the gesture set to be easy to learn, biases introduced by mouse driven interfaces will need to be carried over to the multi-touch domain.
- To account for individual biases caused by the use of devices such as the iPhone or a great deal of time playing computer games, gesture sets could include multiple gestures for the same capabilities. One could even imagine a small set of questions to be asked of a user that would customize the gesture set to their experiences.
- Users expect to provide multiple levels of instruction to the robot. This includes providing a start and destination, providing way points, and providing an explicit path.
- If we use free-form gestures (such as lasso) for robot selection and movement, then there will be an inherent user expectation for free form labels (symbols) and handwriting recognition (instead of virtual keyboard).
- The grammar expressed by the users' gestures are not always complete and may not include an explicit selection step. For example, "Robot A should move to Area B" may be expressed as a drag starting near Robot A and ending in Area B with no explicit selection of the robot itself.

These guidelines become important as we leverage the natural responses from users to increase ease of learning and memorability. User expectations and biases are often different from what we expect as interface engineers. From this preliminary user study we have created a collection of the most natural responses for robot control. The next Chapter describes how we used this distribution of gestures to create an interface that linked the user's intentions to the actual control of robots.

Chapter 8

Multi-Touch Multi-Robot Command and Control

The previous chapters have discussed the design and testing of the individual components of a system for command and control of robots. With all of these pieces assembled, it was important to determine if they could all be engineered into a usable and understandable system that was easy to learn by users who may not have significant time to train and retain the gestures used by the system.

To achieve our goal, we leveraged many of the lessons learned from earlier chapters.

- All of the elements in the interface must be larger than 2 cm for successful interaction with fingers on a tabletop device (Chapter 3).
- Visual affordances from physical devices for on screen elements should be selected carefully, if used at all (Chapter 4).
- Control metaphors must be consistent throughout the interface regardless of underlying implementation details (Chapter 4).
- If possible, use the biomechanics of the human body to build natural interaction methods rather than complex on-screen controls (Chapter 5).

- Take advantage of the biases and muscle memory that users bring to the interface from video games, mobile phones, and other interactions (Chapter 6).
- There are patterns to users' natural interactions that we can model. As in a language's grammar, these patterns may have exceptions, but these variances may be accounted for programmatically (Chapter 7).

Based on these lessons, we created a command and control interface to support our research goal from Chapter 1:

Multi-touch is a rich and interactive method of computer interaction that can provide enhanced learnability in time-critical and safety-critical domains when implemented correctly. By carefully studying the biomechanics of the human hand and leveraging the natural responses of users to multi-touch displays, human-robot interaction can be enhanced for single and multiple robot control. The goal will be to maximize learnability and therefore lessen the amount of training time required for proficient control of the robot or robot teams.

As discussed in Chapter 1, we see a convergence of three key technologies enabling the advancement of command and control for search and rescue operations. Network centric operation, robotics, and multi-touch hardware are the key components that will mature in the near future. We have made some assumptions about the future availability of these advancements in the design of the command and control interface presented in this chapter. Furthermore, our design borrowed from the research discussed in Drury et al. (2010), which explored a hazardous material response training exercise and a mock-up paper prototype for command and control in hazardous material response.

8.1 Technology Capabilities Assumptions

The design of our command and control system for robots rested on several basic assumptions that were based on observations of robot operators, collaborative workspaces, and newly emerging technologies. The deployment of this technology in the field will rely on several key innovations that we expect to see in the coming years.

Tabletop interaction: We assume that field-ready tabletop computing devices will be available in the near future. As mentioned in Chapter 1, we are already seeing limited support for multi-touch on large displays headed to the field for military applications. During planning and managing operations, the design and orientation of a tabletop fosters collaboration. People have been working around tables and sharing information in a natural and productive manner for hundreds of years. Maps and objects can be moved, updated, and referred to easily through the simple movements of grasping hands and pointing fingers. More recently, computers and laptops have begun to replace these traditional table workspaces due to the availability of visualizations and digital data or information. Laptops pose a problem for collaboration since the screens are prohibitively small for group observation and interaction is typically limited to a keyboard or mouse. Our design leverages the Microsoft Surface hardware which combines the best features of both tabletop and laptop scenarios.

Top-down view: We assumed that a top-down view of the disaster area was available. The view could take the form of geo-referenced satellite data as presented in products like Google Earth or NASA Worldwind. Even today, satellite information is available to search teams with an Internet connection or pre-downloaded map data. Ideally, the search team has the ability to receive geo-referenced images from manned or unmanned aerial vehicles over the affected area. Currently, near-real-time aerial data is not available to most search teams, but we

hope in the future that this will change. As discussed in Chapter 1, a Predator aircraft was used to update teams on the ground in near real-time using Google Earth. Additionally, Google demonstrated the ability to update Google Earth shortly after the earthquake in Haiti for organizations assisting in the response.

The top-down view is important because it places the robots and other markers geographically with respect to each other. The location of all of the assets in a command and control scenario is very important, so a top-down view has classically been the view of choice in SAR and military operations. The top-down view not only provides the current location of the assets, but also their movement and graphical representation can provide information about the direction of travel and status of operations.

In a study of radio transmissions during a mock hazardous material response, Drury et al. (2010) showed that 27% of radio transmissions where problems were noted were related to the location, status, and usage of equipment. An additional 11% of the radio transmissions were related to the status of the current situation. Discussions with personnel in the search and rescue field indicated that these numbers were not unique to hazardous material response and were typical of emergency response in general. In the words of one of the participants in the study from Drury et al. (2010), “I can only listen to so many radios.” A top-down view can help provide the command staff with a top down view of the situation and increase awareness of the operation’s progress.

Geo-location of assets: We assumed that GPS tagging and tracking of robots was available. GPS tracking of resources is becoming increasingly common and inexpensive. Tracking of fire and police vehicles is becoming standard. The push for geo-location in emergency 911 calls has flooded the commercial market with cheap GPS enabled cell phones that can be adapted for tracking of personnel on the ground. The robotics field has long used GPS location for unmanned ground vehicles used in outdoor tasks. The above GPS technologies indicate that in the near future it is possible to have a very clear and accurate picture of

all of the resources that can be tasked and monitored in a search and rescue scenario.

Ground robots with reachback: The use of robots with the ability to send back position, video, and sensors was assumed for our interface. Additionally, we assumed that robots were available that had basic navigation, obstacle avoidance, and exploration behaviors sufficient for searching an urban environment. They were also assumed to be able to receive commands wirelessly. Many of these features are available today in field-capable robots. The capabilities of ground robots are increasing and the field is seeing more and more initiatives to bring robots out of the laboratories and into the real-world. As such, we hope that the near future will provide field-ready robots with the capabilities of the robots modeled in our research.

Need for intervention: A basic philosophy surrounding our research strategy was the idea that a solution for multiple robots must also provide manual control for a single robot. We explored single robot control in Chapters 4 and 6, which provided the basis for the solution presented later in this chapter.

Robots can make mistakes in judgement or sensing and will inevitably get themselves stuck in the environment. When this error occurs, the person at the command console will not be able to correct the problem if there is no way to take manual control. The robot teams may be heterogenous, so losing a robot with unique capabilities could limit the successfulness of the mission if those capabilities are needed. Therefore, we believe that it is essential to also provide a means for intervening and manually operating the robot.

Using the above assumptions about technology to frame our engineering constraints, we then proceeded to formulate how we would design a collection of gestures and test our design.

8.2 Gesture Design

Our multi-touch multi-robot gesture set used a bottom-up approach based on the most frequently used gestures described in Chapter 7. Since the gestures were user defined, it follows that the most popular gestures should be the most natural and maximize the ease of learning. In an effort to limit the design to the most popular gestures, we used only the top 90% of the gestures captured in Chapter 7 and shown in Figure 7-3. For selection, these gestures included tap, lasso, double tap, two-finger select, press and hold, and sequence sequence select. For the view changes or movement of the robot, the gestures included drag, two-finger drag, n -finger drag, and waypoints. For changing the map viewpoint, the gestures included single hand finger spread, multi-hand spread¹, multi-hand pinch, and single hand pinch.

For this multi-robot interface, the gestures related to rotation were not used and we also limited the use of specialized screen elements as much as possible. A design goal of our interface was to minimize user interface elements. Any extra graphics, menus, or specialized widgets were only used when there were no other options in the gesture space. In theory, we could build a gesture state machine to interpret any number of complex gestures. However, we did not want the gesture space to be so complex that it was no longer easy to use or remember. To this end, we worked within the following three design guidelines:

1. Use simple and natural gestures wherever possible.
2. Use menus only when guideline 1 is not sufficient, such as gestures that become ambiguous or gestures that are mutually exclusive.
3. Avoid gestures that are very long or complex when guideline 2 can suffice.

Using these three guidelines as a foundation, we kept the on-screen visual clutter to a minimum and created a map-centric interface. Unlike most computer

¹The term spread is synonymous to reverse pinch and is replacing its usage in user interface guidelines such as Villamor et al. (2010).

applications, there were no menus or widgets displayed on the screen. All of the robot commands, map movement, and specialized commands were contained within the gestures themselves or within a small hidden menu that only appeared in a position local to the finger. Although our implementation centered on single user interaction, a benefit of our design is that the interface is insensitive to the angle of the observer. The non-reliance on orientation allows a user or multiple users to approach the surface from any angle and begin interacting with the interface.

8.2.1 Gesture Grammar

The gesture space for the robot actions generally took the grammar of *<subject>* *<verb>* *<predicate>* where:

<subject> was the robot or robots being manipulated,
<verb> was the action or actions to be performed, and
<predicate> was the target or destination of the action.

For example, a tap on the robot (subject) followed by a drag representing the path (verb) that terminated with a double tap at the end (predicate) would cause the selected robot to follow the drawn path and then stop at the final destination. Like any flexible grammar, there were useful exceptions discussed below.

8.2.2 Robot Control Grammar

Selection (as the subject): Selection occurred using one of seven primitives: tap, lasso, double tap, two-finger select, press and hold, sequence select, and n -finger select. Tap and double tap were reduced to a single programmatic select primitive since it could be modeled in the state machine as a re-selection of the same robot on the second tap. For the same reason, tap, two-finger select, and n -finger select were reduced to select since each tap or touch event will fire on the same relative coordinates over the robot's position. We required 250 milliseconds to pass between subsequent touch events before the robot was toggled to a deselected state.

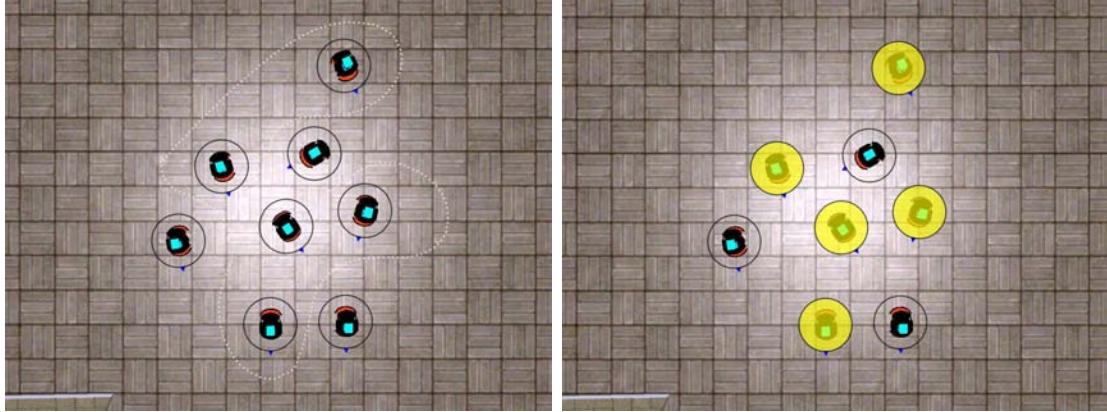


Figure 8-1: An example of the Lasso gesture (left) as dragged on the screen until the finger was lifted. The result of this lasso is shown by the yellow highlighted robots (right).

We used four of the selection gestures above, so three selection primitives remained: sequence select, lasso, and press and hold. Sequence select required the user to select the robots to be affected in sequential order. From a state machine standpoint, as long as additional robots were selected via any of the select methods, then robots were added to the subject list.

Lasso was a path drawn on the ground with a single finger. The gesture began on a coordinate not occupied by a robot (since that could be a tap selection) and was tracked through the drag gesture until the finger was lifted. If the line ended within several inches of the start, the first point and last point of the drag were connected to complete the bounding area. All points within the bounding lasso line are selected and added to the subject list. An example of this selection method is shown in Figure 8-1.

Position (as the verb): The user provided navigation commands to the robot in two ways: a series of waypoints or a path. The user provided waypoints by tapping on the map to indicate where he or she wanted the robot to navigate. Each point was added to an ordered list of waypoints that was sent to the robot upon execution.

A path for the robot was provided with the drag gesture on the surface. The path, projected onto the ground plane, represented the desired path that the robot

should follow. Programmatically, the continuous path was reduced to a close series of waypoints that the robot should travel in the order specified. The preservation of order was important so that loops and intersections were properly followed.

It should be noted that these two gestures could be mixed and may be interrupted. When the finger returned to the screen, the remaining waypoints were simply added to the list. For example, the user could begin specifying a path with the drag gesture. He or she could lift the finger from the drag, wait momentarily, and then begin tapping the ground specifying waypoints. The user could then provide an additional path if more detailed navigation information was needed. The ability to resume adding additional waypoints or paths was important since a user may be formulating the waypoints or paths as he or she is performing the drag gesture. If the user wishes to pause for a moment and consider alternatives, the interface will allow the pause and accept later waypoints and drags as a concatenation to the earlier commands.

Selection (as the predicate): A double tap signified the end of the sentence and the final location of the robot(s) after performing the actions specified in the earlier steps. For example, the user could select a robot, provide waypoints, and then he or she only needed to double tap on the final location to signify the end of the gesture sequence. The robot would then begin moving immediately.

Special cases and caveats: There were a few special cases that we needed to consider in this grammar for robot control. As we showed in Chapter 7, we had some special cases that would break our simplistic grammar.

First, if there were only one or two robots visible on the screen, then the user might omit the selection of the subject and only provide the verb and the predicate. This condition was detectable since the person tapped or dragged on the ground plane near a robot while in the initial state. By programmatically asking the interface if only one two robots were visible, the software could determine if this gesture was a nonsense case or if we need to select the nearby robot and add

waypoints to the waypoint list.

Second, the user might omit the verb and only specify the subject and predicate. An example would be a tap on the robot and a double tap at the destination. Since we terminate with the double tap on the ground plane, this special case was discernible through a state change on double tap if the state machine was waiting for a verb gesture.

Third, the user was allowed to omit the predicate if another unselected robot was selected after the verb. The selection of the next robot, in effect, became the beginning of the next sentence. This particular grammar adjustment was indirectly exposed in the user defined gestures in Chapter 7. Since the paper prototype tested in that experiment placed each of the tasks its their own context, the need for serializing several tasks one after another did not present itself. Interestingly, we observed that the users created a gesture that omitted a subject, and then a verb. We then asked, “When could the users possibly omit the predicate?” We realized a more realistic scenario of tasking of multiple robots would expose a special case.

A diagram of the state machine is shown in Figure 8-2.

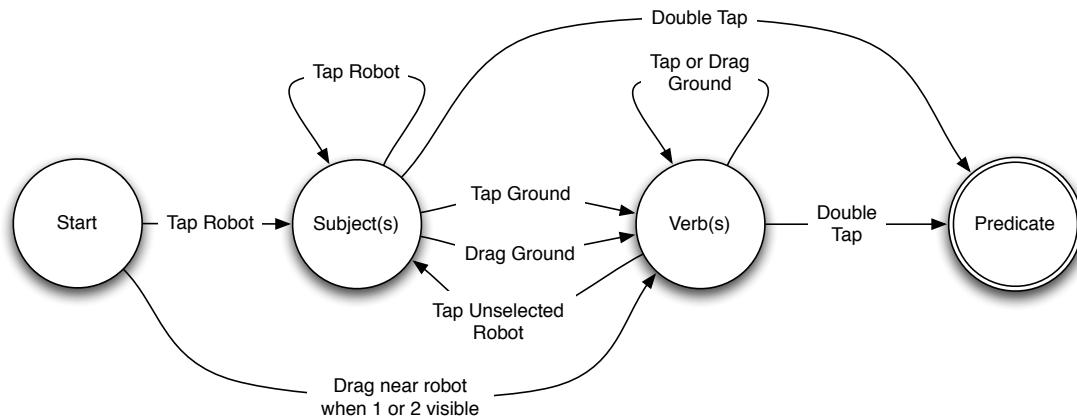
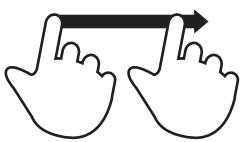


Figure 8-2: State machine for the grammar showing the basic structure and state changes for special cases.

Table 8.1: Touch gesture reference guide for robot selection, movement, and execution. Icons based on (Villamor et al., 2010).

	Name	Location	Description
Selection (Noun)		Tap	Robot Briefly touch robot(s) to select and deselect.
		Double Tap	Robot Rapidly touch finger twice on robot(s) to select and deselect.
		Lasso	Robot Draw a circle around the robot(s) to select. <i>Note: Must be first selection gesture, else could be confused with Path</i>
Action (Verb)		Tap	Ground Briefly touch waypoint(s) that the robot(s) will follow on the way to destination.
		Path	Ground Move one finger over surface without loosing contact on a path robot(s) should follow.
Execution (Pred.)		Double Tap	Ground Rapidly touch ground twice with fingertip to provide the final destination of the robot(s).
		Tap	Robot Briefly touch a new unselected robot to begin executing the previous command(s).
		Press & Hold	Anywhere Select “Execute” to begin executing queued command(s). <i>Explained in Table 8.2</i>

8.2.3 View Control

The user was able to move the view of the map along three degrees of freedom: zoom (on the z -axis) and translate (on the x -axis and y -axis). Zoom allowed the user to adjust the perceived altitude of the viewing camera. Translate moved the camera on the plane parallel to the ground plane in x and y . Translation was useful when the extents of the viewable world extended past the current field of view of the overhead camera.

Zoom: Zoom in was triggered by two fingers simultaneously make contact with the surface and then diverge from one another. We referred to this gesture as a spread in Chapter 7. For the metaphor of the aerial vehicle, zooming in simply lowered the view from the camera in 3D space and placed it closer to the ground. Zoom out was exactly the opposite; the two fingers converge or pinch and the view from the aerial vehicle was raised.

Translate: When two to four fingers contacted the surface and then dragged, a translate action was triggered. Translation moved the camera parallel to the ground in the opposite direction of the drag motion. In effect, the user placed his or her fingers to grab some point on the ground and moved the map as if it was dragged underneath their fingers. The use of two or more fingers in this gesture distinguish it from a lasso. Since users have a tendency to use any number of fingers for gestures, we decided to allow up to four fingers for translation. This freedom allowed users to simply place any four of their fingers on the surface and move the map as if it were a physical object on the table.

It is important to note that translation and zoom could happen at the same time and within the same gesture. For instance, a user could place his or her fingers close together on some object of interest in the upper left hand corner of the screen relative to his or her position. By spreading the fingers apart and moving the fingers closer to the body, the user could zoom in on the object while

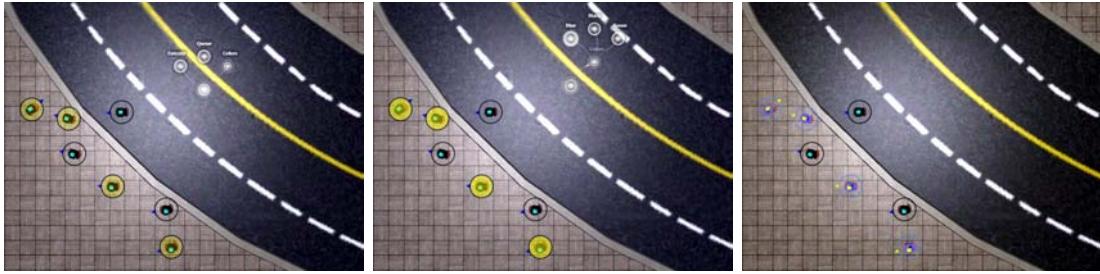


Figure 8-3: After selecting the robots, the user pressed and held his or her finger on the surface for four seconds (left). A menu appeared under the finger (center) where options could be selected such as Color, Queue, and Execute. When the user selected the desired color (right) the finger was lifted and the selected robots were changed.

bringing it to the relative center of the screen in the same motion.

8.2.4 Menu Functions

Having implemented the majority of the user defined gestures, we needed an unambiguous gesture to open a menu. The menu allowed additional mode control and robot identification. The press and hold gesture was often suggested by participants for changing robot properties during the study in Chapter 7. To activate the menu, the user pressed and held his or her finger on the surface for four seconds. Then a graphic appeared underneath the finger as shown in Figure 8-3. The graphic loosely resembled a glowing tree with hierarchical branches that emanate from the finger location. At the first menu layer, the user could choose from Color, Queue, and Execute. The Color menu option expanded to reveal a sub-menu with the color labels for Blue, Green, and Black. When the user selected one of the Color menu sub-options, the currently selected robot(s) changed to the identified color. In our implementation, changing the color of the robot did not change any other physical properties and did not associate the robots' actions with each other in any way. Color was a visual means for managing and talking about the robot teams.

The Queue menu option placed the interface in a special mode for simultaneous execution of commands. By pressing the Queue option, the interface waited before

Table 8.2: Touch gesture reference guide for map movement, menu selection, and joystick control.

	Name	Location	Description
Map Movement	Multi-Finger Drag	Ground	Move two to four fingers on the ground to move camera north and south.
	Spread	Ground	Touch two to four fingers on the ground and move them apart to move camera closer to the ground.
	Pinch	Ground	Touch two to four fingers on the ground and bring them closer together to increase camera altitude.
Menu		Anywhere	Press and hold for: 1) Command queue 2) Execute queue 3) Change color
Command Edit	Fist	Anywhere	Touch one fist with either hand to deselect all robots and reset all waypoints.
	Double Fist	Anywhere	Touch two fists on the surface to halt all robots, deselect all robots, and reset all waypoints.

executing robot movement. All of the functionality of the interface remained the same so that the user could select, provide waypoints, and complete commands for the robots to execute when commanded. Typically, this sequence was repeated for several robots or several teams of robots. The user pressed and held his or her finger to reveal the menu once again and selected execute. At that point, all of the robots were released from their static positions and began executing their assigned tasks.

8.2.5 Stop and Reset Gestures

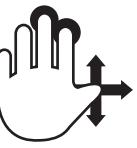
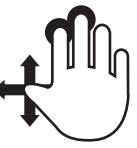
While creating a gesture sequence, the user might make a mistake. If a mistake occurred, the user placed his or her fist on the surface to clear the current gesture sequence. Programmatically, all of the robots are deselected, all of the waypoints are cleared, and the state machine is placed in the starting state.

If the user needed to stop all of the robots that were executing commands and clear the current gesture sequence, the user placed the bottom of both fists on the surface simultaneously. All robot movement was stopped immediately, commands cleared, and the state machine was placed in the starting state. An all-stop signal is commonplace in all fire and rescue operations for safety purposes, so this important gesture was a necessary functionality for any robots designed for the search and rescue domain.

8.2.6 DREAM Controller Integration

At any point, the user could instantiate a DREAM Controller for manual operation or viewing the robot's camera. If the user placed five fingers from the right hand on the surface, a DREAM Controller was created, and as in Chapter 6, it was used to rotate the pan and tilt axis on the camera on the robot. On the panel that extended from the thumb and index finger, a live view from the robot's camera was displayed in a size and orientation appropriate for the user's hand configuration. The thumb control was designed to move the pan and tilt mechanism for the

Table 8.3: Touch gesture reference guide for map movement, menu selection, and joystick control.

	Name	Location	Description
Joystick Control	  Left Joystick	Anywhere	Touch five fingers with the left hand to enable joystick control of the movement of last selected robot and view range sensors.
	  Right Joystick	Anywhere	Touch five fingers with the right hand to enable joystick control of the camera of the last selected robot and view the camera.

camera.² The user then had two simultaneous views of the robot’s situation: one view from the aerial camera above, and one from the semi-transparent DREAM Controller displaying the robot-eye view. If the overhead view of the robot became occluded, the user could relocate the DREAM Controller by lifting the thumb and dragging the index and middle fingers or just by putting his or her hand in another location

The left hand DREAM Controller was also triggered with five fingers on the surface. As in Chapter 6, the left hand was associated with robot’s chassis movement. The thumb controller pad behaved identically to first person shooter games in which the upward direction moved the robot forward, down moved the robot backwards, and left and right rotated the robot in the respective direction. The panel extending from the thumb and index finger displayed the distance panel (described in Chapter 6). The radar-like view of the laser range finder data provided information to the user about the location of the robot relative to the objects around the front of the robot. Since the DREAM Controller functionality

²The pan and tilt camera functionality was not implemented in the virtual robot model at the time of user testing.

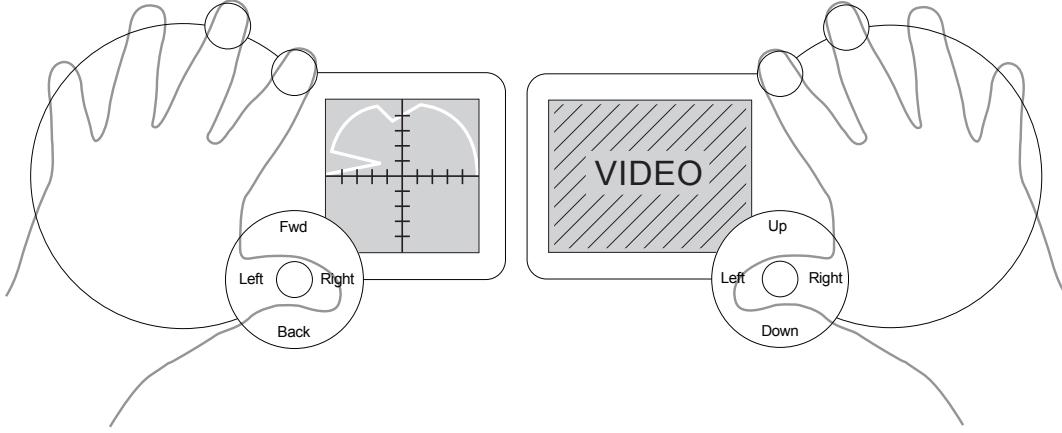


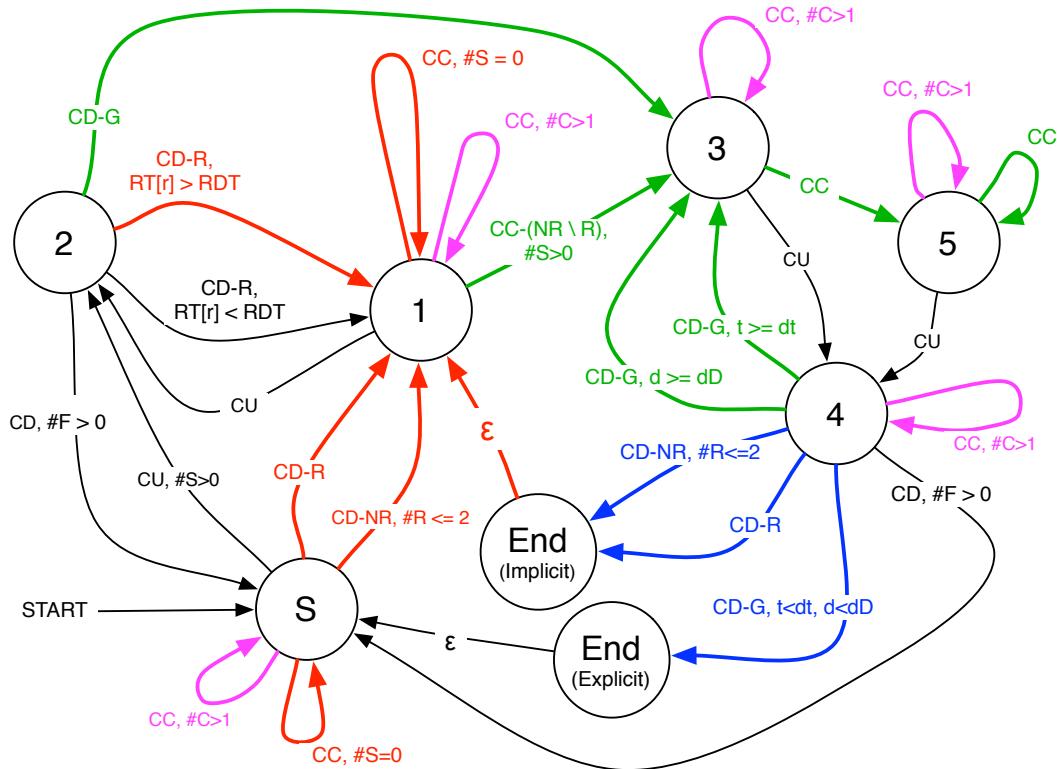
Figure 8-4: Illustration of the DREAM Controller configured for dynamically selected robots in this interface. The left hand controller displays the data from the laser range finder in a radar-like display and allows the user to drive the robot chassis. The right hand shows the video from the robot’s view and controls the pan and tilt of the camera.

was integrated for manual intervention, the laser range range data could be used if the robot became stuck or images if from the camera cannot provide sufficient situation awareness to the user.

8.3 Gesture State Machine Implementation

As demonstrated in the previous section, the deceptively simple grammar grew to include many features and potential paths for execution. The state machine in Figure 8-2 was modified to use the Microsoft Surface specific event model, which enabled the events corresponding to finger up, finger down, and contact changed. These events corresponded to the finger movements for pressing, lifting, and dragging respectively. The modification of the state machine with Microsoft events is shown in Figure 8-5.

Symbolic notation of these states and transitions provide an overview of the events and connecting logic. The implementation details, such as time delays and data structures, are not discussed in this text, but played an important role for the responsiveness and “feel” of the interface.



Transition Key:	Grammar Example:	Robot-Contact Proximity:
Subject - selection(s)	Subject →	G = Ground R = Robot
Verb - waypoint(s)	→ Verb	NR = Near or on Robot FR = Far from Robot
Predicate - execute	→ Predicate	NR \ R = Near Robot
Map Movement *	→ Map Movement	
Other *	→ Other	
* Not a navigation grammatical unit		

Event Types:
CD = Contact Down
CC = Contact Change
CU = Contact Up
Associated with proximity.
Example: CD-G for Contact Down on Ground

General Variables:
#F = # of fists
#C = # of contacts
#R = # of visible robots
#S = # selected robots

Double Tap Variables:
t = time between
d = distance between
RT[r] = Time since the robot r received a CD event

All transitions' include
#F=0, #C=1 unless indicated otherwise.

Robot-Contact Proximity:
G = Ground
R = Robot
NR = Near or on Robot
FR = Far from Robot
NR \ R = Near Robot

DREAM Controllers are created concurrently with this state machine's execution, and supersede this logic while they are active.

Figure 8-5: Finite state machine for the gestures in the multi-robot command and control interface.

8.4 Simulation Design

The harsh reality of ground robotics today made testing with large numbers of robots infeasible. Additionally, the logistics of finding a large city-sized area for the robots to search was non-trivial. Additionally, this experiment was testing the interface and not the robot design or robot behaviors themselves. We needed a simulation architecture that would approach the realism and unpredictability of a real world field test without needing physical robots in large numbers. As such, we used one of the most advanced modeling and simulation environment available to the public: the Microsoft Robotics Developer Studio (MRDS). The MRDS environment is a Windows based architecture that allows for the creation of virtual robots that use the Microsoft .Net Concurrency and Coordination Runtime Libraries for communication and message passing. This service based architecture was more than sufficient for our test and provided all of the functionality required. The three dimensional, hardware-accelerated physics allowed us to simulate eight ActivMedia Pioneer 3DX robots in an urban environment which was approximately 2 acres in area. An aerial view of the simulation is shown in Figure 8-6.

To model the interaction between the human, tabletop, and robots as realistically as possible, we separated the simulation engine from the user interface. The simulation engine ran on a dedicated desktop workstation and managed the creation of robots, the robots' behaviors, and the associated physics related to the robots' movements in the environment. The Microsoft Surface was responsible for interpreting the gestures of the user which were captured through a transparent overlay on the overhead camera view. As shown in Figure 8-7, the robots were graphically outlined with a circle on this layer to show their location in the overhead camera. The small triangle on the outside of the circle represented the direction that the robot chassis faced. This circle was used programmatically to show selection of the robot and was never smaller than 2 cm based on our findings from Chapter 3.

The overhead view was modeled as a virtual camera that can translate on the



Figure 8-6: An aerial view of the urban simulation environment used to test the gesture based command and control interface. The three dimensional city was approximately 2 acres in area and used realistic hardware-accelerated physics for robot movement and interaction with the environment.

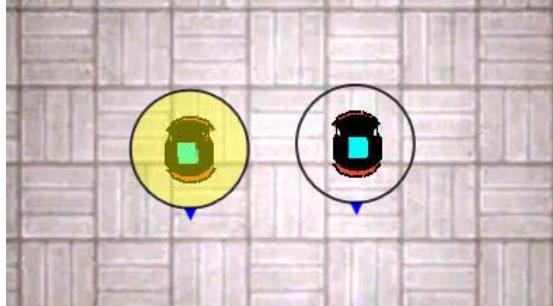


Figure 8-7: Circles were drawn around the robots on a transparent layer outside of the simulation. A selected robot (left) is shown with a yellow highlight and unselected robot (right). In both cases, the blue triangle represents the orientation of the robot.

three axes. The images from the virtual camera were sent as a stream of images to the interface using a UDP protocol over Ethernet. By modeling the camera in this way, as opposed to directly rendering the graphics from the simulation, we more closely model how images might be captured and sent from an aircraft in non-virtual environment. It is important to note that the circles and other graphic overlays were generated independently on the interface and not in the simulation. Since the interface was updated with information regarding the altitude and position of the camera, the software only needed to calculate the required on-screen position for the graphics based on the robots' GPS location and the camera lens characteristics. A side effect of this interface design was that rapid changes to camera position could be updated in real-time in the transparent overlay on the interface. From the user's perspective, the on-screen overlay provided responsive feedback while the aerial camera caught up with the requested motion and transmitted the updated imagery.

8.5 Experiment Design

Our intent was to test the gestures and interface in the most realistic scenario, given the constraints of available personnel, equipment, and scenario scale. Since we had created a realistic world to test the robots, we gathered domain experts in search and rescue to test our hypothesis.

8.5.1 Participants and Procedure

The six participants used for the study were the same six FEMA search and rescue subject matter experts used in Chapter 6. As such, the demographic and self-reported computer and gaming experiences are the same as detailed in Section 6.4.3.

The participant was first briefed on the nature of the study and a general description of the tasks that would be performed. Any questions that the participant had were answered and then he or she was given an informed consent and video consent form to read and sign.

The participant was then trained on the gestures used for the interface. The test administrator demonstrated each of the gestures listed in Table 8.1 through 8.3. The basics of the gesture grammar were discussed to help the participant understand the nature of the gesture sequences. The participant was encouraged to try the gestures during the instruction period and assistance was given if the participant was having problems. The participant was allowed to practice on the gestures until he or she felt that the control of the robots had been sufficiently mastered. Training and practice took 17 minutes ($SD = 8.8$) on average.

At the completion of the training and practice time, the test administrator verbally described a fictional scenario of a dirty bomb attack in the downtown area of a city. Since the scenario and timeline were not real, the spoken description acted as both a summary of the incident and a briefing of where the participant was in the overall timeline. Much like a real briefing, the participant was encouraged to ask questions for clarification on any points. The scenario was described as follows.

As of 8:46 am, the Department of Homeland Security has received a request for federal assistance from Loop City, Nevada. A radiological dispersion event has occurred near the downtown area that has required an evacuation of all buildings and emergency personnel. As of 9:03 am, incident command was been established by the local FBI terrorism

response unit and DHS has directed your FEMA team to work closely with FBI and other law enforcement agencies to secure the crime scene and ensure that there are no live victims still in the downtown area.

Your team's cache was recently augmented with specialized robotic equipment designed to deal with low to medium radiation environments. First, you have a Unmanned Aerial Vehicle capable of hovering at any altitude or position over the city. FAA has cleared the airspace and declared it a no-fly zone for any aircraft, so you have full authority to use it for any visual reconnaissance over the city. Second, you have eight autonomous Unmanned Ground Vehicles capable of traveling through urban terrain. They can be told to navigate to a GPS waypoint or follow a pre-determined path. They have cameras that are capable of detecting humans and sensors for monitoring the radioactivity in the area.

You have arrived on scene and coordinated with command. Your primary goal, as the robot search team leader, is to direct the robots through the city as defined by the planning manager. You are safe to assume that the robots will alert you when they have found a human. In other words, you are directing the robots where to search and not performing the search task yourself. Your secondary goal is to identify the source of the radiological event. Reports from people leaving the scene indicate that the suspects were using a forklift or front-end loader to elevate and disperse the radiological material. It is unclear at this time where the dispersion equipment is located and where the suspects may have placed additional radiological sources. The robots are capable of alerting you if they find a sufficiently strong source of radiation.

The participant was then given a map of the affected city. The map was specifically illustrated and colored to imitate the USGS topographical maps commonly used by search and rescue teams. The map is shown in Figure 8-8. The participant was told to write on the paper map if desired and to make any note that would be

useful to the immediate operation or the command hierarchy. The annotation of maps is a standard practice for search and rescue and is especially useful during a change of personnel.

For the purposes of the scenario, the testing administrator fulfilled the roles of the planning and safety managers in the command hierarchy. The planning manager position meant that the test administrator would provide the search grids and priorities to the participant acting as the robot search team manager. The safety manager role meant that the testing administrator had the authority to direct the participant to immediately change or halt operations at any time.

The participant was instructed that they had 25 minutes to complete the sce-

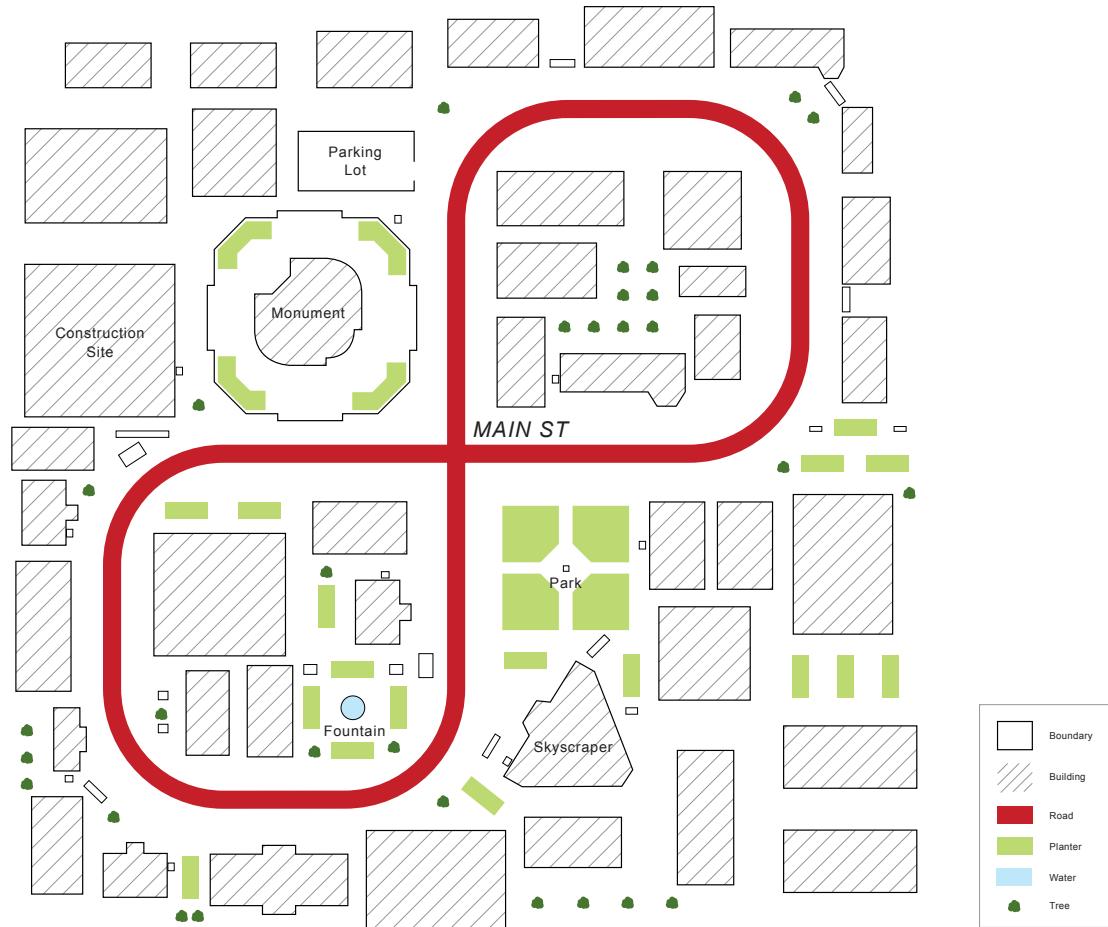


Figure 8-8: Illustration of the fictional Loop City, Nevada, drawn in the style of a USGS topographical map showing major features, highways, and common landmark names. Participants used this map as a reference while directing search operations with the eight simulated robots.

nario. A written timeline of events was pre-scripted and the testing administrator updated the participant on the state of the operations at the times established. The timing script with events was as follows:

Offset (min)	Event to be read
:00	Planning has divided the city into the four quadrants shown on this map [show map].
:00	Create a green team consisting of two robots and a blue team consisting of five robots.
:00	Command the black robot to the center of the Park.
:03	Command the Green Team to search Area A (or C depending on the start location) outside of the main street loop.
:05	Command the Blue Team to search Area A (or C) inside of the main street loop.
:10	Command the black robot to do a detailed search of the Park.
:12	The governor of Nevada has contacted your team leader and wants to know what percentage of the city you have searched at this time.
:15	Use the green and blue teams to search Area D and have them start their search at the same time.
:20	Perform an all-stop immediately.
:20	Command has alerted planning that there is a report of the suspect vehicle at the construction site near the monument. Redirect all of your resources south of the monument for further instruction.
:22	Radiation levels inside the construction site are reported to be high. When two of the robots arrive at the monument, manually teleoperate one the robots into the construction site and visually confirm that the construction vehicle is in the structure.

Each of the events was designed to exercise specific features of the interface and gesture set. First, at time :00, the division of the city is a common practice

and allows personnel to generically talk about areas of the city when they are not familiar with the local names for neighborhoods or districts. In this case, the city was divided into four equal square quadrants, A, B, C, and D, starting in the lower left hand corner and moving clockwise around the map. In densely populated areas, streets are a very common way of dividing search grids. The symmetry of the virtual urban environment meant that the city was effectively divided into four equal quarters of the overall map.

The second event simply divides the robots up into groups that can be referred to collectively. The number of robots in the team was a strategy to mirror the number of robots in the tasks explored in Chapter 7. This design strategy could later be used to see if interaction was different when dealing with one, two, or more than two robots at a time. The next three tasks begin moving the teams of robots through the environment. First the black robot was sent to the park, which minimally required the participant to execute single robot selection, destination, and map movement to see the park area. At :03 and :05, the two groups of robots were sent to different parts of the map on separate searches. The task exercised the gestures for group selection, planned movement, and possibly map movement depending on the view of the aerial camera. To encourage the detailed movement provided by the drag gesture, the black robot (now pre-positioned in the park) needed to be commanded to do a very detailed search of the interior of the park at :10. All of the users relied on the drag gesture for this task and used a variety of search patterns to canvas the area.

It is extremely typical for the search to be interrupted by requests from political figures and the press about the status of the search. The interruption event at :12 was a simple way to have the participant self-report on how well the operation was proceeding and the percentage of the city that had been searched. Updated intelligence reports from outside the search team can often redirect personnel to different locations of the search area even when an earlier task is not completed. The wording of the task at :15 is designed to provide one of these re-tasking of robots that were, by this point, spread all over locations in the first quadrant.

Furthermore, the request that the search be started “at the same time” provided a hint that the command queue functionality of the interface should be used to synchronize robot starting times.

At :20 the experimenter, acting as the safety manager, requires an immediate all-stop of the robots. The command was spoken with authority and without explanation. The intent here was to see if the all-stop gesture was memorable and could be executed quickly. Further details about the all-stop was then provided to the participant. The participant needed to re-group all of the available robots in a location near a construction site so that the participant could manually teleoperate one of the robots to a specific feature within the structure. The final task forced the use of the DREAM Controller in the scenario and helped determine if the reconfiguration of the video and range display panels was useful for inspection tasks. An example of this task is shown in Figure 8-9.

8.5.2 Data Collection

For the experiment, we provided the participants a clipboard with a paper copy of Tables 8.1 through 8.3 and a copy of the map in Figure 8-8. They were also given a pen and highlighter and told that they could mark directly on the sheets. We preserved the resulting sheets for post-hoc analysis.

As in previous experiments, the participants were asked to “think aloud” while using the interface and creating their search strategy. In many cases, the participants became engrossed in the task, and the test administrator would need to prompt the participants to explain what they were thinking or doing. The test administrators worked in a team of three. One person managed the computer console in the event of a program crash or other problem. One person took handwritten notes as the participants interacted with the surface and performed the think aloud method. The third person directly interacted with the participants as the command hierarchy, read any of the descriptive dialog, and answered any questions related to the interface.

We videotaped the session from three views. The video output from the Surface

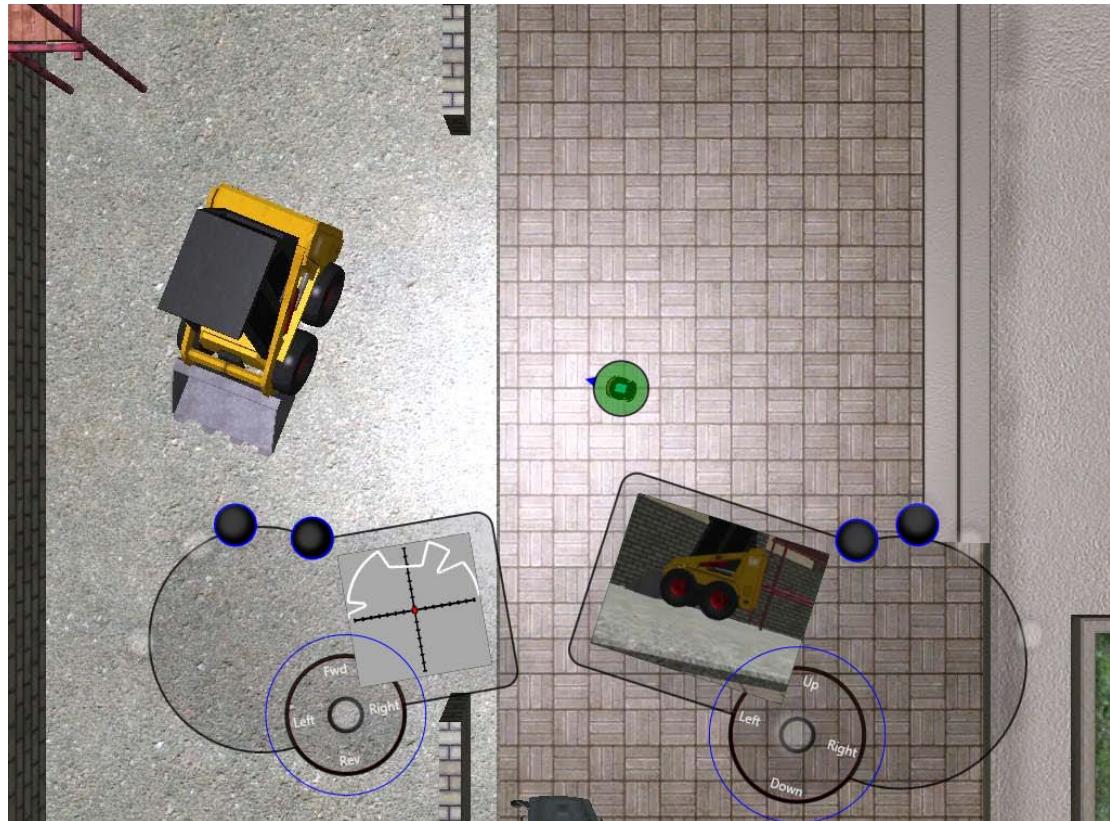


Figure 8-9: Final task of the scenario required the participant to take manual control of the robot (center) and use the DREAM Controller to view the suspect construction vehicle. This screenshot shows the activated DREAM Controller and positions of the fingers are shown lightened circles.

itself was output to a device that converted high resolution computer video to standard definition video. This output was then recorded to digital video tape. The second recording was of the person's hands on the surface of the screen. A digital video camera with wide angle lens was mounted approximately six inches above the Surface near the edge of the screen opposite the participants. This provided an angled view of the hand to help discern when contact occurred. Finally, high definition video was recorded from a view over participant's the shoulder to capture larger gross motion of the arms and body.

We asked the participants semantic differential questions upon the completion of the experiment. The experimenter responsible for note taking verbally asked these questions and recorded responses with handwritten notes.

8.6 Results and Discussion

To provide a cohesive discussion of the results gathered in this user study, we combined the responses to interview questions, participants comments, video observations, and distribution analysis to product the following results. Since this is an exploratory study based on the prototype in Chapter 7, we focused much of the post-hoc analysis on the variances between the two studies and how these differences compared to our expectations. The prototype study had a larger sample population, but the population was taken from the general public. Personnel in the search and rescue domain may exhibit different expectations and biases when interacting with this type of device. As discussed below, there were many patterns that matched, but with gesture based interfaces like our implementation, the difficult parts are often in subtle details.

8.6.1 Gestures as a function of grammar

We first looked at the distribution of gestures within their classifications in the grammar since this can provide insight regarding why qualitative comments or biases were provided in the questionnaire. A graph of the gestures used in this

study are shown in Figure 8-10.

Several observations can be made of this graph relative to the distribution seen in Table 7-3 in Chapter 7 (henceforth referred to as the 2009 study). Lasso was used much less in this study (only 4%) as opposed to 21% in the 2009 study. Additionally, the tap gesture as used for sequence select increased from 5% in the 2009 study to 43% in this study. While the reason for this change is not entirely clear and the sample population not sufficiently large enough to draw any strong conclusions, we can hypothesize several possible reasons. First, the number of robots in group actions was smaller in this study with eight total robots total and the largest sub-group containing five robots. In the 2009 study, there were a total of sixteen robots and eight robots in the largest sub-group. There may be a subtle threshold for when users switch over to the lasso gesture for group actions that was not exposed in this study. In the same way that users omit the subject when there are fewer than three robots on the screen, we may see an effect where lasso becomes the primary gesture for grouping when there are somewhere between eight and sixteen robots on the screen.

Second, an extension of the hypothesis above for the change in the percentage of lasso and sequence select may be further enhanced due to bias introduced by the sentence grammar structure. While planning the next set of operations, the participants may have thought in terms of a sentence with multiple subjects. For example, they might think, “Robot 1, Robot, 2, Robot 3, Robot 4, and Robot 5, follow this path and terminate at this point.” In this sentence, the individual representation of the robots may not seem awkward for five names. While acting out the gesture, performing a sequence select may seem natural since the user can tap each of the robots while recalling the name. When the number of robots in the subject list is increased, the beginning of the sentence becomes long and hard to remember. We hypothesize that this situation is when the lasso gesture becomes useful, since each of the individual robots is not as well defined. In an extension of the grammar, lasso becomes the plural pronoun “you.” Placed in our example sentence above, the large collection of robots to perform the task is substituted

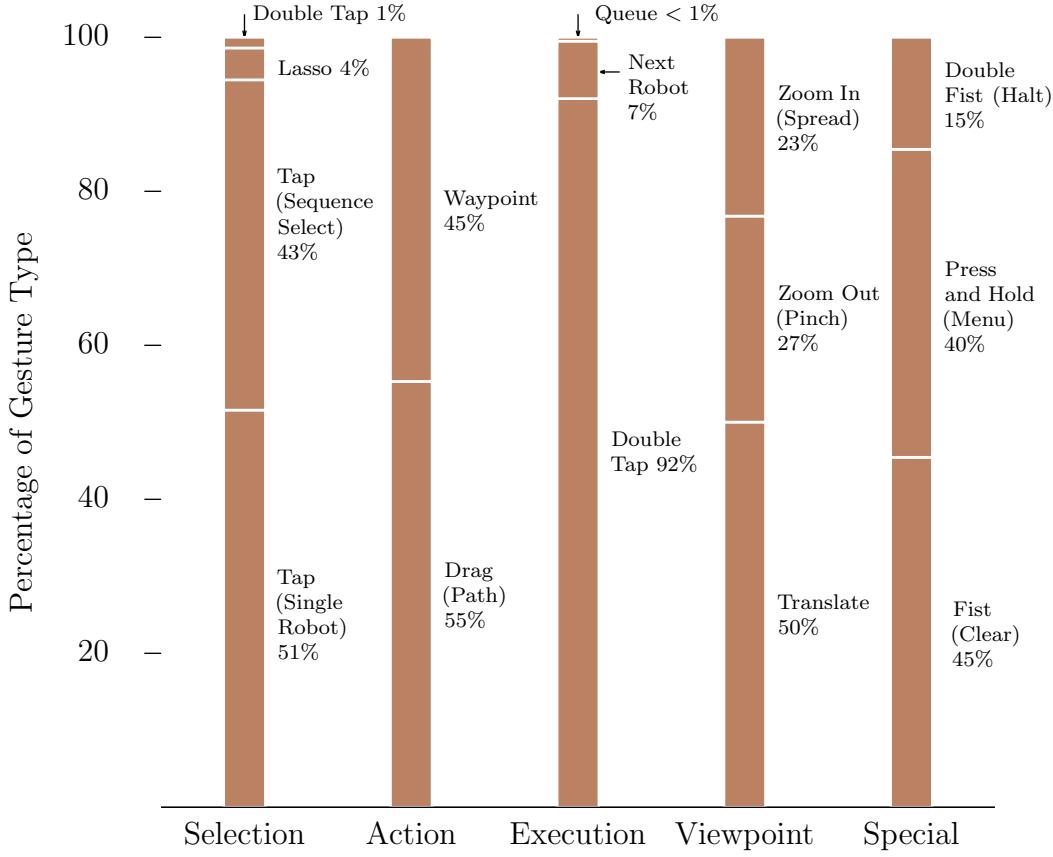


Figure 8-10: Percentages of gestures used, grouped by their position in the grammar discussed in Section 8.2.2.

as follows: “You (group of robots), follow this path and terminate at this point.” Further studies and sample populations would be needed to determine if these hypothesis is correct. A parametric experiment is needed to determine the number of robots that trigger this change in behavior from sequence select to lasso.

A second observation from this data is the dominant use of double tap in the predicate (92%) and the relative lack of double tap in the selection stage (1%). Like the unused on-screen controls described in Chapter 6, we spent a significant amount of time ensuring that the interface software was able to correctly interpret single and double taps for the selection portion of the grammar. We were surprised that this double tap for selection feature was used less than 1% of the time compared to the 7% usage in Chapter 7. Given the dominant use of double click in desktop applications, we expected to see users biased to this double tap selection gesture.

Overall, it should be noted that while this data provided important insights into the gestures used by our target population, the scenario was designed to elicit specific responses from the participants. As such, the distribution of gestures is biased by the scenario itself. For example, special gestures such as double fist, and press and hold, were actions pressed upon the user during the timed scenario. To refine the accuracy of these gesture distributions, longer exercises need to be performed that allow for a wider response of gestures from participants over time. However, in this scripted 25 minute exercise, the data provides a wealth of insight and clarifications about the assumptions and mental models used by the participants.

8.6.2 Gestures as a Function of Assumptions

As mentioned in Chapter 7, the level of tasking will vary based on the user's individual assumptions about the robot's autonomy and capabilities. Although the frequency and depth of discussion varied widely between participants, we did note several data points and themes that may point to the assumptions that participants used while interacting with the interface.

Four of the participants made specific comments about the autonomy of the robots. One participant directly stated that the robots were more autonomous than expected. With respect to both the primary search task and secondary radiation detection tasks, two of the participants made statements related to the robots having obstacle avoidance. One participant volunteered that she felt that the robots would not hurt anyone in the environment. Additionally, she added that the robots were smart enough for this task and, if needed, the robots could plan and navigate around buildings.

Trust in the sensors was a recurring theme during the experiment and in the post-experiment questionnaire. Five of the six participants stated comments related to the sensing capabilities of the robots. Four participants simply stated that they believed the robots would identify the victims or radiation sources with the installed sensors. Since this ability was in the original description, this comment

indicated that the robots behaved in a manner that supported this statement of identification. One participant mentioned that the sensors for victim detection should work without getting really close to the victims, and another participant stated that the radiation sensors needed to be close to the radiation source and would therefore need to stay nearer to structures. One participant mentioned that she assumed the robots had all of the same sensors.

These comments provide an interesting insight to the level of tasking that the participant felt that he or she needed to provide for the robot. In Chapter 7, we discussed that the perceived level of autonomy or trust in the robot's ability to navigate could change the natural interaction method used by the participants. We hypothesized that the participants' trust toward the robots may factor into the similar percentages between the use of drag (for a path) and waypoint. In this data set, drag represented 55% of the gestures and waypoint was 45% of the gestures. This gesture distribution indicated that approximately half of the time the participants believed the robot had sufficient autonomy such that the robot could be given simple waypoints for navigation to a destination.

Interestingly, one participant was observed using waypoints in all but the case of the black robot searching the park in detail. (In the case of the detailed park search, all of the participants used a drag gesture to provide a specific path.) This participant used waypoints for the majority of the navigation gestures and was one of the participants that immediately tasked the robots individually instead of in groups. Interestingly, this participant was one of the most experienced in wide-area search and had the most experience commanding these operations. When asked later why he used waypoints for navigation, he replied that "this is how I would have commanded people or dogs." While providing directions to another human, most people would not give extremely fine-grained information specific to the path that the person should follow. Rather, most people would provide landmarks or waypoints to achieve along the way to the goal.

The experience of this particular participant is important because the difference in his style of interaction may point to a difference between the assumptions

made with extensive command experience versus those with limited command experience. In the case of human searchers, providing explicit paths to follow would be considered micro-managing the process at an unnecessarily low level. An exception to this might be when command has specific information that will help the searchers such as a specific path through an area that is difficult to navigate. This participant demonstrates that experienced search team leaders may make assumptions that cause them to use a different distribution of gestures than less experienced leaders. These differences are important, since a command and control interface like this will need to be used by both experienced and unexperienced leaders.

8.6.3 Motivation for Further Improvement

Based on the comments from participants, there is room for improvement in the interface. Many of the comments centered around the zoom functionality. Four of the six participants mentioned that performing the zoom function was difficult to use. As a result of this, suggestions were made regarding how the participants would rather see the functionality expressed. One participant noted that the zoom was not distinct from the pan gesture, indicating that the mix of the two may have been problematic. The same participant suggested a zoom button that would automatically zoom all the way out to quickly give a high level view of the operation. One participant wanted a gesture similar to a scroll mouse instead of the pinch and spread gestures. A final participant found that the time zooming in and out was too time consuming, stating that a simple on-screen control would suffice.

Admittedly, the coordination of the aerial view and the zoom gesture had some bugs that were realized only just prior to testing. Depending on the speed and persistence of the finger spread or pinch, the detection routine could cause the camera to move erratically for a moment before assuming the correct altitude and position. All of the suggestions by the participants are useful and seem to refer to examples that they have seen work well in the past. Further testing with a

more robust implementation of our zoom functions will need to occur before we are able to determine if this problem is with the gesture itself or a problem with the implementation.

Other interface element problems were noted with less frequency. One participant mentioned having a problem while providing a path for the robot to follow. As a result, he resorted to only providing shorter paths. Another participant mentioned that it was “weird” providing paths around buildings. We believe that the last comment was due to the perspective provided by the aerial camera view. Since the camera was modeled with real lens characteristics, buildings on the edges of the display appear to lean outward due to perspective. Buildings in the middle appear to stand straight up toward the camera. This disparity can be problematic when drawing paths on the ground plane. A solution to this problem is not entirely clear, but this will be a common characteristic of aerial and satellite imagery.

A final point for improvement is the need to manage the situation where the robots are all in very close proximity to each other and the bounding circles overlap. This overlap is due to the 2 cm lower bound for contact established in Chapter 3. This problem reliably expressed itself during the last part of the dirty bomb scenario when the participants were asked to redirect all of their resources to a location south of the monument. All of the users selected the robots using lasso or group select and eventually terminated the grammar with a single destination. Unfortunately, this action caused all of the robots to arrive in the same place. Depending on the frequency of the arrivals, the participant quickly found that it was difficult to select a single robot from the group if the camera was sufficiently zoomed out to cause overlapping circles.

Since we cannot regulate the size of robots or expect to force a lower camera altitude before selection, an alternative must be found that permits selection when the density of robots is high. There are numerous techniques that can be borrowed from the visualization communities related to high density cluster mapping and similar graphics representations. Some example solutions to this problem have been demonstrated by Google Earth and other GIS products. Further investigation will

need to determine if these solutions are natural and compatible with multi-touch interaction.

8.6.4 Information Fusion and Decision Support

A common theme among the comments was the need for more status information in future implementations. Many of the participants made comments about losing track of the waypoints or paths that the robot was following, or had already followed. Note that the interface did show the waypoints or path that the robot was going to follow as the finger was dragged, but this path was erased from the screen to reduce clutter once the predicate was given. Normally, the search team leaders would keep track of this information using pen and paper. In this case, the ease of tasking the robots and the temporary nature of the path visualization may have given a false sense of recorded planning data. One participant commented at the end of his run that “I didn’t use paper at all, and I didn’t feel like I needed it.”

The need for past and future goals is important when earlier decisions have been forgotten and new plans need to be made that may affect or overlap those earlier plans. A simple visualization to remedy this problem may be for the interface to remember the path or waypoints and show them again if the robot is selected while executing the task. In addition, past waypoints can be shown in a different color or shape than the future waypoints.

While only four participants directly commented about the zoom function, we observed that all six used it frequently. This change in perspective may suggest a need to maintain a high level operations view and a lower task level view. It is a common technique in many first person shooter and real-time strategy games to provide a small map of the entire operational area that is coordinated with the information displayed in the main window. The ability to see the map and assets in this broader scope was specifically suggested by one of the participants who had experience with video games.

This type of high level map could be implemented easily in a way similar to the map generated by the single robot in Chapter 6. This smaller map could provide

all of the known area in the scope of operation and be moved or adjusted in size on demand. Additionally, the user could provide a gesture such as a double tap on this operation-level view to swap the two displays much like the ADR mode discussed in Chapter 6.

8.6.5 Questionnaire Results

We asked 11 semantic differential scale questions anchored by opposite phrases such as “helps” or “hinders.” In each question, the participants were asked to respond to a number between 1 and 6 relating to how strongly they felt on that subject. These questions and the results are shown in Table 8.4. In general the results are positive and the participants believe that the interface would bring value to a search operation. We believe that the lowest value for the question “The interface was uncomfortable/comfortable to use” was mostly due to the shape of the Surface since the participants were sitting down and could not place their feet under the table despite it being raised to 27 inches. This inconvenience caused the participants to need to lean from a seated position to interact with the surface of the screen. Future plasma or liquid crystal display multi-touch surfaces would have not exhibited this problem and are already available.

8.6.6 User Expectations and Need for Intervention

All six participants mentioned that the robot did what they expected. While it is not 100% clear what those individual expectations may have been, the robot behaviors appeared to meet a minimum level of approval. When asked if they trusted the robots, all six responded that they did trust the robot, with only one user further qualifying that he trusted the robots to navigate, but not necessarily for sensing.

One of our original design requirements was the ability to directly intervene if a robot should be stuck or become disabled. This experiment verified this need since five out of the six participants mentioned a robot getting stuck. Three of

Table 8.4: List of the semantic differential scale questions and the results from the post experiment interview.

Question	Average	SD						
The interface helped/hindered in performing my primary task hindered <table style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table> helped	1	2	3	4	5	6	4.17	0.75
1	2	3	4	5	6			
The interface helped/hindered in performing my secondary task hindered <table style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table> helped	1	2	3	4	5	6	4.33	0.75
1	2	3	4	5	6			
The interface was difficult/easy to learn difficult <table style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table> easy	1	2	3	4	5	6	4.25	0.84
1	2	3	4	5	6			
The interface was difficult/easy to use difficult <table style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table> easy	1	2	3	4	5	6	4.08	1.10
1	2	3	4	5	6			
The interface was irritating/pleasant to use irritating <table style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table> pleasant	1	2	3	4	5	6	4.00	0.41
1	2	3	4	5	6			
The interface was uncomfortable/comfortable to use uncomfortable <table style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table> comfortable	1	2	3	4	5	6	3.92	0.75
1	2	3	4	5	6			
The interface was inefficient/efficient to use inefficient <table style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table> efficient	1	2	3	4	5	6	4.33	0.63
1	2	3	4	5	6			
The interface has no/much useful functionality none <table style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table> much	1	2	3	4	5	6	4.67	0.52
1	2	3	4	5	6			
The interface was no help/helpful in maintaining safety no help <table style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table> helpful	1	2	3	4	5	6	4.50	0.52
1	2	3	4	5	6			
The interface was hard/easy to understand hard <table style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table> easy	1	2	3	4	5	6	4.42	1.03
1	2	3	4	5	6			
The interface would require more/less time than my current system to maintain awareness of the situation more <table style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table> less	1	2	3	4	5	6	4.50	0.55
1	2	3	4	5	6			

the participants reported robots that flipped over, rolled on their side, or rolled onto their backs (looking at the sky) during the run. Another two participants mentioned that at least one of their robots became disabled or hung up on objects in the environment. In one case, the robot's behavior caused the robot to begin rotating in circles regardless of the waypoint position.

As a testimony to the need for manual operator interaction, two of the participants were able to take manual control of the robots and remove them from the failure state. These rescued robots were then given a task and continued throughout the remainder of the testing period. In all other cases, the robot was not able to be rescued or the participant did not attempt a rescue with the DREAM Controller.

It should be noted that the robots were not designed to fail intermittently and were programmed to use obstacle avoidance with the popular and publicly available vector-field histogram behavior (Borenstein and Koren, 1991) from the Player robot architecture (Gerkey et al., 2003). This algorithm was ported directly from Player to MRDS with no change to the underlying logic. It was hoped that this well tested behavior combined with the simulation environment would provide sufficient robot autonomy to avoid any failures of the robots while they navigated through the environment.

The realism of the simulation and low ground clearance of the Pioneer robots modeled in this simulation performed much like one would expect in the real world. The robots would occasionally not sense an object in the environment and crash into it. In some cases, these objects were bushes or a ramp that was below the laser rangefinder's view. While these instances of errors in sensing and judgement are particular to this virtual world, robots in the real world have much more noise in their sensors and their algorithms need to deal with much more difficult situations.

8.7 Impacts and Implications

This test of our implementation of a command and control interface for multiple robots brought a wealth of information. As discussed above, there is still room for improvement. The responses from our participants was unanimously supportive and encouraging. Comments included “That is a cool interface,” “My strategy improved as it went along,” and “That is it? Can I keep playing?” From a qualitative standpoint this response indicates that the interface was a positive experience and the responders recognized the potential.

Our earlier goal stated that we had created a gesture set will maximize learnability and therefore lessen the amount of training time required for proficient control of the robot or robot teams. While it is difficult to directly measure this goal without a benchmark for comparison, we do have several points that indirectly support this goal. First, the responders were trained on the interface and then given any amount of time to practice before beginning the timed scenario. They were specifically told that there was no time restrictions and that they could practice until they felt that they understood all of the functionality. Participants on average took 17 minutes ($SD = 8.8$) to feel confident and proficient in the interface, including the scripted training time. Also, the scenario itself was designed to run for approximately 25 minutes in length. Our participants were able to command the robots through the search scenario and manually teleoperate one of the robots to a specific point in 27 minutes on average ($SD = 2.8$).

The timing benchmarks above indirectly support our goal since 15 minutes for training seems reasonable for an interface that will presumably allow for better situation awareness and increased operational efficiency. On average the operation ended within two minutes of the expected time, indicating that the interface did not have a large variability in expected versus measured task completion.

We believe that this gesture set and interface design establishes a benchmark and provides an existence proof that robot command and control can be achieved in a gesture-based interface.

Chapter 9

Conclusions and Future Work

Ideally, the next step in this line of research would be to simply place this new interface and control mechanism in the field and let the end users and other developers further iterate on the design. However, technology adoption in application domains such as search and rescue does not happen as quickly as technologists would like. Fortunately, when all of the enabling technologies described in Chapter 1 are mature and field-ready, this research has provided a strong existence proof that multi-touch technology is not only compatible with varying levels of robot control, but may significantly improve the performance of robot operators.

This dissertation has provided several key components of command and control. It has provided a multi-touch interface for teleoperation in the field and a command and control interface for high-level coordination of robot teams. The simultaneous use of these components can be described in a realistic, but fictional scenario based in the application domain of military explosive ordinance disposal (EOD).

When bomb squad technicians are called to the location of a suspicious package or automobile, their goal is to remove or disable the explosive device. The state of the practice involves controlling a robots through large operator control units, weighing twenty to fifty pounds. For example, the iRobot Packbot EOD and Foster Miller Talon operator control units can be seen in Chapter 5 in Figure 5-1. Quickly setting up and using these operator control units while in possible contact

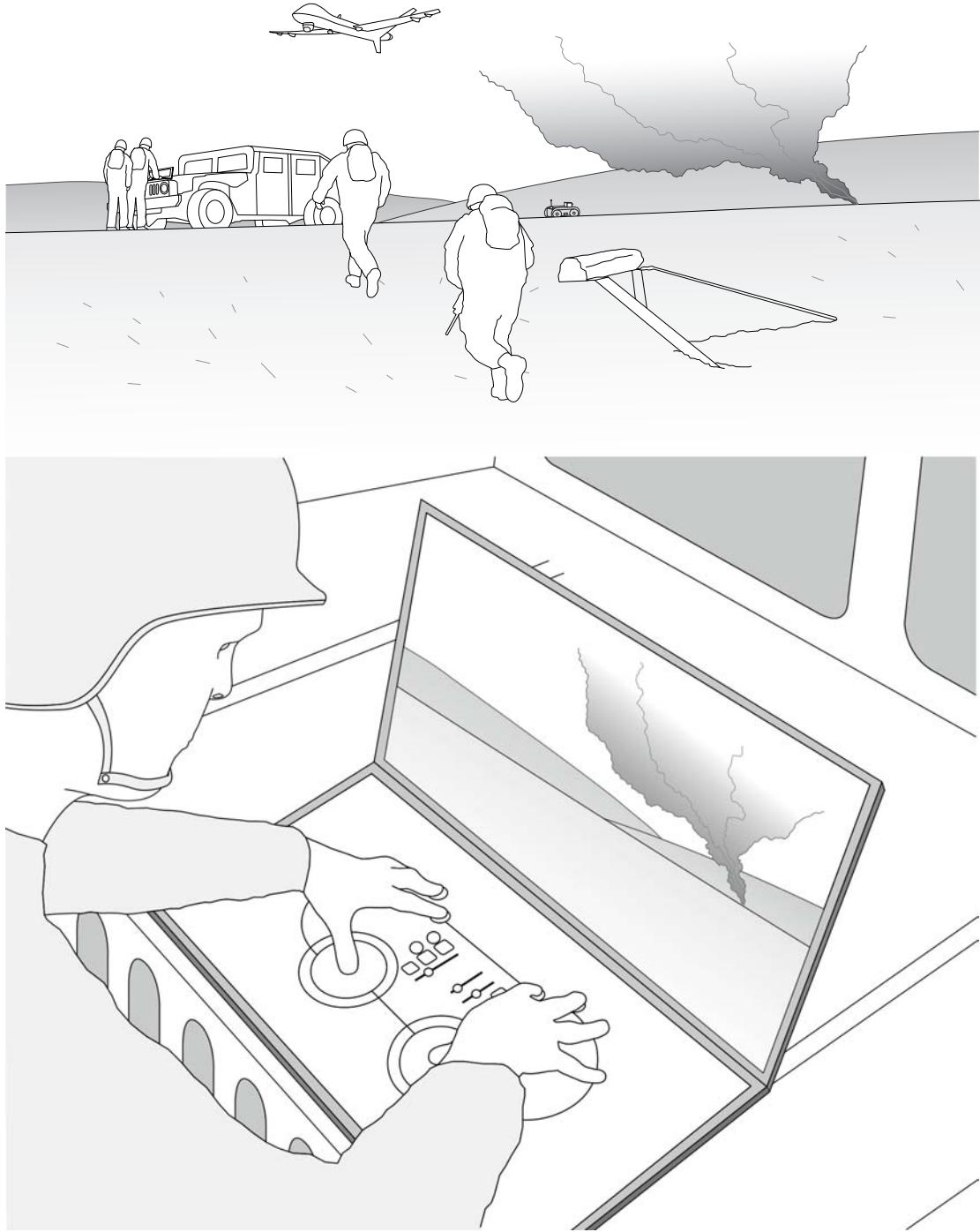


Figure 9-1: Multi-touch technology will be making its way into the field in the very near future. The UAV (above) provides real-time imagery to the command staff back at the base of operation while the robot control unit (below) allows the field technicians to comfortably control the UGV robot and neutralize the threat.

with enemy forces can be daunting and stressful at best.

In our fictional scenario, imagine that instead of the large and heavy controller, the bomb technician has a device that looks like a thin and wide laptop where the keyboard is replaced with a multi-touch surface large enough to accommodate two adult hands. (An artist's interpretation of this fictional scenario is shown in Figure 9-1.) The technician is able to place this device on the tailgate or hood of the vehicle. When it is opened, it shows a display of the EOD robot cameras, manipulator positions, and the state of the robot sensors on the front screen. The technician places both hands on the multi-touch screen and two virtual joysticks are immediately drawn beneath the fingers. Without having to adjust posture for manual switches and dials, the fingers simply glide to the desired positions and chording actions are performed within the comfort and ergonomics of that particular user's bio-mechanics. When the technician needs to relinquish control to a second operator, the system automatically recreates the joysticks for the ergonomics of this new user and the mission can continue without the new operator needing to inspect and recall the positions of the switches and dials.

If the safety of the situation degrades and these technicians radio back to their command and request further assets, the commanders look at the multi-touch table in the command post and identify the assets that are immediately available in the area. After several selection and destination gestures by the commander, the requested resources are en route to the scene and arrival estimations are automatically relayed to the team on the ground via radio. Since the situation may be escalating outside of the view of the field EOD technicians, the commander selects a nearby aerial asset and then puts five fingers down on the display surface. A joystick controller with video display appears underneath the fingers, and a quick movement of the thumb servos the pan and tilt camera on the belly of the aircraft to the location of the operation. Recognizing further complications several blocks away and outside of the view of the EOD technicians, the commander quickly releases control of the camera by simply lifting the hand from the screen, makes a lasso gesture around the remaining human force protection resources, and redirects

them on a path that will provide protection to the evolving incident.

Even five years ago, this narrative story would be safely in the realm of hollywood science fiction. Today, with the enabling technologies described in Chapter 1 and the multi-touch interfaces designed and tested in this dissertation, this scenario can quickly become a reality.

9.1 Contributions

This line of research has provided the following contributions to the field of human robot interaction:

- **Performance model of tabletop touch interfaces based on Fitts's Law:** After a literature and background search, we had no indication when we began working with tabletop devices that they would have performance comparable to traditional mouse interfaces. In Chapter 3 we were able to create a performance model and experimentally show that mean error rate was comparable to the mouse interface for targets of 30 mm and 40 mm, but substantially higher for target diameters of 20 mm and smaller. Additionally, efficient task completion can be expected only when target elements in the user interface are larger than approximately 30 mm in size.
- **Validation of multi-touch human-robot interaction compared to a traditional joystick-based single robot interface:** An interface was designed for robot control that demonstrated comparable performance to traditional joystick-based robot control using a well studied interface design. To our knowledge, our study in Chapter 4 represents the first use of a multi-touch table with a physical agent. We experimentally showed that there was no significant difference for the constructive or destructive performance metrics established for this study.
- **Algorithm for five-point hand identification and finger registration:** In Chapter 5, we created an algorithm that with 97% accuracy can identify

a hand, label the fingers as a thumb, index, middle, ring, or little finger, and label it as a right or left hand. The algorithm can accomplish this identification using only the computationally optimized arctan2 function and basic arithmetic functions. While the algorithm was a means to an end for the DREAM Controller, we believe that the contribution from this algorithm will be useful for design spaces beyond this application.

- **Interface using the dynamically resizing, ergonomic, and multi-touch (DREAM) controller for joystick emulation:** This controller provides the user with four degrees of freedom of control on the thumbs and two buttons on the index and middle finger used for tracking in this implementation. Based on the metrics presented in Chapter 6, we have demonstrated increased performance over a well studied joystick controller implementation for robot control.
- **Taxonomy and coding rule set for multi-touch multi-robot interaction on tabletop surfaces:** In Chapter 7, extensive coding of over 3000 gestures required the creation of a taxonomy and coding rule set for the user generated gesture set. This taxonomy can be used in future research to help codify and identify the gestures from users performing command and control tasks on table top surfaces.
- **A user generated multi-touch gesture set tailored for ease of learning:** Using the distribution of gestures from Chapter 7, we were able to adapt the the most popular gestures into a grammar that permitted the user to command a robot, or groups of robots, to perform navigation tasks using only their fingers and hands.
- **Integrated interface designed specifically for multi-touch interaction that combines high level command and control of robot teams and individual control of single robots:** The interface described in Chapter 8 leveraged the gesture grammar in Chapter 7 to provide high level

control of groups of robots and used an adaptation of the DREAM Controller described in Chapters 5 and 6 to control individual robots and view their sensors.

Of these contributions, the algorithm for five-point hand identification represents the most significant contribution due to the changes that this can enable in general user interface design. The identification of individual fingers and handedness can permit much more rich and generalizable interactions. For instance, if the fingers are identified and labeled, they can represent the “asdf and jkl;” home-row on a computer keyboard or the ivory keys on a piano. Combined with heuristics for hand movement, the identification of the hands and fingers can provide ergonomic emulation of these devices without the need for direct tactile feedback to the user. This algorithm provides a method for a level of interaction not achieved by multi-touch interfaces designed today.

9.2 Future Work

There is some future work that will immediately benefit this line of research.

- **Refinement of the hand detection algorithm:** In Chapter 5, we discuss some of the limitations of the hand and finger detection algorithm. By exploring some of the techniques suggested in Section 5.8, the performance of the algorithm can be improved and should allow the algorithm to achieve nearly 100% accuracy for all hand configurations and angles.
- **Direct comparison of the DREAM Controller and a Playstation or XBox controller:** Chapter 4 and Chapter 6 base their comparisons on a joystick modeled after aircraft flight-controller designs. The inspiration for the DREAM Controller design is based on console gaming-style joysticks that use different muscle groups and ergonomics. It would be useful to perform a direct comparison between the DREAM controller design and the Playstation or XBox controller.

- **Further exploration of gestures as a function of grammar:** As an explorative study, Chapter 8 provided as many questions as it answered. Section 8.6.1 suggested that the higher use of sequence select may indicate a subtle adaptation of the underlying grammar. Additionally, experiments exploring the removal of double tap in the selection stage may reduce mistakes and confusion for the users. Both of these behaviors should be studied to adjust the grammar gesture models.
- **Further exploration of gestures as a function of assumptions:** Section 8.6.2 provided a unique data point that may indicate that responders with different levels of experience may wish to provide different gestures to the robots being controlled. Specifically, the more experienced participant felt that he should provide only high level instructions to the robots and not specific paths. The other participants that largely used drag gestures to direct robots along specific paths. Testing with larger populations may show that behavior is a trend and allow the interface to leverage this information to adjust gesture parameters.

9.3 Limitations of Research

It is important to note the aspects not yet addressed in this line of research. While they are limitations of this research, they also represent long-term future work that will hopefully be continued by future studies and researchers.

First, the hand detection algorithm has only been designed and tested for the two-dimensional case over the surface of a tabletop. This research does not address interaction in the third dimension, often referred to as “above the tabletop” interaction in the literature (Wilson, 2007; Grossman and Wigdor, 2007; Grossman and Balakrishnan, 2008). For all of the richness that can be argued in the case of gestures directly on the tabletop, there is another entire dimension to the gesture interaction that can occur over the table surface. This research has not addressed this dimension.

Second, this research does not address multi-person interaction. In Chapter 1, we framed the motivation for this research in the context of a response in which multiple people would be standing and interacting around a large touch-table that provided many of the affordances of traditional paper maps. The experiments and results presented in this thesis are limited to the single person case, in part, due to the size of the devices used. The literature in this area shows that adding multiple people to the table-top interaction introduce confounders and interesting effects for the interface designer (Scott et al., 2004; Scott and Carpendale, 2006; Tuddenham and Robinson, 2007; Hancock et al., 2006). As multi-touch tabletops become larger and algorithms for detection of individual people become more robust, future investigations can extend the interfaces from this research beyond single-person interaction.

9.4 Generalizations of Research

There are several aspects of this research that can be generalized through further study. First, as mentioned in the section on contributions, the hand detection algorithm can be generalized to allow the emulation of many physical devices that require multi-finger and multi-hand control. We have demonstrated joystick emulation in this dissertation, but conceivably keyboards, mice, and even musical instruments could be emulated using the unique identification of the finger tips. As mentioned at the end of Chapter 6, the ability to dynamically adapt to the users configurations that gives multi-touch interaction a significant advantage over traditional mechanical device design.

It is exciting to think that the generalization of this algorithm can open entirely new types of interactions for multi-touch tabletop interfaces. It is our hope that this algorithm and our demonstration of its use in the DREAM controller will cause the multi-touch community at large to re-evaluate the way that we design interfaces for these devices and seek out other physical devices that may be adapted to multi-touch devices now that fingers and hands can be uniquely identified.

Other generalizations can occur with the DREAM Controller design in demonstrated in Chapters 6 and 8. In our studies, we have limited the design of the DREAM Controller to emulate the dual-gimbal design from Playstation and XBox controllers in an effort to increase ease of learning for users that have experience playing console video games.

A departure from the Playstation and XBox joystick design may provide ergonomic or performance improvements beyond the simple emulation of these physical devices. There is no research in the scientific literature that states that dual-gimbal thumb interaction is the provably correct and most efficient use of the human hand for first-person robot or character movement. This joystick design provides four degrees of freedom through the movement of the thumbs, but this movement may be better served in some way other than proportional control on two axes. There is also a high degree of two-dimensional movement in the index and ring finger that may provide additional input other than the tracking in our implementation.

Finally, in the case of the command and control portion of this research, the work presented can be generalized for heterogeneous teams of robots and humans. We have framed this problem in the context of unmanned ground robots, but one can see this research extended to general command and control for humans, vehicles, and robots of all types. Given an aerial view of the affected area and the geo-location of the resource to be tracked, we can circle the location on the interface and provide all of the functionality demonstrated in the interface from Chapter 8. One can imagine, instead of the location being sent to the robot, that a text-to-speech device can radio to the human agents where they need to rendezvous for their next mission objective.

When dealing with large scale disasters where street signs and landmarks may be destroyed, this top-down view provided by our interface not only helps those in the command structure but those on the ground. Given sufficient reach-back ability, one can imagine the case where the human is provided navigation directions through a handheld device. This generalization of our research would

provide significant value to operations in these heavily destroyed and confusing environments.

9.5 Closing Thoughts

This research was intended to enhance human-robot interaction by carefully studying the biomechanics of the human hand and leveraging the natural responses of users to multi-touch displays. Through this research, we sought to maximize learnability and therefore lessen the amount of training time required for proficient control of the robot or robot teams. Through six user studies and many years of iterative improvements to our interface designs, we have achieved a volume of results and contributions that previously did not exist in the cross section of these research domains.

It is our sincere hope that, as the enabling technologies of robotics, network centric operation, and multi-touch tabletop devices begin to proliferate in all of the emergency response and military command and control domains, this research can help provide a bridge between the technology and the users that will rely on it for future mission successes.

Bibliography

- A. Agarwal, S. Izadi, M. Chandraker, and A. Blake. “High Precision Multi-touch Sensing on Surfaces Using Overhead Cameras.” In *Proceedings of the 2nd Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (Tabletop 2007)*, pages 197–200, 2007.
- D. Alberts and R. Hayes. “The Future of Command and Control: Understanding Command and Control.” *Command and Control Research Program (CCRP) Publication Series*, 2006.
- R. O. Ambrose, H. Aldridge, R. S. Askew, R. R. Burridge, W. Bluethmann, M. Diftler, C. Lovchik, D. Magruder, and F. Rehnmark. “Robonaut: NASA’s Space Humanoid.” *IEEE Intelligent Systems*, 15:57–63, 2000.
- W. Barfield and T. Furness III. *Virtual environments and advanced interface design*. Oxford University Press, Inc. New York, NY, USA, 1995. ISBN 0195075552.
- R. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. Goodrich. “Autonomous Vehicle Technologies for Small Fixed Wing UAVs.” *AIAA Journal of Aerospace Computing, Information, and Communication*, 2(1):92–108, 2005.
- J. Borenstein and Y. Koren. “The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots.” *IEEE Journal of Robotics and Automation*, 7(3):278–288, 1991.
- M. Bruch. “The Multi-Robot Operator Control Unit (MOCU).” In *SPIE Proceed-*

ings 6230: *Unmanned Systems Technology VIII, Defense Security Symposium*, 2006.

S. Bunnell. “Surgery of the Intrinsic Muscles of the Hand Other Than Those Producing Opposition of the Thumb.” *The Journal of Bone and Joint Surgery*, 24(1):1, 1942.

B. Buxton. “Multi-Touch Systems That I Have Known and Loved.” *Microsoft Research*, 2007. <http://www.billbuxton.com/multitouchOverview.html> (accessed 7 October 2010).

S. Card, T. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum, 1st edition, 1983.

Circle Twelve. “DiamondTouch by Circle Twelve.” Website, 2009.

Committee on Planning for Catastrophe. *Successful Response Starts With a Map - Improving Geospatial Support for Disaster Management*. The National Academies Press, 2007.

F. Crescenzi, G. Miranda, F. Persiani, and T. Bombardi. “A First Implementation of an Advanced 3D Interface to Control and Supervise UAV (Uninhabited Aerial Vehicles) Missions.” *Presence: Teleoperators & Virtual Environments*, 18(3): 171–184, 2009.

C. Dang, M. Straub, and E. André. “Hand Distinction for Multi-touch Tabletop Interaction.” In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 101–108, 2009.

P. Dietz and D. Leigh. “DiamondTouch: A Multi-User Touch Technology.” In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 219–226, 2001.

M. Diftler, R. Platt, C. Culbert, R. Ambrose, and W. Bluethmann. “Evolution of the NASA/DARPA Robonaut Control System.” In *Proceedings of the IEEE*

International Conference on Robotics and Automation (ICRA), volume 2, pages 2543–2548, 2003.

S. Douglas, A. Kirkpatrick, and I. MacKenzie. “Testing Pointing Device Performance and User Assessment with the ISO 9241, Part 9 Standard.” In *SIGCHI Conference on Human Factors in Computing Systems*, 1999.

H. Dreyfuss. *Designing for People*. Allworth Press, 1955.

J. L. Drury, M. Micire, and H. A. Yanco. “New Technology Applications in Hazardous Materials Operations.” *Journal of Emergency Management*, 2010. In press.

M. Endsley. “Design and Evaluation for Situation Awareness Enhancement.” In *Proceedings of Human Factors and Ergonomics Society Annual Meeting*, volume 32:2, pages 97–101. Human Factors and Ergonomics Society, 1988.

J. Epps, S. Lichman, and M. Wu. “A Study of Hand Shape Use in Tabletop Gesture Interaction.” In *CHI’06 Extended Abstracts on Human Factors in Computing Systems*, pages 748–753, 2006.

K. A. Ericsson and H. A. Simon. “Verbal Reports as Data.” *Psychological Review*, 87:215–251, 1980.

A. Esenther, C. Forlines, K. Ryall, and S. Shipman. “DiamondTouch SDK: Support for Multi-User, Multi-Touch Applications.” In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2002.

P. Fitts and R. Deninger. “SR Compatibility: Correspondence among Paired Elements within Stimulus and Response Codes.” *Journal of Experimental Psychology*, 48(6):483–92, 1954.

T. Fong and C. Thorpe. “Vehicle teleoperation interfaces.” *Autonomous robots*, 11(1):9–18, 2001. ISSN 0929-5593.

- T. Fong, C. Thorpe, and B. Glass. “PdaDriver: A Handheld System for Remote Driving.” In *Proceedings of the 11th IEEE International Conference on Advanced Robotics (ICAR)*. IEEE, 2003.
- Foster Miller. “Tallon Robots.” Website, 2010. <http://www.talonrobots.com> (accessed 23 June 2010).
- B. Gerkey, R. Vaughan, and A. Howard. “The Player/Stage Project: Tools for Multi-robot and Distributed Sensor Systems.” In *Proceedings of the 11th IEEE International Conference on Advanced Robotics (ICAR)*, pages 317–323, 2003.
- B. Glaser and A. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine de Gruyter, New York, 1967.
- R. Green. “time: Java Glossary.” 2008. Available online at <http://mindprod.com/jgloss/time.html> (accessed 24 March 2010).
- A. Greenfield. *Everyware: The Dawning Age of Ubiquitous Computing*. Peachpit Press, 2006.
- T. Grossman and R. Balakrishnan. “Collaborative Interaction with Volumetric Displays.” In *Proceeding of the 26th Annual ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 383–392, 2008.
- T. Grossman and D. Wigdor. “Going Deeper: A Taxonomy of 3D on the Tabletop.” In *Proceedings of the 2nd Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (Tabletop 2007)*, pages 137–144, 2007.
- J. Gyorfi, E. Buhrke, M. Tarlton, J. Lopez, and G. Valliath. “Applying Telepresence to Incident Management: The Virtual Incident Command Center.” *Presence: Teleoperators & Virtual Environments*, 17(3):231–241, 2008.
- C. Hager-Ross and M. Schieber. “Quantifying the Independence of Human Finger Movements: Comparisons of Digits, Hands, and Movement Frequencies.” *Journal of Neuroscience*, 20(22):8542, 2000.

- J. Y. Han. “Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection.” In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 115–118, 2005.
- J. Y. Han. “Jeff Han Demos His Breakthrough Touchscreen.” Presented at TED2006, 2006. http://www.ted.com/talks/jeff_han_demos_his_breakthrough_touchscreen.html (accessed 7 October 2010).
- M. Hancock, F. Vernier, D. Wigdor, S. Carpendale, and C. Shen. “Rotation and Translation Mechanisms for Tabletop Interaction.” In *Proceedings of the 1st Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (Tabletop 2006)*, pages 79–86, 2006.
- S. T. Hayes, E. R. Hooten, and J. A. Adams. “Multi-Touch Interaction for Tasking Robots.” In *Proceedings of the 5th ACM/IEEE International Conference on Human Robot Interaction (HRI)*, 2010.
- A. Howard. “Multi-robot Simultaneous Localization and Mapping using Particle Filters.” *International Journal of Robotics Research*, 25(12):1243–1256, 2006.
- ISO 2000. “Part 9: Requirements for Non-keyboard Input Devices.” In *ISO 9241-9 International Standard: Ergonomic Requirements for Office Work With Visual Display Terminals (VDTs)*. International Organization for Standardization, 1998.
- S. Izadi, A. Agarwal, A. Criminisi, J. Winn, A. Blake, and Fitzgibbon. “C-Slate: A Multi-Touch and Object Recognition System for Remote Collaboration using Horizontal Surfaces.” In *Proceedings of the 2nd Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (Tabletop 2007)*, pages 3–10, 2007.
- R. Kalawsky. *Science of virtual reality and virtual environments*. Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA, 2004. ISBN 0201427737.

- J. Kato, D. Sakamoto, M. Inami, and T. Igarashi. “Multi-Touch Interface for Controlling Multiple Mobile Robots.” In *Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems*, pages 3443–3448, 2009.
- H. Keskinpala, J. Adams, and K. Kawamura. “PDA-based Human-Robotic Interface.” In *IEEE International Conference on Systems, Man and Cybernetics*, pages 3931–3936, 2003.
- B. Keyes. “Evolution of a Telepresence Robot Interface.” Master’s thesis, University of Massachusetts Lowell, Lowell, MA, 2007.
- B. Keyes, R. Casey, H. Yanco, B. Maxwell, and Y. Georgiev. “Camera Placement and Multi-Camera Fusion for Remote Robot Operation.” In *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics*, 2006.
- H. Koskinen, J. Laarni, and P. Honkamaa. “Hands-On the Process Control: Users Preferences and Associations on Hand Movements.” In *Conference on Human Factors in Computing Systems*, 2008.
- M. Krueger. *Artificial Reality*. Addison-Wesley Professional, 1983.
- M. Krueger. *Artificial Reality II*. Addison-Wesley Professional, 1991.
- M. Krueger, T. Gionfriddo, and K. Hinrichsen. “VIDEOPLACE — An Artificial Reality.” *ACM SIGCHI Bulletin*, 16(4):35–40, 1985.
- G. Landis. “Teleoperation from Mars Orbit: A Proposal for Human Exploration.” *Acta Astronautica*, 62(1):59–65, 2008.
- I. MacKenzie. “Movement Time Prediction in Human-Computer Interfaces.” *Readings in Human-Computer Interaction*, 2:483–493, 1995.
- J. Matejka, T. Grossman, J. Lo, and G. Fitzmaurice. “The Design and Evaluation of Multi-finger Mouse Emulation Techniques.” In *Proceedings of the 27th Inter-*

national Conference on Human Factors in Computing Systems, pages 1073–1082, 2009.

M. Micire, M. Schedlbauer, and H. Yanco. “Horizontal Selection: An Evaluation of a Digital Tabletop Input Device.” In *Proceedings of the 13th Americas Conference on Information Systems*, 2007.

M. Micire, J. L. Drury, B. Keyes, H. A. Yanco, and A. Courtemanche. “Performance of Multi-Touch Table Interaction and Physically Situated Robot Agents.” Technical report, University of Massachusetts Lowell, 2008. Report No. 2008-002. Presented as a poster at the *rd Annual ACM/IEEE Conference on Human-Robot Interaction*.

M. Micire, M. Desai, A. Courtemanche, K. Tsui, and H. Yanco. “Analysis of Natural Gestures for Controlling Robot Teams on Multi-touch Tabletop Surfaces.” In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 41–48, 2009a.

M. Micire, J. Drury, B. Keyes, and H. Yanco. “Multi-Touch Interaction for Robot Control.” In *Proceedings of the 13th ACM International Conference on Intelligent User Interfaces (IUI)*, pages 425–428, 2009b.

M. J. Micire. “Evolution and Field Performance of a Rescue Robot.” *Journal of Field Robotics*, 25(1-2):17–30, 2008.

Microsoft. “Welcome to Microsoft Surface.” Website, 2007. Available at <http://www.microsoft.com/surface> (accessed 6 December 2007).

H. Nguyen, R. Laird, G. Kogut, J. Andrews, B. Fletcher, T. Webber, R. Arrieta, and H. Everett. “Land, Sea, and Air Unmanned Systems Research and Development at SPAWAR Systems Center Pacific.” In *Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7332, page 38, 2009.

J. Nielsen. *Usability Engineering*. Academic Press, San Diego, CA, 1993.

- M. Nielsen, M. Störring, T. Moeslund, and E. Granum. “A Procedure for Developing Intuitive and Ergonomic Gesture Interfaces for Man-machine Interaction.” Technical report, Aalborg University, 2003. Report No. CVMT 03-01, www.cvmt.dk/~fgnet/docs/fgnet_techreport.pdf.
- D. Norman. *The Design of Everyday Things*. Basic Books, New York, N.Y, 1988.
- D. Perzanowski, A. Schultz, W. Adams, E. Marsh, and M. Bugajska. “Building a Multimodal Human-Robot Interface.” *Intelligent Systems*, 16(1):16–21, 2001.
- J. Rekimoto. “SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces.” In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 113–120, 2002.
- A. J. Rowe, K. K. Ligett, and J. E. Davis. “Vigilant Spirit Control Station: A Research Testbed for Multi-UAS Supervisory Control Interfaces.” In *Proceedings of the International Symposium on Aviation Psychology*, 2009.
- D. Saffer. *Designing Gestural Interfaces: Touchscreens and Interactive Devices*. O’Reilly, 1st edition, 2008.
- M. Schedlbauer. “An Extensible Platform for the Interactive Exploration of Fitts’ Law and Related Movement Time Models.” In *CHI’07 Extended Abstracts on Human Factors in Computing Systems*, page 2638. ACM, 2007.
- S. Schipani and E. Messina. “Maze Hypothesis Development in Assessing Robot Performance During Teleoperation.” *National Institute of Standards and Technology NISTIR 7443*, September 2007.
- J. Scholtz, J. Young, J. L. Drury, and H. A. Yanco. “Evaluation of Human-Robot Interaction Awareness in Search and Rescue.” In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans, LA, April 2004.

- S. Scott and S. Carpendale. “Investigating Tabletop Territoriality in Digital Tabletop Workspaces.” Technical report, University of Calgary, 2006. Report No. 2006-836-29.
- S. Scott, M. Sheelagh, T. Carpendale, and K. Inkpen. “Territoriality in Collaborative Tabletop Workspaces.” In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 294–303, 2004.
- B. Shneiderman. “Direct Manipulation: A Step Beyond Programming Languages.” *IEEE Computer*, 16:57–69, 1983.
- P. W. Singer. *Wired for War: The Robotics Revolution and Conflict in the 21st Century*. Penguin Press HC, 1st edition, 2009.
- M. Skubic, S. Blisard, A. Carle, and P. Matsakis. “Hand-Drawn Maps for Robot Navigation.” In *Proceedings of the AAAI Spring Symposium on Sketch Understanding*, 2002.
- M. Skubic, C. Bailey, and G. Chronis. “A Sketch Interface for Mobile Robots.” In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 919–924, 2003.
- C. Snyder. *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. Morgan Kaufmann Publishers, 2003.
- R. Soukoreff and I. MacKenzie. “Towards a Standard for Pointing Device Evaluation, Perspectives on 27 Years of Fitts’ Law Research in HCI.” *International Journal of Human-Computer Studies*, 61(6):751–789, 2004.
- H. Tang, H. Van Brussel, D. Reynaerts, J. Vander Sloten, and P. Koninckx. “A Laparoscopic Robot with Intuitive Interface for Gynecological Laser Laparoscopy.” In *Proceedings on the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, 2003.

- H. Tang, H. Van Brussel, J. Sloten, D. Reynaerts, and P. Koninckx. “Implementation of an Intuitive Writing Interface and a Laparoscopic Robot for Gynaecological Laser Assisted Surgery.” *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 219(4):293–302, 2005.
- R. Taylor. “Human Automation Integration for Supervisory Control of UAVs.” *Virtual Media for Military Applications*, pages 12–1 through 12–10, 2006.
- S. Thompson, J. Slocum, and M. Bohan. “Gain and Angle of Approach Effects on Cursor-Positioning Time with a Mouse in Consideration of Fitts’ Law.” In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 48:5, pages 823–827, 2004.
- S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. “Stanley: The Robot That Won the DARPA Grand Challenge.” *Springer Tracts in Advanced Robotics: The 2005 Grand Challenge*, 36:1–43, 2007.
- TouchTable. “TouchTable.” Website, 2007. Available online at <http://www.touchtable.com> (accessed 6 December 2007).
- E. Tse, C. Shen, S. Greenberg, and C. Forlines. “Enabling Interaction With Single User Applications Through Speech and Gestures on a Multi-User Tabletop.” In *Proceedings of the Working Conference on Advanced Visual Interfaces*, 2006.
- P. Tuddenham and P. Robinson. “Distributed Tabletops: Supporting Remote and Mixed-presence Tabletop Collaboration.” In *Proceedings of the 2nd Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (Tabletop 2007)*, pages 19–26, 2007.

- C. Urmson, J. Anhalt, D. Bartz, M. Clark, T. Galatali, A. Gutierrez, S. Harbaugh, J. Johnston, H. Kato, P. Koon, W. Messner, N. Miller, A. Mosher, K. Peterson, C. Ragusa, D. Ray, B. Smith, J. Snider, S. Spiker, J. Struble, J. Ziglar, and W. Whittaker. “A Robust Approach to High-Speed Navigation for Unrehearsed Desert Terrain.” *Springer Tracts in Advanced Robotics: The 2005 Grand Challenge*, 36:45–102, 2007.
- C. Villamor, D. Willis, and L. Wroblewski. “Touch Gesture Reference Guide.” Website, 2010. <http://www.lukew.com/touch/TouchGestureGuide.pdf>.
- F. Wang and X. Ren. “Empirical Evaluation for Finger Input Properties in Multi-Touch Interaction.” In *Conference on Human Factors in Computing Systems*, 2009.
- F. Wang, X. Cao, X. Ren, and P. Irani. “Detecting and Leveraging Finger Orientation for Interaction with Direct-touch Surfaces.” In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 23–32. ACM, 2009.
- A. Wilson. “Depth-Sensing Video Cameras for 3D Tangible Tabletop Interaction.” In *Proceedings of the 2nd Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (Tabletop 2007)*, pages 201–204, 2007.
- J. Wobbrock, M. Morris, and A. Wilson. “User-defined Gestures for Surface Computing.” In *Conference on Human Factors in Computing Systems*, 2009.
- M. Wu, C. Shen, K. Ryall, C. Forlines, and R. Balakrishnan. “Gesture Registration, Relaxation, and Reuse for Multi-Point Direct-Touch Surfaces.” In *Proceedings of the 1st Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (Tabletop 2006)*, pages 183–190, 2006.
- H. Yanco and J. Drury. “Rescuing Interfaces: A Multi-Year Study of Human-Robot Interaction at the AAAI Robot Rescue Competition.” *Autonomous Robots*, 22(4):333–352, 2006.

H. A. Yanco and J. L. Drury. ““Where Am I?” Acquiring Situation Awareness Using a Remote Robot Platform.” In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, October 2004.

H. A. Yanco, B. Keyes, J. L. Drury, C. W. Nielsen, D. A. Few, and D. J. Bruemmer. “Evolving Interface Design for Robot Search Tasks.” *Journal of Field Robotics*, 24, 2007.

S. Zhai. “Characterizing Computer Input with Fitts’ Law Parameters – The Information and Non-information Aspects of Pointing.” *International Journal of Human-Computer Studies*, 61(6):791–809, 2004.

Appendix A

Coding Classifications for Multi-Touch Robot Control

A.1 Timed Event Coding Guidelines

Proportional Velocity: The participant expects speed to vary based on how far from the origin his/her finger is on the control surface. This finger contact creates a continuous (but not necessarily straight) line as the person drags his/her finger across the touch surface. Discrete events are scored when the finger hovers in an area less than the size of their finger width for more than 500ms. If the person switches fingers, these are coded as separate events. Note that there may be overlap between this movement and the Trackpad movement. The key difference the non-repetitive nature of the movement.

Discrete Velocity: The participant clicks on an area on the control and expects the interface to move in that exact direction with no variance in speed or mixing of degrees of freedom. In many cases, this movement is just a simple button press. Any dragging of the finger on the touch surface greater than the width of his/her finger disqualifies the event as a discrete velocity event. Discrete events are scored when the person touches a button and then lifts his/her finger from the button after holding for more than 500ms.

On Axis: The participant artificially limits himself/herself to a control area of translate and rotate axis, but only one component at a time. This event will occur in conjunction with a proportional or discrete event. The control surface has the effect of shaping a control area like a “+”. The participant cannot deviate from the axis by more than the width of their finger for this event to be scored. Discrete events are counted when participant hovers for more than 500ms or lifts finger from board.

Off Axis: The participant provides both translation and rotation in a single gesture on the drive control or video display. This event will occur in conjunction with a proportional or discrete event and is the inverse of the On Axis event. Axis crossings are allowed and are not scored as a separate event unless the participant stops on the axis and generates a new event based on the above velocity criteria. Discrete events are counted when participant hovers for more than 500ms or lifts finger from board.

Trackpad: The participant puts his/her finger down and moves it with a relatively constant speed, but in short bursts in the direction of motion. The finger is lifted between subsequent pushes and placed back in the starting position of the last movement. Discrete events are scored if the participant performs two or more of these motions within two second period and has a one second pause before next maneuver.

Ring: The participant’s fingers enter the the outer ring of the drive control. Discrete events are scored when the person has more than half of his/her finger in the wheel and then lifts the finger or moves away from the outer ring or wheel. This action should be combined with the proportional or discrete velocity events.

A.2 Notes that may be of interest:

Artificial limiting of speed: The participant artificially bounds the motion of his/her finger due to visual elements in the participant interface. For instance, the inner buttons bound the participant from placing his/her fingers on the outer edge of the ring, giving 100% rotation or translation. Discrete events are scored when the person stalls at the artificial boundary for more than 500ms.

Page movement model (only applies to camera): The participant wants to drag image directly, like a physical or PDF document. This event occurs in the “grab document” model where the image is “pulled to center” as opposed to “move window” model where the window is panned to center.

Appendix B

Biographical Sketch of Author

Mark Micire has worked for over a decade to bring robots and other technologies to emergency response and search and rescue. He is certified in multiple aspects of search and rescue including technical search, technical rescue, hazardous material response, and is a nationally certified fire fighter. He is active in the search and rescue community as a search specialist for the Massachusetts FEMA search and rescue team. Mark was a technical search robot operator during the World Trade Center Disaster and was a technical search specialist for Florida Task Force Three during the Hurricane Katrina response in Biloxi, Mississippi. His recent research leverages multi-touch tabletop, cell phone, and robot technology to bridge the gap between responders in the field and the incident command structure that supports them.